BALANCED LEARNING FOR DOMAIN ADAPTIVE SEMANTIC SEGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Unsupervised domain adaptation (UDA) for semantic segmentation aims to transfer knowledge from a labeled source domain to an unlabeled target domain, improving model performance on the target dataset without additional annotations. Despite the effectiveness of self-training techniques in UDA, they struggle to learn each class in a balanced manner due to inherent class imbalance and distribution shift in both data and label space between domains. To address this issue, we propose Balanced Learning for Domain Adaptation (BLDA), a novel approach to directly assess and alleviate class bias without requiring prior knowledge about the distribution shift between domains. First, we identify over-predicted and under-predicted classes by analyzing the distribution of predicted logits. Subsequently, we introduce a post-hoc approach to align the positive and negative logits distributions across different classes using anchor distributions and cumulative density functions. To further consider the network's need to generate unbiased pseudo-labels during self-training, we couple Gaussian mixture models to estimate logits distributions online and incorporate logits correction terms into the loss function. Moreover, we leverage the resulting cumulative density as domain-shared structural knowledge to connect the source and target domains. Extensive experiments on two standard UDA semantic segmentation benchmarks demonstrate that BLDA consistently improves performance, especially for under-predicted classes, when integrated into existing methods. Our work highlights the importance of balanced learning in UDA and effectively mitigates class bias in domain adaptive semantic segmentation.

1 INTRODUCTION

Semantic segmentation is a fundamental computer vision task that assigns a semantic label to each pixel in an image, enabling comprehensive image understanding. Despite the remarkable progress in recent research (Long et al., 2015a; Chen et al., 2017b; Cheng et al., 2021; 2022), the performance of these methods often significantly drops when applied to new target datasets. This performance degradation stems from the differences between the source and target domains, challenging the generalization ability of these models. To address this issue, unsupervised domain adaptation (UDA) techniques have been extensively studied. UDA aims to bridge the domain gap by transferring knowledge from a labeled source domain to an unlabeled target domain, thereby improving the model's performance on the target dataset without requiring additional annotations.

In previous work, self-training techniques (Tranheden et al., 2021; Hoyer et al., 2022a) have been naturally introduced into UDA tasks to fully utilize the large amount of unlabeled target domain data, becoming a mainstream paradigm. This paradigm constructs a teacher network using a temporal aggregation mechanism, treats its predictions on the target domain as pseudo-labels, and gradually guides the student network's learning. Despite achieving remarkable results, these methods struggle to learn each class in a balanced manner. Generally, the inherent class imbalance in segmentation datasets (Cordts et al., 2016) (Fig.1(a)) leads networks to produce biased predictions towards head classes, often studied as the long-tail problem (Van Horn & Perona, 2017; Buda et al., 2018; Liu et al., 2019). However, in UDA, data and label distribution shifts between the training and test data complicate the class bias. The network's bias towards classes does not entirely depend on the differences in class sample distribution. As shown in Fig.1(b), when a network trained on the source domain is tested on the target domain, the performance degradation varies greatly across classes, distinguishing easy-to-transfer and hard-to-transfer classes. These factors jointly determine

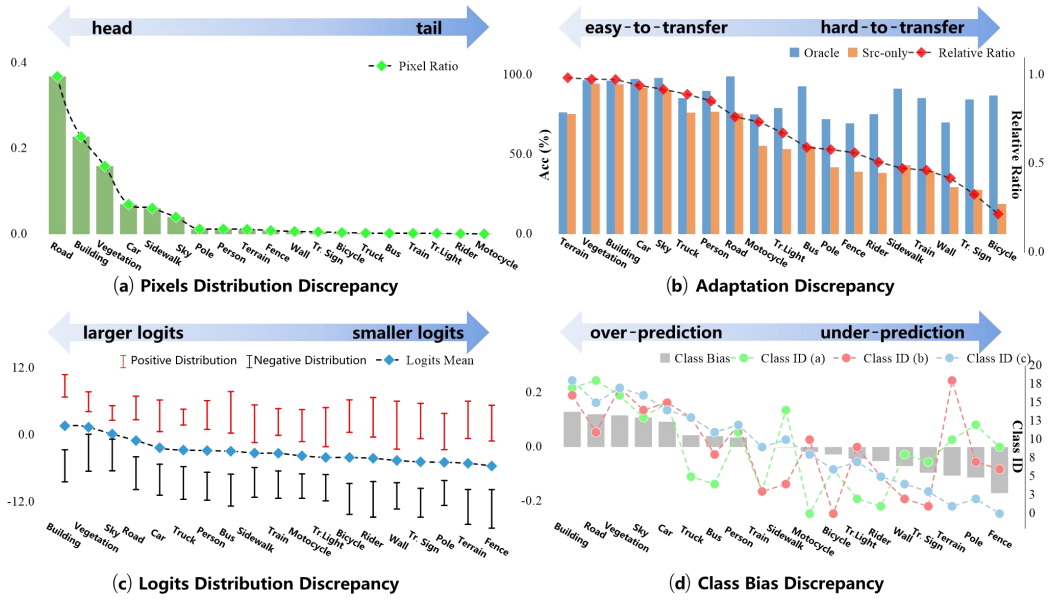
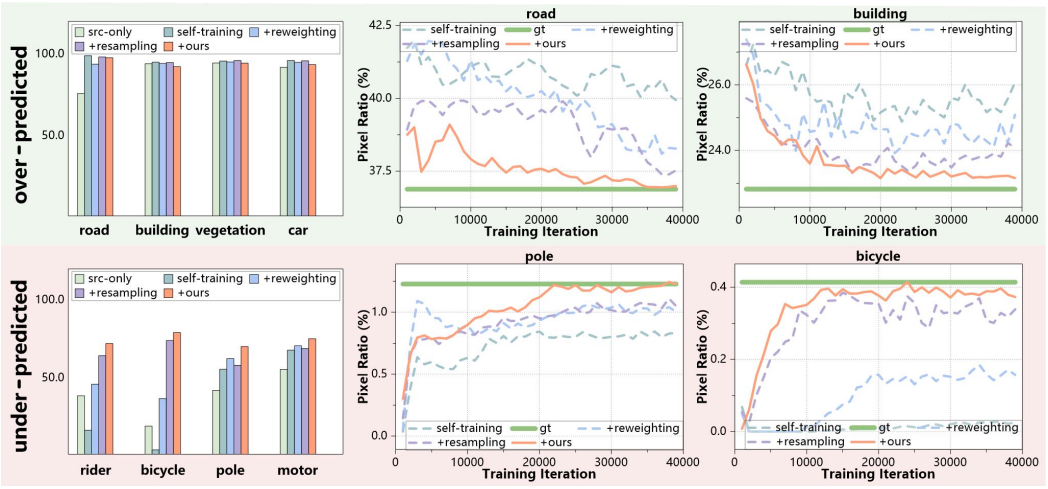


Figure 1: Demonstration of factors that cause class bias. (a) The inherent class imbalance problem in segmentation datasets. (b) The differences in transfer difficulty across classes in cross-domain settings. "Oracle' represents the performance under full supervision, while "Src-only' represents training with the source domain and testing it on the target domain. (c) The differences in logits distributions predicted for each class by the network, including "positive distribution" and "negative distribution". (d) Bias assessment for different classes via Eq.4. The corresponding class IDs of (a), (b), and (c) are mapped in descending order onto this figure.

the network's different biases towards each class in target domain, resulting in over-prediction and under-prediction. Furthermore, confirmation bias (Guo et al., 2017) causes self-training techniques to exacerbate this phenomenon. Fig.2(a) shows the severe deterioration of classes like *rider* and *bicycle* after self-training, widening the performance gap across classes. Therefore, achieving balanced learning for each class in UDA is a challenging and worthwhile exploration.

Existing strategies to reduce model bias towards different classes can be broadly categorized into re-weighting (Cui et al., 2019; Lin et al., 2017; Cao et al., 2019; Truong et al., 2023; Buda et al., 2018) and re-sampling (Hoyer et al., 2022a; Araslanov & Roth, 2021; He et al., 2008; 2021; Guan et al., 2022). To compare these methods, we take self-training as the baseline method in Fig.2 and implement re-weighting (Cui et al., 2019) and re-sampling (Hoyer et al., 2022a) techniques, respectively. We observe the class bias through class-wise accuracy (Fig.2(a)) and the frequency of pseudo-labels generated on the target domain during training (Fig.2(b)). Loss re-weighting aims to assign different weights to classes, making the model pay more attention to tail classes. Although intuitive, the update frequency of each class to the network still varies greatly, with some classes remaining challenging to learn effectively, resulting in unstable performance in self-training. In contrast, sample re-sampling proves more effective by directly adjusting the class sample distribution during training, significantly enhancing the performance of tail classes. Despite their empirical solid performance, these methods are heuristic and rely on the assumption that the test and training data share the same distribution in both data and label space. However, in the UDA setting, these assumptions are invalid because (1) the class distributions of the source and target domains differ, and the target domain's prior class distribution is unavailable; (2) the data distributions also differ, leading to varying transfer difficulties across classes in cross-domain settings. This raises the question: *How to assess and alleviate class bias directly without requiring prior knowledge about the distribution shift between the two domains?*

In this work, we propose to assess the degree of class bias by analyzing the distribution of logits predicted by the network (Sec.3.3.2). Fig.1(c) shows that the network exhibits differences in the predicted logits distributions for different classes, directly leading to class bias. Fig.1(d) illustrates that the ranking of class bias highly coincides with the ranking of logit distribution differences, i.e.,



(a) Class-wise Accuracy (b) Pseudo-label Frequency

Figure 2: In UDA, class bias can be expressed as over-predicted classes and under-predicted classes. (a) Class-wise accuracy under different training settings. (b) Frequency of pseudo-labels generated by the network for different classes during training.

over-predicted classes have larger logit values, while under-predicted classes have smaller logit values. This assessment approach prompts us to propose BLDA, a method to achieve balanced learning for domain adaptive semantic segmentation by balancing the logits distribution. First, we consider a post-hoc approach to adjust the logits (Sec.3.3.3). We set shared anchor distributions for the positive and negative logits distributions and align the class-wise logits distributions with the anchor distributions based on the cumulative density function mapping. Furthermore, to generate unbiased pseudo-labels for classes during self-training, we propose an online logit adjustment method (Sec.3.3.4). This strategy couples Gaussian mixture models to estimate the logits distributions online during training and incorporates logit correction terms into the loss function to replace the post-hoc method. Moreover, we find that the resulting cumulative density can measure the discrimination difficulty of different sample points in each class, which is a domain-shared structural knowledge that can be used as an auxiliary loss to connect the two domains, further enhancing domain adaptation performance (Sec.3.3.5). As shown in Fig.2, our method can be integrated into existing self-training-based UDA paradigms and effectively balance the prediction bias across classes.

Our contributions can be summarized as follows: (1) We propose statistically analyzing the predicted logits to directly assess the network's bias towards classes. (2) We propose BLDA to estimate logit correction terms online during UDA training to achieve balanced learning for each class. (3) We demonstrate that BLDA can be easily integrated into existing UDA methods and consistently improves performance on two standard UDA semantic segmentation benchmarks, significantly enhancing the performance of under-predicted classes and confirming the effectiveness of our method.

2 Related Work

2.1 Domain Adaptive Semantic Segmentation

Unsupervised domain adaptation (UDA) aims to transfer semantic knowledge learned from labeled source domains to unlabeled target domains. Due to the ubiquity of domain gaps, UDA methods have been widely studied in various computer vision tasks, such as image classification, object detection, and semantic segmentation. UDA is crucial for semantic segmentation to avoid laborious pixel-wise annotation in new target scenarios. Recent UDA approaches for semantic segmentation can be categorized into two main paradigms: adversarial training-based methods (Toldo et al., 2020; Tsai et al., 2018; Chen et al., 2018; Ganin & Lempitsky, 2015; Hong et al., 2018; Long et al., 2015b) and self-training-based methods (Tranheden et al., 2021; Hoyer et al., 2022a; Araslanov & Roth, 2021; Zhang et al., 2021). Adversarial training-based methods learn domain-invariant representations

through a min-max optimization game, where a feature extractor is trained to confuse a domain discriminator, aligning feature distributions across domains. Self-training-based methods, which have come to dominate the field due to the domain-robustness of Transformers (Bhojanapalli et al., 2021), generate pseudo labels for target images based on a teacher-student optimization framework. The success of this paradigm relies on generating high-quality pseudo labels, with strategies such as entropy minimization (Chen et al., 2019) and consistency regularization (Hoyer et al., 2023) being developed for this purpose. However, due to the inherent class imbalance and distribution shift in both data and label space between domains, networks often produce complicated class bias, which is further exacerbated by confirmation bias in the self-training paradigm. Our method focuses on balanced learning in UDA training.

2.2 Class-Imbalanced Learning

Class imbalance is a common problem in semantic segmentation, where the number of samples per class varies significantly. Existing methods address this issue through re-weighting or re-sampling techniques. Re-weighting methods assign different weights to classes during training, giving higher importance to under-represented classes (Cui et al., 2019; Lin et al., 2017; Cao et al., 2019). Re-sampling techniques modify the class distribution in the training data by over-sampling minority classes or under-sampling majority classes (He et al., 2008; 2021). In UDA for semantic segmentation, some approaches have introduced these strategies to alleviate class bias (Hoyer et al., 2022a; Araslanov & Roth, 2021; Li et al., 2022). However, these methods are still empirical and focus on the single-domain setting, which follows the assumptions that the test data and training data share the same distribution in both data space and label space, without considering the additional challenges posed by domain shift in UDA. In this work, we aim to access class bias directly and achieve balanced learning for each class with no prior knowledge about the distribution shift between domains.

3 Method

3.1 Problem Definition

In unsupervised domain adaptation for semantic segmentation, the network is simultaneously trained on labeled source domain data and unlabeled target domain data. To be specific, the source domain can be denoted as $D_s = \{(x_i^S, y_i^S)\}_{i=1}^{N_S}$, where $x_i^S \in X_S$ represents an image with $y_i^S \in Y_S$ as the corresponding pixel-wise one-hot label covering C classes. The target domain can be denoted as $D_t = \{(x_i^T)\}_{i=1}^{N_T}$, which shares the same label space but has no access to the target label Y_T.

3.2 Revisiting Self-training in UDA

Self-training-based pipelines for UDA segmentation consist of a supervised branch for the source domain and an unsupervised branch for the target domain. For the supervised branch, loss $\mathcal{L}^s$ can only be calculated on the source domain to train a neural network f_θ:

$$\mathcal{L}^s = \frac{1}{N_S} \sum_{i=1}^{N_S} \frac{1}{HW} \sum_{j=1}^{H \times W} \ell_{ce}(f_\theta(x_{ij}^S), y_{ij}^S), \tag{1}$$

where ℓ_{ce} denotes the cross-entropy loss. Unsupervised branch introduces teacher-student framework to generate pseudo-labels $\hat{y}_{ij}^T = \mathrm{argmax}(g_\phi(x_{ij}^T))$ with the teacher model g_ϕ for target domain:

$$\mathcal{L}^u = \frac{1}{N_T} \sum_{i=1}^{N_T} \frac{1}{HW} \sum_{j=1}^{H \times W} q(p_{ij}) \ell_{ce}(f_\theta(x_{ij}^T), \hat{y}_{ij}^T), \tag{2}$$

where we define $q(p_{ij})$ as a quality estimate conditioned on confidence $p_{ij} = max(g_\phi(x_{ij}^T))$ for pseudo labels, which gradually strengthens with increasing accuracy of models and can be implemented with threshold filtering or a weighting function. After each training step, the teacher model g_ϕ is updated with the exponentially moving average of the weights of f_θ. Then, the overall objective function is a combination of supervised loss and unsupervised loss as $\mathcal{L} = \mathcal{L}^s + \mathcal{L}^u$.

3.3 BALANCED LEARNING FOR DOMAIN ADAPTIVE SEMANTIC SEGMENTATION

3.3.1 OVERVIEW

In this section, we first propose to assess the network's prediction bias towards each class by statistically analyzing the distribution of logits (Sec.3.3.2). Based on the above analysis, we define a post-hoc method to balance the network's predictions (Sec.3.3.3). Furthermore, we introduce online logits adjustment tailored for the UDA training process (Sec.3.3.4). Finally, we introduce cumulative density estimation as domain-shared knowledge to bridge the two domains (Sec.3.3.5).

3.3.2 ASSESSING PREDICTION BIAS FROM LOGITS DISTRIBUTION

Given a label space $\mathcal{Y} = [C] = \{1, 2, ..., C\}$, the segmentation network can be seen as a scorer $f_\theta : x_{ij} \to \mathcal{R}^C$ that assigns class-wise scores, also known as logits, to a pixel x_{ij} from image x_i. To investigate the distribution of logits obtained by the network for different classes, we can analyze it from the perspective of the confusion matrix. The confusion matrix is a $C \times C$ matrix M, where each element M_{cl} represents the number of pixels with ground truth label c predicted as class l. We replace each element M_{cl} in the confusion matrix M with the corresponding set of logits, i.e., the logits predicted for class l for all pixels with ground truth label c, to obtain the logits set matrix $\mathcal{M}$. We then use $\mathcal{M}$ to assess prediction bias.

Definition 1. Element in logits set matrix, $\mathcal{M}_{cl}$.

$$\mathcal{M}_{cl} = \{f_\theta(x_{ij})[l] \mid y_{ij} = c\}, \tag{3}$$

where $f_\theta(x_{ij})[l]$ represents the logit value predicted by the network for class l of pixel x_{ij}, and y_{ij} is the ground truth label of pixel x_{ij}. In the resulting $C \times C$ matrix $\mathcal{M}$, diagonal elements $\mathcal{M}_{ll}$ represent the "positive logits distribution" for class l, while off-diagonal elements $\mathcal{M}_{cl}$ $(c \neq l)$ represent the "negative logits distribution" for class l with respect to class c.

Definition 2. Bias of the network towards class l, $\mathrm{Bias}(l)$.

$$\mathrm{Bias}(l) = \frac{1}{C} \sum_{c \in [C]} \mathbb{P}(\arg\max_{c' \in [C]} f_\theta(x)[c'] = l | y = c) - \frac{1}{C}, \tag{4}$$

where $\mathbb{P}(\arg\max_{c' \in [C]} f_\theta(x)[c'] = l | y = c)$ represents the probability that the network predicts a sample from class c as class l. This definition measures the average difference between the probability of predicting class l and the uniform probability $1/C$ across all classes. A positive bias indicates over-prediction, while a negative bias indicates under-prediction for class l.

Let P_{cl} denote the distribution of $\mathcal{M}_{cl}$. Assuming each distribution P_{cl} is independent, we can estimate the $\mathbb{P}(l|c)$ by comparing the logit values:

$$\mathbb{P}(\arg\max_{y' \in [C]} f_\theta(x)[y'] = l | c) \approx \int_{-\infty}^{\infty} P_{cl}(z) \prod_{y' \neq l} \left(\int_{-\infty}^{z} P_{cy'}(t) dt \right) dz. \tag{5}$$

Combining Eq.4 and Eq.5, for an unbiased network, i.e., $\mathrm{Bias}(l) = 0$ for all $l \in [C]$, a sufficient condition is that they have the same positive and negative distributions. This means the network's prediction performance is consistent across all classes. Fig.1(d) shows a direct correlation between logit distribution differences and class bias, indicating that variations in logit distributions lead to class bias in the network's predictions.

3.3.3 POST-HOC CLASS BALANCING

Generally, the network tends to produce larger logits for over-predicted classes and smaller logits for under-predicted classes, as shown in Fig.3 (a). Reweighting/resampling strategies can alleviate this gap by making the network pay more attention to tail classes and reducing the emphasis on head classes during training, as illustrated in Fig.3(b). However, as shown in Fig.1, class bias does not fully correlate with the inherent class imbalance problem, especially in the UDA setting, where different distribution shifts exist in both data and label space between domains. Furthermore, these methods are empirical and lack generalization capability across various scenarios.

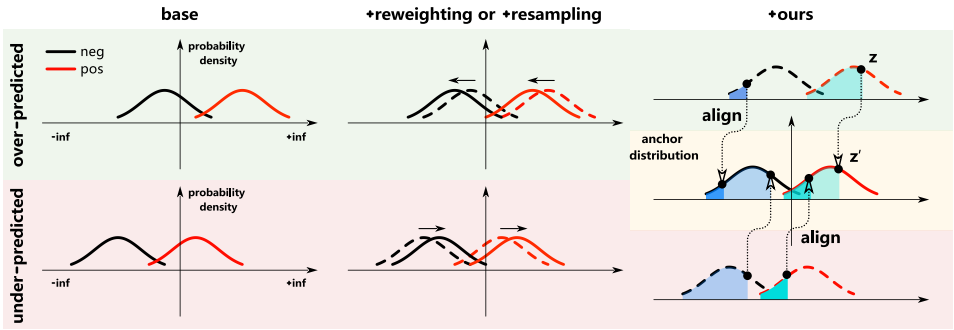


Figure 3: Illustration of proposed post-hoc class balancing. (a) The logits distributions of over-predicted and under-predicted classes. (b) Reweighting/resampling strategies alleviate class imbalance by adjusting the training emphasis on different classes. (c) Our post-hoc logits adjustment method aligns the logits distributions of all classes with anchor distributions to achieve balanced prediction.

Based on the above analysis, to balance the network's prediction capabilities across classes, we adjust the network's predictions in a post-hoc manner. Specifically, we define an anchor distribution P_p for the positive logits distribution and an anchor distribution P_n for the negative logits distribution. We then align all the logits distributions with corresponding anchor distributions, as shown in Fig.3 (c). To preserve the relative ordering of logits, i.e., structural information within each distribution, we align them in a point-wise way via the cumulative distribution function (CDF). Let $F_{cl}(z)$, $F_p(z)$ and $F_n(z)$ be the CDFs of P_{cl}, P_p and P_n, respectively. We align the logit value z from P_{cl} to P_p or P_n as follows:

$$z' = \begin{cases} F_p^{-1}(F_{cl}(z)), & \text{if } c = l \\ F_n^{-1}(F_{cl}(z)), & \text{if } c \neq l \end{cases} \tag{6}$$

where z' is the aligned value of z with respect to the anchor distribution. For brevity, we define an offset for logit as $\Delta_{cl}(z) = z' - z$. Considering the probability estimate that $p(y_{ij} = c|x_{ij}) \propto \exp(f_\theta(x_{ij})[c])$, we can obtain revised prediction results for each pixel x_{ij} by:

$$\widetilde{y}_{ij} = \text{argmax}_{c\in[C]} \frac{\exp(f_\theta(x_{ij})[c] + \tau\Delta_{cc}(f_\theta(x_{ij})[c]))}{\exp(f_\theta(x_{ij})[c] + \tau\Delta_{cc}(f_\theta(x_{ij})[c])) + \sum_{c'\neq c}\exp(f_\theta(x_{ij})[c'] + \tau\Delta_{cc'}(f_\theta(x_{ij})[c']))}, \tag{7}$$

where τ is a scaling factor (the derivation of Eq.7 is detailed in Appendix A). When $\tau = 1$, the model produces balanced predictions. However, to achieve optimal performance on specific evaluation metrics (e.g., mIoU), we need to adjust the value of τ. We discuss the choice of τ in detail in the experimental section. In this way, the network generates balanced predictions for different classes.

Discussion about anchor distribution. In our experiments, we use the global positive and negative logits distributions on the source domain to estimate the anchor distribution for both source and target domains. This choice is based on two key considerations: (1) The network tends to produce larger logits for over-predicted classes and smaller logits for under-predicted classes. By using the global logits distribution, we can effectively measure the average learning degree of the network across all classes. Aligning each class-specific distribution with this global distribution can help neutralize the class bias of the network, ensuring a more balanced learning process. (2) When estimating the logits distributions for each class, there exist varying degrees of statistical errors. According to Bernstein inequalities, estimating the global logits distribution can reduce estimation errors to a certain extent and accelerate convergence rates. In our case, the global distribution, being more robust and stable, serves as a reliable anchor distribution for subsequent alignment.

3.3.4 Online Logits Adjustment for UDA

While the post-hoc logits adjustment method introduced in Sec.3.3.3 can effectively balance the network predictions across different classes, it is performed after training the model. In the UDA setting, it is significant to incorporate the logits balancing mechanism directly into the training process. By doing so, the model can learn to make more balanced predictions while adapting to the target domain through pseudo labels.

To achieve this, we propose an online logit adjustment method tailored for UDA training. The key to this method lies in the online estimation of the logits distributions. We employ Gaussian Mixture Models (GMMs) to model these distributions. Considering that the source and target domains have inherently different logits distributions, we maintain two sets of GMMs separately, with each set containing $C \times C \times K$ Gaussian components, where C denotes the number of classes and K denotes the number of Gaussian components per element in $\mathcal{M}$. Formally, we define:

$$P_{cl}^s = \sum_{k=1}^{K} \pi_{clk}^s \mathcal{N}(\mu_{clk}^s, \sigma_{clk}^s), \quad P_{cl}^t = \sum_{k=1}^{K} \pi_{clk}^t \mathcal{N}(\mu_{clk}^t, \sigma_{clk}^t), \tag{8}$$

where P_{cl}^s and P_{cl}^t represent the estimated logits distributions for class l when the ground truth label or pseudo label is c in the source and target domains, respectively. The parameters π_{clk}^s, μ_{clk}^s, σ_{clk}^s and π_{clk}^t, μ_{clk}^t, σ_{clk}^t denote the mixing coefficient, mean, and standard deviation of the k-th Gaussian component in the corresponding GMM. We update the GMM parameters during each training iteration using the logits obtained from the current mini-batch via the Expectation-Maximization (EM) algorithm (McLachlan & Krishnan, 2007). Since the network f_θ gradually evolves during training, we adopt a momentum-based EM update strategy. Specifically, we directly use the GMM parameters $\hat{\phi}_{cl}$ estimated in the latest iteration as the initialization $\phi_{cl}^{(0)}$ for the current iteration. After T EM loops, the current iteration is completed, and a momentum update is adapted with $\phi_{cl}^{(T)} \leftarrow (1-\tilde{\tau}^n)\phi_{cl}^{(T)} + \tilde{\tau}^n \hat{\phi}_{cl}$, where n represents the number of iterations that have not been updated since the last parameter update for ϕ_{cl}. We also implement GMMs to estimate the anchor distributions P_p and P_n using the source domain data's global positive and negative logits. The algorithm flow is detailed in Appendix D.

After estimating the logits distributions using GMMs, we compute the adjusted logits offset Δ^S and Δ^T for source and target domains, respectively. These offsets are then used to adjust the cross-entropy loss for both domains:

$$\begin{aligned}
\widetilde{\mathcal{L}}^s &= -\frac{1}{N_S} \sum_{i=1}^{N_S} \sum_{j=1}^{H \times W} \log \frac{\exp(f_\theta(x_{ij}^S)[y_{ij}^S] - \tau \Delta_{y_{ij}^S, y_{ij}^S}^S (f_\theta(x_{ij}^S)[y_{ij}^S]))}{\sum_{c=1}^{C} \exp(f_\theta(x_{ij}^S)[c] - \tau \Delta_{y_{ij}^S, c}^S (f_\theta(x_{ij}^S)[c]))}, \\
\widetilde{\mathcal{L}}^u &= -\frac{1}{N_T} \sum_{i=1}^{N_T} \sum_{j=1}^{H \times W} q(p_{ij}) \log \frac{\exp(f_\theta(x_{ij}^T)[\hat{y}_{ij}^T] - \tau \Delta_{\hat{y}_{ij}^T, \hat{y}_{ij}^T}^T (f_\theta(x_{ij}^T)[\hat{y}_{ij}^T]))}{\sum_{c=1}^{C} \exp(f_\theta(x_{ij}^T)[c] - \tau \Delta_{\hat{y}_{ij}^T, c}^T (f_\theta(x_{ij}^T)[c]))}.
\end{aligned} \tag{9}$$

In contrast to Eq.7, where the logits are adjusted post-hoc, Eq.9 directly incorporates the offset into the learning process of the logits. This approach is equivalent to learning a scorer of the form $g(x)[y] = f(x)[y] - \tau \Delta_y(x)$. Consequently, we have $\mathrm{argmax} f(x)[y] = g(x)[y] + \tau \Delta_y(x)$, which can be seen as analogous to the post-hoc adjustment. By employing these adjusted losses, we achieve a two-fold benefit. Firstly, the anchor distribution can serve as a reference distribution to balance the learning progress between classes within both domains. Secondly, since the pseudo-label-based loss in the target domain has a gradually increasing weight, using a shared anchor distribution allows the logits distribution of the target domain to gradually align with that of the source domain, thus establishing a connection between the two domains.

3.3.5 Bridging Domains through Cumulative Density Estimation

Furthermore, for each sample pixel, we can query the corresponding positive cumulative distribution value F_{cc} based on its label c, which ranges from 0 to 1. The positive distribution measures the discriminative ability of a class, and we find that this cumulative distribution value indicates the difficulty of the sample pixel belonging to that class. This structural knowledge depends only on the context of the pixel and is not affected by the image style, making it domain-invariant. To further bridge the two domains, we add an extra regression head to the network to predict this value as an additional auxiliary task.

Specifically, for each sample point x_{ij}^S in the source domain, we can query the corresponding positive cumulative distribution value based on its true label y_{ij}^S: $d_{ij}^S = F_{y_{ij}^S, y_{ij}^S}(f_\theta(x_{ij}^S)[y_{ij}^S])$, where $F_{y_{ij}^S, y_{ij}^S}$ is the positive cumulative distribution function for class y_{ij}^S. Similarly, for each sample point x_{ij}^T in the target domain, we can query the corresponding positive cumulative distribution value based on its

Table 1: UDA segmentation performance on GTA.→CS. using the mIoU (%) evaluation metric, where the improvement is marked as **bold**. The results are acquired based on CNN-based model (He et al., 2016; Chen et al., 2017a), denoted as C, and Transformer-based model (Xie et al., 2021), denoted as T.

Method	Arch.	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Veg	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motor	Bike	mIoU	std
DACS (Tranheden et al., 2021)	C	89.9	39.7	87.9	39.7	39.5	38.5	46.4	52.8	88.0	44.0	88.8	67.2	35.8	84.5	45.7	50.2	0.2	27.3	34.0	52.1	24.4
ProDA (Zhang et al., 2021)	C	87.8	56.0	79.7	46.3	44.8	45.6	53.5	53.5	88.6	45.2	82.1	70.7	39.2	88.8	45.5	50.4	1.0	48.9	56.4	57.5	21.2
CPSL (Li et al., 2022)	C	92.3	59.5	84.9	45.7	29.7	52.8	61.5	59.5	87.9	41.6	85.0	73.0	35.5	90.4	48.7	73.9	26.3	53.8	53.9	60.8	20.3
TransDA (Chen et al., 2022)	T	94.7	64.2	89.2	48.1	45.8	50.1	60.2	40.8	90.4	50.2	93.7	76.7	47.6	92.5	56.8	60.1	47.6	49.6	55.4	63.9	18.5
DAFormer(Hoyer et al., 2022a)	T	95.7	70.2	89.4	53.5	48.1	49.6	55.8	59.4	89.9	47.9	92.5	72.2	44.7	92.3	74.5	78.2	65.1	55.9	61.8	68.3	16.8
+BLDA	T	95.4	**78.3**	88.3	**54.0**	**55.2**	**55.7**	**60.3**	**65.2**	89.2	47.3	91.1	71.4	**44.8**	91.6	74.3	**83.4**	**73.2**	**59.3**	**67.1**	**70.7**	**15.5**
HRDA (Hoyer et al., 2022b)	T	96.5	74.4	91.0	61.6	51.5	57.1	63.9	69.3	91.3	48.4	94.2	79.0	52.9	93.9	84.1	85.7	75.9	63.9	67.5	73.8	15.4
+BLDA	T	96.4	**77.6**	90.7	**63.3**	**57.9**	**62.1**	**66.5**	**72.5**	91.3	**52.2**	**94.4**	76.9	**57.3**	93.5	**86.2**	**87.7**	**79.9**	**66.8**	**68.9**	**75.6**	**13.8**
MIC (Hoyer et al., 2023)	T	97.4	80.1	91.7	61.2	56.9	59.7	66.0	71.3	91.7	51.4	94.3	79.8	56.1	94.6	85.4	90.3	80.4	64.5	68.5	75.9	14.8
+BLDA	T	97.1	**82.6**	91.6	**64.7**	**61.0**	**64.9**	**68.0**	**74.8**	91.2	**56.6**	92.4	**80.0**	54.7	**95.7**	**87.3**	88.8	**82.6**	64.2	**72.0**	**77.1**	**13.5**

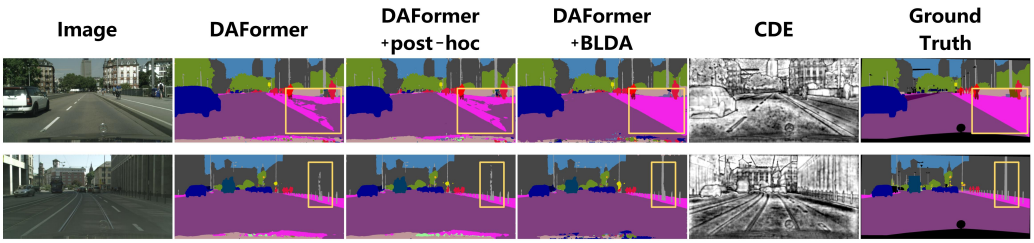


Figure 4: Qualitative results. Note that the yellow boxes mark regions improved by BLDA.

pseudo label $\hat{y}^T_{ij}$: $d^T_{ij} = F_{\hat{y}^T_{ij},\hat{y}^T_{ij}}(f_\theta(x^T_{ij})[\hat{y}^T_{ij}])$, where $F_{\hat{y}^T_{ij},\hat{y}^T_{ij}}$ is the positive cumulative distribution function for the class corresponding to the pseudo label $\hat{y}^T_{ij}$. We then add an extra regression head h_ϕ to the network to predict the cumulative distribution value for each sample point. The corresponding regression losses for the source and target domains can be defined as:

$$\mathcal{L}^S_{reg} = \frac{1}{N_S}\sum_{i=1}^{N_S}\sum_{j=1}^{H\times W}|h_\phi(\tilde{f}_\theta(x^S_{ij})) - d^S_{ij}|^2, \mathcal{L}^T_{reg} = \frac{1}{N_T}\sum_{i=1}^{N_T}\sum_{j=1}^{H\times W} q(p_{ij})|h_\phi(\tilde{f}_\theta(x^T_{ij})) - d^T_{ij}|^2, \quad (10)$$

where $|\cdot|^2$ denotes the L2 loss, and $\tilde{f}_\theta(x_{ij})$ denote the features extracted by the network f_θ. Finally, the overall training objective can be expressed as $\mathcal{L} = \widetilde{\mathcal{L}}^s + \widetilde{\mathcal{L}}^u + \lambda(\mathcal{L}^S_{reg} + \mathcal{L}^T_{reg})$, where λ is a hyperparameter balancing the cumulative density estimation loss.

4 Experiments

4.1 Experimental Setup

Datasets. Following standard UDA protocols, we evaluate our method on two widely used benchmarks that involve transferring knowledge from a synthetic domain to a real domain in a street scene setting. Specifically, we use GTAv/SYNTHIA (Ros et al., 2016; Richter et al., 2016) as the labeled source domain and Cityscapes (Cordts et al., 2016) as the unlabeled target domain. GTAv contains 24,966 synthetic images with a resolution of 1914×1052, while SYNTHIA consists of 9,400 synthetic images with a resolution of 1280×960.

Implementation Details. Our method can be built with different self-training-based frameworks. For thorough evaluation, we apply BLDA to three strong baseline methods, i.e., DAFormer (Hoyer et al., 2022a) , HRDA (Hoyer et al., 2022b), and MIC (Hoyer et al., 2023), with MiT-B5 (Xie et al., 2021) pretrained on ImageNet-1k (Deng et al., 2009) as the backbone. BLDA is implemented based on MMSegmentation (Contributors, 2020). All experiments are trained for 40K iterations and a batch size of 2, with one or two RTX-3090 (24 GB memory) GPUs, depending on the complexity of used UDA frameworks. We train the network with an AdamW optimizer with learning rates of 6×10^{-5} for the encoder and 6×10^{-4} for the decoder, a weight decay of 0.01, and linear learning rate warm-up for the first 1.5K iterations. The input images are rescaled and randomly cropped to

Table 2: UDA segmentation performance on SYN.→CS. using the mIoU (%) evaluation metric, where the improvement is marked as **bold**. Note that the mIoUs on are calculated over 16 classes.

Method	Arch.	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Veg	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motor	Bike	mIoU	std
DACS (Tranheden et al., 2021)	C	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	27.5
ProDA (Zhang et al., 2021)	C	87.8	45.7	84.6	37.1	0.6	44.0	54.6	37.0	88.1	-	84.4	74.2	24.3	88.2	-	51.1	-	40.5	45.6	55.5	25.6
CPSL (Li et al., 2022)	C	87.2	43.9	85.5	33.6	0.3	47.7	57.4	37.2	87.8	-	88.5	79.0	32.0	90.6	-	49.4	-	50.8	59.8	57.9	25.5
TransDA (Chen et al., 2022)	T	90.4	54.8	86.4	31.1	1.7	53.8	61.1	37.1	90.3	-	93.0	71.2	25.3	92.3	-	66.0	-	44.4	49.8	59.3	26.5
DAFormer(Hoyer et al., 2022a)	T	84.5	40.7	88.4	41.5	6.5	50.0	55.0	54.6	86.0	-	89.8	73.2	48.2	87.2	-	53.2	-	53.9	61.7	60.9	22.1
+BLDA	T	80.7	**44.9**	85.6	**45.1**	**9.6**	**54.3**	**60.2**	**58.7**	**87.7**	-	**92.3**	**75.7**	**51.1**	**87.3**	-	**62.7**	-	**59.9**	**65.8**	**64.0**	**20.6**
HRDA (Hoyer et al., 2022b)	T	85.2	47.7	88.8	49.5	4.8	57.2	65.7	60.9	85.3	-	92.9	79.4	52.8	89.0	-	64.7	-	63.9	64.9	65.8	21.4
+BLDA	T	83.9	**54.9**	87.5	**53.1**	**11.5**	**63.2**	**69.4**	**64.4**	**87.2**	-	**93.1**	79.1	**54.7**	88.3	-	**69.1**	-	**64.2**	**65.7**	**67.9**	**19.3**
MIC (Hoyer et al., 2023)	T	86.6	50.5	89.3	47.9	7.8	59.4	66.7	63.4	87.1	-	94.6	81.0	58.9	90.1	-	61.9	-	67.1	64.3	67.3	21.0
+BLDA	T	86.1	**61.2**	**89.8**	47.2	**10.2**	**62.6**	**70.3**	**67.1**	**90.0**	-	94.4	**81.4**	56.4	**90.5**	-	**67.2**	-	64.8	**66.3**	**69.1**	**20.4**

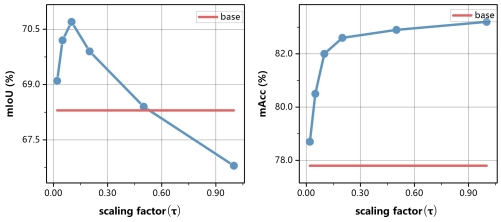


Figure 5: Study of the different evalution metrics with respect to scaling factor τ.

Table 3: BLDA ablation study of different components built with DAFormer.

None	Post-hoc	OLA_S	OLA	CDE	mIoU	mAcc
✓					68.3	77.8
	✓				69.2	80.3
		✓			68.9	79.5
			✓		70.2	81.8
			✓	✓	**70.7**	**82.0**

512×512 following the same data augmentation in DAFormer (Hoyer et al., 2022a), and the EMA coefficient for updating the teacher net is set to be 0.999. We set temperature coefficient $\tau = 0.1$ and loss weight $\lambda = 0.2$ respectively. For more details, please refer to the Appendix.

4.2 Comparison with Existing Methods

Comparative Evaluation. We compare BLDA to existing state-of-the-art UDA approaches on the GTA.→CS. and SYN.→CS. benchmarks. In addition to the widely adopted mean intersection-over-union (mIoU) metric, we also report the mean accuracy (mAcc) metric, which is equivalent to measuring the balanced error (Menon et al., 2020), i.e., the average of each class's error rate, and is more suitable to be used to assess the balance among classes. We discuss these two metrics in detail in Appendix C. Additionally, we calculate the standard deviation of IoU and Acc for each class to reflect the balanced degree of performance across classes.

Evaluation Results. Tab.1-2 shows BLDA consistently improves the performance of all baseline methods on two benchmarks by a large margin, ranging from 1.2% to 3.1%. Furthermore, significant improvements are obtained for under-predicted classes, such as *sidewalk*, *fence*, *pole*, *light*, and *sign*, which demonstrates that BLDA can mitigate the class bias with decreased standard deviation and thus bring the performance gains. In Table 4, the same phenomenon is observed, and the improvement in mAcc is more significant, ranging from 2.9% to 4.2%.

4.3 Ablation Study

We conduct a series of ablation studies on the GTA.→CS. benchmark built with DAFormer (Hoyer et al., 2022a). Please refer to the Appendix for further analysis, where we provide a deeper study of parameter settings and more visualization results.

Influence of scaling factor. As illustrated in Fig. 5, we explore the influence of different τ values on evaluation metrics. The mAcc metric gradually increases as τ grows, reaching its peak performance when $\tau = 1$. Interestingly, the mIoU metric does not demonstrate a perfectly positive correlation with the rise in mAcc. This discrepancy arises from the fact that the mIoU calculation is heavily impacted by the imbalanced distribution of the test set, whereas mAcc serves as a class-balanced metric. We comprehensively explain this phenomenon in Appendix C. Our method, which models class-balanced learning, effectively boosts mAcc. However, to achieve improvements in mIoU, selecting a smaller scaling factor τ is necessary.

Table 4: UDA segmentation performance on GTA.→CS. using the mAcc (%) evaluation metric, where the improvement is marked as **bold**.

Method	Arch.	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Veg	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motor	Bike	mAcc	std
DAFormer(Hoyer et al., 2022a)	T	99.1	74.8	95.2	61.0	53.1	59.8	70.7	68.6	96.0	63.4	98.7	84.2	69.9	95.5	88.9	84.1	74.0	70.0	69.7	77.8	14.2
+BLDA	T	98.3	**86.7**	93.4	**70.9**	**61.9**	**66.0**	**74.0**	**78.9**	95.4	59.0	98.6	**85.7**	**73.1**	95.3	**89.8**	**89.3**	**85.4**	**77.2**	**78.6**	**82.0**	**11.9**
HRDA (Hoyer et al., 2022b)	T	99.1	85.6	96.0	72.5	56.3	69.4	83.9	79.6	96.1	59.1	98.5	89.8	60.7	96.2	91.3	89.4	80.2	77.0	80.9	82.2	13.2
+BLDA	T	**99.2**	**87.2**	95.0	64.9	**68.2**	**72.7**	**88.3**	79.5	95.7	**65.7**	**98.9**	87.9	**79.6**	95.5	**93.8**	**92.5**	**87.0**	**83.1**	**81.2**	**85.1**	**10.7**
MIC (Hoyer et al., 2023)	T	99.5	87.1	96.0	73.2	65.3	68.5	81.0	74.8	96.8	58.9	98.8	85.7	81.3	96.7	91.4	91.0	78.1	79.0	77.5	83.2	11.6
+BLDA	T	98.9	**90.7**	95.4	**74.6**	**70.9**	**73.5**	**88.9**	**82.8**	96.4	**64.8**	98.8	**89.0**	80.8	96.3	**93.8**	**92.8**	**88.2**	**86.5**	**78.8**	**86.3**	**9.8**

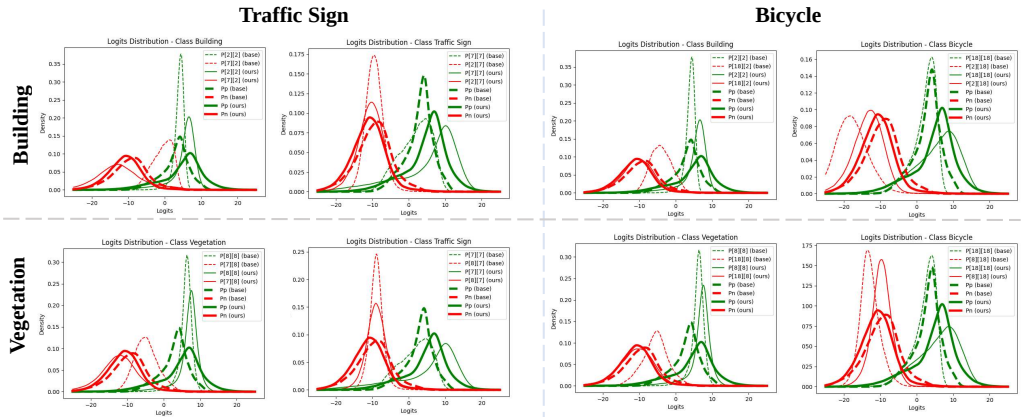


Figure 6: Comparison of Logits Distribution. We choose {*building* (2), *vegetation* (8)} as over-predicted classes, and {*traffic sign* (7), *bicycle* (18)} as under-predicted classes for visualization. Note that the anchor distribution is counted separately at baseline and in our method.

Effectiveness of Components. In Tab.10, we delve into the various components of BLDA. By solely applying the post-hoc method to adjust the predictions, we observe a minor performance improvement of 0.9%. When introducing online logit adjustment exclusively during source domain image training (OLA_S), the improvement is comparatively modest at 0.6%. However, by simultaneously performing adjustments in both domains (OLA), we witness a significant performance boost of 1.9%, suggesting that this strategy effectively captures the disparity in learning degrees between the domains. Lastly, the extra supervison from cumulative distribution estimation (CDE) further models the shared structural information across the domains (shown in Fig.4), producing additional performance gains.

Qualitative Results. Fig.4 presents the qualitative results. We observe that for under-predicted classes, such as *sidewalk* and *pole*, the baseline method struggles to recognize them accurately. While the post-hoc method can slightly improve the performance, our proposed BLDA approach significantly enhances the ability to predict these classes.

Comparison of Logits Distribution. In Fig.6, we visualize the positive distribution and negative distribution corresponding to the over-predicted classes { *building* (2), *vegetation* (8)} and under-predicted classes { *traffic sign* (7), *bicycle* (18) } on the Cityscapes Val. set. In the baseline method, the positive and negative logits of classes *building* and *vegetation* are larger than anchor distribution, while this phenomenon is reversed in classes *traffic light* and *bicycle*, which leads to class bias. Our method reduces this distribution difference by aligning with the anchor distribution and achieves class-balanced learning.

5 Conclusion

In this work, we present Balanced Learning for Domain Adaptation (BLDA), a novel approach to address class bias in unsupervised domain adaptation (UDA) for semantic segmentation. BLDA analyzes logits distributions to assess prediction bias and introduces an online logits adjustment mechanism to balance class learning in both source and target domains. Our method effectively mitigates class bias, promotes balanced learning, and enhances generalization to the target domain. Experimental results demonstrate consistent performance improvements on standard UDA benchmarks.

REFERENCES

Nikita Araslanov and Stefan Roth. Self-supervised augmentation consistency for adapting semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15384–15394, 2021.

Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10231–10241, 2021.

Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural networks*, 106:249–259, 2018.

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems*, 32, 2019.

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 transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017a.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017b.

Minghao Chen, Hongyang Xue, and Deng Cai. Domain adaptation for semantic segmentation with maximum squares loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2090–2099, 2019.

Runfa Chen, Yu Rong, Shangmin Guo, Jiaqi Han, Fuchun Sun, Tingyang Xu, and Wenbing Huang. Smoothing matters: Momentum transformer for domain adaptive semantic segmentation. *arXiv preprint arXiv:2203.07988*, 2022.

Yuhua Chen, Wen Li, and Luc Van Gool. Road: Reality oriented adaptation for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7892–7901, 2018.

Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34:17864–17875, 2021.

Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1290–1299, 2022.

Peng Chu, Xiao Bian, Shaopeng Liu, and Haibin Ling. Feature space augmentation for long-tailed data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pp. 694–710. Springer, 2020.

MMSegmentation Contributors. Mmsegmentation: Openmmlab semantic segmentation toolbox and benchmark, 2020.

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9268–9277, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pp. 1180–1189. PMLR, 2015.

Dayan Guan, Jiaxing Huang, Aoran Xiao, and Shijian Lu. Unbiased subclass regularization for semi-supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9968–9978, 2022.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pp. 1322–1328. Ieee, 2008.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Ruifei He, Jihan Yang, and Xiaojuan Qi. Re-distributing biased pseudo labels for semi-supervised semantic segmentation: A baseline investigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6930–6940, 2021.

Weixiang Hong, Zhenzhen Wang, Ming Yang, and Junsong Yuan. Conditional generative adversarial network for structured domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1335–1344, 2018.

Lukas Hoyer, Dengxin Dai, and Luc Van Gool. Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9924–9935, 2022a.

Lukas Hoyer, Dengxin Dai, and Luc Van Gool. Hrda: Context-aware high-resolution domain-adaptive semantic segmentation. In *European Conference on Computer Vision*, pp. 372–391. Springer, 2022b.

Lukas Hoyer, Dengxin Dai, Haoran Wang, and Luc Van Gool. Mic: Masked image consistency for context-enhanced domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11721–11732, 2023.

Xinyue Huo, Lingxi Xie, Wengang Zhou, Houqiang Li, and Qi Tian. Focus on your target: A dual teacher-student framework for domain-adaptive semantic segmentation. *arXiv preprint arXiv:2303.09083*, 2023.

Ying Jin, Ximei Wang, Mingsheng Long, and Jianmin Wang. Minimum class confusion for versatile domain adaptation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pp. 464–480. Springer, 2020.

Jaehyung Kim, Jongheon Jeong, and Jinwoo Shin. M2m: Imbalanced classification via major-to-minor translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13896–13905, 2020.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11593–11603, 2022.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2537–2546, 2019.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015a.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pp. 97–105. PMLR, 2015b.

Geoffrey J McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2007.

Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. *arXiv preprint arXiv:2007.07314*, 2020.

Duo Peng, Ping Hu, Qiuhong Ke, and Jun Liu. Diffusion-based image translation with label guidance for domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 808–820, 2023.

Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *arXiv preprint arXiv:1710.06924*, 2017.

Harsh Rangwani, Sumukh K Aithal, Mayank Mishra, Arihant Jain, and Venkatesh Babu Radhakrishnan. A closer look at smoothness in domain adversarial training. In *International conference on machine learning*, pp. 18378–18399. PMLR, 2022.

Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 102–118. Springer, 2016.

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3234–3243, 2016.

Fengyi Shen, Akhil Gurram, Ziyuan Liu, He Wang, and Alois Knoll. Diga: Distil to generalize and then adapt for domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15866–15877, 2023.

Marco Toldo, Umberto Michieli, Gianluca Agresti, and Pietro Zanuttigh. Unsupervised domain adaptation for mobile semantic segmentation based on cycle consistency and feature alignment. *Image and Vision Computing*, 95:103889, 2020.

Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. Dacs: Domain adaptation via cross-domain mixed sampling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1379–1389, 2021.

Thanh-Dat Truong, Ngan Le, Bhiksha Raj, Jackson Cothren, and Khoa Luu. Fredom: Fairness domain adaptation approach to semantic scene understanding. In *IEEE/CVF Computer Vision and Pattern Recognition (CVPR)*, 2023.

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7472–7481, 2018.

Grant Van Horn and Pietro Perona. The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*, 2017.

Kaihong Wang, Donghyun Kim, Rogerio Feris, and Margrit Betke. Cdac: Cross-domain attention consistency in transformer for domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11519–11529, 2023.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34:12077–12090, 2021.

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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12414–12424, 2021.

Dong Zhao, Shuang Wang, Qi Zang, Dou Quan, Xiutiao Ye, Rui Yang, and Licheng Jiao. Learning pseudo-relations for cross-domain semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19191–19203, 2023.

A Derivation of Eq.7

Considering probabilities estimate that $p(y_{ij} = c|x_{ij}) \propto \exp(f_\theta(x_{ij})[c])$, the discriminant probability for pixel x_{ij} can be presented as:

$$p(y_{ij} = c|x_{ij}) = \frac{\exp(f_\theta(x_{ij})[c]}{\sum_{c'} \exp(f_\theta(x_{ij})[c'])}. \tag{11}$$

Given offset for logits as $\Delta_{cl}(z) = z' - z$ and the label c for pixel x_{ij}, we can obtain revised class-conditional discriminant probability for each pixel x_{ij} by:

$$\widetilde{p}(y_{ij} = c|x_{ij}, c) = \frac{\exp(f_\theta(x_{ij})[c] + \Delta_{cc}(f_\theta(x_{ij})[c]))}{\sum_{c'} \exp(f_\theta(x_{ij})[c'] + \Delta_{cc'}(f_\theta(x_{ij})[c']))}. \tag{12}$$

Then, following Bayes rule, the revised posterior is derived as:

$$\widetilde{p}(y_{ij} = c|x_{ij}) = \frac{p(c)\widetilde{p}(y_{ij} = c|x_{ij}, c)}{\sum_{c'} p(c')\widetilde{p}(y_{ij} = c|x_{ij}, c')}. \tag{13}$$

So we can obtain revised prediciton results for each pixel x_{ij} by:

$$\widetilde{y}_{ij} = \text{argmax}_{c\in[C]} p(c)\widetilde{p}(y_{ij} = c|x_{ij}, c). \tag{14}$$

Since the class probabilities $p(c)$ are typically set as a uniform prior, i.e., $p(c) = \frac{1}{c}$, Eq.14 can be rewritten as:

$$\begin{aligned}\widetilde{y}_{ij} &= \text{argmax}_{c\in[C]} p(y_{ij} = c|x_{ij}, c) \\ &= \text{argmax}_{c\in[C]} \frac{\exp(f_\theta(x_{ij})[c] + \Delta_{cc}(f_\theta(x_{ij})[c]))}{\exp(f_\theta(x_{ij})[c] + \Delta_{cc}(f_\theta(x_{ij})[c])) + \sum_{c'\neq c} \exp(f_\theta(x_{ij})[c'] + \Delta_{cc'}(f_\theta(x_{ij})[c']))}.\end{aligned} \tag{15}$$

For Eq.6, we add a scaling factor τ to be adjusted for specific evaluation metrics.

B Discussion about Eq.9

For a deeper understanding of the loss function in Eq.9, we can rewrite it as:

$$\widetilde{\mathcal{L}}^s = \frac{1}{N_S} \sum_{i=1}^{N_S} \sum_{j=1}^{H\times W} \log \left(1 + \sum_{c\neq y_{ij}^S} \left(\frac{\exp\left(\Delta^S_{y_{ij}^S, y_{ij}^S}(f_\theta(x_{ij}^S)[y_{ij}^S])\right)}{\exp\left(\Delta^S_{y_{ij}^S, c}(f_\theta(x_{ij}^S)[c])\right)} \right)^\tau \exp\left(f_\theta(x_{ij}^S)[c] - f_\theta(x_{ij}^S)[y_{ij}^S]\right) \right), \tag{16}$$

which can be interpreted as a standard cross-entropy loss with an adaptive margin. Specifically, if the class y_{ij}^S is an over-predicted class, it is reasonable to assume that for most other classes c, $\Delta^S_{y_{ij}^S, y_{ij}^S}(f_\theta(x_{ij}^S)[y_{ij}^S]) < \Delta^S_{y_{ij}^S, c}(f_\theta(x_{ij}^S)[c])$. This implies that:

$$\left(\frac{\exp\left(\Delta^S_{y_{ij}^S, y_{ij}^S}(f_\theta(x_{ij}^S)[y_{ij}^S])\right)}{\exp\left(\Delta^S_{y_{ij}^S, c}(f_\theta(x_{ij}^S)[c])\right)} \right)^\tau < 1 \tag{17}$$

In the loss function Eq.16, this ratio is used to scale the term $\exp\left(f_\theta(x_{ij}^S)[c] - f_\theta(x_{ij}^S)[y_{ij}^S]\right)$. When y_{ij}^S is an over-predicted class, the ratio is less than 1, which reduces the weight of the term $\exp\left(f_\theta(x_{ij}^S)[c] - f_\theta(x_{ij}^S)[y_{ij}^S]\right)$. Consequently, for over-predicted classes, the loss function imposes a smaller penalty for misclassification. Conversely, if y_{ij}^S is an under-predicted class, it can be assumed that for most other classes c, $\Delta^S_{y_{ij}^S, y_{ij}^S}(f_\theta(x_{ij}^S)[y_{ij}^S]) > \Delta^S_{y_{ij}^S, c}(f_\theta(x_{ij}^S)[c])$. This leads to a ratio greater than 1, which increases the weight of the term $\exp\left(f_\theta(x_{ij}^S)[c] - f_\theta(x_{ij}^S)[y_{ij}^S]\right)$, thereby imposing a larger penalty for misclassification of under-predicted classes. In summary, by introducing the margin-based ratio term, the loss function Eq.16 adaptively adjusts the strength of the penalty based on the difficulty of the classes. This approach helps to mitigate the class bias problem and enhances the model's performance on under-predicted classes, leading to a more balanced and accurate classification. Moreover, since the logit offset term Δ is updated online during the training process, it aligns well with the self-training paradigm in UDA.

C Discussion about Evaluation Metrics

In this section, we investigate the impact of class imbalance on the evaluation metrics mIoU and mAcc in the context of semantic segmentation. We consider a multi-class problem with C classes, where the number of samples in the i-th class is denoted as N_i. Let P_{ij} represent the probability of classifying a sample from the i-th class as the j-th class in the confusion matrix. For the purpose of this analysis, we assume that the probabilities P_{ii} and P_{kk} are balanced across all classes, i.e., $P_{ii} = P_{kk} = p, \forall i, k \in 1, 2, ..., C$, and focus solely on the effect of class imbalance in terms of sample numbers. Under this assumption, the calculation formula for mAcc can be written as:

$$mAcc = \frac{1}{C}\sum_{i=1}^{C}\frac{N_i \cdot P_{ii}}{\sum_{j=1}^{C} N_i \cdot P_{ij}} = \frac{1}{C}\sum_{i=1}^{C}\frac{p}{\sum_{j=1}^{C} P_{ij}} \tag{18}$$

As evident from the equation above, the sample numbers N_i cancel out in the calculation of mAcc, making it independent of the class imbalance in terms of sample numbers. Therefore, mAcc remains unaffected by class imbalance under the given assumptions. On the other hand, the calculation formula for mIoU is given by:

$$mIoU = \frac{1}{C}\sum_{i=1}^{C}\frac{N_i \cdot P_{ii}}{\sum_{j=1}^{C} N_i \cdot P_{ij} + \sum_{j=1}^{C} N_j \cdot P_{ji} - N_i \cdot P_{ii}} \tag{19}$$

In contrast to mAcc, the sample numbers N_i do not cancel out in the calculation of mIoU. Consequently, when the classes are imbalanced, i.e., the sample numbers N_i vary significantly across classes, the IoU of classes with larger sample numbers will dominate the overall mIoU result. To illustrate the impact of class imbalance on mIoU, let us consider a case where class k is a head class with a significantly larger sample number N_k compared to a tail class i with sample number N_i. The performance of class i will be greatly affected by class k through the term $N_k \cdot P_{ki}$ in the denominator of mIoU. For class i to have a fair contribution to mIoU, the probability P_{ki} needs to be very small. This implies that a balanced classifier may not be optimal for maximizing mIoU under class imbalance. Therefore, our proposed method implements a scaling factor τ to modulate the contributions of introduced logits offset. In contrast, mAcc is inherently unaffected by class imbalance, as the sample numbers N_i cancel out in its calculation formula. This means that a balanced classifier is indeed optimal for maximizing mAcc, and our method, which aims to balance the contributions of different classes, aligns well with this objective and can achieve consistent improvements.

D Implementation Details of Online Logits Distribution Estimation

In this section, we provide the pseudo label to explain implementation details of online logits distribution estimation, as shown in Alg.1. For computational efficiency, in each iteration, we sample N_{sample} logits for each element in M_{cl}^S and M_{cl}^T, where N_{sample} is the minimum sample number of classes in the minibatch (shoule be greater than or equal to N_{min}, where we set it as 100). This way, we obtain $C \times C \times N_{sample}$ logits, and then update the $C \times C$ GMMs simultaneously in a parallel manner. Since not all classes may be updated in each iteration, we maintain a variable n for each GMM that is not updated, to record the number of iterations since its last update. This n is used to adjust the momentum factor using $\tilde{\tau}$ in the EMA update, in order to match the update speed of the network.

In the algorithm, the cumulative distribution functions (CDFs) F_{cl}^s, F_{cl}^t, F_p, and F_n are computed using the estimated Gaussian mixture models (GMMs). These CDFs describe the cumulative probability distribution of the corresponding GMMs. For the source and target domain GMMs P_{cl}^s and P_{cl}^t, their CDFs can be represented as: $F_{cl}^s(z) = \sum_{k=1}^{K} \pi_{cl,k}^s \cdot \Phi(\frac{z-\mu_{cl,k}^s}{\sigma_{cl,k}^s})$ $F_{cl}^t(z) = \sum_{k=1}^{K} \pi_{cl,k}^t \cdot \Phi(\frac{z-\mu_{cl,k}^t}{\sigma_{cl,k}^t})$, where $\Phi(\cdot)$ is the CDF of the standard normal distribution, and $\pi_{cl,k}^s$, $\mu_{cl,k}^s$, and $\sigma_{cl,k}^s$ are the weight, mean, and standard deviation of the k-th component of the source domain GMM P_{cl}^s, respectively. $\pi_{cl,k}^t$, $\mu_{cl,k}^t$, and $\sigma_{cl,k}^t$ are the corresponding parameters for the target domain GMM P_{cl}^t. Similarly, the CDFs for the anchor GMMs P_p and P_n are: $F_p(z) = \sum_{k=1}^{K} \pi_{p,k} \cdot \Phi(\frac{z-\mu_{p,k}}{\sigma_{p,k}})$

Algorithm 1 Online Logits Adjustment for UDA

Input: Source domain $D_s = (x_i^S, y_i^S)_{i=1}^{N_S}$, target domain $D_t = (x_i^T)_{i=1}^{N_T}$, number of classes C, number of Gaussian components K, momentum factor $\tilde{\tau}$, scaling factor τ, minimum number of elements N_{min}, number of EM Loop T.
Output: Model parameters θ.

Initialize model parameters θ, source GMMs P_{cl}^s, target GMMs P_{cl}^t, anchor GMMs P_p and P_n for all $c, l \in [C]$.
while not converged **do**
  Sample a mini-batch of source data $(x_i^S, y_i^S)_{i=1}^{B_S}$ and target data $(x_i^T)_{i=1}^{B_T}$.
  Compute logits $f_\theta(x_{ij}^S)$ and $f_\theta(x_{ij}^T)$ for source and target samples.
  Compute pseudo-labels for target samples: $\hat{y}_{ij}^T = \text{argmax}_{c\in[C]} f_\theta(x_{ij}^T[c])$.
  Compute matrices M_{cl}^S and M_{cl}^T based on the source labels and target pseudo-labels:
  $M_{cl}^S = \{f_\theta(x_{ij}^S)[l] \mid y_{ij}^S = c\}$
  $M_{cl}^T = \{f_\theta(x_{ij}^T)[l] \mid \hat{y}_{ij}^T = c\}$
  Update source GMMs P_{cl}^s, target GMMs P_{cl}^t, anchor GMMs P_p and P_n using the momentum-based EM algorithm:
  for $c, l \in [C]$ **do**
    if $|M_{cl}^S| > N_{min}$ **then**
      Initialize $\phi_{cl}^{s,(0)} \leftarrow \hat{\phi}_{cl}^s$ from the latest iteration.
      for $t = 1, \ldots, T$ **do**
        Update $\phi cl^{s,(t)}$ using the EM algorithm with current logits.
      end for
      $\hat{\phi} cl^s \leftarrow (1-\tilde{\tau}^n)\phi_{cl}^{s,(T)} + \tilde{\tau}^n \hat{\phi} cl^s$, where n is the number of iterations since the last update.
    end if
    if $|M_{cl}^S| > N_{min}$ **then**
      Initialize $\phi_{cl}^{t,(0)} \leftarrow \hat{\phi}_{cl}^t$ from the latest iteration.
      for $t = 1, \ldots, T$ **do**
        Update $\phi cl^{t,(t)}$ using the EM algorithm with current logits.
      end for
      $\hat{\phi}_{cl}^t \leftarrow (1-\tilde{\tau}^n)\phi_{cl}^{t,(T)} + \tilde{\tau}^n \hat{\phi}_{cl}^t$, where n is the number of iterations since the last update.
    end if
  end for
  Update anchor GMMs P_p and P_n using the global positive and negative logits from the source domain:
  Initialize $\phi_p^{(0)} \leftarrow \hat{\phi}_p$, $\phi_n^{(0)} \leftarrow \hat{\phi}_n$ from the latest iteration.
  for $t = 1, \ldots, T$ **do**
    Update $\phi_p^{(t)}$, $\phi_n^{(t)}$ using the EM algorithm with current global logits.
  end for
  $\hat{\phi}_p \leftarrow (1-\tilde{\tau})\phi_p^{(T)} + \tilde{\tau}\hat{\phi}_p$
  $\hat{\phi}_n \leftarrow (1-\tilde{\tau})\phi_n^{(T)} + \tilde{\tau}\hat{\phi}_n$
  Compute cumulative distributions F_{cl}^s, F_{cl}^t, F_p, F_n using the estimated GMMs.
  Compute logits offsets for source domain: $\Delta_{cl}^S(z) = \begin{cases} F_p^{-1}(Fcl^s(z)) - z, & \text{if } c = l \\ F_n^{-1}(Fcl^s(z)) - z, & \text{if } c \neq l \end{cases}$
  Compute logits offsets for target domain: $\Delta_{cl}^T(z) = \begin{cases} F_p^{-1}(F_{cl}^t(z)) - z, & \text{if } c = l \\ F_n^{-1}(F_{cl}^t(z)) - z, & \text{if } c \neq l \end{cases}$
  Compute the adjusted losses $\widetilde{\mathcal{L}}^s$ and $\widetilde{\mathcal{L}}^u$ using Eq. equation 8 with Δ_{cl}^S and Δ_{cl}^T.
  Update model parameters θ by minimizing $\widetilde{\mathcal{L}}^s + \widetilde{\mathcal{L}}^u$ using an optimizer (e.g., SGD or Adam).
end while
return Model parameters θ.

$F_n(z) = \sum_{k=1}^K \pi_{n,k} \cdot \Phi(\frac{z-\mu_{n,k}}{\sigma_{n,k}})$. The inverse function of a CDF, denoted as $F^{-1}(\cdot)$, represents the

Table 5: Parameter study of K.

K	mIou (%)
1	70.2
3	70.5
5	**70.7**
10	70.6

Table 6: Parameter study of $\tilde{\tau}$.

$\tilde{\tau}$	mIou (%)
0	68.6
0.9	70.0
0.99	**70.7**
0.999	70.3

Table 7: Parameter study of T.

T	mIou (%)
1	70.4
3	**70.7**
5	**70.7**
10	70.5

Table 8: Parameter study of λ.

λ	mIou (%)
0.05	70.3
0.2	**70.7**
0.5	70.4
1	69.9

value of the variable corresponding to a given cumulative probability. For a given logit value z, by computing $F_p^{-1}(F_{cl}^s(z))$ and $F_n^{-1}(F_{cl}^s(z))$, we obtain the corresponding logit values of the positive anchor distribution P_p and the negative anchor distribution P_n at the cumulative probability $F_{cl}^s(z)$. Then, the difference between these values and the original logit value z is used as the logits offset $\Delta_{cl}^S(z)$ for the source domain. Similarly, by computing $F_p^{-1}(F_{cl}^t(z))$ and $F_n^{-1}(F_{cl}^t(z))$, we obtain the logits offset $\Delta_{cl}^T(z)$ for the target domain. To efficiently compute the CDF and its inverse function, we use the Abramowitz-Stegun formula to approximate the CDF in the form of a polynomial and employ interpolation methods to estimate the inverse function.

E Influence of Parameters Setting

In this section, we further study the influence of parameters setting introduced in BLDA, i.e., number of Gaussian components K, momentum factor $\tilde{\tau}$, number of EM Loop T for GMM estimation and λ for cumulative density estimation loss. All experiments are conducted with DAFormer (Hoyer et al., 2022a) on GTA→CS.

Number of Gaussian Components K. As shown in Tab.5, we find that BLDA can work well even when $K = 1$, since the logit distribution is naturally close to Gaussian. The model achieves the best performance when $K = 5$. A larger K allows for more flexibility in modeling the logits distributions but may also introduce noise. We choose $K = 5$ as a balance between model capacity and robustness.

Momentum Factor $\tilde{\tau}$. The momentum factor $\tilde{\tau}$ controls the speed of updating the GMM parameters. When $\tilde{\tau} = 0$, performance becomes erratic because the logits from the current iteration alone are not sufficient to model the distribution of all logits. A larger $\tilde{\tau}$ leads to slower updates, retaining the previously estimated distribution and making the estimation more stable but less adaptive. As presented in Tab.6, setting $\tilde{\tau}$ to 0.99 yields the best performance, suggesting that a relatively stable estimation of the logits distributions is beneficial for the adaptation process.

Number of EM Loop T. The number of EM loops T determines the number of iterations used to update the GMM parameters in each training step. Tab.7 shows that the model is not sensitive to the choice of T, since the convergence rate of GMM is faster than the rate of network update, and it can be estimated well even when $T = 1$. We choose $T = 3$ for stable performance while considering computational efficiency.

Cumulative Density Estimation Loss Weight λ. The weight λ balances the cumulative density estimation loss with the segmentation loss. A higher λ enforces stronger domain alignment through the cumulative density functions. As shown in Tab.8, $\lambda = 0.2$ provides the best performance gain. An overly large λ may distract the model from learning the primary segmentation task, leading to performance degradation.

F Extended Experiment on Image Classification

To demonstrate the generality of BLDA, we implement BLDA based on MIC (Hoyer et al., 2023) with ResNet-101 on the VisDA-2017 (Peng et al., 2017) UDA classification benchmark in Tab.9, and our method still achieves improvements. In the classification task, the dataset does not have severe class distribution differences like segmentation. However, as we point out in Fig.1, the transfer difficulty differences between domains still lead to severe class bias in this task, and our method can effectively alleviate this and achieve more balanced predictions.

Table 9: Image classification accuracy in % on VisDA-2017 for UDA, where the improvement is marked as **bold**.

Method	Plane	Bcycl	Bus	Car	Horse	Knife	Mcyle	Persn	Plant	Sktb	Train	Truck	Mean
MCC (Jin et al., 2020)	88.1	80.3	80.5	71.5	90.1	93.2	85.0	71.6	89.4	73.8	85.0	36.9	78.8
SDAT (Rangwani et al., 2022)	95.8	85.5	76.9	69.0	93.5	97.4	88.5	78.2	93.1	91.6	86.3	55.3	84.3
MIC (Hoyer et al., 2023)	96.7	88.5	84.2	74.3	96.0	96.3	90.2	81.2	94.3	95.4	88.9	56.6	86.9
+BLDA	96.2	**90.3**	82.8	**81.2**	95.7	**96.7**	**93.4**	**86.5**	**95.7**	94.3	**91.0**	**65.7**	**89.1**

G VISUALIZATION OF ESTIMATED GMMS

In this section, we visualize the learned GMMs for target domain, i.e., M_{cl}^T for all $c, l \in [C]$. Fig. 7 presents the Estimated GMMs built with DAFormer, and Fig. 8 presents the estimated GMMs with introducing BLDA. We find that the estimated GMMs can accurately model logits distribution and our method reduces the difference in logits distribution across classes, thus achieving balanced learning.

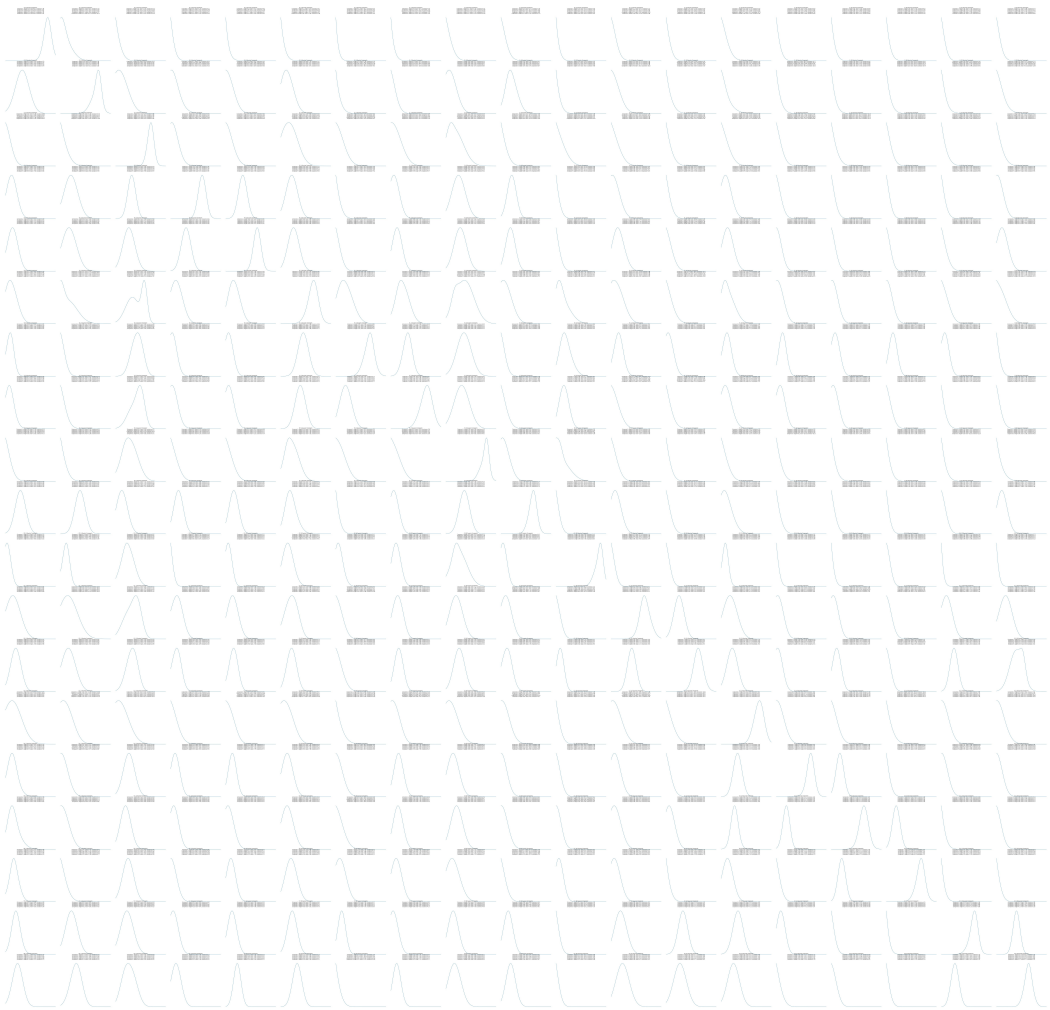


Figure 7: Estimated GMMs on target domain built with DAFormer.

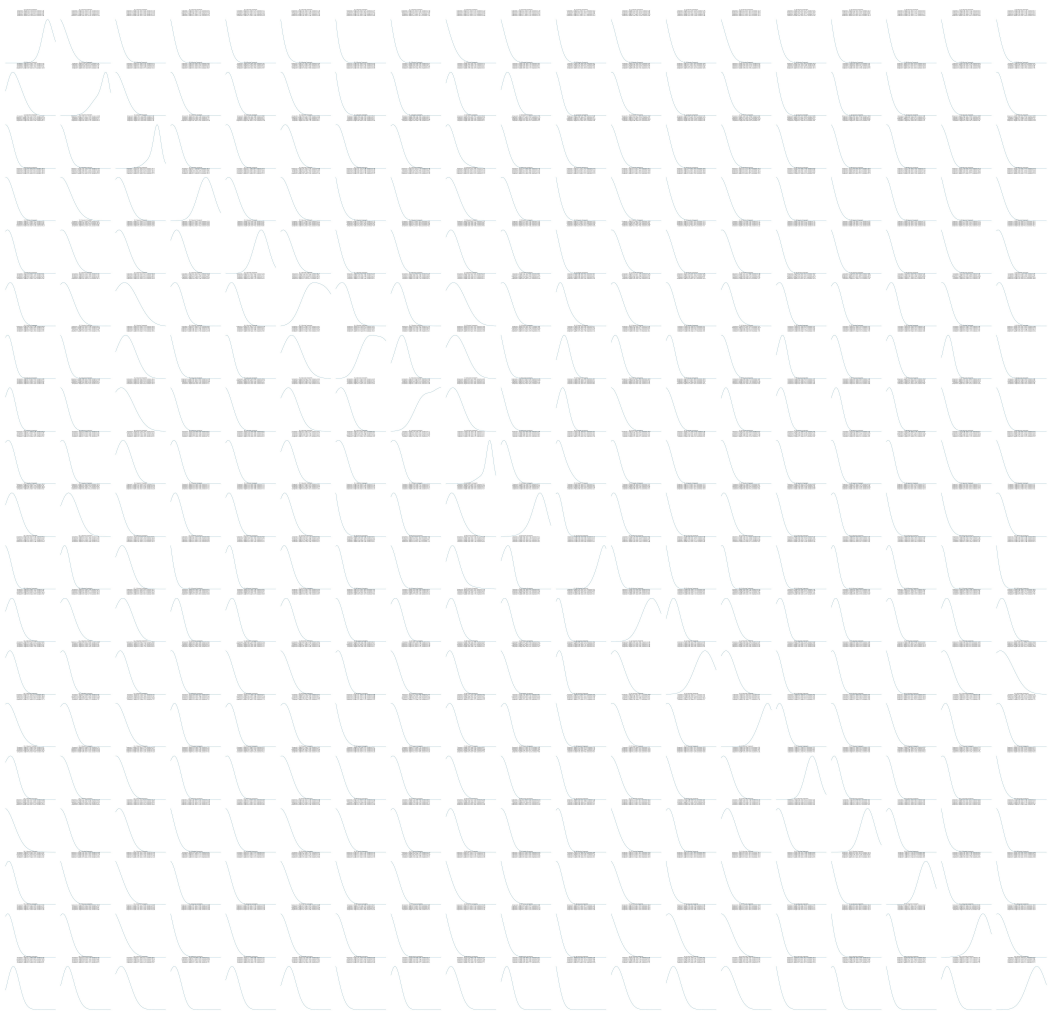


Figure 8: Estimated GMMs on target domain built with BLDA.

H MORE VISUALIZATION RESULTS OF LOGITS DISTRIBUTION

In this section, we provide more visualization results to compare logits distribution built with our method. As shown in Fig.9, for over-predicted classes, the network predicts larger positive logits and negative logits (column 1, 3) , while for under-predicted classes, the network predicts smaller logits (column 2, 4). This difference in logits distribution leads to the class prediction bias. Our method reduces this difference through aligning with the anchor distribution and achieves class-balanced learning.

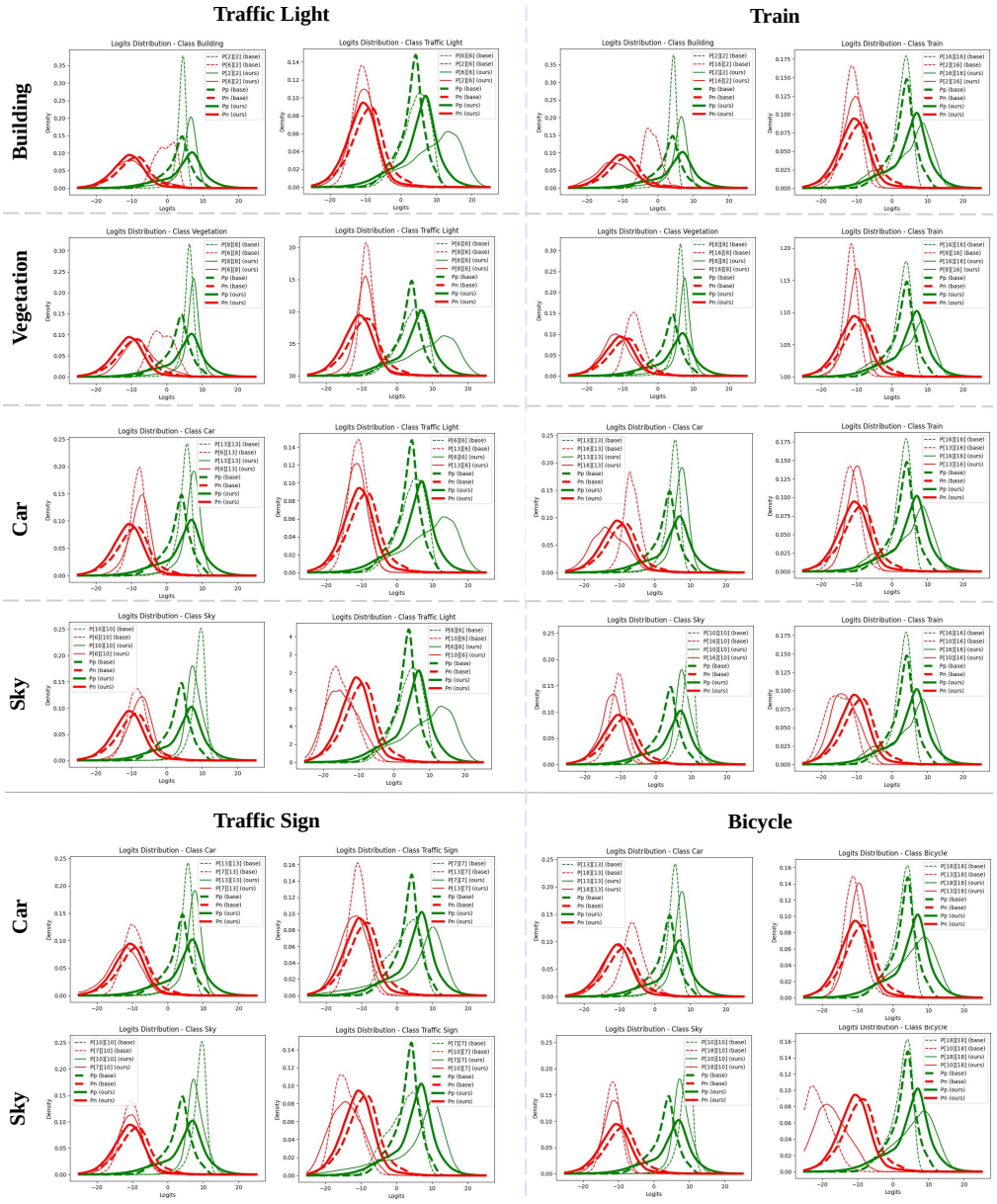


Figure 9: Comparison of Logits Distribution. We choose {*building* (2), *vegetation* (8), *car* (13) ,*sky* (10)} as over-predicted classes, and {*train* (16), *bicycle* (18), *traffic light* (6), *traffic sign* (7) } as under-predicted classes for visualization. Note that the anchor distribution is counted separately at baseline and our method.

I MORE QUALITATIVE RESULTS

In this section, we provide more qualitative results between our method and other competitors on GTA→CS. As shown in Fig.10, when previous methods fail to recognize the classes that are under-predicted and suffer severe performance decline in UDA (e.g. *sidewalk*, *pole*, *fence*, *terrain*, *bike*,*sign*), BLDA shows significant improvement on them, thereby demonstrating the effectiveness of our method.

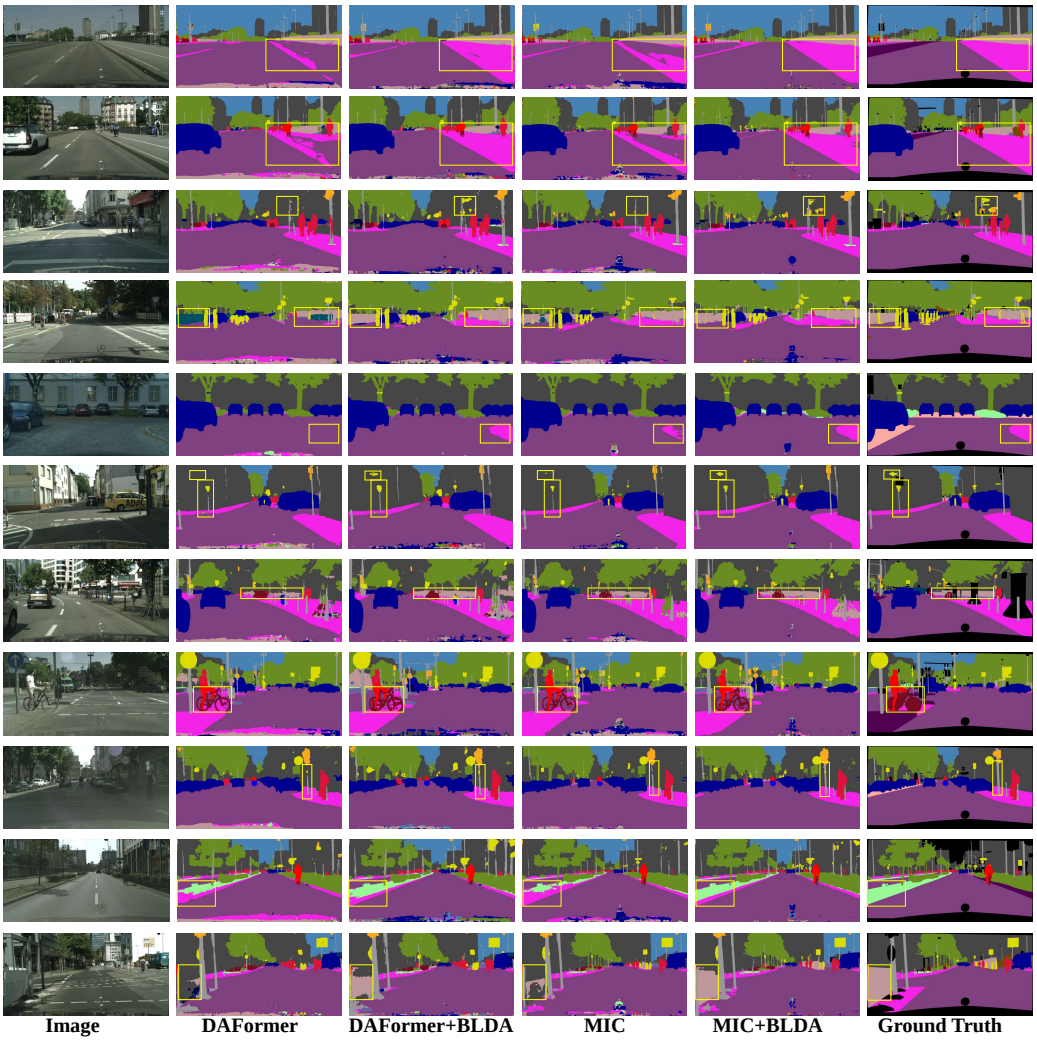


Figure 10: More qualitative comparison with DAFormer and MIC. The yellow boxes mark regions improved by BLDA.

J LIMITATION AND SOCIETY IMPACT

Our method analyzes the class bias in domain adaptive semantic segmentation through logits distribution statistics and propose a method to implement online logits adjustment tailored for the UDA training process, which can be easily built with exiting methods and demonstrate consistent improvements. Although BLDA achieves remarkable performance, we balance the class under the assumption that each logits distribution in $\mathcal{M}$ is independent, without considering correlation between classes. How to model this correlation and mitigate the class bias further is still to be resolved. Within

this paper, we present an approach for domain adaptive semantic segmentation, a pivotal research area in the realm of computer vision, with no apparent negative societal implications known thus far.

K EXTENDED EXPERIMENTS

Table 10: UDA segmentation performance on SYN.→CS. using the mAcc (%) evaluation metric, where the improvement is marked as **bold**.

Method	Arch.	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Veg	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motor	Bike	mAcc	std
DAFormer(Hoyer et al., 2022a)	T	89.8	90.2	96.2	33.8	8.3	51.9	63.1	57.8	95.1	-	98.4	86.3	63.6	96.7	-	83.4	-	55.4	61.4	70.7	25.1
+BLDA	T	87.3	**95.6**	94.9	**38.8**	**14.1**	**57.5**	**70.1**	**63.3**	**97.8**	-	**98.9**	**90.0**	**68.0**	**97.7**	-	**95.7**	-	**63.8**	**67.5**	**75.1**	**23.6**
HRDA (Hoyer et al., 2022b)	T	90.3	85.2	96.0	69.4	8.0	70.6	81.2	69.6	94.7	-	99.1	88.0	68.9	97.4	-	93.8	-	71.3	74.0	78.6	21.2
+BLDA	T	89.4	**94.5**	**95.4**	**75.5**	**24.5**	**78.2**	**85.8**	**74.7**	**97.6**	-	98.8	**88.9**	**71.6**	97.4	-	**96.9**	-	**72.6**	**76.2**	**82.4**	**17.9**
MIC (Hoyer et al., 2023)	T	89.9	87.4	96.2	71.3	8.4	66.7	81.5	69.7	95.6	-	98.5	89.1	73.0	97.3	-	93.5	-	78.2	71.5	79.2	21.2
+BLDA	T	89.4	**97.2**	96.0	**73.4**	**17.9**	**72.1**	**87.6**	**75.8**	**97.4**	-	98.3	**91.5**	72.5	**97.7**	-	**96.2**	-	**79.3**	**75.6**	**82.4**	**19.4**

Table 11: UDA segmentation performance on GTA.→CS. using the mIoU (%) evaluation metric, where the improvement is marked as **bold**. The results are acquired based on CNN-based model (He et al., 2016; Chen et al., 2017a), denoted as C, and Transformer-based model (Xie et al., 2021), denoted as T. $*$ denotes the reproduced result.

Method	Arch.	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Veg	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motor	Bike	mIoU	std
DACS* (Tranheden et al., 2021)	C	93.0	52.0	87.8	29.4	38.3	37.7	45.0	53.3	87.9	46.3	90.2	67.8	38.0	89.0	51.1	51.1	0.0	10.7	19.4	52.1	27.1
+BLDA	C	92.9	**67.5**	87.1	**36.3**	**39.3**	**41.2**	**50.7**	**58.5**	87.3	45.5	87.7	**69.1**	**40.4**	88.3	45.3	**53.5**	**1.2**	**10.8**	**36.3**	**54.7**	**25.6**
DAFormer* (Hoyer et al., 2022a)	C	95.7	69.9	87.2	35.6	36.7	37.0	49.4	52.8	87.3	44.1	87.9	69.0	42.2	86.5	40.0	51.7	0.2	41.1	54.0	56.2	24.0
+BLDA	C	95.6	**73.6**	86.7	**41.0**	**40.2**	**43.3**	**51.1**	**62.4**	86.2	43.7	87.4	68.3	39.2	**86.6**	**43.6**	45.6	**1.0**	**49.4**	**59.4**	**58.1**	**23.2**
CDAC* (Wang et al., 2023)	T	96.1	72.8	90.5	55.2	48.0	51.8	57.1	61.8	90.8	50.4	91.9	73.2	46.9	93.6	80.9	78.6	58.2	56.9	64.5	69.2	16.7
+BLDA	T	**96.6**	**78.1**	90.0	**57.9**	**52.5**	**55.1**	**58.7**	**64.5**	90.1	**50.8**	90.9	**73.3**	**47.5**	93.2	75.1	**80.0**	**65.3**	**60.7**	**68.9**	**71.0**	**15.4**

Table 12: UDA segmentation performance on GTA.→CS. using the mAcc(%) evaluation metric, where the improvement is marked as **bold**. The results are acquired based on CNN-based model (He et al., 2016; Chen et al., 2017a), denoted as C, and Transformer-based model (Xie et al., 2021), denoted as T. $*$ denotes the reproduced result.

Method	Arch.	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Veg	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motor	Bike	mAcc	std
DACS* (Tranheden et al., 2021)	C	97.9	58.9	94.9	40.7	49.4	46.1	51.7	57.2	94.5	66.9	98.6	82.0	62.2	93.3	82.9	79.0	0.0	74.4	20.0	65.8	26.5
+BLDA	C	97.1	**80.5**	93.4	**54.1**	39.2	**50.5**	**64.6**	**67.4**	93.7	**71.5**	**99.0**	**83.4**	58.8	**95.0**	73.0	76.2	**1.0**	73.5	**39.2**	**69.0**	**24.2**
DAFormer* (Hoyer et al., 2022a)	C	98.4	78.3	93.6	43.2	45.4	45.0	61.5	59.5	93.8	65.3	98.0	80.1	69.2	92.2	77.0	85.3	0.2	69.7	60.8	69.3	23.8
+BLDA	C	97.0	**84.8**	93.0	**55.1**	**56.1**	**55.4**	**71.0**	**74.5**	93.2	**73.7**	97.7	**84.8**	**77.2**	92.0	**80.1**	**85.5**	**1.1**	**77.3**	**73.0**	**74.9**	**21.6**
CDAC* (Wang et al., 2023)	T	99.1	78.0	94.6	69.3	52.9	61.2	71.7	68.7	95.8	59.3	99.0	85.0	70.2	96.0	88.3	85.5	71.1	75.4	74.7	78.7	13.8
+BLDA	T	**98.1**	82.5	**94.0**	**74.8**	**66.7**	**68.0**	**76.8**	74.1	**95.5**	67.9	98.3	**87.1**	72.9	**95.6**	**90.8**	**88.8**	**71.8**	80.8	**82.9**	**82.5**	**11.5**

Table 13: Performance comparison on GTA.→CS. grouped by over-prediction and under-prediction classes. The results are acquired based on CNN-based model (He et al., 2016; Chen et al., 2017a), denoted as C, and Transformer-based model (Xie et al., 2021), denoted as T.

Metrics		IoU: mean/std		Acc: mean/std	
Methods	Arch.	over-prediction	under-prediction	over-prediction	under-prediction
DACS (Tranheden et al., 2021)	C	68.7/28.9	37.0/13.1	80.3/29.2	52.3/14.5
+BLDA	C	68.0/28.6	42.7/14.3	79.1/29.0	59.9/13.4
DAFormer (Hoyer et al., 2022a)	C	67.3/29.6	46.3/10.0	79.8/29.0	59.8/11.2
+BLDA	C	66.8/29.3	50.3/10.9	80.5/28.6	69.8/10.0
CDAC (Wang et al., 2023)	T	83.8/11.6	56.5/7.6	90.4/8.5	68.1/7.6
+BLDA	T	83.8/10.1	59.5/8.7	91.1/7.8	74.7/5.7
DAFormer (Hoyer et al., 2022a)	T	83.3/10.3	54.7/7.3	90.6/8.0	66.1/6.2
+BLDA	T	84.2/8.5	57.8/9.4	92.3/4.7	72.6/8.0
HRDA (Hoyer et al., 2022b)	T	87.8/6.8	61.0/8.0	93.0/5.6	72.5/10.1
+BLDA	T	88.6/6.2	64.5/7.2	93.9/4.0	77.0/8.2
MIC (Hoyer et al., 2023)	T	89.5/5.9	63.6/8.0	92.7/6.6	74.7/8.0
+BLDA	T	89.6/5.3	66.4/8.0	94.4/3.6	79.2/7.9

Table 14: Computational resource requirements comparison

Methods	GPU Memory (MB)	Time per iter (s)	Total Time (h)
DAFormer (Hoyer et al., 2022a)	9,807	1.32	14.5 (40K iters)
+BLDA	12,655	1.59	17.7 (40K iters)
DACS (Tranheden et al., 2021)	11,078	0.52	35.5 (250K iters)
+BLDA	14,354	0.81	56.1 (250K iters)
CDAC (Wang et al., 2023)	35,443	1.66	18.7 (40K iters)
+BLDA	35,443	1.95	22.3 (40K iters)

Table 15: Comparison with other class-imbalanced methods on GTA.→CS. using the mIoU (%) evaluation metric, where baseline is based on DAFormer with uniform class sampling.

Method	Arch.	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Veg	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motor	Bike	mIoU	std
baseline	T	95.2	67.4	89.0	43.7	47.5	46.7	55.7	55.0	89.3	47.2	90.7	71.4	44.5	92.3	76.1	78.5	62.1	55.4	64.1	66.9	17.6
[1] (Lin et al., 2017)	T	87.6	40.1	87.9	49.2	41.2	45.7	52.1	50.9	89.2	45.0	90.7	70.8	41.1	89.9	56.6	70.5	60.3	54.6	59.5	62.3	18.1
[2] (Cui et al., 2019)	T	96.0	70.6	89.6	53.8	48.3	50.2	53.3	60.8	89.7	49.4	90.9	70.2	42.6	91.7	65.7	77.8	60.1	54.4	65.2	67.4	16.8
[3] (Hoyer et al., 2022a)	T	95.7	70.2	89.4	53.5	48.1	49.6	55.8	59.4	89.9	47.9	92.5	72.2	44.7	92.3	74.5	78.2	65.1	55.9	61.8	68.3	16.8
[4] (Menon et al., 2020)	T	94.7	66.9	89.2	53.2	47.0	49.0	51.2	56.0	89.6	50.1	87.2	69.0	41.9	92.0	71.8	75.7	62.3	50.4	64.0	66.4	16.9
+BLDA	T	94.5	72.1	88.4	48.3	51.6	52.9	59.5	64.5	88.1	51.5	90.5	72.4	47.6	94.0	75.9	80.8	65.6	53.7	67.7	69.5	15.9

Table 16: Comparison with other class-imbalanced methods on GTA.→CS. using the mAcc (%) evaluation metric, where baseline is based on DAFormer with uniform class sampling.

Method	Arch.	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Veg	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motor	Bike	mAcc	std
baseline	T	98.3	75.4	94.9	53.8	57.8	53.6	70.2	59.7	96.2	57.6	99.1	83.5	67.6	95.5	87.1	84.3	73.8	74.5	77.5	76.9	15.3
[1] (Lin et al., 2017)	T	91.7	63.3	94.6	66.4	50.2	54.4	63.9	55.1	95.6	66.9	98.6	84.9	58.5	92.6	84.4	81.9	70.0	68.8	68.2	74.2	15.2
[2] (Cui et al., 2019)	T	98.8	77.7	94.1	71.4	56.8	63.7	77.2	71.2	95.3	63.5	99.0	84.1	72.0	94.6	90.1	89.2	68.5	76.4	80.2	80.2	12.6
[3] (Hoyer et al., 2022a)	T	99.1	74.8	95.2	61.0	53.1	59.8	70.7	68.6	96.0	63.4	98.7	84.2	69.9	95.5	88.9	84.1	74.0	70.0	69.7	77.8	14.2
[4] (Menon et al., 2020)	T	97.9	79.9	95.0	64.4	64.7	66.6	74.3	72.6	96.5	66.1	99.7	87.9	72.5	96.3	90.5	91.3	85.7	74.3	72.4	81.5	12.2
+BLDA	T	97.5	79.4	93.4	65.5	68.0	63.9	77.2	75.5	94.9	68.1	99.0	85.2	71.4	95.2	89.2	87.6	80.8	77.0	83.6	81.7	10.9

Table 17: The JS divergence between the P_{cl} and corresponding anchor distribution P_p and P_n, where the results on P_n is averaged over the $C-1$ negative components.

Method	Anchor	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Veg	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motor	Bike	mean	std
baseline	P_p	0.412	0.289	0.381	0.286	0.293	0.137	0.325	0.142	0.429	0.273	0.596	0.130	0.217	0.300	0.225	0.164	0.190	0.145	0.180	0.270	0.119
+BLDA	P_p	0.367	0.246	0.261	0.153	0.212	0.221	0.146	0.234	0.281	0.173	0.310	0.244	0.070	0.273	0.284	0.181	0.125	0.207	0.125	0.217	0.072
baseline	P_n	0.517	0.327	0.315	0.394	0.305	0.280	0.422	0.288	0.412	0.404	0.433	0.278	0.396	0.279	0.387	0.382	0.401	0.430	0.394	0.371	0.064
+BLDA	P_n	0.231	0.165	0.153	0.135	0.209	0.118	0.224	0.119	0.248	0.246	0.240	0.186	0.158	0.170	0.199	0.208	0.176	0.162	0.136	0.183	0.041

Table 18: Ablation on different selection criteria for anchor distributions on GTA.→CS. using the mIoU (%) evaluation metric. OLA denotes online logits adjustment.

Anchor	Strategy	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Veg	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motor	Bike	mIoU	std
DAFormer (Hoyer et al., 2022a)	-	95.7	70.2	89.4	53.5	48.1	49.6	55.8	59.4	89.9	47.9	92.5	72.2	44.7	92.3	74.5	78.2	65.1	55.9	61.8	68.3	16.8
target (global)	post-hoc	95.7	77.0	87.8	61.0	53.7	54.3	56.1	61.2	87.2	50.2	90.4	74.4	43.4	90.9	73.3	82.5	56.6	54.8	68.3	69.4	15.8
source (building)	post-hoc	95.8	77.1	90.0	59.8	54.5	51.6	56.6	59.8	86.2	48.2	90.6	75.4	43.7	90.4	72.0	79.5	58.9	53.7	69.2	69.1	16.0
source (fence)	post-hoc	95.6	77.5	88.0	58.4	55.0	51.2	56.9	57.7	88.7	48.3	90.5	75.2	43.8	90.1	71.7	80.6	61.1	53.9	65.5	68.9	16.0
source (global)	post-hoc	95.7	77.1	88.6	60.5	55.3	48.5	57.3	60.9	89.5	47.2	91.0	72.9	43.7	91.3	73.7	80.8	61.1	55.3	63.8	69.2	16.2
source (building)	OLA	95.5	77.9	88.5	60.4	59.1	53.3	57.9	60.4	88.3	49.2	90.3	76.1	43.6	90.3	76.4	85.7	56.2	57.9	68.8	70.3	15.8
source (fence)	OLA	95.6	81.2	88.2	57.9	57.0	51.6	54.2	60.2	87.9	50.2	90.3	76.3	44.3	90.0	75.0	82.4	57.7	56.0	70.6	69.8	16.0
source (global)	OLA	95.4	78.3	88.3	54.0	55.2	55.7	60.3	65.2	89.2	47.3	91.1	71.4	44.8	91.6	74.3	83.4	73.2	59.3	67.1	70.7	15.5

Table 19: Ablation on different selection criteria for anchor distributions on GTA.→CS. using the mAcc (%) evaluation metric. OLA denotes online logits adjustment.

Anchor	Strategy	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Veg	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motor	Bike	mAcc	std
DAFormer (Hoyer et al., 2022a)	-	99.1	74.8	95.2	61.0	53.1	59.8	70.7	68.6	96.0	63.4	98.7	84.2	69.9	95.5	88.9	84.1	74.0	70.0	69.7	77.8	14.2
target (global)	post-hoc	98.7	85.6	92.2	65.9	67.4	67.3	75.1	68.6	90.8	66.8	95.5	91.8	67.3	93.1	92.7	95.7	62.0	84.5	80.2	81.1	12.4
source (building)	post-hoc	98.7	89.6	92.2	73.5	67.5	66.3	76.5	67.0	90.8	63.8	95.5	90.7	68.3	93.1	92.6	95.6	60.3	79.9	80.9	81.2	12.5
source (fence)	post-hoc	98.5	85.3	91.3	74.1	66.7	66.2	82.3	68.4	90.5	63.4	95.5	88.9	62.1	92.5	91.9	94.9	63.8	83.5	79.2	81.0	12.2
source (ours)	post-hoc	98.6	90.0	91.6	71.0	66.5	64.1	78.5	68.2	91.2	65.1	96.4	91.9	64.8	92.7	92.5	95.1	67.6	82.2	77.4	81.3	12.4
source (building)	OLA	98.6	84.7	92.6	71.4	66.2	68.8	76.6	71.0	91.2	67.3	95.8	94.6	69.7	93.5	93.2	96.3	59.7	79.5	80.2	81.6	12.2
source (fence)	OLA	98.6	84.8	92.2	71.6	64.1	66.9	78.3	68.2	90.9	68.2	95.6	92.9	65.4	93.3	93.0	95.9	60.4	82.3	84.8	81.4	12.6
source (ours)	OLA	98.3	86.7	93.4	70.9	61.9	66.0	74.0	78.9	95.4	59.0	98.6	85.7	73.1	95.3	89.8	89.3	85.4	77.2	78.6	82.0	11.9

L EXTENDED DISCUSSION

L.1 DOMAIN ADAPTIVE SEMANTIC SEGMENTATION

Unsupervised domain adaptation (UDA) transfers semantic knowledge learned from labeled source domains to unlabeled target domains. Due to the ubiquity of domain gaps, UDA methods have been widely studied in various computer vision tasks, such as image classification, object detection, and semantic segmentation. UDA is crucial for semantic segmentation to avoid laborious pixel-wise annotation in new target scenarios.

Recent UDA approaches for semantic segmentation fall into two main paradigms: adversarial training-based methods (Toldo et al., 2020; Tsai et al., 2018; Chen et al., 2018; Ganin & Lempitsky, 2015; Hong et al., 2018; Long et al., 2015b) and self-training-based methods (Tranheden et al., 2021; Hoyer et al., 2022a; Araslanov & Roth, 2021; Zhang et al., 2021). Adversarial training-based methods learn domain-invariant representations through a min-max optimization game, where a feature extractor is trained to confuse a domain discriminator, aligning feature distributions across domains. Self-training-based methods, which have become dominant in the field due to the domain-robustness of Transformers (Bhojanapalli et al., 2021), generate pseudo labels for target images based on a teacher-student optimization framework. The success of this paradigm depends on generating high-quality pseudo labels, with strategies such as entropy minimization (Chen et al., 2019) and consistency regularization (Hoyer et al., 2023) being developed for this purpose.

Recently, several approaches have been proposed to tackle the challenges in UDA for semantic segmentation: DTS (Huo et al., 2023) employs a Dual Teacher-Student Framework, promoting the self-training paradigm to fully adapt models to the target domain by exploring different mix strategies. CDAC (Wang et al., 2023) introduces consistency constraints in attention to enhance the model's cross-domain performance. RTea (Zhao et al., 2023) defines proxy tasks based on structural information and incorporates them into the self-training paradigm as additional supervision signals. Peng et al. (2023) apply Diffusion-based image translation techniques to directly mitigate the distribution differences between the target and source domains. DiGA (Shen et al., 2023) integrates distillation strategies and self-training through multi-stage training.

However, due to the inherent class imbalance and distribution shift in both data and label space between domains, networks often exhibit complex class biases, which are further amplified by the confirmation bias inherent in the self-training paradigm. Our method aims to achieve balanced learning in UDA training to mitigate these issues and improve the overall performance of domain adaptation for semantic segmentation.

L.2 CLASS-IMBALANCED LEARNING

Class imbalance is a prevalent issue in semantic segmentation, where the number of samples per class varies significantly. Existing methods tackle this problem through re-weighting or re-sampling techniques. Re-weighting methods assign different weights to classes during training, giving higher importance to under-represented classes (Cui et al., 2019; Lin et al., 2017; Cao et al., 2019; Liu et al., 2019). Re-sampling techniques modify the class distribution in the training data by over-sampling minority classes or under-sampling majority classes (He et al., 2008; 2021; He & Garcia, 2009; Kim et al., 2020; Chu et al., 2020).

Recent works have proposed various approaches to address class imbalance in different tasks. For object detection, Lin et al. (2017) propose a re-weighting method that assigns weights to samples based on their confidence, with harder samples receiving higher weights. In classification, Cui et al. (2019) define an effective number of samples for each class to weight different classes, while Menon et al. (2020) introduce a logit adjustment method that incorporates correction terms in logits, determined by prior knowledge of class distribution. For segmentation, Hoyer et al. (2022a) present a re-sampling technique that samples based on prior knowledge of class distribution, with more sampling for rare classes. Truong et al. (2023) considers class-imbalance as fairness problem and propose to model the context of structural dependency to tackle it.

In UDA for semantic segmentation, some approaches have introduced these strategies to alleviate class bias (Hoyer et al., 2022a; Araslanov & Roth, 2021; Li et al., 2022). However, these methods are still empirical and focus on the single-domain setting, which assumes that the test data and training data share the same distribution in both data space and label space, without considering the additional challenges posed by domain shift in UDA.

In contrast to these methods, our approach aims to directly address class bias and achieve balanced learning for each class without relying on prior knowledge about the distribution shift between domains. We evaluate class bias through logit sets in the form of a confusion matrix and explicitly balance components in the matrix using anchor distributions and cumulative density functions, implemented in an online self-training paradigm. Our method does not depend on prior knowledge of class distribution and instead directly models class bias based on the actual logit distribution, making it more adaptable to the challenges posed by domain shift in UDA.

M CORE CODE

We provide our core code in BLDA to demonstrate our implementation.