Measure the Feature Universe: Topology-based Pseudo Labeling and Gravity Consistency for Source-Free Domain Adaptation

Supplementary Material

In the supplementary material, we list the parts requiring further explanation in the order mentioned in the main manuscript. Its contents are as follows:

- **Pseudo code of the feature traversal algorithm:** This section provides the detailed pseudo code and explanation of the proposed feature traversal algorithm, introduced in Section 2.1, for refining pseudo-labels on the cosine k-NN graph.
- **Class centroid variations:** This section provides a detailed description of the various class centroid measures introduced in Section 2.3 and employed in our proposed pseudo labeling procedure.
- **Dataset details:** This section provides descriptions and statistics of the various datasets and benchmarks used in our experiments, as introduced in Section 3.1.
- **Implementation details:** This section provides the training setups and hyper-parameter settings used in our adaptation experiments, as described in Section 3.5.
- **Ablation study details:** This section provides ablation experiments on the number of virtual features and the loss weights in the objective.
- **Pseudo label accuracy comparison with existing methods:** This section provides a comparison of pseudo-label accuracy between our method and representative SFDA baselines, as shown in Tab. 7 of Section 3.6.
- **Additional analysis on pseudo-label refinement:** This section provides additional comparisons with alternative pseudo-labeling baselines to evaluate the effectiveness of the proposed feature traversal algorithm for pseudo-label refinement.
- **Qualitative analysis of feature traversal algorithm:** This section provides representative feature traversal paths to qualitatively illustrate how the proposed method differs from conventional kNN-based pseudo-label assignment.
- **Computational overhead:** This section provides a comparison of computational cost, including training time and peak GPU memory usage, between our approach and existing methods.
- **Additional result of gravity consistency:** This section provides additional visualizations and analyses of the proposed gravity consistency in the feature space, as shown in Fig. 3 of Section 3.7.

Algorithm 1 Feature traversal algorithm

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ on Z_t'; initial node label $\tilde{y}_i$; reliable-node confidence mask $\mathcal{M}$; set of real nodes $Z_t \subset \mathcal{V}$; max hops H.
Ensure: Refined pseudo labels $\hat{y}_i$ for $i \in Z_t$.

for each real node $i \in Z_t$ **do**
  $Q \leftarrow$ queue with $(i, 0)$; Visited $\leftarrow \emptyset$
  $\hat{y}_i \leftarrow \varnothing$; $y_{\text{last}} \leftarrow \tilde{y}_i$
  while Q not empty **do**
    $(u, h) \leftarrow \text{pop}(Q)$; $y_{\text{last}} \leftarrow \tilde{y}_u$
    if $\mathcal{M}_u = 1$ **then**
      $\hat{y}_i \leftarrow \tilde{y}_u$; **break**
    end if
    if $h = H$ **then break**
    end if
    for each v such that $(u, v) \in \mathcal{E}$ **do**
      if $v \notin$ Visited **then**
        $\text{push}(Q, (v, h + 1))$;
        Visited $\leftarrow$ Visited $\cup \{v\}$;
      end if
    end for
  end while
  if $\hat{y}_i = \varnothing$ **then**
    $\hat{y}_i \leftarrow y_{\text{last}}$
  end if
end for
return $\hat{y}_i$

A. Pseudo code of the feature traversal algorithm

Algorithm 1 provides a detailed description of the proposed feature traversal algorithm. The goal of the algorithm is to correct inaccurate or unreliable initial pseudo labels $\hat{y}_i$ by traversing the neighborhood structure of cosine k-nearest neighborhood (k-NN) graph and identifying reliable nodes that offer more stable label information. To achieve this, we perform a feature traversal on a cosine k-NN graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, constructed from real and virtual features. Each real node i is assigned an initial pseudo label $\hat{y}_i$, and every node is associated with a reliability mask $\mathcal{M} = 1(\text{or}0)$. The traversal begins by initializing a queue Q with the pair $(u = i, h = 0)$, where u, i, and h denote the current node, the index of the starting node in the graph, and the current hop depth, respectively. Here, a "hop" refers to

one step of movement from a node to one of its neighbors in the k-NN graph. Thus, each pair (u, h) represents the node currently being visited and the number of hops taken from the initial node. A set $Visited$ is maintained to prevent revisiting the same nodes during traversal. The algorithm proceeds until one of two termination conditions is met: (1) the hop count reaches the predefined maximum depth H, or (2) the current node u is a reliable node with $\mathcal{M}_u = 1$. If a reliable node is found during traversal, its label $\tilde{y}_u$ is immediately assigned as the refined pseudo-label $\hat{y}_i$. If the traversal terminates without discovering any reliable node, the algorithm assigns $\hat{y}_i$ using the most recently observed label y_{last}. This strategy mitigates extreme errors from noisy pseudo labels by incorporating information from the nearest reachable neighbor in the graph $\mathcal{G}$ rather than naively retaining the original label.

B. Class centroid variations

Analysis on probability-weighted mean. As mentioned in Section 2 of the manuscript, our framework employs two distinct types of class centroids in the feature embedding space, each designed for a different stage of the learning pipeline. For universe-aware pseudo labeling (UP), we use the geometric k-means centroid $\mu^{(k)}$ (Section 2.1) to parametrize the class-wise Gaussian distributions from which virtual features are sampled. In contrast, volume-aware pseudo supervision (VP) uses the probability-weighted mean centroid $\mu'^{(k)}$ (Section 2.3) to estimate the perceptual manifold volume [19] and to compute class-balanced weights. These two centroids capture complementary perspectives of the embedding space. The k-means centroid $\mu^{(k)}$ represents a geometric prototype that reflects the local structure of the feature manifold, making it suitable for defining realistic class-wise Gaussian distributions in UP. The probability-weighted mean centroid $\mu'^{(k)}$, on the other hand, emphasizes high-confidence regions by aggregating feature vectors with soft probability weights, following the standard practice established since SHOT [16]. Because their roles are fundamentally different, our design intentionally assigns them to different components of the pipeline rather than using a single centroid type throughout. To validate this design choice, we conducted an ablation study in which we independently substituted the centroid type used in UP and VP. The results in Tab. 8 clearly support our design rationale. Configurations using the k-means centroid (KM) in the UP stage generally achieve stronger performance than those using the probability-weighted mean (PW), even though the difference varies across adaptation tasks. This confirms that UP benefits from a geometry-aware prototype that better aligns with the underlying class-wise structure when constructing Gaussian-based virtual features. On the other hand, using PW in the VP stage yields slightly higher performance than KM, especially when UP is based on KM. This is expected, as VP relies on estimating perceptual manifold volume, a process that is naturally aligned with probability-weighted aggregation. The best average accuracy (74.4%) is obtained when UP uses KM while VP uses PW, precisely matching our intended division of roles. Overall, the ablation results validate both components of our method: (1) UP requires a geometric centroid to model the feature manifold and generate virtual samples, and (2) VP benefits from probability-weighted centroids that reflect confidence-weighted feature statistics.

Additional consideration for class centroid. Among the possible alternatives for defining class-wise centroids in the feature embedding space for universe-aware pseudo labeling, we also considered another option beyond the probability-weighted mean and the K-means–based centroid. We considered an alternative, widely used notion of class prototype: the classifier weight vectors. For the weight of classifier f, the k-th row vector $w^{(k)} \in \mathbb{R}^D$ is often interpreted as a prototype in the feature space, because the class logit is obtained by the inner product with the feature vectors $\langle w^{(k)}, z \rangle$. Motivated by this, we constructed an additional variant where the centroids used in universe-aware pseudo labeling (UP) are derived from the classifier weights learned on the labeled source domain. To make the classifier weights compatible with the target feature distribution, we applied a feature-wise z-score normalization and affine mapping to the target feature statistics. Let $W \in \mathbb{R}^{K \times D}$ denote the classifier weight matrix, where K is the number of classes and D is the feature dimension. We first compute the mean and standard deviation of the weights across classes:

$$\begin{aligned} \mu_w &= \frac{1}{K} \sum_{k=1}^{K} w^{(k)} \in \mathbb{R}^D, \\ \sigma_w &= \sqrt{\frac{1}{K} \sum_{k=1}^{K} \left(w^{(k)} - \mu_w\right)^2} \in \mathbb{R}^D, \end{aligned} \tag{14}$$

and similarly the mean and standard deviation of all target features $\{z_i\}_{i=1}^N$:

$$\begin{aligned} \mu_z &= \frac{1}{N} \sum_{i=1}^{N} z_i \in \mathbb{R}^D, \\ \sigma_z &= \sqrt{\frac{1}{N} \sum_{i=1}^{N} (z_i - \mu_z)^2} \in \mathbb{R}^D. \end{aligned} \tag{15}$$

We then map each classifier weight vector into the target feature embedding space via

$$\tilde{w}^{(k)} = (w^{(k)} - \mu_w) \odot (1/\sigma_w) \odot \sigma_z + \mu_z, \tag{16}$$

Table 8. Accuracies (%) on the 12 Office-Home adaptation tasks when varying the class centroids used in universe-aware pseudo labeling (UP) and volume-aware pseudo supervision (VP). "PW" and "KM" denote the probability-weighted mean centroid $\mu'^{(k)}$ and the k-means geometric centroid $\mu^{(k)}$, respectively.

UP	VP	A→C	A→P	A→R	C→A	C→P	C→R	P→A	P→C	P→R	R→A	R→C	R→P	Avg.
PW	PW	59.6	**81.6**	82.7	70.4	79.4	81.0	70.1	58.6	83.3	74.6	61.5	84.7	74.0
PW	KM	**60.0**	80.7	**82.8**	71.0	79.8	81.1	**70.3**	58.4	83.1	**74.7**	61.9	85.5	74.1
KM	PW	59.7	81.4	**82.8**	**71.2**	**80.4**	**81.3**	**70.3**	**58.7**	**83.5**	**74.7**	**63.1**	**85.7**	**74.4**
KM	KM	59.5	80.9	82.4	71.0	**80.4**	80.9	69.5	58.6	82.4	74.3	62.5	85.2	74.0

Table 9. Accuracies (%) on the 12 Office-Home adaptation tasks when varying the class centroids used in universe-aware pseudo labeling (UP) and volume-aware pseudo supervision (VP). "PW", "KM", and "CW" denote the probability-weighted mean centroid $\mu'^{(k)}$, the k-means geometric centroid $\mu^{(k)}$, and the classifier-weighted centroid $\tilde{w}^{(k)}$, respectively.

UP	VP	A→C	A→P	A→R	C→A	C→P	C→R	P→A	P→C	P→R	R→A	R→C	R→P	Avg.
PW	PW	59.6	**81.6**	82.7	70.4	79.4	81.0	70.1	58.6	83.3	74.6	61.5	84.7	74.0
CW	PW	59.5	**81.6**	82.7	70.6	80.0	81.0	69.9	58.3	83.1	74.2	62.8	85.2	74.1
KM	PW	**59.7**	81.4	**82.8**	**71.2**	**80.4**	**81.3**	**70.3**	**58.7**	**83.5**	**74.7**	**63.1**	**85.7**	**74.4**

where $\odot$ denote the element-wise multiplication. The resulting vectors $\tilde{w}^{(k)}$ are finally used as k-th class centroids in place of the original geometric centroids when defining the class-wise Gaussian distributions and constructing the feature universe for universe-aware pseudo-labeling. Table 9 compares the accuracy(%) obtained when universe-aware pseudo labeling (UP) uses the classifier weight-based centroid $\tilde{w}^{(k)}$ and the k-means centroid $\mu^{(k)}$, while the volume-aware pseudo supervision (VP) stage consistently employs the probability-weighted mean $\mu'^{(k)}$. The results show that K-means centroids $\mu^{(k)}$ consistently achieve stronger average performance (74.4%) compared to classifier weight–based centroids (74.1%), with clear gains in several adaptation directions. Interestingly, a comparable pattern also appears in the additional rows of Tab. 9, where using the probability-weighted mean for UP similarly results in noticeable degradation in several adaptation directions such as C→A, C→P, and R→C. Such consistency across both experiments suggests that certain adaptation directions are especially sensitive to how well the centroid represents the underlying target-domain feature structure. In these cases, the classifier-weight prototype—being primarily shaped by discriminative decision boundaries—fails to capture the geometric layout of the feature manifold. Likewise, the probability-weighted mean centroid, which emphasizes high-confidence regions by construction, tends to overlook low-density yet semantically important areas of the class distribution. As a result, both alternatives provide suboptimal prototypes for modeling class-wise Gaussian distributions in the UP stage, whereas the k-means centroids remain more reliable due to their direct alignment with the embedding geometry.

C. Dataset details

To assess the performance of our source-free domain adaptation framework, we conducted experiments on three commonly used benchmarks: Office-Home [31], VisDA-C [22], and DomainNet-126 [23]. The **Office-Home** dataset contains 15,500 images distributed across four domains (Art, Clipart, Product, and Real-World), sharing a total of 65 categories. **VisDA-C** is a large-scale synthetic-to-real benchmark comprising over 152K images from 12 categories; the training split consists of synthetically generated renderings, while the validation split is sourced from real images in MS-COCO. **DomainNet-126** is a curated subset of the DomainNet dataset, including 145K images across 126 classes, collected from four domains (Clipart, Painting, Real, and Sketch).

Table 10. Hyperparameter configurations of our method for each adaptation task on the Office-Home and DomainNet-126 datasets.

Office-Home							**DomainNet-126**						
Task	λ	H	top-k	α	β	γ	Task	λ	H	top-k	α	β	γ
A→C	5	5	7	0.30	1.35	1.0	C→P	15	15	7	0.35	1.20	1.0
A→P	20	20	7	0.30	1.40	1.0	C→R	15	15	7	0.20	1.20	1.0
A→R	5	5	7	0.30	1.35	1.0	C→S	15	15	7	0.20	1.30	1.0
C→A	20	20	7	0.30	1.35	1.0	P→C	15	15	7	0.20	1.35	1.0
C→P	20	20	7	0.30	1.40	1.0	P→R	15	15	7	0.20	1.25	1.0
C→R	20	20	7	0.30	1.35	1.0	P→S	15	15	7	0.20	1.20	1.0
P→A	5	5	7	0.30	1.35	1.0	R→C	15	15	7	0.20	1.35	1.0
P→C	10	10	7	0.30	1.35	1.0	R→P	15	15	7	0.20	1.35	1.0
P→R	20	20	7	0.30	1.30	1.0	R→S	15	15	7	0.20	1.20	1.0
R→A	5	5	7	0.30	1.35	1.0	S→C	15	15	7	0.20	1.20	1.0
R→C	20	20	7	0.30	1.45	1.0	S→P	15	15	7	0.20	1.20	1.0
R→P	20	20	7	0.35	1.35	1.0	S→R	15	15	7	0.30	1.20	1.0

Table 11. General training configurations of our method for each adaptation task on the Office-Home, VisDA-C, and DomainNet-126 datasets.

Hyper-parameters	**Office-Home**	**VisDA-C**	**DomainNet-126**
Batch size	64	64	64
Optimizer	SGD	SGD	SGD
Momentum	0.9	0.9	0.9
Weight-decay	0.001	0.001	0.001
Learning rate	0.01	0.001	0.001
Epochs	15	15	15

Table 12. Ablation study on hyperparameter configurations for the Office-Home dataset. We vaired the number of virtual features γ per class and the loss terms α, β in Eq. 13 to analyze their influence on adaptation performance across all 12 adaptation tasks.

λ	α	β	A→C	A→P	A→R	C→A	C→P	C→R	P→A	P→C	P→R	R→A	R→C	R→P	Avg.
0	0.30	1.35	**60.5**	80.9	82.9	70.8	80.9	80.4	68.5	57.8	82.8	74.2	61.7	84.0	73.8
5	0.30	1.35	59.9	80.8	**83.0**	**71.3**	80.6	80.4	**70.1**	**58.9**	82.8	**74.9**	61.5	85.0	74.1
10	0.30	1.35	59.5	80.9	82.8	71.0	80.2	80.1	70.0	**58.9**	82.9	74.7	62.7	85.1	74.1
20	0.30	1.35	59.5	80.9	82.6	**71.2**	80.0	**81.1**	69.8	58.2	83.4	74.1	62.6	85.6	74.1
20	0.20	1.35	59.9	80.2	82.5	70.9	80.5	80.6	69.8	58.3	82.4	74.5	62.4	84.8	73.9
20	0.25	1.35	59.2	80.6	82.4	71.1	80.2	80.6	70.0	58.1	83.2	74.1	61.8	85.3	73.9
20	0.30	1.35	59.5	80.9	82.6	**71.2**	80.0	81.1	69.8	58.2	83.4	74.1	62.6	85.6	74.1
20	0.35	1.35	59.5	80.9	82.6	**71.2**	80.0	80.6	69.5	58.1	**83.5**	73.7	62.4	**85.7**	73.9
20	0.30	1.30	59.3	80.5	82.6	70.8	80.2	80.9	69.9	57.8	**83.5**	74.3	62.3	85.5	74.0
20	0.30	1.35	59.5	80.9	82.6	**71.2**	80.0	81.1	69.8	58.2	83.4	74.1	62.6	85.6	74.1
20	0.30	1.40	59.2	**81.1**	82.7	71.1	**80.4**	80.9	70.0	58.2	83.4	74.1	62.5	**85.7**	74.1
20	0.30	1.45	59.5	80.6	82.8	71.1	79.7	**81.1**	**70.1**	58.0	83.1	**74.3**	**63.1**	85.2	74.0
-	-	-	**59.9**	**81.1**	**83.0**	**71.2**	**80.4**	**81.1**	**70.1**	**58.9**	**83.5**	**74.9**	**63.1**	**85.7**	**74.4**

Table 13. Ablation study on hyperparameter configurations for the DomainNet-126 dataset.

λ	α	β	C→P	C→R	C→S	P→C	P→R	P→S	R→C	R→P	R→S	S→C	S→P	S→R	Avg.
0	0.20	1.20	65.2	80.1	64.4	75.4	83.4	69.5	75.8	72.4	66.5	76.2	70.2	80.5	73.3
5	0.20	1.20	65.5	79.8	64.1	75.2	83.4	69.3	76.0	72.4	66.2	76.4	70.3	**80.6**	73.3
15	0.20	1.20	65.7	79.8	64.3	75.2	83.4	**69.4**	75.8	72.3	**67.0**	**76.5**	70.2	80.3	73.3
30	0.20	1.20	66.0	80.0	64.5	74.9	83.4	69.3	75.9	72.3	66.7	75.8	69.8	80.5	73.2
15	0.20	1.20	65.7	**80.6**	**64.8**	75.1	83.4	68.9	76.2	72.0	66.1	75.7	**70.3**	80.4	73.3
15	0.25	1.20	66.1	80.4	64.4	74.4	83.4	68.8	76.0	71.6	65.6	75.9	69.9	80.4	73.1
15	0.30	1.20	65.8	80.5	64.1	74.3	**83.5**	68.5	75.7	71.3	65.3	75.4	69.5	**80.6**	72.9
15	0.35	1.20	**66.2**	80.5	63.8	74.0	83.3	68.1	75.6	71.1	64.4	74.6	69.2	**80.6**	72.6
15	0.20	1.20	65.7	**80.6**	**64.8**	75.1	83.4	68.9	76.2	72.0	66.1	75.7	**70.3**	80.4	73.3
15	0.20	1.25	66.0	**80.6**	64.6	75.2	**83.5**	68.9	76.2	72.1	66.1	75.7	70.2	80.3	73.3
15	0.20	1.30	65.9	80.0	**64.8**	75.0	**83.5**	69.3	76.3	72.3	66.5	75.8	70.2	80.5	73.3
15	0.20	1.35	66.1	80.2	64.7	**75.3**	**83.5**	**69.4**	**76.5**	**72.5**	66.6	75.9	70.1	80.3	73.1
-	-	-	**66.2**	**80.6**	**64.8**	**75.3**	**83.5**	**69.4**	**76.5**	**72.5**	**67.0**	**76.5**	**70.3**	**80.6**	**73.6**

D. Implementation details

This section provides additional implementation details regarding the hyper-parameter configurations used in our experiments, along with a description of how Tabs. 10 and 11 are related.

Data augmentation. For target model training, we adopt a two-view augmentation strategy composed of one weak view and one strong view. The weak augmentation follows a standard image classification pipeline consisting of resizing the input image to 256×256, applying a random crop of size 224×224, random horizontal flipping, and ImageNet normalization. The strong augmentation includes random resized cropping to 224×224, color jittering, random grayscale conversion, Gaussian blur, random horizontal flipping, and ImageNet normalization. These two augmented views are jointly used for target model training.

Hyper-parameter configuration overview. Table 10 summarizes the final hyper-parameter configurations adopted for each adaptation direction on the Office-Home [31] and DomainNet-126 [23] datasets. These configurations are directly derived from the comprehensive ablation studies reported in Tab. 12(Office-Home) and Tab. 13(DomainNet-126). A detailed analysis of these ablations is presented in the subsequent sections of this supplementary material. In Tabs. 12 and 13, we systematically vary three key hyper-parameters: (1)λ–the number of virtual features generated per class, used during universe-aware pseudo labeling, (2)α–loss weight for the weighted cross-entropy term $\mathcal{L}_{\text{CE}}$, and (3)β–loss weight for the gravity consistency term $\mathcal{L}_{\text{GV}}$. These task-specific hyper-parameters are used for all the manuscript results.

General training configuration. The overall training setup used across all benchmarks is summarized in Table 11. For Office-Home, VisDA-C, and DomainNet-126, we adopt a unified optimization scheme with a batch size of 64, SGD as the optimizer, and momentum and weight decay set to 0.9 and 0.001, respectively. We use a fixed training schedule of 15 epochs for all datasets, and the backbone feature encoder is kept consistent within each benchmark: ResNet-50 is used for Office-Home and DomainNet-126, whereas VisDA-C follows prior SFDA works and adopts a ResNet-101 backbone. The only dataset-specific difference in optimization lies in the learning rate of the target model: Office-Home employs a learning rate of 0.01, while both VisDA-C and DomainNet-126 use 0.001. These consistent settings ensure that performance differences arise from the proposed method itself.

Table 14. Comparison of adaptation accuracies(%) across different training epochs on the Office-Home and DomainNet-126 benchmarks. For each adaptation, we report the performance of SHOT [16], TPDS/GKD baselines, and our method using feature encoder checkpoints obtained at epochs {3, 6, 9, 12, 15}. The results demonstrate that our approach consistently outperforms existing SFDA methods across most tasks and remains robust throughout the entire training trajectory, confirming the stability and effectiveness of the proposed topology-aware pseudo labeling framework.

Office-Home							**DomainNet-126**						
Task	Method	3	6	9	12	15	Task	Method	3	6	9	12	15
A→C	SHOT	53.95	54.80	54.96	55.07	55.07	C→P	SHOT	62.99	63.22	63.22	63.18	63.18
	TPDS	56.20	57.07	57.16	57.50	57.37		GKD	60.96	61.31	61.40	61.42	61.42
	Ours	**57.30**	**58.81**	**59.43**	**59.73**	**59.66**		Ours	**63.47**	**64.69**	**65.04**	**65.24**	**65.63**
A→P	SHOT	78.42	78.58	78.80	78.80	78.82	C→R	SHOT	78.48	78.78	78.75	78.75	78.76
	TPDS	78.04	79.00	79.18	79.18	79.27		GKD	77.75	77.75	77.63	77.53	77.51
	Ours	**80.40**	**80.90**	**81.01**	**81.26**	**81.46**		Ours	**78.60**	**79.42**	**79.81**	**80.21**	**80.41**
A→R	SHOT	80.86	81.20	81.36	81.18	81.25	C→S	SHOT	60.33	60.30	60.19	60.20	60.20
	TPDS	81.23	81.63	81.98	81.90	81.90		GKD	60.48	60.28	60.39	60.44	60.43
	Ours	**81.57**	**82.17**	**82.86**	**82.79**	**82.76**		Ours	**62.39**	**63.65**	**64.23**	**64.34**	**64.50**
C→A	SHOT	68.97	69.26	69.20	69.02	68.93	P→C	SHOT	68.07	68.46	68.49	68.45	68.46
	TPDS	**69.14**	**70.33**	70.52	70.24	69.84		GKD	69.20	69.50	69.52	69.54	69.65
	Ours	68.93	70.13	**70.62**	**71.12**	**71.28**		Ours	**72.56**	**74.38**	**74.83**	**75.00**	**75.15**
C→P	SHOT	78.26	79.71	78.87	78.80	78.89	P→R	SHOT	81.07	80.87	80.74	80.68	80.70
	TPDS	76.35	78.37	78.09	78.46	78.46		GKD	81.43	81.41	81.38	81.38	81.37
	Ours	**78.62**	**79.93**	**80.15**	**80.36**	**80.18**		Ours	**82.98**	**83.05**	**83.11**	**83.10**	**83.25**
C→R	SHOT	78.24	79.78	79.83	79.78	79.00	P→S	SHOT	61.80	61.90	61.87	61.83	61.79
	TPDS	78.24	79.85	79.80	80.28	79.75		GKD	63.19	63.44	63.40	63.33	63.30
	Ours	**79.64**	**80.33**	**80.54**	**80.95**	**81.20**		Ours	**66.78**	**68.11**	**68.35**	**68.54**	**68.79**
P→A	SHOT	67.46	69.04	69.82	69.36	69.22	R→C	SHOT	67.61	68.05	68.06	68.01	68.00
	TPDS	67.66	69.68	69.66	69.59	69.02		GKD	68.20	68.33	68.41	68.37	68.38
	Ours	**68.31**	**69.76**	**70.13**	**70.29**	**70.17**		Ours	**73.51**	**75.54**	**76.04**	**76.38**	**76.37**
P→C	SHOT	55.31	54.20	54.39	54.48	54.52	R→P	SHOT	68.46	68.43	68.34	68.31	68.31
	TPDS	54.55	56.01	55.97	56.04	56.11		GKD	68.36	68.34	68.38	68.43	68.42
	Ours	**55.37**	**57.64**	**58.35**	**58.35**	**58.65**		Ours	**71.50**	**71.89**	**80.16**	**80.16**	**80.35**
P→R	SHOT	81.31	81.59	81.61	81.68	81.66	R→S	SHOT	58.61	58.65	58.62	58.63	58.62
	TPDS	81.12	81.85	81.95	82.49	82.46		GKD	59.46	59.54	59.56	59.68	59.69
	Ours	**81.89**	**82.62**	**82.97**	**83.36**	**83.47**		Ours	**63.75**	**65.98**	**66.65**	**66.65**	**66.97**
R→A	SHOT	73.47	73.71	73.59	73.71	73.63	S→C	SHOT	70.81	70.78	70.77	70.73	70.77
	TPDS	**74.17**	**74.91**	**74.41**	**74.87**	74.54		GKD	71.44	71.44	71.53	71.58	71.61
	Ours	72.81	73.84	74.29	74.45	**74.62**		Ours	**73.28**	**75.34**	**75.84**	**76.03**	**76.27**
R→C	SHOT	57.46	58.58	58.85	58.83	58.90	S→P	SHOT	65.10	64.95	64.93	64.90	64.89
	TPDS	59.36	60.16	60.32	61.03	60.92		GKD	65.42	65.16	65.06	65.04	65.03
	Ours	**60.44**	**62.84**	**62.91**	**62.98**	**63.21**		Ours	**68.56**	**69.43**	**69.73**	**69.81**	**69.93**
R→P	SHOT	82.92	83.23	83.31	83.82	83.33	S→R	SHOT	77.78	77.77	77.71	77.66	77.65
	TPDS	82.36	83.26	83.89	84.23	84.23		GKD	77.62	77.64	77.70	77.72	77.73
	Ours	**83.71**	**85.13**	**85.40**	**85.49**	**85.60**		Ours	**78.78**	**79.81**	**80.16**	**80.16**	**80.35**

E. Ablation study details

Effect of λ (the number of virtual features). Across both Office-Home and DomainNet-126 (Tabs. 12 and 13), increasing λ from 0 to a moderate value consistently improves accuracy, confirming that virtual features help stabilize pseudo labeling by better approximating class topology. Performance saturates around $\lambda \approx$ 15–20, and additional virtual samples bring no further gains.

Effect of α (weight of $\mathcal{L}_{CE}$). With λ and β fixed, varying α shows a mild but clear tendency: too small a value weakens classification supervision, while too large a value suppresses the effect of regularization. Both benchmarks exhibit a stable region around $\alpha \approx 0.20$–0.30, where peak performance was achieved. We therefore adopted $\alpha = 0.30$ for Office-Home and $\alpha = 0.20$ for DomainNet-126.

Effect of β (weight of $\mathcal{L}_{GV}$). Accuracy varies only slightly as β changes within 1.20–1.40 on both benchmarks. Very small β weakens gravity consistency, whereas overly large values introduce excessive regularization. The chosen value $\beta = 1.35$ lies near the center of this stable region, showing that the gravity term is effective without requiring fine-grained tuning.

F. Pseudo label accuracy comparison with existing methods

As shown in Tab. 14, we compared the pseudo-label accuracy of our method against representative SFDA baselines on both the Office-Home and DomainNet-126 datasets. For a fair and meaningful evaluation, we included only the methods that satisfy the following criteria: (1) they use a comparable training schedule with the same number of epochs, (2) official implementations are publicly available so that we can fully reproduce them in our environment,

Table 15. Accuracy (%) comparison of label propagation algorithms on the Office-Home dataset.

	Venue	A→C	A→P	A→R	C→A	C→P	C→R	P→A	P→C	P→R	R→A	R→C	R→P	Avg.
Initial PL	-	59.2	79.2	81.7	69.8	79.3	79.3	67.4	57.4	82.5	73.3	62.4	84.4	73.0
Label propagation	CMU'02	60.0	80.7	81.7	**71.3**	77.4	79.1	70.4	57.5	81.0	74.7	60.4	83.9	73.2
Label spreading	NeurIPS'04	**60.1**	80.2	81.8	70.1	78.2	80.2	**70.9**	57.8	82.9	74.3	60.4	85.4	73.5
Ours		59.9	**81.1**	**83.0**	71.2	**80.4**	**81.1**	70.1	**58.9**	**83.5**	**74.9**	**63.1**	**85.7**	**74.4**

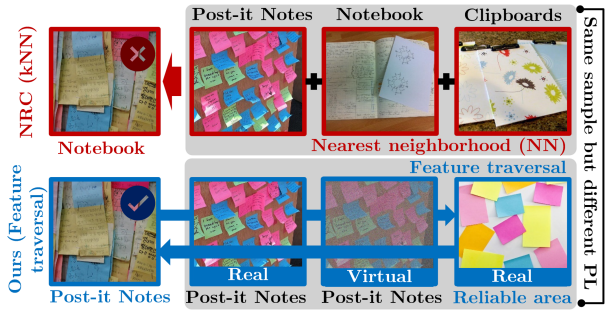

Figure 4. Comparison between conventional kNN-based pseudo-label assignment and the proposed feature traversal strategy.

and (3) they achieve competitive performance on the corresponding dataset. Based on these criteria, TPDS [28] was selected for Office-Home [31] and GKD [27] was selected for DomainNet-126 [23], while SHOT [16] was included as a common baseline. All methods were trained using identical backbones and the same number of training epochs (3, 6, 9, 12, and 15). After each epoch, we compared the accuracy of the pseudo labels generated by each method during training. Across most adaptation tasks in both datasets, our method consistently achieved higher pseudo-label accuracy than SHOT, TPDS, and GKD. As training proceeds, the gap either persists or widens, demonstrating that our universe-aware pseudo-labeling strategy produces more reliable pseudo-labels from the beginning and effectively avoids noise accumulation. In contrast, prior methods often exhibit stagnation or fluctuation in pseudo-label accuracy across epochs. Our approach steadily improves pseudo-label accuracy, providing a stable supervision signal throughout training. This continuous refinement of pseudo labels by our pseudo-labeling strategy and gravity consistency directly contributes to the performance gains reported in the manuscript.

G. Additional analysis on pseudo-label refinement.

We provide additional comparisons to examine the effectiveness of the proposed feature traversal algorithm for pseudo labeling. Specifically, we replaced our feature traversal algorithm with several alternative baselines, including the initial pseudo labels and representative graph-based propagation methods [36, 38]. As shown in Tab. 15, the proposed method yields the best average performance over all adaptation tasks. This result indicates that standard pseudo-label assignment alone is not sufficient to fully utilize the structural information of the target feature space, whereas our method enables more reliable pseudo-label refinement.

Table 16. Comparison of computational overhead. TT, TTB, and PGM denote the total training time, training time per batch, and peak GPU memory usage, respectively.

Method	TT (s) ↓	TTB (s) ↓	PGM (GB) ↓
SHOT	597.33	0.5771	7.759
TPDS	1144.18	1.1055	7.759
Our	589.40	0.5695	12.884

H. Qualitative analysis of feature traversal algorithm.

Figure 4 presents representative feature traversal paths sampled from the actual feature traversal process of the proposed method. These examples are provided to qualitatively examine how the proposed traversal differs from conventional kNN-based pseudo-label assignment [34]. Existing kNN-based approaches determine labels only from local neighbors, which can be problematic when the neighboring samples are distributed near a decision boundary. By contrast, our method performs iterative feature traversal to move from the query feature toward a more reliable region of the embedding space. Once a reliable anchor is reached, its label is propagated back to the query feature. This behavior demonstrates that the proposed method can mitigate the limitation of purely local pseudo-label assignment and better exploit the geometry of the feature manifold.

I. Computational overhead

As shown in Tab. 16, our method achieves the fastest iteration speed and the shortest overall training time among the compared approaches [16, 28], but it requires noticeably higher peak GPU memory due to the computation of class-wise covariance matrices and the construction of virtual features. While this memory overhead does not hinder performance in our experimental setting, it may limit scalability on devices with constrained GPU resources. Overall, our approach provides competitive computational efficiency at the cost of increased memory usage.

J. Additional result of gravity consistency

Figure 5 visualizes the four gravity consistency types (A–D), originally illustrated conceptually in Fig. 1 of the manuscript, now extracted from multiple adaptation tasks on the Office-Home dataset. Each target sample is categorized into one of the four types based on the feature similarity and logit consistency between weakly and strongly augmented views. Across different transfer directions, we consistently observe that the proportion of type A samples (highly consistent in both feature and logit space) increases over training, while type D samples (inconsistent in both spaces) progressively decrease. This indicates that gravity consistency effectively drives unstable samples (types C and D) toward more reliable states, especially type A, in a wide range of adaptation scenarios. In addition, when compared to standard Kullback–Leibler (KL)-based consistency regularization (CR) under the same settings, our gravity consistency (GV) produces a noticeably larger shift of samples into type A. With KL-based CR, the growth of type A is slower and often plateaus, whereas GV leads to a faster and more substantial increase of type A, along with more frequent direct transitions from type D to type A. These observations suggest that GV goes beyond simply matching predictions between two views; it acts as a joint regularizer in feature and logit space that pulls samples toward structurally coherent regions of the manifold. Consequently, GV yields higher-quality pseudo labels throughout training, which in turn contributes to the improved adaptation performance reported in the manuscript.

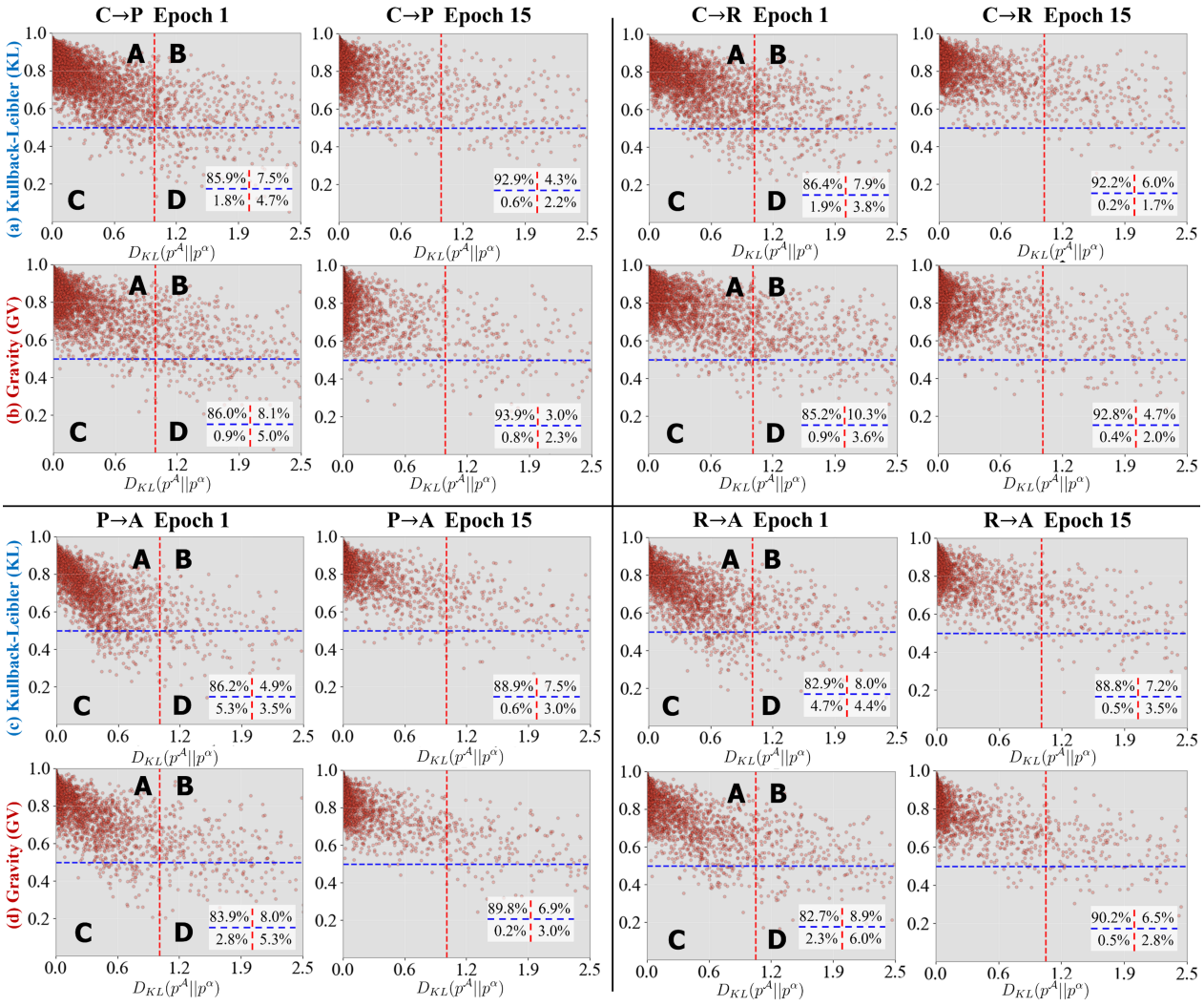

Figure 5. Visualization of the four gravity consistency types (A–D) across multiple adaptation tasks on the Office-Home dataset. The numbers in the four regions of the proportion box at the bottom-right of the scatter plot denote the proportion of samples of each type, defined by feature similarity and logit consistency between weakly and strongly augmented views. Here, the labels at the top of each plot denote the corresponding adaptation tasks C→P, C→R, P→A, and R→A, where A, C, P, and R indicate the Art, Clipart, Product, and Real-World domains of Office-Home, respectively. Rows (a) and (c) show results obtained using standard KL-based consistency regularization (CR), while rows (b) and (d) illustrate the outcomes when applying our gravity consistency (GV). Compared to CR, GV yields a faster and larger increase in type A samples (high feature and logit agreement) and a stronger reduction of unreliable types (especially type D), illustrating that GV more effectively drives target samples toward structurally coherent and reliable regions of the feature space.