Progressive Distribution Bridging: Unsupervised Adaptation for Large-scale Pre-trained Models via Adaptive Auxiliary Data

Supplementary Material

A. More Discussions

A.1. Fairness Considerations for PDB vs. UDA

As formulated in Sec. 3.1, both UDA methods and our proposed PDB utilize auxiliary datasets to facilitate model adaptation to the unlabeled target domain. However, UDA employs static labeled source domains, whereas PDB adaptively constructs auxiliary datasets tailored to the target domain through sophisticated retrieval algorithms from LAION-400M. Evidently, they inherently share similarities and differences at the data level, necessitating further discussion regarding "fair comparison" or "effectiveness validation". We now discuss this issue progressively from basic to advanced perspectives.

Is there an inherent advantage in sampling data from the extensive LAION-400M compared to utilize UDA source domains? No, there is no "inherent" advantage. Using LAION-400M is more challenging than UDA's source domains. UDA's source domains are typically well-curated, labeled, and closed-set, while LAION-400M is assumption-free, weakly labeled, and noisy. Despite the increased difficulty, PDB overcomes the challenges and creates advantages, even besting annotation-reliant UDA on OfficeHome and VisDA. PDB demonstrates the potential of constructing adaptive auxiliary data in the current era of large pre-trained models and large-scale datasets.

Why does PDB achieve superior performance compared to UDA on OfficeHome and VisDA, while failing to demonstrate advantages on DomainNet? This is because annotations play a more significant role on more challenging datasets. DomainNet is considerably more difficult than OfficeHome/VisDA, thus UDA possesses a natural advantage over annotation-free paradigms like UFT/PDB.

Why not align the auxiliary data budgets between UDA and PDB for a "fair" comparison? This is related to performance saturation. As shown in Tab. 8, PDB employs more auxiliary samples than UDA on Office-Home, fewer on DomainNet, and markedly fewer on average across all benchmarks. At this point, one might argue that PDB's advantage on OfficeHome stems from using more auxiliary data. However, examining the VisDA results reveals this is not the case. Compared to UDA, PDB achieves better performance on VisDA while using significantly less auxiliary data. For this intricate and complex issue, we believe the first aspect to question is: is UDA's auxiliary data budget the most reasonable? Obviously, the answer is negative. Using such a massive amount of auxiliary data but with significant domain gaps on VisDA, which has only 12 categories, yields minimal benefits. Therefore, we did not align PDB's budget with UDA's, but instead set a budget of $N_{max} = 100$ per class based on PDB's performance saturation phenomenon and hyperparameter consistency.

A.2. Differences Between PDB and Active DA

Active DA selects ambiguous and representative samples from the target domain for annotation, in addition to using labeled source data. Its goal is to maximize target domain performance under a fixed annotation budget. Our work addresses the limitations of UFT and UDA in the context of VLMs, focusing on constructing a more suitable "source" domain from large-scale datasets. Key differences include: (1) Our setting is annotation-free and has no oracle access. (2) Our sample pool's label space is open. (3) Our sample pool includes a wide range of styles. We propose semantic filters for the first two challenges, the style controller for the third. While both Active DA and our method involve "sample selection", they differ significantly in terms of motivations, selection criteria, and encountered challenges.

A.3. Differences Between PDB and REACT/RA-CLIP

Our work focuses on UFT/UDA's intrinsic limitations for VLM "adaptation"; REACT [52], in contrast, operates before such "adaptation" (see REACT's Introduction). REACT is like continual pre-training for specific domains. REACT lacks adaptivity, retrieving the same data for domains of different styles. In contrast, PDB constructs adaptive and progressive data streams. Besides, REACT's training is costly (10k samples/class, contrastive loss), whereas our semantic filters enable data efficiency and more direct supervision (0.1k samples/class, CE loss). On the other hand, RA-CLIP [84] enables models to enhance zero-shot performance via test-time knowledge retrieval. Conversely, PDB relies on no external knowledge at test time, aiming to push performance to its limits for a specific scenario.

B. More Results

B.1. Ablation Study on Training Loss

We evaluate the effectiveness of GCE loss on DomainNet, as shown in Tab. A3. For domains Q and C, which have poor initial performance, GCE loss demonstrates better robustness. However, for domain R with high initial performance, standard CE loss performs slightly better. We attribute this

	→Ar	→Cl	→Pr	→Rw	**Avg**
CLIP [63]	82.9	67.8	89.0	89.8	82.4
POUF-FT [74]	84.4	73.8	92.7	91.7	85.7
PDB-TO	87.3	76.9	92.9	91.5	87.2
PDB	**87.9**	**78.1**	**93.2**	**92.1**	**87.8**

Table A1. Performance comparison between our strong baseline (PDB-TO) and our full method (PDB). Ar, Cl, Pr and Rw are domains in OfficeHome [77]. In "PDB-TO", the model is adapted with only target domain optimization.

	→C	→I	→P	→Q	→R	→S	**Avg**
CLIP [63]	70.1	46.4	61.7	13.7	82.9	62.6	56.2
POUF-FT [74]	73.8	**55.7**	68.6	18.8	84.6	66.4	61.3
PDB-TO	75.0	54.6	71.1	20.9	84.7	67.0	62.2
PDB	**75.9**	55.2	**71.6**	**22.8**	**85.5**	**67.6**	**63.1**

Table A2. Performance comparison between our strong baseline (PDB-TO) and our full method (PDB) on DomainNet [61].

to the fact that the performance of semantic filters partially depends on initial performance. Thus, applying semantic filters on domains with poor initial performance introduces more noisy samples, which requires more robust adaptation process.

On the other hand, Tabs. A1 and A2 present the performance comparison between our strong baseline (PDB-TO) and full method (PDB). For reference, we also include the results of POUF [74]. As shown in the tables, our PDB-TO outperforms the UFT method POUF by margins of 1.5% and 0.9% on OfficeHome [77] and DomainNet [61], respectively, which has demonstrated the effectiveness of our $l_{\text{CM-IDCo}}$ (Eq. (7)). Furthermore, by incorporating our adaptive auxiliary datasets, PDB achieves additional gains of 0.6% and 0.9% over the strong baseline (PDB-TO).

	→Q	→R	→C	**Avg**
CLIP [63]	13.7	82.9	70.1	55.6
PDB-SO w/ CE	18.1	**85.3**	74.0	59.1
PDB-SO	**19.1**	85.2	**74.8**	**59.7**

Table A3. Ablation study on source domain training loss. Q, R and C are domains in DomainNet [61]. In "PDB-SO w/ CE", we replace the GCE loss with the standard CE loss.

B.2. Analysis of Retrieval Size

Tab. A4 analyzes PDB's scalability with respect to the number of retrieved samples. Our experiments demonstrate that retrieving a set of 20 samples per class is sufficient to obtain significant performance improvements relative to PDB-TO. Furthermore, while increasing samples per class from 20 to 300 provides slight performance gains. The observed saturation in performance is consistent with findings reported in other few-shot learning methods (*e.g.*, CoOp). We opt for $N_{max} = 100$ as our default setting to maintain a good balance between performance and efficiency.

On the other hand, our method requires significantly fewer samples compared to REACT [52], which requires at least 3000 samples per class on ImageNet-1K [13] to achieve significant improvements. We attribute this advantage to our well designed cascaded semantic filters (Sec. 4.1) and the adoption of pseudo-label training loss (Sec. 4.3), where the latter provides more direct supervision compared to image-text contrastive training [52]. It is worth noting that our performance gains are measured against the stronger baseline (PDB-TO), not CLIP zero-shot inference.

	Retrieval Size	→Q	→R	→C	avg
CLIP [63]	0	13.70	82.90	70.10	55.60
PDB-TO	0	20.85	84.73	75.01	60.20
PDB	20×345	22.78	85.22	75.64	61.11
	100×345 †	22.82	**85.51**	75.86	61.40
	200×345	22.90	85.46	**76.05**	61.47
	300×345	**23.13**	85.45	75.91	**61.49**

Table A4. Performance scaling of PDB with respect to the number of retrieved samples. † denotes our default experimental setting, *i.e.* $N_{max} = 100$. In "PDB-TO", the model is adapted with only target domain optimization.

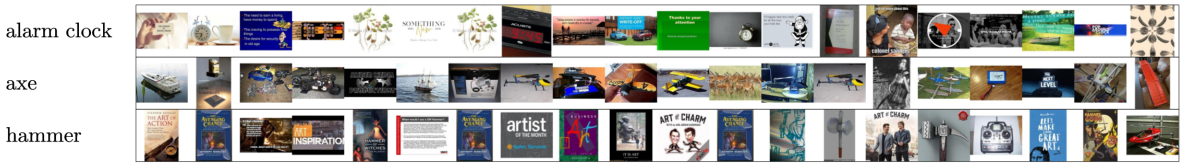
C. More Visualization

C.1. Analysis of CLIP-based Retrieval Algorithm

As shown in Fig. A1, we present the visualization results of four different basic retrieval algorithms (T2T, I2T, T2I and I2I). We randomly sampled 20 instances each from the retrieval results of "alarm clock", "axe" and "hammer", where instances with top 100 matching similarity are recalled. As illustrated in Fig. A1(a) and (b), retrieval based on matching caption in the image-text pairs, *i.e.* T2T and I2T, recalls almost no useful samples, which is consistent with the arguments presented in the main paper (Sec. 3.2). For T2I retrieval, limited by imperfect image-text alignment from pre-training, only succeeds in retrieving a small number of correct samples for "alarm clock". Additionally, although I2I effectively identifies "alarm clock" through image similarity matching, it still fails to handle categories with lower initial performance, such as "axe" and "hammer".

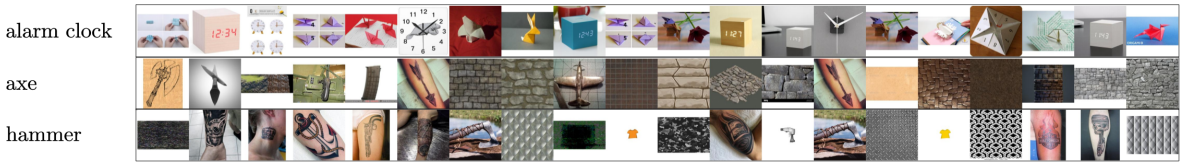
C.2. Analysis of Cascaded Semantic Filters

While the effectiveness of each filter has been demonstrated through quantitative experimental results in the main paper, *i.e.* Tab. 5, we provide visualization here for more intuitive insights. As default, we follow the same visualization strategy as described in the previous subsection (Sec. C.1).

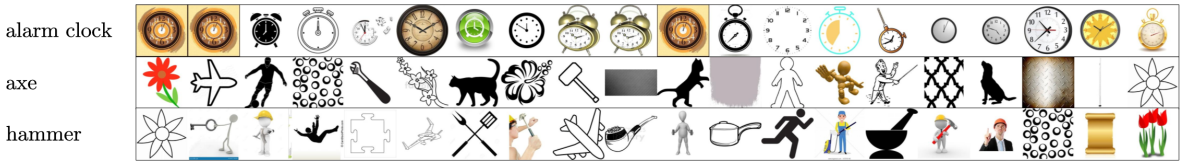


(a) T2T

(b) I2T



(c) T2I



(d) I2I

Figure A1. Visualization of results from four different basic CLIP-based retrieval algorithms (Sec. 3.2). All results comes from the Clipart domain of Mini-DomainNet.

Fig. A2 presents examples from three categories - "axe", "hammer", and "string bean" - selected from the Clipart domain of Mini-DomainNet. Note that "axe" and "hammer" share are visually similar objects, and "string bean" exhibits poor initial classification accuracy. A comparison between Fig. A1 (a) and Fig. A2 (a) demonstrates how Concept Matching (CM) filter enhances vanilla T2T retrieval performance by eliminating a substantial amount of noisy samples. Comparing Fig. A2 (a) and (b) shows that GMM effectively eliminates text-only images, *e.g.* results of "hammer". Fig. A2 (c) demonstrates the effect of style controller (Sec. 4.2). Finally, Fig. A2 (d) clearly shows that I2I retrieval fails to handle visually similar categories ("axe" and "hammer") and hard categories ("string bean").

C.3. Class-wise Performance Curves

As stated in the main paper Sec. 4.1, we propose T2T retrieval as the base retrieval algorithm due to its superior properties on visually similar and hard categories in unsupervised adaptation. Here, we compare the performance of PDB and PDB-TO on these specific categories to verify our claims. As shown in Fig. A3, through three representative categories - "axe", "hammer", and "string bean" - PDB consistently achieves significantly better final results than PDB-TO. Besides, these results also confirm that auxiliary data is essential when dealing with poor initial performance.

C.4. Dealing with Textual Ambiguity

As shown in the Fig. A4, we present retrieval results for the ambiguous category "Marker". This word can refer to both "road sign/milestone" and "writing marker/pen", with these meanings being significantly different from each other. We mitigate the ambiguity issue by incorporating minimal visual information ($\alpha = 0.1$) into the concept prototypes.

D. Implementation Details

D.1. LoRA Fine-tuning

Drawing inspiration from recent studies [6, 15, 68], we utilize LoRA [28] to adapt both the image and text encoders of the CLIP [63] model for effective target domain adaptation. In the Transformer [76] architecture, LoRA implements a low-rank decomposition scheme $W_0 + \Delta W = W_0 + BA$ to constrain the updates of pre-trained linear layer weights

(a) T2T + CM ($\alpha = 0.0$)

(b) T2T + CM + GMM ($\alpha = 0.0$)

(c) T2T + Style + CM ($\alpha = 0.7$)

(d) I2I + CM ($\alpha = 1.0$)

Figure A2. Visualization of retrieval results using different semantic filters. "CM" denotes Concept Matching Filter (Sec. 4.1). "GMM" denotes heuristic GMM-based filter (Sec. 4.1). "Style" denotes Style Filter (Sec. 4.2). All results comes from the Clipart domain of Mini-DomainNet.

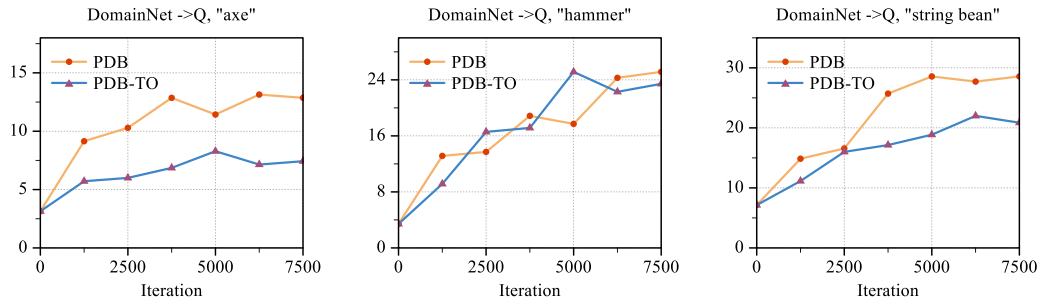


Figure A3. Performance curves of specific categories, *i.e.* "axe", "hammer" and "string bean". Note that the former two constitute visually similar classes, while the latter represents a hard class. In "PDB-TO", the model is adapted with only target domain optimization.

$W_0 \in \mathbb{R}^{m \times n}$. The residual weights are represented by matrices $B \in \mathbb{R}^{m \times r}$ and $A \in \mathbb{R}^{r \times n}$, where the rank r is much smaller than $\min(m, n)$. The training process keeps the base weight matrix W_0 fixed, with optimization performed exclusively on residual matrices A and B. Following the original LoRA [28] paper, we initialize these matrices accordingly and equate the scaling factor α with rank r. Our implementation sets $r = 8$ and only apply LoRA [28] to the self-attention modules, specifically the query (W_q) and value (W_v) weight matrices. For model inference, the residual weights A and B are merged into W_0.

Figure A4. Visualization of retrieval results with ambiguous class name. The results comes from the Clipart domain of OfficeHome [77] and the class name is "Marker". For both (a) and (b), we employ the same experimental configuration except the value of α: "T2T + Style + CM + GMM".

Algorithm 1: Our Retrieval Algorithm

Input: Image encoder f, text encoder g and j-th class name t_j. Unlabeled target data $\mathcal{D}^t = \{\mathbf{x}_i^t\}_{i=1}^{N_t}$ and large-scale image-text pairs $\mathcal{D}^{all} = \{(\mathbf{x}_i, c_i)\}_{i=1}^{N}$. $\alpha \in \{0.1, 0.4, 0.7\}$.

Output: Auxiliary dataset $\mathcal{D}_{j,\alpha}^2$ of j-th category with specific α.

Perform T2T Retrieval via pre-built 16GB KNN index: $\mathcal{D}_j^0 = \{(\mathbf{x}_i, c_i) | \text{ Top-K}_{(\mathbf{x}_i, c_i) \in \mathcal{D}^{all}} \, g(t_j)^T g(c_i)\}$.

Get stylized queries $\mathbf{Q}^{style}$ for all categories via Eq. (6), and perform Style Filter: $\mathcal{D}_{j,\alpha}^{sty} = \{(\mathbf{x}_i, c_i) | \text{ Top-K}_{(\mathbf{x}_i, c_i) \in \mathcal{D}_j^0} \, f(\mathbf{x}_i)^T \mathbf{q}_j^{style}\}$.

Perform Concept Matching Filter with stylized queries: $\mathcal{D}_{j,\alpha}^1 = \{(\mathbf{x}_i, c_i) \in \mathcal{D}_{j,\alpha}^{sty} | \delta_j(f(\mathbf{x}_i)^T \mathbf{Q}^{style}) > \gamma\}$.

if $\alpha = 0.1$ **then**

  Performing GMM clustering in Eq. (4) and filter out text-only images via GMM filter: $\mathcal{D}_{j,\alpha}^2 = \{(\mathbf{x}_i, c_i) \in \mathcal{D}_{j,\alpha}^1 | f(\mathbf{x}_i) \in \mathcal{C}_k\}$.

else

  $\mathcal{D}_{j,\alpha}^2 = \mathcal{D}_{j,\alpha}^1$.

return $\mathcal{D}_{j,\alpha}^2$

D.2. Retrieval Algorithm

To demonstrate our retrieval algorithm with greater rigor, we present its pseudocode implementation in Algorithm 1.