Adapt via Bayesian Nonparametric Clustering: Fine-Grained Classification for Model Recycling Under Domain and Category Shift

Anonymous authors
Paper under double-blind review

Abstract

Recycling pretrained classification models for new domains, known as Source-Free Domain Adaptation (SFDA), has been extensively studied under the closed-set assumption that source and target domains share identical label spaces. However, this assumption does not hold when unseen classes appear in the target domain. Addressing this category shift is challenging, as unknown target classes usually arise with no prior knowledge of their identities or number, and becomes particularly difficult in the source-free setting, where access to source data is unavailable. Most existing methods treat all unknown classes as a single group during both training and evaluation, limiting their capacity to model the underlying structure within the unknown class space. In this work, we present Adapt via Bayesian Nonparametric Clustering (ABC), a novel framework designed for SFDA scenarios where unknown target classes are present. Unlike prior methods, ABC explicitly achieves fine-grained classification of unknown target classes, offering a more structured vision of the problem. Our method first identifies high-confidence target samples likely to belong to known source classes. Using these as guidance, we develop a guided Bayesian nonparametric clustering approach that learns distinct prototypes for both known and unknown classes without requiring the number of unknown classes *a priori*, and assigns target samples accordingly. We further introduce a training objective that refines the source model by encouraging prototype-based discriminability and local prediction consistency. Experiments show that our method achieves competitive performance on standard benchmarks while simultaneously providing effective clustering of unknown classes.

1 Introduction

Deep learning models usually require large labeled datasets, but due to costly annotation (Ganin et al., 2016), labeled data in a source domain is often used to train models for unlabeled data in target domains. However, due to domain shift (Torralba & Efros, 2011), these models often fail to generalize, motivating the development of Unsupervised Domain Adaptation (UDA) techniques to bridge the gap. Traditional UDA methods typically assume no category shift, meaning the label sets of the source and target domains, denoted as $\mathcal{C}_s$ and $\mathcal{C}_t$, are identical, i.e., $\mathcal{C}_s = \mathcal{C}_t$, a setting known as closed-set domain adaptation. With the progress of UDA, increasing attention has been paid to the more realistic category shift scenarios that relax this assumption, allowing $\mathcal{C}_s \not\supseteq \mathcal{C}_t$, and thus permit the presence of unknown target classes. While many methods have been proposed to handle different category shift settings, most still follow traditional UDA approaches that require access to source data. This requirement limits the practicality of UDA methods due to factors such as privacy concerns, bandwidth constraints in distributed systems, and the lack of access to source data, for example when it is sensitive, too large to transmit, or lost due to corruption (Kundu et al., 2020). In such cases, it is desirable to recycle the source model without access to the original source data. These challenges have motivated the development of source-free domain adaptation (SFDA) methods, which perform UDA without the source data. Despite their potential, existing SFDA research has primarily focused on the closed-set setting. More realistic scenarios with label mismatches, where unknown target classes may be present (as illustrated in Figure 1), remain underexplored due to their increased complexity and ambiguity, with only a few works addressing these challenges (Liang et al., 2021; Qu et al., 2023; Liang et al., 2020).

While these prior works have demonstrated effective performance in classifying known classes and effectively detecting samples from unknown classes, they typically frame the problem within a standard classification setting, treating all unknown classes as a single unified "unknown" category. However, collapsing all unknown classes into one label risks overlooking the latent structure within them, potentially hindering transfer learning performance. Furthermore, with the growing interest and development in deep clustering (Min et al., 2018), where unlabeled samples are automatically grouped based on their learned embeddings, there is increasing practical value in partitioning samples from unknown classes into finer-grained clusters. This not only enhances interpretability, but also supports downstream tasks that benefit from more granular classification. For example, in medical imaging, it may reveal previously unobserved patient subpopulations when adapting a diagnosis model to data from a new cohort; in robotics, it could help systems distinguish novel sensory patterns in unfamiliar settings; and in security, it can facilitate identifying and separating unseen attacks or anomalous behaviors when deploying a recognition model to a new environment, enabling more effective handling of different unknown patterns.

A promising strategy is to integrate clustering into the adaptation process, enabling the model to simultaneously classify known classes and cluster unknown samples into distinct latent groups. This joint task presents additional challenges: not only is the identity of unknown classes unavailable during adaptation, but the number of unknown classes is also unknown. The number of ways to cluster n samples into k non-empty subsets is given by the Stirling number of the second kind $S(n, k)$, whereas allowing an unknown number of subsets yields the Bell number $B_n = \sum_{k=0}^{n} S(n, k)$, which grows much more rapidly with n. Traditional methods that tune the number of clusters K (e.g., using internal measures such as the Silhouette score (Rousseeuw, 1987) or Elbow methods (Thorndike, 1953)) often require repeatedly refitting the clustering model across a set of candidate K values, which can introduce non-negligible computational overhead and make integration into the SFDA pipeline less straightforward. More critically, such tuning procedures require defining a search space for the number of clusters K, e.g., for a grid search, which relies heavily on prior knowledge of the true K and whose range depends on the dataset scale. For instance, target-domain datasets with many classes demand much broader search ranges than those with only a few, making grid design not very practical for real-world scenarios where prior knowledge of the number of target-domain classes is usually unavailable.

In this paper, we propose a novel algorithm to *A̲dapt via B̲ayesian Nonparametric C̲lustering (ABC)*, which simultaneously classifies known-class samples and clusters unknown ones, while recycling a source model when unknown classes arise in the new domain. ABC first identifies target samples with high confidence of belonging to known classes. Using these detected samples as anchors, a guided Bayesian nonparametric clustering (BNC) model is introduced to assign the remaining samples either to existing known classes or to newly formed unknown classes, while simultaneously learning prototypes for all classes, both known and unknown. We further incorporate a prototype regularizer, which aligns samples with their assigned prototypes, and a local consistency regularizer, which promotes prediction agreement within local neighborhoods, together facilitating iterative refinement of the source model for adaptation to target samples. In our model, the BNC component automatically learns the number of unknown classes, allowing their structure to emerge *a posteriori* and eliminating the need for costly grid searches to determine K. We conduct extensive experiments and analyses, and demonstrate its effectiveness and solid performance compared to competitive baselines.

2 Related Work

Unsupervised Domain Adaptation (UDA): UDA methods improve the generalization ability of deep neural networks across different domains and mitigate the performance degradation caused by domain shift. Among the most widely adopted approaches are discrepancy minimization methods(Tzeng et al., 2014; Long et al., 2015; Sun et al., 2016; Long et al., 2017; 2016; Chen et al., 2019) and adversarial training techniques (Ganin & Lempitsky, 2015; Ganin et al., 2016; Tzeng et al., 2017; Zhang et al., 2018; Gonçalves et al., 2014). To reduce domain shift, discrepancy minimization methods introduce a divergence measure between data distributions across domains and minimize it, while adversarial training, inspired by generative adversarial networks (GANs)(Goodfellow et al., 2014), leverages a domain discriminator to classify samples as originating

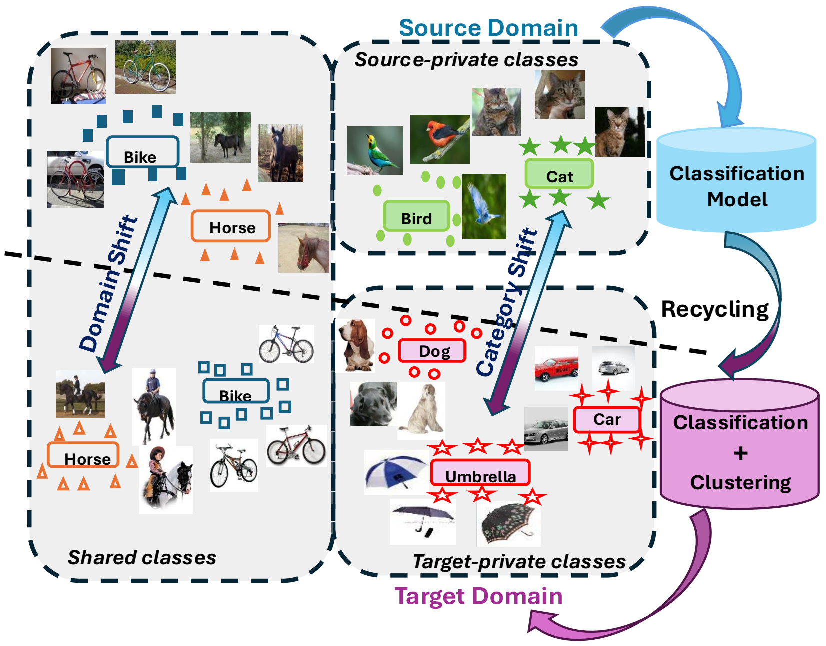


Figure 1: An illustration of the SFDA problem with unknown target classes (with both source- and target-private classes displayed) and the goal of our method.

from the source or target domain, thereby encouraging the learning of domain-invariant features (Ganin & Lempitsky, 2015; Ganin et al., 2016). More recent research has begun to address the UDA problem under category shift. Research has examined partial(Cao et al., 2018a;b; Wang et al., 2020; Li et al., 2021), open-set(Liu et al., 2022; Panareda Busto & Gall, 2017; Saito et al., 2018; Jang et al., 2022; Li et al., 2021), and open-partial domain adaptation scenarios(Qu et al., 2022; You et al., 2019; Li et al., 2021). Furthermore, there is a growing interest in developing general methods that can handle different settings (Chen et al., 2022; Saito et al., 2020; Li et al., 2021). Many of these approaches detect samples from unknown classes by identifying uncertain instances based on indicators derived from prediction outputs, such as entropy values (You et al., 2019; Jang et al., 2022), while some methods leverage consensus clustering to identify shared classes (Li et al., 2021).

Source-free Domain Adaptation (SFDA): SFDA focuses on adapting a model pre-trained on the source domain to a target domain without requiring access to the source data. Many methods have been proposed to address this problem under conventional settings, typically assuming no category shift between source and target domains (Li et al., 2021; Liang et al., 2020; Qu et al., 2022; Yang et al., 2021b; Ding et al., 2022; Yang et al., 2022). Some approaches utilize the source model to extract anchor features and then assign pseudo-labels based on these anchors (Ding et al., 2022), while others adapt the source model by promoting consistency among local neighborhood features in the target domain (Yang et al., 2022). Only a few works have tackled SFDA under category shift conditions, where target classes may not completely overlap with source classes (Qu et al., 2023; Kundu et al., 2020; Liang et al., 2021). Although these methods represent significant progress, they often treat all unknown classes uniformly in both modeling and evaluation, and seldom explicitly explore the clustering behavior of samples from unknown classes in the target domain.

Bayesian Nonparametric Clustering (BNC): Clustering aims to group similar observations into a fixed number of clusters. Common methods like K-Means and finite mixture models require specifying the number of clusters, K, in advance(JAIN et al., 1999; Reynolds, 2015). While internal validation approaches can help tune this parameter by evaluating internal metrics across different values of K(Wang & Ye, 2024), they often require repeated model refitting, selection of evaluation criteria, and careful design of a search range.

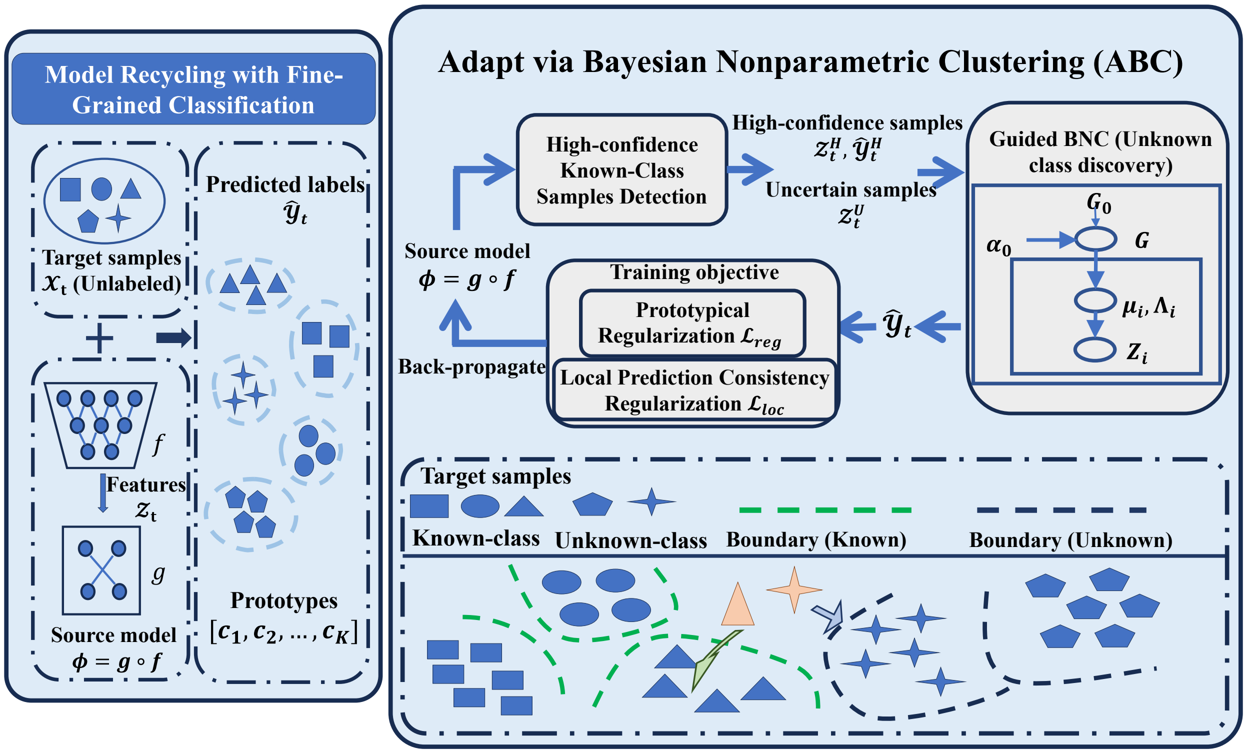

Model Recycling with Fine-Grained Classification
Target samples $\mathcal{X}_t$ (Unlabeled)
Predicted labels $\hat{y}_t$
f
Features z_t
g
Source model $\phi = g \circ f$
Prototypes $[c_1, c_2, \ldots, c_K]$
Adapt via Bayesian Nonparametric Clustering (ABC)
High-confidence Known-Class Samples Detection
High-confidence samples $z_t^H, \hat{y}_t^H$
Uncertain samples z_t^U
Guided BNC (Unknown class discovery)
G_0
α_0
G
μ_i, Λ_i
Z_i
Source model $\phi = g \circ f$
Back-propagate
Training objective
Prototypical Regularization $\mathcal{L}_{reg}$
Local Prediction Consistency Regularization $\mathcal{L}_{loc}$
$\hat{y}_t$
Target samples
Known-class
Unknown-class
Boundary (Known)
Boundary (Unknown)

Figure 2: An overview of Adapt via Bayesian Nonparametric Clustering (ABC).

BNC addresses this limitation by inferring the number of clusters as part of the model (Hjort et al., 2010), allowing clustering without predefining the number of clusters. BNC encompasses a flexible family of mixture models, such as Dirichlet Process Mixtures (Ferguson, 1973; Antoniak, 1974), Pitman–Yor Process Mixtures (Pitman & Yor, 1997), and Normalized Generalized Gamma Process Mixtures (Lijoi et al., 2007), which can approximate complex data distributions. Advances in scalable inference techniques, such as variational inference (Blei & Jordan, 2006; Ni et al., 2020b) and consensus Monte Carlo (Ni et al., 2020c;a), have enabled BNC to handle large datasets, making it applicable in large-scale scenarios. Recent advances have integrated BNC with deep neural networks, enabling automatic discovery of the number of clusters when clustering high-dimensional data(Wang et al., 2021; Ronen et al., 2022). Together, these developments make BNC accessible to a wide range of large-scale and high-dimensional data settings.

3 Methodology

In SFDA, neither source data $\mathcal{X}_s$, source labels $\mathcal{Y}_s$, nor target labels $\mathcal{Y}_t$ are accessible during adaptation. To address scenarios with unknown classes, we consider the setting where the source and target class sets $\mathcal{C}_s$ and $\mathcal{C}_t$ satisfy $\mathcal{C}_s \cap \mathcal{C}_t \neq \emptyset$, and $\mathcal{C}_t \setminus \mathcal{C}_s \neq \emptyset$. For clarity, we define $\mathcal{C} = \mathcal{C}_s \cap \mathcal{C}_t$ as the *common label space*, $\mathcal{C}_s^c = \mathcal{C}_s \setminus \mathcal{C}_t$ as the *source-private label space*, and $\mathcal{C}_t^c = \mathcal{C}_t \setminus \mathcal{C}_s$ as the *target-private label space*. We will refer to the classes within $\mathcal{C}_s$ as known classes and those within $\mathcal{C}_t^c$ as unknown classes. While prior work typically treats classes in $\mathcal{C}_t^c$ as a single category c_{unk}, we aim to distinguish these classes individually, denoting the set of discovered unknown classes by $\hat{\mathcal{C}}_t^c$. Our learning objective is twofold: to classify target instances as the estimated label $\hat{y}_i^t \in \mathcal{C}$ if they belong to known classes, and to assign them to discovered groups $\hat{\mathcal{C}}_t^c$ if they belong to unknown classes. The overall workflow of the proposed algorithm is illustrated in Figure 2, with detailed steps described in the following sections.

3.1 High-Confidence Common Class Samples

We first focus on detecting samples most likely to belong to the common label set $\mathcal{C}$. Let the source model be denoted as $\phi = g \circ f$, where f represents the feature extractor mapping $x_i \in \mathcal{X}_s$ to a feature representation $z_i \in \mathbb{R}^m$, and g is the classifier that maps z_i to source classes $\mathcal{C}_s = \{1, 2, \cdots, K_s\}$, where K_s represents the number of source classes. Let $\mathsf{w}^g \in \mathbb{R}^{m \times K_s}$ denote the weight matrix of the classifier g, where each column $\mathsf{w}_k^g \in \mathbb{R}^m$ corresponds to the weight vector associated with source class k. Each w_k^g can be interpreted as an anchor representing the k-th source class in the feature space, and has been leveraged to initialize source cluster centers for spherical k-means clustering in previous SFDA methods, where target instances are assigned to source classes by maximizing cosine similarity to these centers (Ding et al., 2022). The cosine similarity between a target feature z_i and each source class anchor w_k^g, defined as $cos(z_i, w_k^g) = \frac{{z_i}^\top \mathsf{w}_k^g}{\|z_i\| \|\mathsf{w}_k^g\|}$, measures proximity to known class k. Consequently, the maximum similarity over all source classes provides an interpretable measure of the sample's affiliation with the known classes. A low maximum similarity indicates the sample may not belong to any known class.

The source model's output is often used to initialize the detection and classification of known-class samples and can provide informative indicators, denoted as κ, for identifying known-class versus unknown-class samples (Jang et al., 2022; Qu et al., 2023). Building on the discussion above, we define $\kappa_i = \max_k cos(z_i, w_k^g)$ and model its marginal distribution using a mixture of two Beta distributions, a choice that is both natural and effective in many prior UDA work targeting unknown classes.

$$p(\kappa) = \pi_{\text{known}} p(\kappa \mid y \in \mathcal{C}_s) + \pi_{\text{unkown}} p(\kappa \mid y \notin \mathcal{C}_s) \tag{1}$$

where π_{known} and π_{unknown} represent the probabilities $P(y \in \mathcal{C}_s)$ and $P(y \notin \mathcal{C}_s)$, respectively. Given this Beta mixture model, we can estimate the posterior probability $P(y \in \mathcal{C}_s \mid \kappa)$, which represents the confidence that a sample belongs to the source classes. The parameters of the Beta distributions, along with π_{known} and π_{unknown}, are estimated by fitting the Beta mixture model using the Expectation-Maximization (EM) algorithm (Arazo et al., 2019; Jang et al., 2022). Once the parameters are obtained, the posterior probability $P(y \in \mathcal{C}_s \mid \kappa)$ is given by: $P(y \in \mathcal{C}_s \mid \kappa) = \pi_{\text{known}} p(\kappa \mid y \in \mathcal{C}_s) / [\pi_{\text{known}} p(\kappa \mid y \in \mathcal{C}_s) + \pi_{\text{unkown}} p(\kappa \mid y \notin \mathcal{C}_s)]$. The decision boundary $P(y \in \mathcal{C}_s \mid \kappa) > P(y \notin \mathcal{C}_s \mid \kappa)$, which indicates that the posterior probability of belonging to the known classes exceeds that of the unknown classes, corresponds to $P(y \in \mathcal{C}_s \mid \kappa) > 0.5$. Consequently, we can identify high-confidence common-class samples by applying a threshold $P(y \in \mathcal{C}_s \mid \kappa) > \delta$ with $\delta = 0.5$ as the default. The threshold δ can be slightly adjusted around 0.5 using an auto-tuning strategy, and the details of this implementation will be discussed in Section 4.3. For samples that meet this criterion, we assign their predicted label to the class with the highest cosine similarity, computed as $\arg\max_k cos(z_i, w_k^g)$. We denote the identified *high-confidence samples* and their estimated labels as $\mathcal{X}_t^H$ and $\hat{\mathcal{Y}}_t^H$, respectively. Based on $\hat{\mathcal{Y}}_t^H$, we can derive an estimate of the common label set $\mathcal{C}$, denoted as $\widehat{\mathcal{C}}$. The remaining target samples, for which the source-class confidence is insufficient, are considered as *uncertain samples* and denoted by $\mathcal{X}_t^U$. We use $\mathcal{Z}_t^H$ and $\mathcal{Z}_t^U$ to denote the embeddings extracted by the encoder corresponding to $\mathcal{X}_t^H$ and $\mathcal{X}_t^U$, respectively.

3.2 Guided Bayesian Nonparametric Clustering

Uncertain samples may include both instances outside the common label set (i.e., from target-private label space) and those within it that received low-confidence predictions. High-confidence common-class samples can serve as anchors or references to guide the clustering of uncertain samples, making the problem more aligned with weakly-supervised clustering. Futhermore, clustering samples from unknown classes requires a method that does not rely on a predefined number of clusters, as this information is usually not available. To this end, we propose a guided Bayesian nonparametric clustering (GBNC) framework that automatically discovers previously unseen classes while simultaneously classifies samples into known classes.

Given the embeddings $\mathcal{Z}_t^U = \{z_1^U, \ldots, z_{n_U}^U\}$ of uncertain samples with size n_U, we consider leveraging Bayesian nonparametric clustering (BNC) technique to estimate the corresponding cluster labels $\mathcal{Y}_t^U = \{y_1^U, \ldots, y_{n_U}^U\}$, as it can both handle unknown class discovery and incorporate weak supervision effectively. Specifically, we build our model on the Dirichlet Process Mixture (DPM) framework, a widely used Bayesian nonparametric approach known for its simplicity of implementation and extensive literature support. The DPM-based formulation naturally accommodates a flexible and potentially unbounded number of clusters, allowing the model to adjust automatically to the number of clusters as the data grows. Our method can also be extended to other Bayesian nonparametric mixture models, such as Pitman–Yor process mixtures and Normalized Generalized Gamma process mixtures, which, like the DPM, allow the number of clusters to emerge from the data. The Dirichlet Process (DP) is a stochastic process that defines a distribution over distributions, meaning that each draw from it is itself a probability distribution. A key property of the DP is its almost surely discrete realizations, which naturally induces clustering by allowing parameter sharing among individual data points and eliminates the need to specify the number of clusters in advance. A DPM constructed from the observed data $\mathcal{Z}_t^U$ can be formulated within a Bayesian hierarchical framework,

$$z_i^U|\eta_i \sim p(z_i^U|\eta_i), \quad \eta_i|G \sim G, \quad G \sim DP(\alpha, G_0). \tag{2}$$

$p(\cdot \mid \eta_i)$ denotes the likelihood function, following a probability distribution parameterized by η_i, where η_i is drawn from a DP-distributed prior G. The DP prior is characterized by the concentration parameter α and the base distribution G_0, which specifies a prior probability distribution over the parameters. The base distribution G_0 is typically chosen to be conjugate to the likelihood to make the inference tractable and thus simplify computation. Due to the discrete nature of G, multiple η_i associated with different observations may take the same value, naturally forming clusters. Cluster assignments arise from these shared values, meaning that $y_i^U = y_j^U$ when $\eta_i = \eta_j$. The concentration parameter α is directly related to the prior expected number of clusters, which can be set in a noninformative manner, as discussed in our Section 4.2.

We denote the labels generated in Section 3.1 for the high-confidence samples as $\hat{\mathcal{Y}}_t^H = \{\hat{y}_1^H, \dots, \hat{y}_{n_H}^H\}$, where n_H is their number. These labels are assumed to be reliably assigned to known classes and are therefore treated as fixed. Accordingly, for the embeddings $\mathcal{Z}_t^H = \{z_1^H, \dots, z_{n_H}^H\}$, we can directly model their distribution given these fixed labels as $z_i^H \sim p(z_i^H|\eta^\star_{\hat{y}_i^H})$, where $\eta^\star_{\hat{y}_i^H}$ denotes the parameters associated with the cluster corresponding to the predicted label $\hat{y}_i^H$. Let $|\widehat{\mathcal{C}}|$ denote the number of estimated common classes, i.e., the number of unique classes in $\hat{\mathcal{Y}}_t^H$. To address the potential mismatch in cluster indices between $\hat{\mathcal{Y}}_t^H$ and $\mathcal{Y}_t^U$, we assume that all $\hat{y}_i^H$ take values from the index set $\{1, 2, \dots, |\widehat{\mathcal{C}}|\}$. In practice, we include an additional step to align and remap the estimated labels $\hat{\mathcal{Y}}_t^H$ to this index range.

By marginalizing out G in Equation 2, the hierarchical model can be reformulated as an infinite mixture model, as in Equation A2, also known as the stick-breaking representation (with more details in Appendix A). We then integrate this representation with the distribution of z_i^H to yield:

$$\begin{aligned} &z_i^U|y_i^U = k \sim p(z_i^U|\eta_k^\star), \quad z_i^H \sim p(z_i^H|\eta^\star_{\hat{y}_i^H}), \quad \eta_k^\star \sim G_0 \\ &y_i^U|\pi \sim Cat(\boldsymbol{\pi}), \quad \boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_{|\widehat{\mathcal{C}}|}, \cdots), \quad \pi_k = \upsilon_k \prod_{j=1}^{k-1}(1-\upsilon_j), \quad \upsilon_k \sim Beta(1, \alpha) \end{aligned} \tag{3}$$

where $\boldsymbol{\pi}$ is a probability vector over clusters for $k = 1, 2, \dots$. The categorical distribution $\mathrm{Cat}(\boldsymbol{\pi})$ assigns cluster memberships to the uncertain samples with probabilities $P(y_i^U = k) = \pi_k$. $Beta(\cdot)$ denotes the beta distribution.

We model the mixture components using a multivariate Gaussian distribution, $p(\cdot|\eta_k^\star) = N(\cdot|\mu_k, \Lambda_k)$, where Λ_k is the inverse covariance matrix (i.e., precision matrix). We adopt a conjugate Normal-Wishart prior, $G_0 = N(\mu_k|\zeta, \tau\Lambda_k) \times W(\Lambda_k|b, \Omega)$, where τ is a scaling factor, b is the degrees of freedom, and Ω is the scale matrix. The overall model is therefore parameterized by $\{\upsilon_k, \mu_k, \Lambda_k\}_{k=1}^\infty$ and latent cluster labels $\{y_i^U\}_{i=1}^{n_U}$. The constructive stick-breaking representation allows us to derive a tractable joint distribution for posterior estimation. For computational efficiency, we choose Variational Bayes (VB) to approximate the posterior of these parameters. The joint probability distribution of all random variables can be factorized as follows, where boldface symbols denote vectorized forms of the corresponding variables in Equation 4.

$$p(\mathbf{Z}^U, \mathbf{Z}^H, \boldsymbol{Y}^U, \boldsymbol{\upsilon}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = p(\mathbf{Z}^U \mid \mathbf{Y}^U, \boldsymbol{\mu}, \boldsymbol{\Lambda}) p(\mathbf{Z}^H \mid \boldsymbol{\mu}, \boldsymbol{\Lambda}) p(\mathbf{Y}^U \mid \boldsymbol{\upsilon}) p(\boldsymbol{\upsilon}) p(\boldsymbol{\mu} \mid \boldsymbol{\Lambda}) p(\boldsymbol{\Lambda}) \tag{4}$$

The goal of VB is to find a variational distribution that minimizes the Kullback–Leibler (KL) divergence to the true posterior. We assume a fully factorized variational family, as in Equation 5, which allows us to apply mean-field approximation to yield tractable updates in our VB implementation (Bishop & Nasrabadi, 2006).

$$q(\{\upsilon_k\}_{k=1}^{K^*-1}, \{\mu_k\}_{k=1}^{K^*}, \{\Lambda_k\}_{k=1}^{K^*}, \{y_i^U\}_{i=1}^{n_U}) = \prod_{k=1}^{K^*-1} q(\upsilon_k) \prod_{k=1}^{K^*} q(\mu_k, \Lambda_k) \prod_{i=1}^{n_U} q(y_i^U), \tag{5}$$

where $q(\upsilon_k)$, $q(\mu_k, \Lambda_k)$, and $q(y_i^U)$ denote the variational distributions for the corresponding variables and form the factors of the factorized variational distribution. This structure enables efficient optimization via coordinate ascent. Following the standard approach of applying a truncated stick-breaking representation in VB for DPMs (Blei & Jordan, 2006), we truncate the stick-breaking process at level K^* such that $q(\upsilon_{K^*} = 1) = 1$. The value K^* serves as a practical upper bound, fundamentally different from the number of clusters K in tuning-based methods which is repeatedly adjusted, and consistent results can be expected with any sufficiently large K^* (e.g., 100). In our model, the variational distributions are specified as: $q(\upsilon_k) = Beta(\upsilon_k|\gamma_{k1}, \gamma_{k2})$, $q(\mu_k, \Lambda_k) = N(\mu_k|m_k, \tau_k\Lambda_k) \times W(\Lambda_k|c_k, D_k)$, and $q(y_i^U) = Cat(\phi_i)$, where $\phi_i = (\phi_{i1}, \phi_{i2}, \dots)$. The variational parameters $\{m_k, \tau_k, c_k, D_k, \gamma_{k1}, \gamma_{k2}\}_k$ and $\{\phi_i\}_i$ are updated via the VB algorithm. Both the full conditional and variational distributions belong to the exponential family, allowing the variational distributions for the parameters to be derived from their full conditionals (Blei & Jordan, 2006). Building on this, we derive our VB algorithm, with detailed derivations, update steps, and further implementation details provided in Appendix B. Finally, with the estimated ϕ_i, each uncertain sample is assigned a predicted label $\hat{y}_i^U = \arg\max_k \phi_{ik}$, forming $\hat{\mathcal{Y}}_t^U = \{\hat{y}_1^U, \dots, \hat{y}_{n_U}^U\}$.

3.3 Optimization and Inference

Prototype-based Regularization: To enhance target sample separation, we alternate clustering updates with end-to-end training using pseudo-cluster labels generated in Sections 3.1 and 3.2, optimized under a clustering-based objective. Inspired by prior work on non–source-free settings (Li et al., 2021), we incorporate a prototype-based regularizer that promotes intra-cluster compactness and inter-cluster separation. Specifically, we first compute a centroid c_k for each cluster k, used as its prototype, based on all target samples and their predicted labels $\hat{\mathcal{Y}}_t = [\hat{\mathcal{Y}}_t^H, \hat{\mathcal{Y}}_t^U]$, where $\hat{\mathcal{Y}}_t^H$ and $\hat{\mathcal{Y}}_t^U$ are obtained from Sections 3.1 and 3.2, respectively. Let $\mathbf{c} = [c_1, c_2, \dots, c_K]$ represent the estimated prototypes, computed as the L2-normalized centroids of the obtained target clusters. These prototypes are dynamically updated during each training iteration, after each update of $\hat{\mathcal{Y}}_t = \{\hat{y}_1^t, \hat{y}_2^t, \cdots\}$. Before applying the regularizer, we convert each predicted label $\hat{y}_i^t$ into its one-hot representation, where $\hat{y}_{i,k}^t = 1$ if $\hat{y}_i^t = k$. The prototypical regularizer for a minibatch B is then defined as:

$$\mathcal{L}_{\text{reg}} = -\sum_{i \in B} \sum_{k=1}^{K} \hat{y}_{i,k}^t \log \hat{p}_{(i,k)}^t \tag{6}$$

where, $\hat{p}_{(i,k)}^t = \frac{\exp\left((\tilde{z}_i^t)^\top c_k/0.1\right)}{\sum_{k=1}^{K} \exp\left((\tilde{z}_i^t)^\top c_k/0.1\right)}$, and $\tilde{z}_i^t$ denotes the L2-normalized feature z_i of target sample i. To gradually regulate the influence of $\mathcal{L}_{\text{reg}}$ and avoid premature over-expansion, we adopt the ramp-up schedule from (Li et al., 2021), using a dynamic weight $\lambda_{\text{reg}} = e^{-3\cdot(1-\frac{l}{L})}$, where l is the current step and L the total training steps, yielding the term $\lambda_{\text{reg}}\mathcal{L}_{\text{reg}}$ in the training objective.

Local Prediction Consistency Regularization: To further refine the source classifier g and align similar target samples, we introduce a local consistency loss $\mathcal{L}_{\text{loc}}$, which enforces prediction consistency within local neighborhoods.

$$\mathcal{L}_{\text{loc}} = -\sum_{i \in B} \sum_{j \in \mathcal{N}_i} \log(g_i^t)^\top g_j^t \tag{7}$$

g_i^t denotes the softmax prediction of a target sample from the source classifier g, representing a probability distribution over source classes. The set $\mathcal{N}_i$ refers to the local neighborhood of target sample i, defined as its top K_{nn} nearest neighbors based on cosine similarity in the feature space. To implement the nearest-neighbor computations during minibatch training, we follow prior work (Yang et al., 2021a; 2022; 2021b) and maintain two memory banks: one storing all target features and the other storing their corresponding softmax predictions, both updated incrementally using minibatch data. During training, g_j^t is retrieved from the memory bank and treated as a fixed reference, so gradients are only backpropagated through g_i^t. Similar to methods that align neighboring samples by maximizing the dot product of their predictions (e.g., Yang et al. (2022)), this regularizer maximizes the dot product between a query sample's log prediction and the predictions of its neighbors. This encourages consistency among nearby samples while reducing the tendency toward overly peaked (i.e., one-hot-like) softmax outputs over the source classes, so that high-entropy (less confident) predictions are aligned more softly. Since gradients are only propagated through g_i^t, our objective also resembles $\text{KL}(g_j^t \| g_i^t)$, minimizing the KL divergence between a fixed neighbor distribution and the query sample.

Training Objective and Inference: The final training objective integrates two components: the prototypical regularizer $\mathcal{L}_{\text{reg}}$ and the local prediction consistency regularizer $\mathcal{L}_{\text{loc}}$, balanced by their respective weighting coefficients λ_{reg} and λ_{loc}. The overall loss function is formulated as: $\mathcal{L} = \lambda_{\text{reg}}\mathcal{L}_{\text{reg}} + \lambda_{\text{loc}}\mathcal{L}_{\text{loc}}$. With this overall objective, the workflow of ABC, comprising the iterative steps detailed in Sections 3.1, 3.2, and 3.3, is outlined in Algorithm 1. During inference, with the final set of prototypes $[c_1, c_2, \ldots, c_K]$, each target sample is assigned the label of the most similar prototype based on cosine similarity between its L2-normalized feature and the normalized prototypes, according to $\hat{y}_i^t = \arg\max_k \langle \tilde{z}_i^t, c_k \rangle$.

Algorithm 1 Adapt via Bayesian Nonparametric Clustering

```
 1: Input: Unlabeled target samples $\mathcal{X}_t$; pretrained classification model $\phi = g \circ f$
 2: Output: Predicted labels $\hat{\mathcal{Y}}_t$; prototypes $\{c_1, c_2, \ldots, c_K\}$
 3: for $epoch = 1, 2, \ldots$ do
 4:     a. Identify high-confidence samples $\mathcal{X}_t^H$ with pseudo-labels $\hat{\mathcal{Y}}_t^H$
 5:     b. Identify uncertain samples $\mathcal{X}_t^U$
 6:     c. Estimate labels $\hat{\mathcal{Y}}_t^U$ for $\mathcal{X}_t^U$ via guided Bayesian nonparametric clustering
 7:     d. Generate prototypes $\mathbf{c} = [c_1, \ldots, c_K]$
 8:     for $iteration = 1, 2, \ldots$ do
 9:         Backpropagate $\phi$ w.r.t. the loss $\mathcal{L}$ and update $\mathbf{c}$
10:     end for
11: end for
```

4 Experiments

4.1 Experiment Setup

Datasets: We evaluate on three widely-used benchmark datasets: 1. *Office-31* (Saenko et al., 2010), comprises more than 4k images from 31 categories across three domains: *Amazon* (**A**), *DSLR* (**D**), and *Webcam* (**W**). 2. *Office-Home* (Venkateswara et al., 2017), includes about 15,500 images categorized into 65 classes across four domains: *Artistic* (**Ar**), *Clip Art* (**Cl**), *Product* (**Pr**), and *Real-World* (**Rw**); 3. VisDA-C (Peng et al., 2017) is a challenging synthetic-to-real benchmark with 12 object categories, featuring around 152k synthetic images in the source domain and 55k real-world images in the target domain. In the main experiments, we focus on the setting where neither $\mathcal{C}_s$ nor $\mathcal{C}_t$ is a subset of the other ($\mathcal{C}_s \nsubseteq \mathcal{C}_t$ and $\mathcal{C}_s \nsupseteq \mathcal{C}_t$), which is known as open-partial domain adaptation (OPDA) (Qu et al., 2023). We follow the class split settings used in prior OPDA works (Qu et al., 2023; Liang et al., 2020), using 10/10/11

Table 1: H-score (%) on Office-Home, Office-31, and VisDA under OPDA settings. 'SF' indicates source-free methods, and 'FG' refers to fine-grained discovery of unknown classes. **Bold blue numbers** highlight top non-SF methods, and **bold red numbers** highlight leading SF methods (excluding the GBNC-only module).

(a) Office-Home

Methods	SF	FG	A2C	A2P	A2R	C2A	C2P	C2R	P2A	P2C	P2R	R2A	R2C	R2P	Avg
UAN (You et al., 2019)	✗	✗	51.6	51.7	54.3	61.7	57.6	61.9	50.4	47.6	61.5	62.9	52.6	65.2	56.6
CMU (Fu et al., 2020)	✗	✗	56.0	56.9	59.2	67.0	64.3	67.8	54.7	51.1	66.4	68.2	57.9	69.7	61.6
DCC (Li et al., 2021)	✗	✗	58.0	54.1	58.0	**74.6**	70.6	77.5	64.3	**73.6**	74.9	**81.0**	**75.1**	80.4	70.2
OVANet (Saito & Saenko, 2021)	✗	✗	62.8	75.6	78.6	70.7	68.8	75.0	71.3	58.6	80.5	76.1	64.1	78.9	71.8
GATE (Chen et al., 2022)	✗	✗	**63.8**	**75.9**	**81.4**	74.0	**72.1**	**79.8**	**74.7**	70.3	**82.7**	79.1	71.5	**81.7**	**75.6**
Best Epoch (Best H-score)															
SHOT-O (Liang et al., 2020)	✓	✗	43.2	55.9	57.2	57.5	52.8	60.4	55.5	45.1	62.6	57.8	43.9	58.2	54.1
GLC (Qu et al., 2023)	✓	✗	56.8	78.5	**89.7**	64.4	73.2	**89.0**	66.2	53.6	**88.1**	**79.5**	52.6	**84.3**	**72.9**
ABC	✓	✓	**57.7**	**80.2**	84.7	**71.5**	**81.2**	82.7	**70.0**	**55.4**	83.9	70.2	**57.2**	79.8	**72.9**
Final Epoch															
SHOT-O (Liang et al., 2020)	✓	✗	30.7	33.0	38.3	49.8	29.7	37.5	50.1	35.9	43.9	53.7	32.5	38.8	39.4
GLC (Qu et al., 2023)	✓	✗	**56.4**	**78.1**	**88.8**	16.2	73.1	**88.3**	65.6	13.8	**87.2**	**76.9**	15.1	**83.5**	61.9
GBNC-only	-	-	53.7	74.9	82.7	56.9	68.7	73.2	62.6	51.9	80.1	61.1	54.6	77.4	66.5
ABC	✓	✓	54.6	75.4	82.1	**68.2**	**73.8**	81.0	**69.3**	**53.8**	83.8	68.9	**55.9**	72.3	**69.9**

(b) Office-31 and VisDA

Methods	SF	FG	Office-31							VisDA
			A2D	A2W	D2A	D2W	W2A	W2D	Avg	S2R
UAN (You et al., 2019)	✗	✗	59.7	58.6	60.1	70.6	60.3	71.4	63.5	34.8
CMU (Fu et al., 2020)	✗	✗	68.1	67.3	71.4	79.3	72.2	80.4	73.1	32.9
DCC (Li et al., 2021)	✗	✗	**88.5**	78.5	70.2	79.3	75.9	88.6	80.2	43.0
OVANet (Saito & Saenko, 2021)	✗	✗	85.8	79.4	80.1	**95.4**	**84.0**	**94.3**	86.5	53.1
GATE (Chen et al., 2022)	✗	✗	87.7	**81.6**	**84.2**	94.8	83.4	94.1	**87.6**	**56.4**
Best Epoch (Best H-score)										
SHOT-O (Liang et al., 2020)	✓	✗	62.1	58.8	65.5	80.9	59.8	72.7	66.6	32.3
GLC (Qu et al., 2023)	✓	✗	81.2	83.7	87.5	90.2	85.7	90.2	86.4	**63.9**
ABC	✓	✓	**86.4**	**96.0**	**88.2**	**95.3**	**89.5**	**93.4**	**91.5**	61.6
Final Epoch										
SHOT-O (Liang et al., 2020)	✓	✗	55.5	55.8	59.6	75.9	54.6	68.5	61.6	17.8
GLC (Qu et al., 2023)	✓	✗	69.8	81.5	80.7	88.4	75.5	84.6	80.0	**62.3**
GBNC-only	-	-	81.2	91.4	84.9	85.7	79.4	93.0	85.9	35.0
ABC	✓	✓	**84.3**	**96.0**	**86.3**	**92.2**	**85.0**	**86.6**	**88.4**	59.5

shared/source-private/target-private classes for Office-31, 10/5/50 for Office-Home, and 6/3/3 for VisDA-C. We also discuss the case where $\mathcal{C}_s \subseteq \mathcal{C}_t$, known as open-set domain adaptation (OSDA) (Qu et al., 2023), in Appendix D.2.

Evaluation Protocol: We follow the standard evaluation protocol used in prior works (Chen et al., 2022; Liang et al., 2021; Qu et al., 2023), reporting the harmonic mean (H-score) between the average accuracy on known classes and the accuracy on unknown-class samples, along with the average per-class accuracy (mean accuracy), treating the unknown class as a single category. Since these widely used metrics treat all unknown ground-truth classes as a single category, we additionally report Normalized Mutual Information (NMI; described in Appendix C) to evaluate fine-grained classification. NMI measures the alignment between predicted and ground-truth labels across individual classes, independent of class label permutations, thereby treating each unknown class separately. We compare our method against representative non-SFDA baselines in terms of H-score, with their results reprinted from previous work (Chen et al., 2022; Qu et al., 2023). Additionally, we evaluate against two recent SFDA methods: SHOT (Liang et al., 2020) and GLC (Qu et al., 2023), with results reproduced using their publicly available codebases. As both methods predict all unknown samples as a single "unknown" class and do not directly produce fine-grained classifications, we apply the *DPM*, using the same configuration as in our model, to the unknown target samples predicted by SHOT and GLC, and then combine the resulting cluster assignments with their known-class predictions to compute NMI against the ground-truth labels.

Implementation Details: For a fair comparison, we adopt the same backbone, ResNet-50 pretrained (He et al., 2016) on ImageNet (Deng et al., 2009), in line with existing benchmark methods SHOT (Liang et al.,

Table 2: NMI (%) on Office-Home, Office-31, and VisDA datasets under OPDA settings.

(a) Office-Home

Methods	A2C	A2P	A2R	C2A	C2P	C2R	P2A	P2C	P2R	R2A	R2C	R2P	Avg
Best Epoch													
SHOT-O (Liang et al., 2020) + DPM	**49.1**	66.4	63.8	56.2	64.0	63.0	56.7	47.6	63.0	58.3	**49.9**	68.0	58.7
GLC (Qu et al., 2023) + DPM	**49.1**	**75.3**	73.7	57.4	**75.9**	74.0	60.2	47.4	73.8	63.2	49.3	74.6	64.5
ABC	48.0	74.1	**76.0**	**61.3**	75.0	**74.1**	**61.1**	**48.4**	**74.2**	**63.3**	48.7	**74.7**	**64.9**
Final Epoch													
SHOT-O (Liang et al., 2020) + DPM	47.4	62.3	60.6	56.8	61.3	60.6	57.3	46.7	59.8	56.6	**47.5**	62.5	56.6
GLC (Qu et al., 2023) + DPM	**49.1**	**75.6**	**74.0**	48.9	**75.8**	**74.1**	61.1	35.2	72.2	**64.2**	40.8	**74.2**	62.1
GBNC-only	49.8	73.3	74.3	49.5	67.6	64.4	55.5	46.4	70.2	57.7	49.0	73.9	61.0
ABC	48.1	73.9	**74.0**	**60.8**	73.6	72.5	**61.8**	**47.4**	**73.3**	62.2	47.2	74.1	**64.1**

(b) Office-31 and VisDA

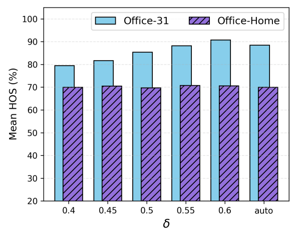
Methods	Office-31							VisDA
	A2D	A2W	D2A	D2W	W2A	W2D	Avg	S2R
Best Epoch								
SHOT-O (Liang et al., 2020) + DPM	77.6	77.3	68.2	82.1	67.6	82.8	75.9	39.9
GLC (Qu et al., 2023) + DPM	79.4	82.5	68.0	82.1	66.7	81.8	76.7	**46.8**
ABC	**80.7**	**84.8**	**69.0**	**85.8**	**69.1**	**83.8**	**78.9**	**46.8**
Final Epoch								
SHOT-O (Liang et al., 2020) + DPM	79.5	79.2	70.7	82.5	68.9	83.6	77.4	44.7
GLC (Qu et al., 2023) + DPM	**81.6**	83.0	69.3	84.0	**70.4**	**82.3**	78.4	**48.0**
GBNC-only	79.0	84.0	69.2	83.1	71.0	83.3	78.3	32.4
ABC	81.2	**84.8**	**70.9**	**85.1**	70.0	81.5	**78.9**	46.9

2020) and GLC (Qu et al., 2023). We pretrain the source model using the code provided by GLC (Qu et al., 2023), following the source model pretraining procedure recommended in SHOT (Liang et al., 2020) to apply a 90/10 split for training and validation. Instead of a random split, we employ stratified sampling to maintain the class distribution across both training and validation sets, reducing the impact of class imbalance during source pretraining. Due to space constraints, implementation details and configuration settings of our method are provided in Appendix C.

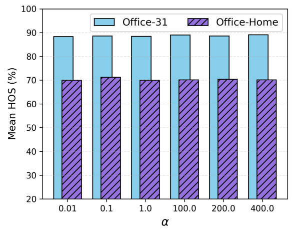
4.2 Experiment Results

H-score, mean accuracy, and NMI results for all three datasets are reported in Tables 1a, 1b, A1, A2, 2a, and 2b, respectively. Due to the absence of standardized epoch selection criteria in SFDA methods, results are reported for both the **Best Epoch** (the highest H-score) and the **Final Epoch** (the last training epoch) for the two SFDA baselines and ours. As shown in Tables 1a,1b,A1, and A2, ABC outperforms GLC and SHOT in terms of average H-score and mean accuracy on both Office-31 and Office-Home. Notably, although GLC has been among the strongest baselines, in the Final Epoch setting ABC surpasses it by a substantial margin, achieving at least an 8% improvement in average H-score across both datasets and over a 12% gain in average mean accuracy on Office-31. Furthermore, ABC delivers performance comparable to non-SFDA methods on Office-Home, and achieves higher average H-score on Office-31, regardless of whether results are taken from the best or final epoch. This demonstrates that ABC can achieve strong adaptation performance even without access to source data. On VisDA, while GLC slightly outperforms ABC, the gap is small, and ABC remains among the top-performing methods; both ABC and GLC significantly outperform SHOT, and surpass all compared non-SFDA methods.

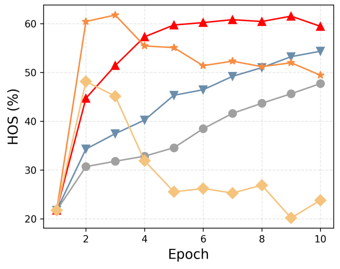
From Tables 2a and 2b, which report NMI between true and predicted labels, including unknown classes, we observe that ABC achieves the highest average NMI on Office-31 and Office-Home, while showing slightly lower average NMI on VisDA compared to GLC+DPM. Note that the compared methods group all unknown classes into a single category and do not inherently support fine-grained (FG) classification by explicitly predicting labels for individual unknown classes. The NMI results of these methods are obtained by applying the DPM module, which is central to our approach, to their predicted unknown samples. Compared to appending a



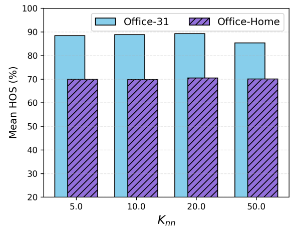
(a) Sensitivity to δ



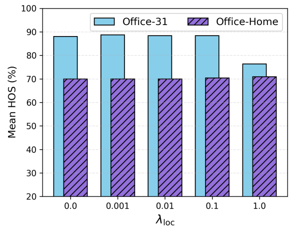
(b) Sensitivity to α



(a) HOS(%) vs. Epoch

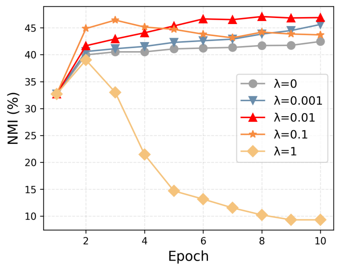


(c) Sensitivity to K_{nn}



(d) Sensitivity to λ_{loc}

Figure 3: Hyperparameter sensitivity (H-score) on Office-31 and Office-Home under OPDA.



(b) NMI(%) vs. Epoch

Figure 4: Performance curve on VisDA for different λ_{loc}.

separate BNC step to existing methods, our approach integrates it into a unified framework, enabling joint optimization, minimizing error propagation between clustering and adapting the source model, and eliminating the need for post hoc processing. We also evaluate the GBNC component in isolation (GBNC-only), which performs competitively on Office-31 and Office-Home but significantly worse on VisDA, demonstrating both its effectiveness and the added value of the full learning framework. While our experimental validation primarily focuses on the OPDA setting, we also provide an additional evaluation under the Open-Set Domain Adaptation (OSDA) setting in the Appendix, where the source-private class set is empty (i.e., $\mathcal{C}_s \subseteq \mathcal{C}_t$). As shown in Appendix D.2, ABC also performs well in this setting, suggesting that it can generalize effectively to other scenarios involving unknown classes.

4.3 Experiment Analysis

To better understand the robustness and reliability of our method, we conduct comprehensive sensitivity and ablation analyses on key hyperparameters. Average results across tasks are reported in the main paper; per-task results are in Appendix D.3. The results demonstrate that our method achieves robust performance across a wide range of hyperparameter values.

δ for High-Confidence Sample Selection: To select high-confidence samples, we apply $P(y \in \mathcal{C}_s \mid \kappa) > \delta$. Setting δ too high results in too few selected samples, potentially omitting entire common classes, while a low δ may include too many, increasing the risk of misclassified unknown-class samples. The natural decision boundary, where $P(y \in \mathcal{C}_s \mid \kappa) > P(y \notin \mathcal{C}_s \mid \kappa)$, suggests a default threshold of $\delta = 0.5$, but it need not always be fixed at this value. For Office-31 and Office-Home, we tune δ per epoch within the range $[0.4, 0.6]$ (with a step size of 0.05), selecting the value that maximizes the Silhouette score (Rousseeuw, 1987) based on the embedded data and the estimated labels from the GBNC. To assess sensitivity, we also rerun our algorithm using fixed values $\delta \in \{0.4, 0.45, 0.5, 0.55, 0.6\}$ and compare the results to our tuning strategy (denoted as "auto"). As shown in Figures 3a, A4, A5, and A6, performance remains relatively stable around $\delta = 0.5$. The auto-tuning strategy yields similar results to fixed settings, further mitigating sensitivity to the choice of δ without requiring manual selection.

DPM Hyperparameter α: The concentration parameter α plays a key role in the DPM model, controlling its ability to infer the number of clusters. Following a common practice, we set $\alpha = \frac{\alpha^*}{K^*}$ with $\alpha^* = 1$ as a noninformative prior, allowing the influence of the hyperparameters to diminish with increasing sample size. To assess sensitivity, we further report the H-score on Office-31 and Office-Home using $\alpha^* \in \{0.01, 0.1, 1, 100, 200, 400\}$ (i.e., $\alpha \in \{10^{-4}, 10^{-3}, 10^{-2}, 1, 2, 4\}$ for $K^* = 100$), as shown in Figures 3b, with additional metrics provided in Figures A4, A5, and A6. Results demonstrate robust performance, supporting the use of a non-informative prior. While α can also be estimated via empirical Bayes or assigned a hyperprior, where the hyperparameters are either inferred from the data or exert less influence higher in the model hierarchy, the performance stability across values suggests that our choice achieves reliable performance while providing both efficiency and simplicity, making the modeling process more controllable. In general, we prefer non-informative priors in our model to let the data drive the clustering process while keeping the implementation simple and efficient.

K_{nn} for the Calculation of $\mathcal{L}_{\text{loc}}$: For Office-31 and Office-Home, we evaluate performance with different number of nearest neighbors $K_{nn} \in \{5, 10, 20, 50\}$, reporting H-scores in Figures 3c and additional metrics in Figures A4, A5, and A6. Our method demonstrates robustness to the choice of K_{nn}, with H-scores remaining stable across tested values. Given the size of these two datasets, this range of K_{nn} is appropriate: too few neighbors (e.g., 1 or 2) can lead to unstable estimates, especially when those neighbors are misclassified, while too many may introduce noise by incorporating samples from different classes, as noted in similar settings (Yang et al., 2022). For larger datasets like VisDA, we adopt a higher K_{nn} to account for its much larger target domain (over 50k samples).

λ_{loc} for Local Prediction Consistency Regularizer: We evaluate the impact of $\lambda_{\text{loc}} \in \{0, 0.001, 0.01, 0.1, 1\}$. H-scores on Office-31 and Office-Home are shown in Figures 3d, with NMI and mean accuracy provided in Figures A4, A5, and A6. Results show that performance remains generally stable, indicating robustness to λ_{loc}, though larger values may degrade performance on Office-31. For VisDA, we analyze performance across training epochs (Figure 4). λ_{loc} shows a stronger impact: small λ_{loc} values performs better compared to $\lambda_{\text{loc}} = 0$, and it shows more stable, gradual convergence while larger values lead to faster convergence but compromise final performance. Across small λ_{loc} values, metrics steadily improve over time, indicating stable and effective learning. This suggests that $\mathcal{L}_{\text{loc}}$ can be particularly helpful when baseline adaptation performance is weak, typically in challenging tasks, as in VisDA.

5 Conclusion

In this paper, we propose Adapt via Bayesian Nonparametric Clustering (ABC) for SFDA with unknown target classes, enabling both discovery and fine-grained classification of unknown classes. Through the development of a guided Bayesian nonparametric clustering algorithm, our approach first identifies high-confidence target samples from known classes and then uses them as anchors to effectively cluster known-class samples and discover previously unseen ones, without requiring prior knowledge of the number of unknown classes or specifying it in advance. A prototype-based regularizer and a local prediction-consistency term are incorporated to further refine the source model for adaptation to the target data. Experiments demonstrate that ABC achieves solid performance across various benchmarks and evaluation metrics. By enabling the identification and classification of unknown classes in new domains without access to source data, ABC has the potential to provide important practical values across a range of real-world applications.

References

Charles E Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, pp. 1152–1174, 1974.

Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *International conference on machine learning*, pp. 312–321. PMLR, 2019.

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

David M Blei and Michael I Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 2006.

Silvia Bucci, Mohammad Reza Loghmani, and Tatiana Tommasi. On the effectiveness of image rotation for open set domain adaptation, 2020. URL `https://arxiv.org/abs/2007.12360`.

Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Partial transfer learning with selective adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2724–2732, 2018a.

Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 135–150, 2018b.

Chao Chen, Zhihong Chen, Boyuan Jiang, and Xinyu Jin. Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation. In *AAAI Conference on Artificial Intelligence*, 2019.

Liang Chen, Yihang Lou, Jianzhong He, Tao Bai, and Minghua Deng. Geometric anchor correspondence mining with uncertainty modeling for universal domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16134–16143, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE CVPR*, pp. 248–255. IEEE, 2009.

Ning Ding, Yixing Xu, Yehui Tang, Chao Xu, Yunhe Wang, and Dacheng Tao. Source-free domain adaptation via distribution estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7212–7222, 2022.

Thomas S Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2): 209–230, 1973.

Bo Fu, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Learning to detect open classes for universal domain adaptation. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV*, pp. 567–583, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58554-9. doi: 10.1007/978-3-030-58555-6_34. URL `https://doi.org/10.1007/978-3-030-58555-6_34`.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pp. 1180–1189, 2015.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

André R Gonçalves, Puja Das, Soumyadeep Chatterjee, Vidyashankar Sivakumar, Fernando J Von Zuben, and Arindam Banerjee. Multi-task sparse structure learning. In *CIKM*, pp. 451–460. ACM, 2014.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pp. 2672–2680, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Nils Lid Hjort, Chris Holmes, Peter Müller, and Stephen G Walker. *Bayesian Nonparametrics.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, 2010. doi: 10.1017/CBO9780511802478.

AK JAIN, MN MURTY, and PJ FLYNN. Data clustering: A review. *ACM Computing Surveys*, 31(3), 1999.

JoonHo Jang, Byeonghu Na, Dong Hyeok Shin, Mingi Ji, Kyungwoo Song, and Il-Chul Moon. Unknown-aware domain adversarial learning for open-set domain adaptation. *Advances in Neural Information Processing Systems*, 35:16755–16767, 2022.

Jogendra Nath Kundu, Naveen Venkat, R Venkatesh Babu, et al. Universal source-free domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4544–4553, 2020.

Guangrui Li, Guoliang Kang, Yi Zhu, Yunchao Wei, and Yi Yang. Domain consensus clustering for universal domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9757–9766, 2021.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International conference on machine learning*, pp. 6028–6039. PMLR, 2020.

Jian Liang, Dapeng Hu, Jiashi Feng, and Ran He. Umad: Universal model adaptation under domain and category shift. *arXiv preprint arXiv:2112.08553*, 2021.

Antonio Lijoi, Ramses H Mena, and Igor Prunster. Controlling the reinforcement in Bayesian non-parametric mixture models. *Journal of the Royal Statistical Society: Series B*, 69(4):715–740, 2007.

Zhengfa Liu, Guang Chen, Zhijun Li, Yu Kang, Sanqing Qu, and Changjun Jiang. Psdc: A prototype-based shared-dummy classifier model for open-set domain adaptation. *IEEE Transactions on Cybernetics*, 53(11): 7353–7366, 2022.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. In *ICML*, pp. 97–105. JMLR. org, 2015.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *NeurIPS*, pp. 136–144, 2016.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, pp. 2208–2217. JMLR. org, 2017.

Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

Yang Ni, Yuan Ji, and Peter Müller. Consensus monte carlo for random subsets using shared anchors. *Journal of Computational and Graphical Statistics*, pp. 1–12, 2020a.

Yang Ni, David Jones, and Zeya Wang. Consensus variational and Monte Carlo algorithms for Bayesian nonparametric clustering. In *2020 IEEE International Conference on Big Data (Big Data)*, pp. 204–209. IEEE, 2020b.

Yang Ni, Peter Müller, Maurice Diesendruck, Sinead Williamson, Yitan Zhu, and Yuan Ji. Scalable Bayesian nonparametric clustering and classification. *Journal of Computational and Graphical Statistics*, 29(1): 53–65, 2020c.

Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In *Proceedings of the IEEE international conference on computer vision*, pp. 754–763, 2017.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Vincent Dubourg, Joris Van Meerbergen, Rémi Weiss, and et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL `http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf`.

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.

Jim Pitman and Marc Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, pp. 855–900, 1997.

Sanqing Qu, Guang Chen, Jing Zhang, Zhijun Li, Wei He, and Dacheng Tao. Bmd: A general class-balanced multicentric dynamic prototype strategy for source-free domain adaptation. In *European conference on computer vision*, pp. 165–182. Springer, 2022.

Sanqing Qu, Tianpei Zou, Florian Röhrbein, Cewu Lu, Guang Chen, Dacheng Tao, and Changjun Jiang. Upcycling models under domain and category shift. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20019–20028, 2023.

Douglas Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, pp. 827–832, 2015.

Meitar Ronen, Shahaf E Finder, and Oren Freifeld. Deepdpm: Deep clustering with an unknown number of clusters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9861–9870, 2022.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV*, pp. 213–226. Springer, 2010.

Kuniaki Saito and Kate Saenko. Ovanet: One-vs-all network for universal domain adaptation. In *Proceedings of the ieee/cvf international conference on computer vision*, pp. 9000–9009, 2021.

Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 153–168, 2018.

Kuniaki Saito, Donghyun Kim, Stan Sclaroff, and Kate Saenko. Universal domain adaptation through self supervision. *Advances in neural information processing systems*, 33:16282–16292, 2020.

Jayaram Sethuraman. A constructive definition of dirichlet priors. *Statistica sinica*, pp. 639–650, 1994.

Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *AAAI Conference on Artificial Intelligence*, 2016.

Robert L Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953.

A Torralba and AA Efros. Unbiased look at dataset bias. In *IEEE CVPR*, pp. 1521–1528. IEEE Computer Society, 2011.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *IEEE CVPR*, volume 1, pp. 4, 2017.

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5018–5027, 2017.

Zeya Wang and Chenglong Ye. Deep clustering evaluation: How to validate internal clustering validation measures. *arXiv preprint arXiv:2403.14830*, 2024.

Zeya Wang, Baoyu Jing, Yang Ni, Nanqing Dong, Pengtao Xie, and Eric Xing. Adversarial domain adaptation being aware of class relationships. In *ECAI 2020*, pp. 1579–1586. IOS Press, 2020.

Zeya Wang, Yang Ni, Baoyu Jing, Deqing Wang, Hao Zhang, and Eric Xing. Dnb: A joint learning framework for deep bayesian nonparametric clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):7610–7620, 2021.

Shiqi Yang, Joost Van de Weijer, Luis Herranz, Shangling Jui, et al. Exploiting the intrinsic neighborhood structure for source-free domain adaptation. *Advances in neural information processing systems*, 34: 29393–29405, 2021a.

Shiqi Yang, Yaxing Wang, Joost Van De Weijer, Luis Herranz, and Shangling Jui. Generalized source-free domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8978–8987, 2021b.

Shiqi Yang, Shangling Jui, Joost Van De Weijer, et al. Attracting and dispersing: A simple approach for source-free domain adaptation. *Advances in Neural Information Processing Systems*, 35:5802–5815, 2022.

Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Universal domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2720–2729, 2019.

Weichen Zhang, Wanli Ouyang, Wen Li, and Dong Xu. Collaborative and adversarial network for unsupervised domain adaptation. In *IEEE CVPR*, pp. 3801–3809, 2018.

Appendix

Appendix	Description
Appendix A	More Details about Guided Nonparametric Bayesian Clustering
Appendix B	Approximate Posterior Inference via Variational Bayes
Appendix C	Additional Experiment Setup
Appendix D	Additional Results

A More Details about Guided Nonparametric Bayesian Clustering

Treating the embedding data $\mathcal{Z}_t^U = \{z_1^U, \ldots, z_{n_U}^U\}$ as observed data, a Dirichlet Process Mixture (DPM) model for $\mathcal{Z}_t^U$ can be naturally represented within a Bayesian hierarchical framework (same as Equation 2),

$$z_i^U|\eta_i \sim p(z_i^U|\eta_i), \quad \eta_i|G \sim G, \quad G \sim DP(\alpha, G_0). \tag{A1}$$

The probability function $p(\cdot \mid \eta_i)$ is parameterized by η_i, which is drawn from a DP-distributed prior G. In our method, we adopt a Dirichlet Process Gaussian Mixture Model (DPGMM), where $p(\cdot \mid \eta_i)$ is a multivariate Gaussian, and η_i consists of the mean and covariance parameters of the Gaussian. Equation A1 can be equivalently formulated as an infinite mixture model in Equation A2 by marginalizing out G.

$$\begin{aligned} &z_i^U|y_i^U = k \sim p(z_i^U|\eta_k^\star), \quad y_i^U|\pi \sim Cat(\boldsymbol{\pi}), \quad \boldsymbol{\pi} = (\pi_1, \pi_2, \cdots) \\ &\pi_k = \upsilon_k \prod_{j=1}^{k-1}(1-\upsilon_j), \quad \upsilon_k \sim Beta(1,\alpha), \quad \eta_k^\star \sim G_0 \end{aligned} \tag{A2}$$

where $\eta_k^\star$ represents the parameters of the distribution associated with the mixture component k and $\boldsymbol{\pi}$ is a probability vector specifying the mixing distribution over clusters $k = 1, 2, \ldots$. The categorical distribution, denoted by $Cat(\cdot)$, assigns cluster memberships according to $y_i^U \mid \boldsymbol{\pi} \sim Cat(\boldsymbol{\pi})$, which is equivalent to the probability expression $P(y_i^U = k) = \pi_k$. Equation A2 adopts the stick-breaking representation, where the DP prior G is explicitly expressed as an infinite mixture of point masses weighted by π_k (Sethuraman, 1994; Blei & Jordan, 2006). In Equation A2, the mixing proportions π_k can be viewed as being generated by sequentially "breaking" a unit-length stick into infinitely many pieces, with each piece determined by υ_k from the remaining stick.

For the embedding data from high-confidence samples, $\mathcal{Z}_t^H = \{z_1^H, \ldots, z_n^H\}$, we treat their labels $\hat{\mathcal{Y}}_t^H = \{\hat{y}_1^H, \ldots, \hat{y}_n^H\}$ obtained during the high-confidence sample selection step, as fixed, giving:

$$z_i^H \sim p(z_i^H|\eta_{\hat{y}_i^H}^\star). \tag{A3}$$

With the alignment mentioned in Section 3.2, we can combine Equation A3 with Equation A2 to obtain (same as Equation 3):

$$\begin{aligned} &z_i^U|y_i^U = k \sim p(z_i^U|\eta_k^\star), \quad z_i^H \sim p(z_i^H|\eta_{\hat{y}_i^H}^\star) \\ &y_i^U|\pi \sim Cat(\boldsymbol{\pi}), \quad \boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_{|\widehat{\mathcal{C}}|}, \cdots) \\ &\pi_k = \upsilon_k \prod_{j=1}^{k-1}(1-\upsilon_j), \quad \upsilon_k \sim Beta(1,\alpha), \quad \eta_k^\star \sim G_0 \end{aligned} \tag{A4}$$

where $p(\cdot|\eta_k^\star) = N(\cdot|\mu_k, \Lambda_k)$. We place a conjugate Normal-Wishart prior, $N(\mu_k|\zeta, \tau\Lambda_k) \times W(\Lambda_k|b, \Omega)$ on (μ_k, Λ_k), where τ, b and Ω are Bayesian hyperparameters.

B Approximate Posterior Inference via Variational Bayes

Our model is parameterized by four sets of parameters and latent variables: $\{\upsilon_k\}_{k=1}^{\infty}$, $\{\mu_k\}_{k=1}^{\infty}$, $\{\Lambda_k\}_{k=1}^{\infty}$, and $\{y_i^U\}_{i=1}^{n_U}$, where n_U denotes the size of $\mathcal{Z}_t^U$. We employ Variational Bayes (VB) to approximate the posterior distribution. VB approximates the true posterior by minimizing the Kullback–Leibler (KL) divergence between a variational distribution and the posterior. To facilitate this, we apply the mean filed approximation and assume a fully factorized variational distribution (Bishop & Nasrabadi, 2006),

$$q(\{\upsilon_k\}_{k=1}^{K^*-1}, \{\mu_k\}_{k=1}^{K^*}, \{\Lambda_k\}_{k=1}^{K^*}, , \{y_i^U\}_{i=1}^{n_U}) = \prod_{k=1}^{K^*-1} q(\upsilon_k) \prod_{k=1}^{K^*} q(\mu_k, \Lambda_k) \prod_{i=1}^{n_U} q(y_i^U), \tag{A5}$$

The stick-breaking process is truncated at a value K^* with $q(\upsilon_{K^*} = 1) = 1$ for computation purpose, a construction known as the truncated stick-breaking representation. It should be noted that K^* does not represent a pre-determined number of clusters, as in conventional clustering algorithms like K-Means. Instead, it serves as a practical upper bound on the number of clusters, and any sufficiently large value (e.g., 100) is generally expected to yield consistent results. We now present the model specifications.

$$\begin{aligned}
z_i^U \mid y_i^U, \boldsymbol{\mu}, \boldsymbol{\Lambda} &\sim \mathcal{N}(\mu_{y_i^U}, \Lambda_{y_i^U}) \\
z_i^H \mid \boldsymbol{\mu}, \boldsymbol{\Lambda} &\sim \mathcal{N}(\mu_{\hat{y}_i^H}, \Lambda_{\hat{y}_i^H}) \quad (\hat{y}_i^H \text{ considered as fixed}) \\
y_i^U \mid \boldsymbol{\pi} &\sim \text{Cat}(\boldsymbol{\pi}) \\
\mu_k \mid \Lambda_k &\sim \mathcal{N}(\zeta, \tau\Lambda_k) \\
\Lambda_k &\sim \mathcal{W}(b, \Omega) \\
\pi_k &= \upsilon_k \prod_{j=1}^{k-1} (1 - \upsilon_j) \\
\upsilon_k &\sim \text{Beta}(1, \alpha)
\end{aligned} \tag{A6}$$

Given Equation A6, the joint distribution of all random variables, expressed in their vectorized forms, can be factorized as:

$$p(\mathbf{Z}^U, \mathbf{Z}^H, \boldsymbol{Y}^U, \boldsymbol{\upsilon}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = p(\mathbf{Z}^U \mid \mathbf{Y}^U, \boldsymbol{\mu}, \boldsymbol{\Lambda}) p(\mathbf{Z}^H \mid \boldsymbol{\mu}, \boldsymbol{\Lambda}) p(\mathbf{Y}^U \mid \boldsymbol{\upsilon}) p(\boldsymbol{\upsilon}) p(\boldsymbol{\mu} \mid \boldsymbol{\Lambda}) p(\boldsymbol{\Lambda}) \tag{A7}$$

The full conditional distributions in our model belong to the exponential family. Accordingly, we select variational distributions also from the exponential family: $q(\upsilon_k) = Beta(\upsilon_k|\gamma_{k1}, \gamma_{k2})$, $q(\mu_k, \Lambda_k) = N(\mu_k \mid m_k, \tau_k\Lambda_k) W(\Lambda_k \mid c_k, D_k)$, $q(y_i^U) = Cat(\phi_i)$, and apply mean-field variational inference for exponential families (Blei & Jordan, 2006). Variational inference typically proceeds via a coordinate ascent algorithm, updating the parameters of each variational distribution sequentially. Under the mean-field framework, the variational parameters for each distribution are updated by computing the expected natural parameters of the corresponding full conditional, evaluated with respect to the current variational distributions. For completeness, we provide below the detailed update steps for our model.

Variational distributions:

$$\begin{aligned}
&q(\mu_k, \Lambda_k) = N(\mu_k \mid m_k, \tau_k\Lambda_k) W(\Lambda_k \mid c_k, D_k) \\
&q(\upsilon_k) = Beta(\upsilon_k \mid \gamma_{k1}, \gamma_{k2}) \\
&q(y_i^U = k) = \phi_{ik}^{I(y_i^U = k)}
\end{aligned} \tag{A8}$$

Update variational parameters (τ_k, m_k, c_k, D_k):

To simplify the notations, we let $n_k = \sum_{i=1}^{n_U} I(y_i^U = k)$, $\hat{n}_k = E_q[n_k] = \sum_{i=1}^{n_U} \phi_{ik}$, and $\hat{n}_k^H = \sum_{i=1}^{n_H} I(\hat{y}_i^H = k)$. Also, we set

$$\overline{z}_k = \frac{1}{n_k + \hat{n}_k^H}[\sum_{i=1}^{n_U} I(y_i^U = k) z_i^U + \sum_{i=1}^{n_H} I(\hat{y}_i^H = k) z_i^H] \tag{A9}$$

$$S_k = \frac{1}{n_k + \hat{n}_k^H}[\sum_{i=1}^{n_U} I(y_i^U = k)\left(z_i^U - \overline{z}_k\right)\left(z_i^U - \overline{z}_k\right)^{\mathrm{T}} + \sum_{i=1}^{n_H} I(\hat{y}_i^H = k)\left(z_i^H - \overline{z}_k\right)\left(z_i^H - \overline{z}_k\right)^{\mathrm{T}}] \tag{A10}$$

For the likelihood of the parameters associated with cluster k, given the embedding data z_i^U and z_i^H, we have

$$\begin{aligned}
&\prod_{i=1}^{n_U} N(z_i^U \mid \mu_k, \Lambda_k)^{I(y_i^U = k)} \prod_{i=1}^{n_H} N(z_i^H \mid \mu_k, \Lambda_k)^{I(\hat{y}_i^H = k)} \\
&\propto \exp\left(-\frac{1}{2}\sum_{i=1}^{n_U} I(y_i^U = k)\{(z_i^U - \mu_k)^\top \Lambda_k (z_i^U - \mu_k) - \log|\Lambda_k|\}\right) \\
&\times \exp\left(-\frac{1}{2}\sum_{i=1}^{n_H} I(\hat{y}_i^H = k)\{(z_i^H - \mu_k)^\top \Lambda_k (z_i^H - \mu_k) - \log|\Lambda_k|\}\right) \\
&\propto \exp\left(-\frac{n_k + \hat{n}_k^H}{2}\operatorname{tr}(S_k\Lambda_k) - \frac{n_k + \hat{n}_k^H}{2}(\bar{z}_k - \mu_k)^\top \Lambda_k (\bar{z}_k - \mu_k) + \frac{n_k + \hat{n}_k^H}{2}\log|\Lambda_k|\right)
\end{aligned} \tag{A11}$$

Let d denote the dimensionality of the embedding data. We can easily show that the full conditional distribution of each μ_k, Λ_k.

$$\begin{aligned}
p(\mu_k, \Lambda_k \mid \cdot) &\propto \prod_{i=1}^{n_U} N(z_i^U \mid \mu_k, \Lambda_k)^{I(y_i^U = k)} \prod_{i=1}^{n_H} N(z_i^H \mid \mu_k, \Lambda_k)^{I(\hat{y}_i^H = k)} N(\mu_k \mid \zeta, \tau\Lambda_k) W(\Lambda_k \mid b, \Omega) \\
&\propto \exp\left(-\frac{n_k + \hat{n}_k^H}{2}\operatorname{tr}(S_k\Lambda_k) - \frac{n_k + \hat{n}_k^H}{2}(\bar{z}_k - \mu_k)^\top \Lambda_k (\bar{z}_k - \mu_k) + \frac{n_k + \hat{n}_k^H}{2}\log|\Lambda_k|\right) \\
&\times \exp\left(-\frac{1}{2}\tau(\mu_k - \zeta)^\top \Lambda_k (\mu_k - \zeta) + \frac{1}{2}\log|\Lambda_k|\right) \\
&\times \exp\left(-\frac{1}{2}\operatorname{tr}(\Omega^{-1}\Lambda_k) + \frac{b - d - 1}{2}\log|\Lambda_k|\right) \\
&\propto \exp\left(-\frac{1}{2}[(n_k + \hat{n}_k^H)(\bar{z}_k - \mu_k)^\top \Lambda_k (\bar{z}_k - \mu_k) + \tau(\mu_k - \zeta)^\top \Lambda_k (\mu_k - \zeta)]\right. \\
&\left. -\frac{1}{2}\operatorname{tr}(((n_k + \hat{n}_k^H)S_k + \Omega^{-1})\Lambda_k) + \frac{1}{2}(n_k + \hat{n}_k^H + b - d)\log|\Lambda_k|\right) \\
&\propto \exp\left(-\frac{1}{2}[(\tau + n_k + \hat{n}_k^H)(\mu_k - \frac{\tau\zeta + (n_k + \hat{n}_k^H)\bar{z}_k}{\tau + n_k + \hat{n}_k^H})^\top \Lambda_k (\mu_k - \frac{\tau\zeta + (n_k + \hat{n}_k^H)\bar{z}_k}{\tau + n_k + \hat{n}_k^H})\right. \\
&\qquad + \frac{(n_k + \hat{n}_k^H)\tau}{\tau + n_k + \hat{n}_k^H}(\bar{z}_k - \zeta)^\top \Lambda_k (\bar{z}_k - \zeta)] \\
&\qquad \left. -\frac{1}{2}\operatorname{tr}((\Omega^{-1} + (n_k + \hat{n}_k^H)S_k)\Lambda_k) + \frac{1}{2}(n_k + \hat{n}_k^H + b - d)\log|\Lambda_k|\right) \\
&\propto \exp\left(-\frac{1}{2}(\tau + n_k + \hat{n}_k^H)(\mu_k - \frac{\tau\zeta + (n_k + \hat{n}_k^H)\bar{z}_k}{\tau + n_k + \hat{n}_k^H})^\top \Lambda_k (\mu_k - \frac{\tau\zeta + (n_k + \hat{n}_k^H)\bar{z}_k}{\tau + n_k + \hat{n}_k^H})\right. \\
&\qquad -\frac{1}{2}\operatorname{tr}((\Omega^{-1} + (n_k + \hat{n}_k^H)S_k + \frac{(n_k + \hat{n}_k^H)\tau}{\tau + n_k + \hat{n}_k^H}(\bar{z}_k - \zeta)(\bar{z}_k - \zeta)^\top)\Lambda_k) \\
&\qquad \left. +\frac{1}{2}(n_k + \hat{n}_k^H + b - d)\log|\Lambda_k|\right)
\end{aligned} \tag{A12}$$

Thus the natural parameters are

$$\eta_1 = \text{vec}(\Omega^{-1} + (n_k + \hat{n}_k^H)S_k + \frac{(n_k + \hat{n}_k^H)\tau}{\tau + n_k + \hat{n}_k^H}(\bar{z}_k - \zeta)(\bar{z}_k - \zeta)^\top + \tag{A13}$$
$$\frac{(\tau\zeta + (n_k + \hat{n}_k^H)\bar{z}_k)(\tau\zeta + (n_k + \hat{n}_k^H)\bar{z}_k)^\top}{\tau + n_k + \hat{n}_k^H}) \tag{A14}$$
$$\eta_2 = \tau\zeta + (n_k + \hat{n}_k^H)\bar{z}_k = \sum_i^{n_U} I(y_i^U = k)z_i^U + \sum_i^{n_H} I(\hat{y}_i^H = k)z_i^H + \tau\zeta, \tag{A15}$$
$$\eta_3 = n_k + \hat{n}_k^H + \tau, \tag{A16}$$
$$\eta_4 = n_k + \hat{n}_k^H + b - d \tag{A17}$$

Solve τ_k

$$\begin{aligned} \tau_k &= (\tau + \hat{n}_k + \hat{n}_k^H) \\ \Rightarrow \tau_k &= \tau + \hat{n}_k + \hat{n}_k^H \end{aligned} \tag{A18}$$

Solve m_k

$$\begin{aligned} \tau_k m_k &= E_q[\sum_i^{n_U} I(y_i^U = k)z_i^U + \sum_i^{n_H} I(\hat{y}_i^H = k)z_i^H + \tau\zeta] \\ \tau_k m_k &= \tau\zeta + \sum_{i=1}^{n_U} \phi_{ik} z_i^U + \sum_{i=1}^{n_H} z_i^H I(\hat{y}_i^H = k) \\ \Rightarrow m_k &= \frac{\tau\zeta + \sum_{i=1}^{n_U} \phi_{ik} z_i^U + \sum_{i=1}^{n_H} z_i^H I(\hat{y}_i^H = k)}{\tau + \hat{n}_k + \hat{n}_k^H} \end{aligned} \tag{A19}$$

Solve c_k

$$\begin{aligned} c_k - d &= E_q[n_k + \hat{n}_k^H + b - d] \\ c_k - d &= b - d + \hat{n}_k + \hat{n}_k^H \\ \Rightarrow c_k &= \hat{n}_k + \hat{n}_k^H + b \end{aligned} \tag{A20}$$

Solve D_k

$$D_k^{-1} + \tau_k m_k m_k^\top = E_q[\Omega^{-1} + (n_k + \hat{n}_k^H) S_k + \frac{(n_k + \hat{n}_k^H)\tau}{\tau + n_k + \hat{n}_k^H}(\bar{z}_k - \zeta)(\bar{z}_k - \zeta)^\top \tag{A21}$$

$$+ \frac{(\tau\zeta + (n_k + \hat{n}_k^H)\bar{z}_k)(\tau\zeta + (n_k + \hat{n}_k^H)\bar{z}_k)^\top}{\tau + n_k + \hat{n}_k^H}]$$

$$D_k^{-1} + \tau_k m_k m_k^\top = E_q[\Omega^{-1} + \sum_{i=1}^{n_U} I(y_i^U = k)\left(z_i^U - \overline{z}_k\right)\left(z_i^U - \overline{z}_k\right)^{\mathrm{T}} \tag{A22}$$

$$+ \sum_{i=1}^{n_H} I(\hat{y}_i^H = k)\left(z_i^H - \overline{z}_k\right)\left(z_i^H - \overline{z}_k\right)^{\mathrm{T}} + (n_k + \hat{n}_k^H)\overline{z}_k\overline{z}_k^\top + \tau\zeta\zeta^\top]$$

$$D_k^{-1} + \tau_k m_k m_k^\top = E_q[\Omega^{-1} + (\sum_{i=1}^{n_U} I(y_i^U = k) z_i^U (z_i^U)^\top + \sum_{i=1}^{n_H} I(\hat{y}_i^H = k) z_i^H (z_i^H)^\top)$$

$$- 2\overline{z}_k(\sum_{i=1}^{n_U} I(y_i^U = k)(z_i^U)^\top + \sum_{i=1}^{n_H} I(\hat{y}_i^H = k)(z_i^H)^\top) + 2(n_k + \hat{n}_k^H)\overline{z}_k\overline{z}_k^\top + \tau\zeta\zeta^\top]$$

$$D_k^{-1} + \tau_k m_k m_k^\top = E_q[\Omega^{-1} + (\sum_{i=1}^{n_U} I(y_i^U = k) z_i^U (z_i^U)^\top + \sum_{i=1}^{n_H} I(\hat{y}_i^H = k) z_i^H (z_i^H)^\top + \tau\zeta\zeta^\top]$$

$$D_k^{-1} = \Omega^{-1} + \sum_{i=1}^{n_U} \phi_{ik} z_i^U (z_i^U)^\top + \sum_{i=1}^{n_H} I(\hat{y}_i^H = k) z_i^H (z_i^H)^\top + \tau\zeta\zeta^\top - \tau_k m_k m_k^\top \tag{A23}$$

If we further set:

$$\overline{z}_k^* = \frac{1}{\hat{n}_k + \hat{n}_k^H}[\sum_{i=1}^{n_U} \phi_{ik} z_i^U + \sum_{i=1}^{n_H} I(\hat{y}_i^H = k) z_i^H] \tag{A24}$$

$$S_k^* = \frac{1}{\hat{n}_k + \hat{n}_k^H}[\sum_{i=1}^{n_U} \phi_{ik}\left(z_i^U - \overline{z}_k^*\right)\left(z_i^U - \overline{z}_k^*\right)^{\mathrm{T}} + \sum_{i=1}^{n_H} I(\hat{y}_i^H = k)\left(z_i^H - \overline{z}_k^*\right)\left(z_i^H - \overline{z}_k^*\right)^{\mathrm{T}}] \tag{A25}$$

We can rearrange Equation (A23) to have:

$$D_k^{-1} = \Omega^{-1} + (\hat{n}_k + \hat{n}_k^H) S_k^* + \frac{(\hat{n}_k + \hat{n}_k^H)\tau}{\tau + \hat{n}_k + \hat{n}_k^H}(\bar{z}_k^* - \zeta)(\bar{z}_k^* - \zeta)^\top \tag{A26}$$

Update variational parameters $(\gamma_{k1}, \gamma_{k2})$:

Given $\hat{n}_k = E_q[n_k] = \sum_{i=1}^{n_U} \phi_{ik}$.

$$\gamma_{k1} = E_q[n_k + 1] = \hat{n}_k + 1 \tag{A27}$$

$$\gamma_{k2} = E_q[\sum_{j>k} n_j + \alpha] = \sum_{j>k} \hat{n}_j + \alpha \tag{A28}$$

Update variational parameters $\phi_i = (\phi_{i1}, \phi_{i2}, \dots)$:

$$\begin{aligned}
\log(\phi_{ik}) &\propto E_q[\log \pi_k + \log N(z_i^U|\mu_k, \Lambda_k)] \\
&\propto E_q\big[\log \upsilon_k + \sum_{j=1}^{k-1} \log(1-\upsilon_j) + \frac{1}{2}\log|\Lambda_k| - \frac{1}{2}(z_i^U - \mu_k)^T \Lambda_k (z_i^U - \mu_k)\big] \\
&\propto \psi(\gamma_{k1}) - \psi(\gamma_{k1} + \gamma_{k2}) + \sum_{j=1}^{k-1}[\psi(\gamma_{j2}) - \psi(\gamma_{j1} + \gamma_{j2})] \\
&\quad + \frac{1}{2}\big[\psi_d(\frac{c_k}{2}) + \log|D_k|\big] - \frac{1}{2}[d\tau_k^{-1} + c_k(z_i^U - m_k)^\top D_k (z_i^U - m_k)] \qquad \text{(A29)}
\end{aligned}$$

where $\psi(\cdot)$ and $\psi_d(\cdot)$ represent the digamma and multivariate digamma functions, respectively.

Given these update steps for the variational parameters, we employ a coordinate descent algorithm to iteratively refine them until a convergence criterion is met, such as the convergence of the Evidence Lower Bound (ELBO), which is typically used as the objective in standard Dirichlet Process Mixture (DPM) models (Blei & Jordan, 2006). For initialization, we assign clusters using a variant of the KMeans algorithm, adapted to our setting (standard KMeans is commonly used in the variational Bayes (VB) procedure for DPMs). In this variant, we incorporate the labels of high-confidence samples during updates of the source-class cluster centers and modify the KMeans++ initialization (Arthur & Vassilvitskii, 2006) to consider class-center information from these samples at the start of KMeans. Finally, using the estimated values of ϕ_{ik}, the predicted labels for the uncertain samples, $\hat{\mathcal{Y}}_t^U = \{\hat{y}_1^U, \dots, \hat{y}_{n_U}^U\}$, are determined by $\hat{y}_i^U = \arg\max_k \phi_{ik}$.

C Additional Experiment Setup

Normalized Mutual Information (NMI). NMI is a standard metric for evaluating the agreement between two label assignments, invariant to permutations of the class labels. Given label assignments $\mathcal{Y}_a$ and $\mathcal{Y}_b$, NMI can be computed by scaling the mutual information between $\mathcal{Y}_a$ and $\mathcal{Y}_b$ by the arithmetic mean of their respective entropies:

$$NMI(\mathcal{Y}_a; \mathcal{Y}_b) = \frac{2 \times I(\mathcal{Y}_a; \mathcal{Y}_b)}{H(\mathcal{Y}_a) + H(\mathcal{Y}_b)}, \qquad \text{(A30)}$$

where $I(\cdot;\cdot)$ represents the mutual information between two random variables and $H(\cdot)$ represents the entropy of a random variable. NMI takes values between 0, suggesting no mutual information, and 1, indicating perfect alignment, with higher values reflecting better correspondence between the two label assignments. In our evaluation, we compute $NMI(\mathcal{Y}; \hat{\mathcal{Y}})$, comparing ground-truth labels $\mathcal{Y}$ with predicted cluster labels $\hat{\mathcal{Y}}$ across all samples. To evaluate NMI for the compared methods in our experiments, which requires estimated labels for unknown-class samples, we apply the *DPM* component, using the same model configurations as in our approach, to the unknown target samples predicted by SHOT and GLC, then combine the resulting cluster assignments with their known-class predictions to compute NMI against the ground-truth labels.

Additional Settings: During target model adaptation, we use the SGD optimizer with a momentum of 0.9 and weight decay of 0.001, following the conventional inverse learning rate decay schedule defined as $\eta_l = \frac{\eta_0}{(1+10\cdot\frac{l}{L})^{0.75}}$, where η_0 is the initial learning rate. η_0 is set to 10^{-2} for the classifier on Office-31, 10^{-4} for Office-Home, and 10^{-3} for VisDA, with a scaling factor of 0.1 applied to the extractor and bottleneck layers. The batch size is fixed at 64 for all benchmark datasets. The weight for λ_{loc} is set to 0.01. The number of nearest neighbors, K_{nn}, is set to 5 for both Office-31 and Office-Home, and 100 for VisDA to account for its considerably larger data scale. We run Algorithm 1 for 10 epochs across all three datasets. For high-confidence sample selection, we use the default threshold value $\delta = 0.5$ for the VisDA dataset. For the other two smaller datasets, we tune δ over the range $[0.4, 0.6]$ in increments of 0.05, selecting the value that maximizes the Silhouette score (Rousseeuw, 1987) computed on the embedded data and the estimated labels produced by the GBNC. This choice is supported by the sensitivity analysis in Section 4.3. The hyperparameters in the DP base distribution G_0 are set to $\zeta = \bar{z}$, which is the mean of embedding data, $\tau = 1$, $b = d$, and $\Omega^{-1} = 0.001 \times d \times I_d$, where d is the dimension of the embedding, which is 256 in our model. K^*

is chosen with 100 for all the datasets. The VB algorithm for our GBNC model is implemented by adapting the DPM implementation provided in the *scikit-learn* library (Pedregosa et al., 2011). All experiments are conducted on a server with eight NVIDIA RTX A5500 GPUs and AMD EPYC 7413 24-Core processors.

D Additional Results

This section presents additional results, including tables, figures, and sensitivity analyses, that were omitted from the main paper due to space constraints.

D.1 Additional Results for OPDA

In this section, we report the mean accuracy of all methods under the OPDA settings, as shown in Table A1 and Table A2 for the Office-Home, Office-31, and VisDA datasets. These results are consistent with the trends observed in H-scores and NMI metrics (Tables 1a, 2a, 1b, 2b). Our method achieves the highest average mean accuracy on Office-Home and Office-31, and performs comparably to GLC, the best-performing method, on VisDA.

Table A1: Mean Accuracy (%) on the Office-Home dataset under OPDA settings.

Methods	SF	FG	A2C	A2P	A2R	C2A	C2P	C2R	P2A	P2C	P2R	R2A	R2C	R2P	Avg
Best Epoch															
SHOT-O (Liang et al., 2020)	✓	✗	36.1	55.5	56.9	50.8	52.9	60.6	50.4	39.9	62.3	55.0	36.5	56.0	51.0
GLC (Qu et al., 2023)	✓	✗	46.1	73.8	89.2	57.6	66.3	**89.8**	55.1	46.7	**88.1**	75.2	48.0	79.7	67.9
ABC	✓	✓	**52.5**	**80.7**	**90.2**	**74.5**	**82.6**	88.8	**71.8**	**48.6**	86.8	**79.1**	**50.9**	**81.9**	**74.0**
Final Epoch															
SHOT-O (Liang et al., 2020)	✓	✗	41.3	54.7	55.8	54.6	53.5	61.8	54.3	44.5	60.0	55.9	41.8	57.1	52.9
GLC (Qu et al., 2023)	✓	✗	45.8	73.4	**91.1**	58.2	66.1	**89.8**	54.9	29.2	**88.2**	**78.3**	40.9	**81.9**	66.4
GBNC-only	-	-	45.8	81.9	90.3	66.0	71.7	83.0	66.1	44.7	85.9	73.4	47.5	81.3	69.8
ABC	✓	✓	**48.3**	**74.2**	88.0	**73.3**	**72.3**	86.8	**72.3**	**46.4**	86.3	76.4	**50.1**	71.2	**70.5**

Table A2: Mean Accuracy (%) on the Office-31 and VisDA datasets under OPDA settings.

Methods	SF	FG	**Office-31**							**VisDA**
			A2D	A2W	D2A	D2W	W2A	W2D	Avg	S2R
Best Epoch										
SHOT-O (Liang et al., 2020)	✓	✗	58.6	55.5	55.8	77.4	52.9	64.8	60.8	30.9
GLC (Qu et al., 2023)	✓	✗	79.5	85.3	88.6	91.4	87.4	**95.6**	87.9	**69.1**
ABC	✓	✓	**90.2**	**94.5**	**88.7**	**91.9**	**89.8**	94.1	**91.5**	62.6
Final Epoch										
SHOT-O (Liang et al., 2020)	✓	✗	59.1	63.6	55.3	72.9	51.8	65.7	61.4	37.4
GLC (Qu et al., 2023)	✓	✗	81.6	83.0	69.3	84.0	70.4	82.3	78.4	**64.4**
GBNC-only	-	-	80.4	91.1	86.7	77.2	79.3	94.8	84.9	29.9
ABC	✓	✓	**88.2**	**94.5**	**89.9**	**87.4**	**89.7**	**93.8**	**90.6**	63.2

D.2 Results for OSDA

In this section, we present results under the OSDA setting for the Office-31 dataset, reporting H-score, mean accuracy, and NMI in Tables A3, A4, and A5, respectively. We follow the standard experimental split used in prior work (Qu et al., 2023), with 10 shared, 0 source-private, and 11 target-private classes. The results are consistent with our observations from the OPDA settings, with our method achieving solid performance across metrics. Notably, SHOT shows significantly improved performance under the OSDA setting, achieving the highest average mean accuracy.

D.3 Additional Sensitivity Analysis Results

In this section, we present an extensive sensitivity analysis of the key hyperparameters used in our model under the OPDA setting, including λ_{loc}, α, K_{nn}, and δ. These results are visualized across multiple figures

Table A3: H-score (%) on the Office-31 dataset under OSDA settings.

Methods	SF	FG	Office-31						
			A2D	A2W	D2A	D2W	W2A	W2D	Avg
OSBP (Saito et al., 2018)	✗	✗	82.4	82.7	75.1	97.2	73.7	91.1	83.7
ROS (Bucci et al., 2020)	✗	✗	82.4	82.1	77.9	96.0	77.2	**99.7**	85.9
CMU (Fu et al., 2020)	✗	✗	52.6	55.7	76.5	75.9	65.8	64.7	65.2
DANCE (Saito et al., 2020)	✗	✗	84.9	78.8	79.1	78.8	68.3	88.9	79.8
DCC (Li et al., 2021)	✗	✗	58.3	54.8	67.2	89.4	85.3	80.9	72.6
OVANet (Saito & Saenko, 2021)	✗	✗	**90.5**	**88.3**	**86.7**	**98.2**	**88.3**	98.4	**91.7**
GATE (Chen et al., 2022)	✗	✗	88.4	86.5	84.2	95.0	86.1	96.7	89.5
Best Epoch (Best H-score)									
SHOT-O (Liang et al., 2020)	✓	✗	89.1	87.2	82.7	88.0	79.6	86.2	85.4
GLC (Qu et al., 2023)	✓	✗	82.9	77.1	**92.1**	**95.6**	92.9	86.2	87.8
ABC	✓	✓	**90.1**	**89.9**	90.1	90.1	**93.1**	**94.3**	**91.3**
Final Epoch									
SHOT-O (Liang et al., 2020)	✓	✗	85.2	81.3	66.5	79.9	71.5	85.9	78.3
GLC (Qu et al., 2023)	✓	✗	82.1	75.9	**91.9**	**95.6**	**92.7**	84.0	87.0
GBNC-only	-	-	88.8	88.3	89.8	90.3	91.4	93.4	90.3
ABC	✓	✓	**85.6**	**89.6**	89.6	87.6	89.1	**91.0**	**88.8**

Table A4: Mean Accuracy (%) on the Office-31 dataset under OSDA settings.

Methods	SF	FG	Office-31						
			A2D	A2W	D2A	D2W	W2A	W2D	Avg
Best Epoch									
SHOT-O (Liang et al., 2020)	✓	✗	90.5	**91.9**	82.4	89.8	81.9	85.2	86.9
GLC (Qu et al., 2023)	✓	✗	76.5	67.4	**91.7**	**94.3**	**91.5**	77.9	83.2
ABC	✓	✓	**91.4**	87.6	90.1	83.8	90.7	**91.7**	**89.2**
Final Epoch									
SHOT-O (Liang et al., 2020)	✓	✗	**93.3**	**94.1**	90.3	94.2	81.7	86.9	**90.0**
GLC (Qu et al., 2023)	✓	✗	74.4	65.6	**91.8**	**94.3**	**91.4**	74.9	82.0
GBNC-only	-	-	93.9	85.0	85.2	84.3	90.7	91.5	88.4
ABC	✓	✓	87.0	91.2	89.1	82.4	89.3	**93.2**	88.7

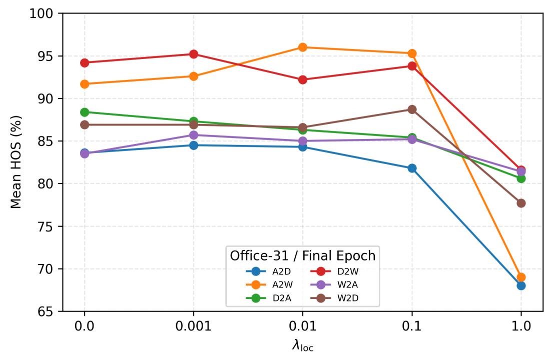
Table A5: NMI (%) on the Office-31 dataset under OSDA settings.

Methods	Office-31						
	A2D	A2W	D2A	D2W	W2A	W2D	Avg
Best Epoch							
SHOT-O (Liang et al., 2020) + DPM	**80.5**	81.7	69.8	80.8	69.7	79.9	77.0
GLC (Qu et al., 2023) + DPM	79.6	79.4	69.0	83.2	69.2	80.6	76.8
ABC	80.1	**83.2**	**70.2**	**84.0**	**71.7**	**82.1**	**78.6**
Final Epoch							
SHOT-O (Liang et al., 2020) + DPM	**81.0**	81.2	**70.2**	80.4	69.0	79.9	76.9
GLC (Qu et al., 2023) + DPM	79.8	79.3	68.6	**83.2**	68.8	80.7	76.7
GBNC-only	79.8	82.5	68.8	82.7	69.9	81.8	77.6
ABC	79.7	**83.8**	69.7	83.6	**69.8**	**82.2**	**78.1**

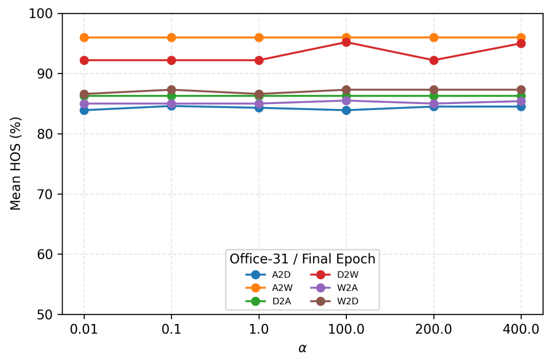
(Figures A1–A6) and cover metrics such as H-score, NMI, and mean accuracy at the Final Epoch on the Office-31 and Office-Home datasets.

To provide a comprehensive view of hyperparameter effects, we present two complementary visualization formats. Specifically, line plots (Figures A1–A3) illustrate per-task performance variations as each hyperparameter changes, highlighting the response patterns across different domain adaptation scenarios. In contrast, bar plots (Figures A5 and A6) summarize the average performance across all tasks, offering a high-level overview of model robustness.

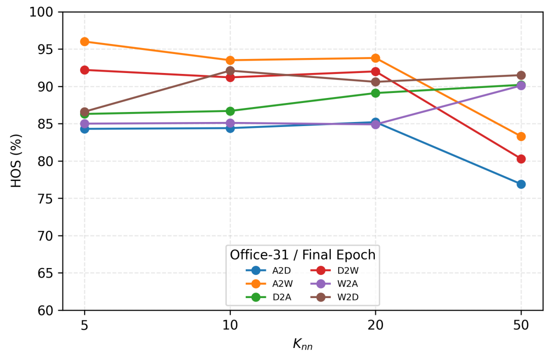
Overall, the results show that the performance of our model remains stable across a broad range of hyperparameter values, with only moderate sensitivity observed in specific cases. For instance, the performance (measured by NMI and H-score) remains stable when varying α, while it is slightly more sensitive to the choice of K_{nn}. The results validate our hyperparameter choices and indicate that ABC performs well without extensive tuning, making it practical for real-world deployment.



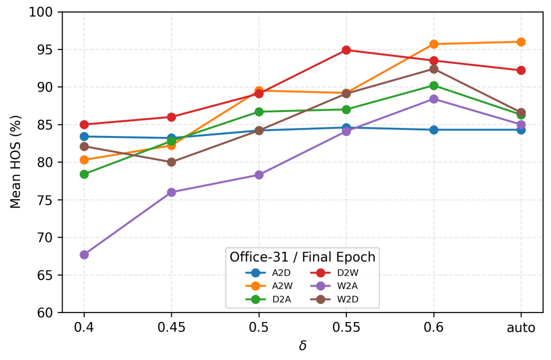
(a) Office-31: Sensitivity on λ_{loc}



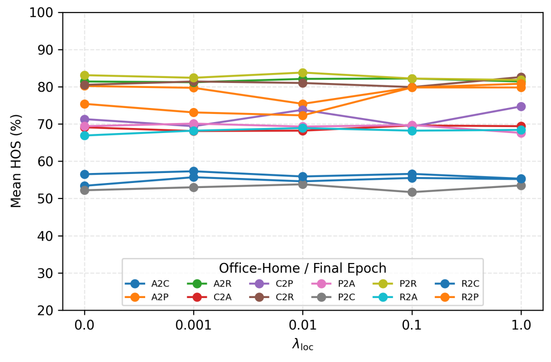
(b) Office-31: Sensitivity on α



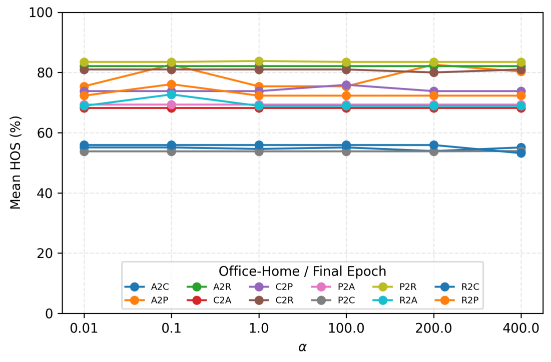
(c) Office-31: Sensitivity on K_{nn}



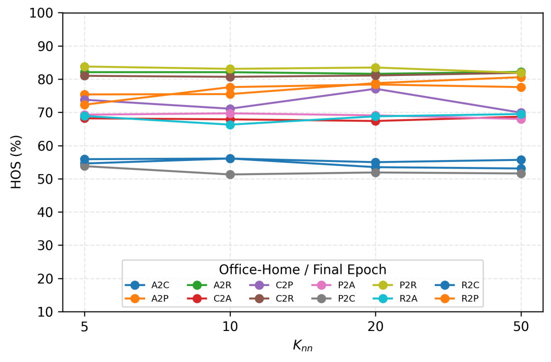
(d) Office-31: Sensitivity on δ



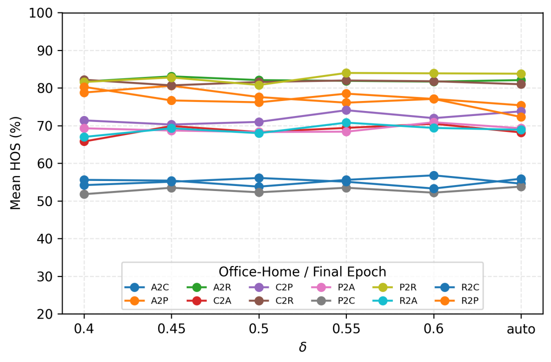
(e) Office-Home: Sensitivity on λ_{loc}



(f) Office-Home: Sensitivity on α

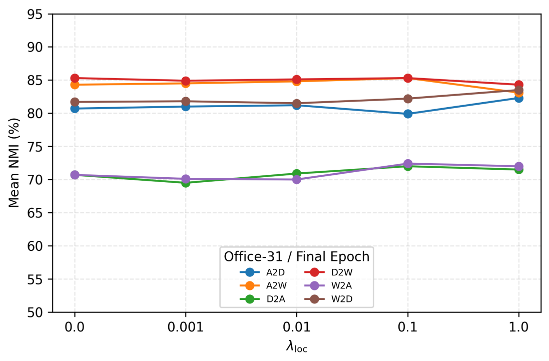


(g) Office-Home: Sensitivity on K_{nn}

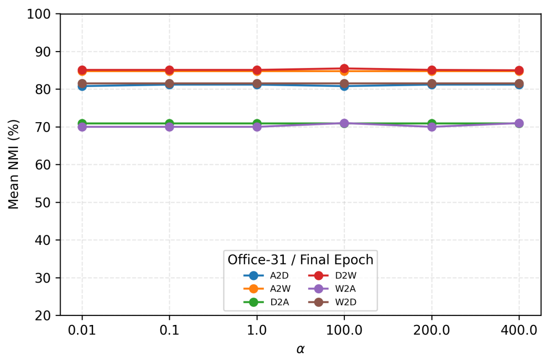


(h) Office-Home: Sensitivity on δ

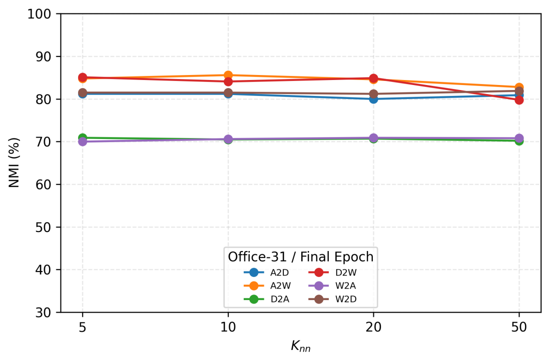
Figure A1: Hyperparameter sensitivity analysis per task (H-score) of λ_{loc}, α, K_{nn}, and δ on Office-31 and Office-Home under the OPDA setting.



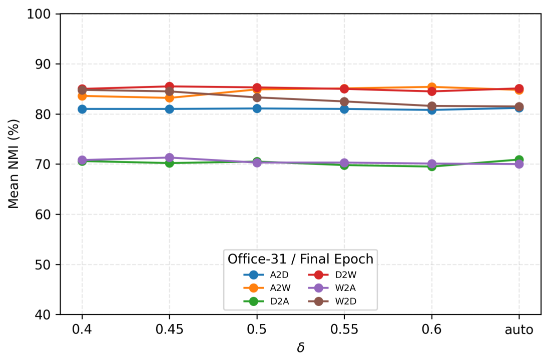
(a) Office-31: Sensitivity on λ_{loc}



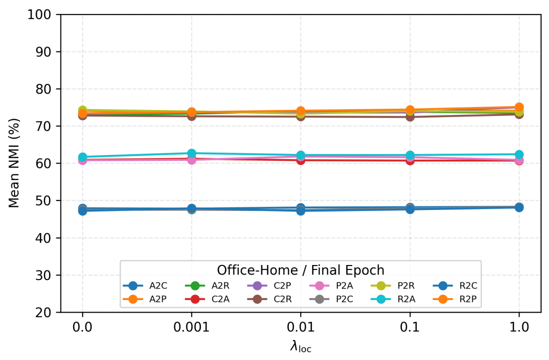
(b) Office-31: Sensitivity on α



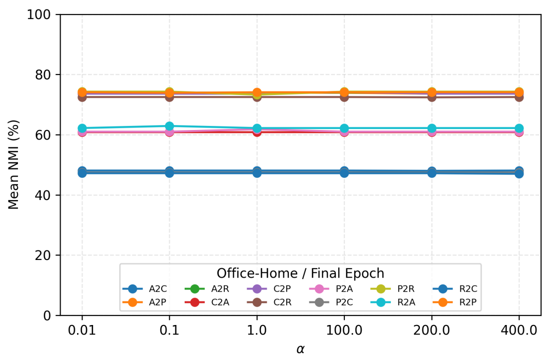
(c) Office-31: Sensitivity on K_{nn}



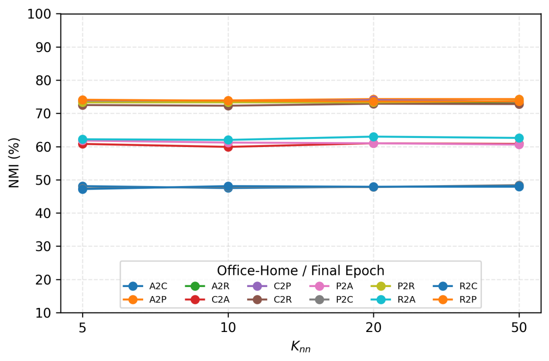
(d) Office-31: Sensitivity on δ



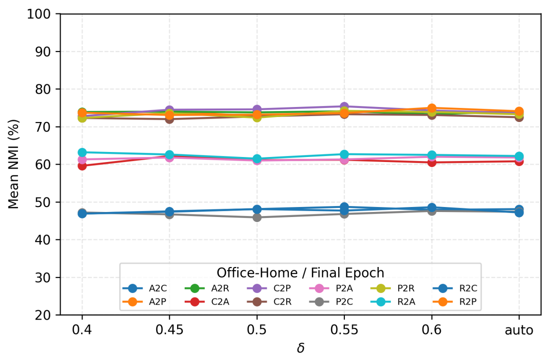
(e) Office-Home: Sensitivity on λ_{loc}



(f) Office-Home: Sensitivity on α

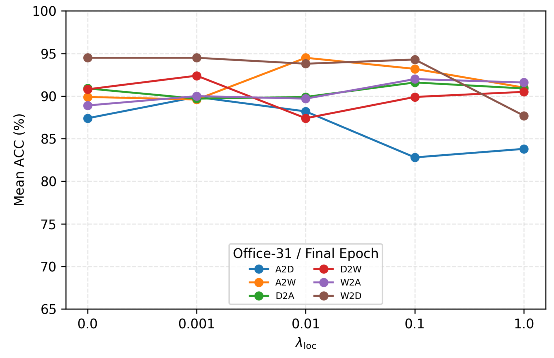


(g) Office-Home: Sensitivity on K_{nn}

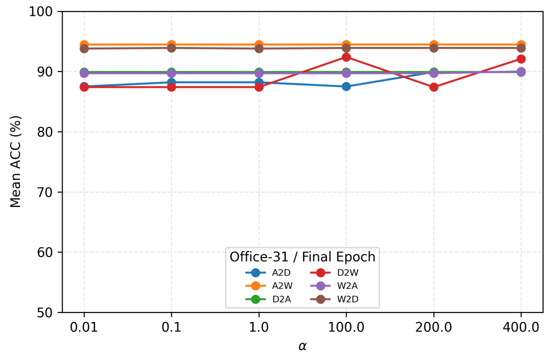


(h) Office-Home: Sensitivity on δ

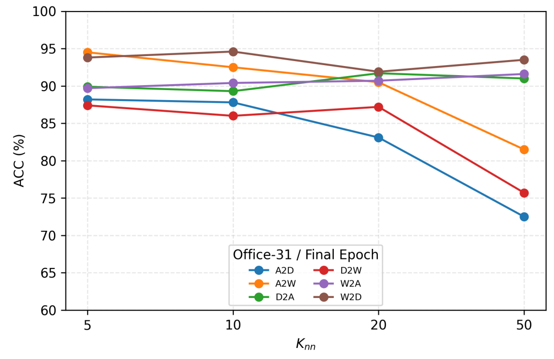
Figure A2: Hyperparameter sensitivity analysis per task (NMI) of λ_{loc}, α, K_{nn}, and δ on Office-31 and Office-Home under the OPDA setting.



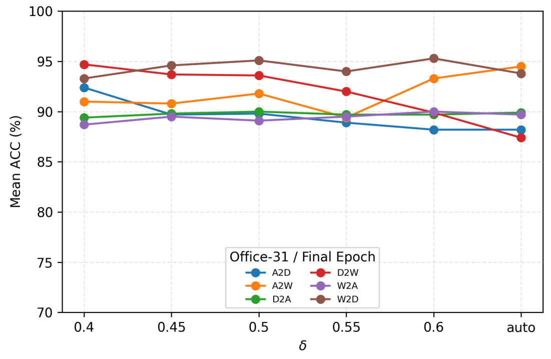
(a) Office-31: Sensitivity on λ_{loc}



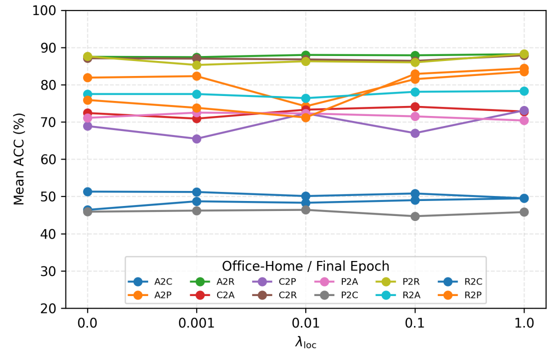
(b) Office-31: Sensitivity on α



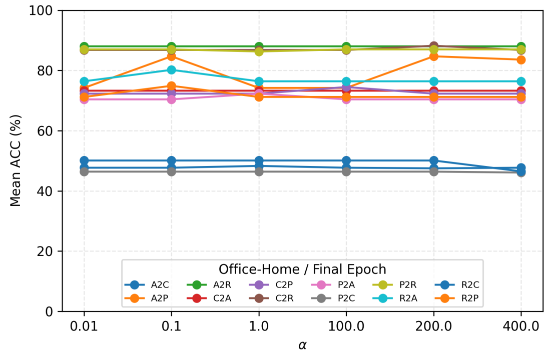
(c) Office-31: Sensitivity on K_{nn}



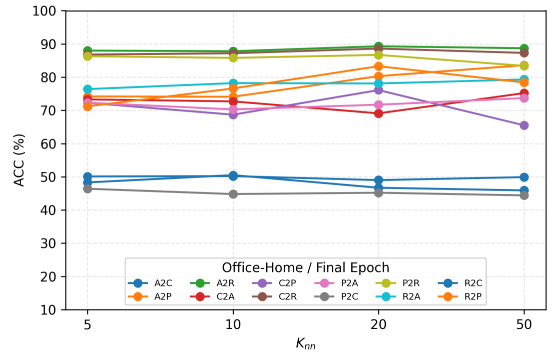
(d) Office-31: Sensitivity on δ



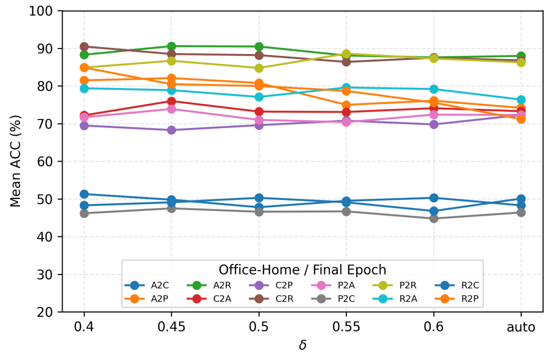
(e) Office-Home: Sensitivity on λ_{loc}



(f) Office-Home: Sensitivity on α

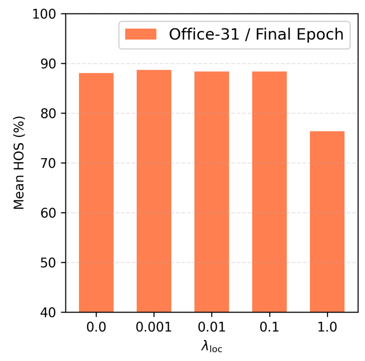


(g) Office-Home: Sensitivity on K_{nn}

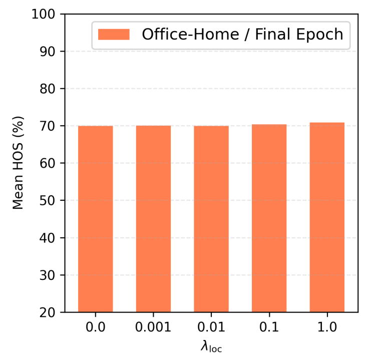


(h) Office-Home: Sensitivity on δ

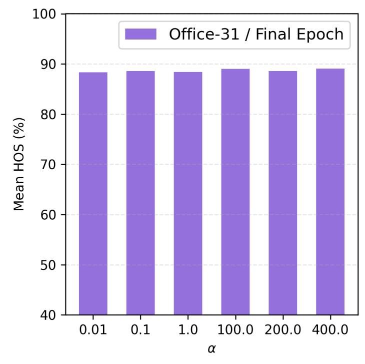
Figure A3: Hyperparameter sensitivity analysis per task (mean accuracy) of λ_{loc}, α, K_{nn}, and δ on Office-31 and Office-Home under the OPDA setting.



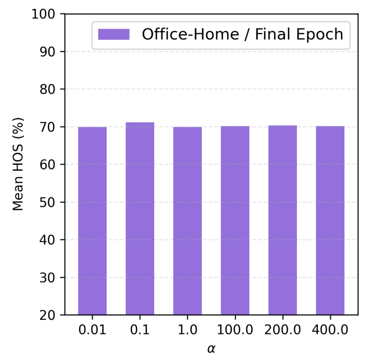
(a) Office-31: λ_{loc}



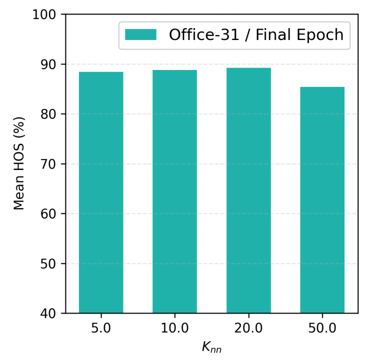
(b) Office-Home: λ_{loc}



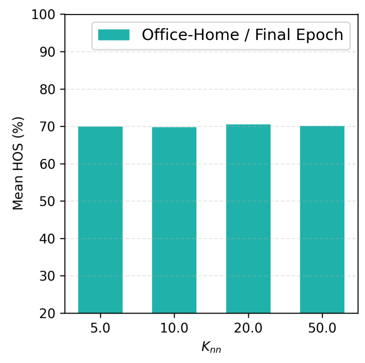
(c) Office-31: α



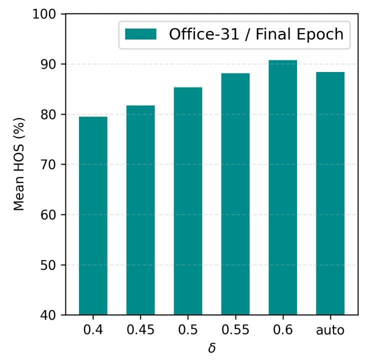
(d) Office-Home: α



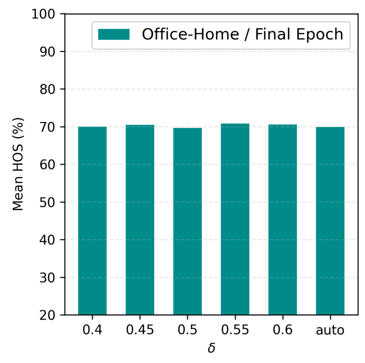
(e) Office-31: K_{nn}



(f) Office-Home: K_{nn}

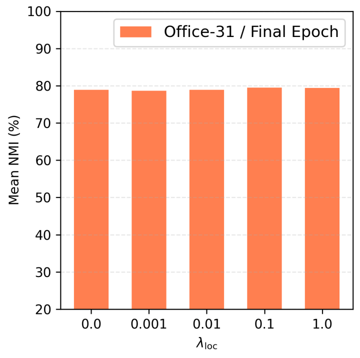


(g) Office-31: δ

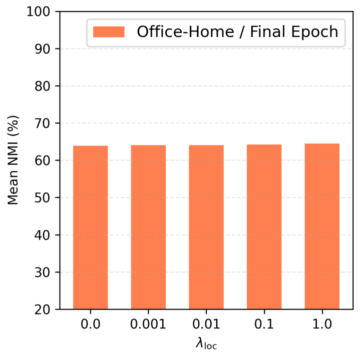


(h) Office-Home: δ

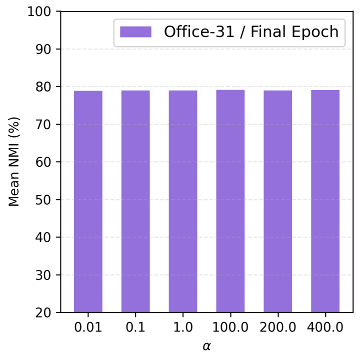
Figure A4: Hyperparameter sensitivity (H-score) of λ_{loc}, α, K_{nn}, and δ on Office-31 and Office-Home under OPDA.



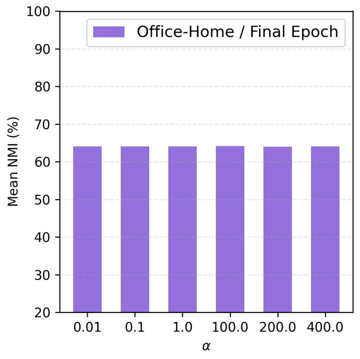
(a) Office-31: λ_{loc}



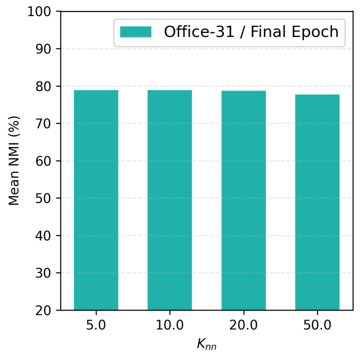
(b) Office-Home: λ_{loc}



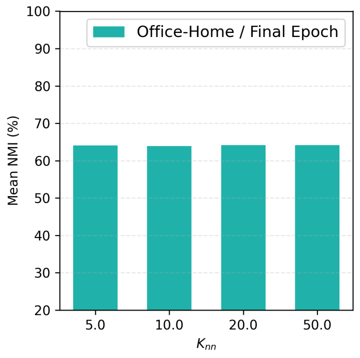
(c) Office-31: α



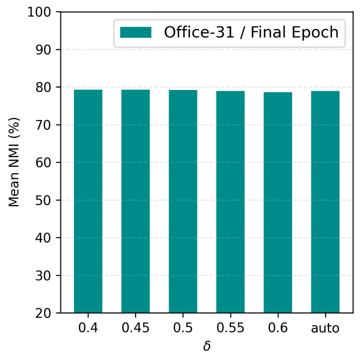
(d) Office-Home: α



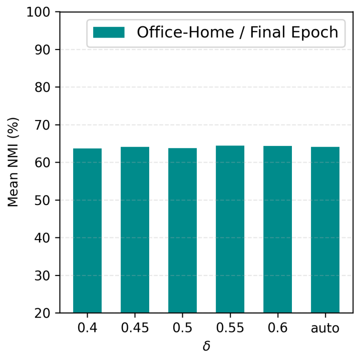
(e) Office-31: K_{nn}



(f) Office-Home: K_{nn}

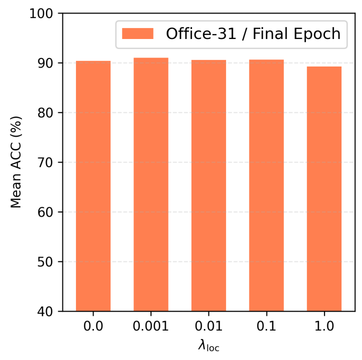


(g) Office-31: δ

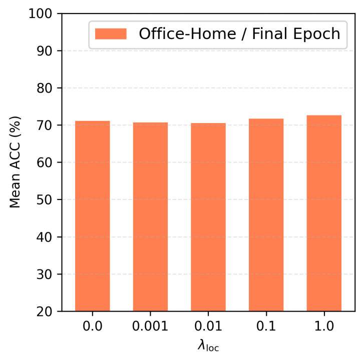


(h) Office-Home: δ

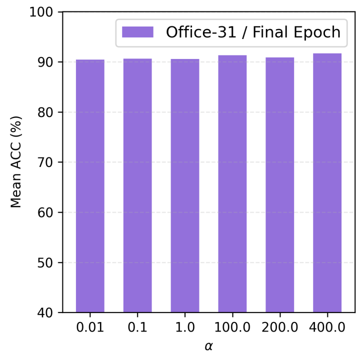
Figure A5: Hyperparameter sensitivity (NMI) of λ_{loc}, α, K_{nn}, and δ on Office-31 and Office-Home under OPDA.



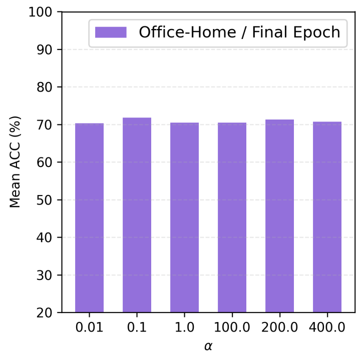
(a) Office-31: λ_{loc}



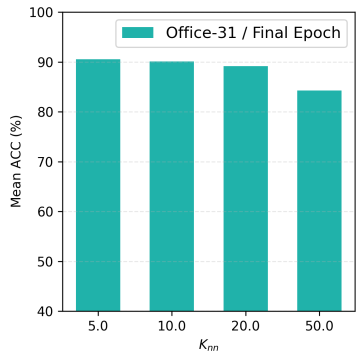
(b) Office-Home: λ_{loc}



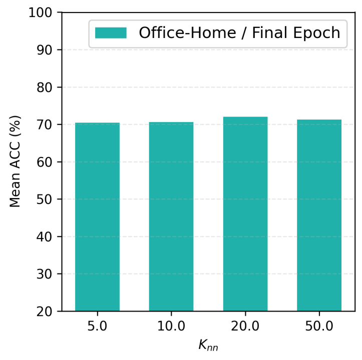
(c) Office-31: α



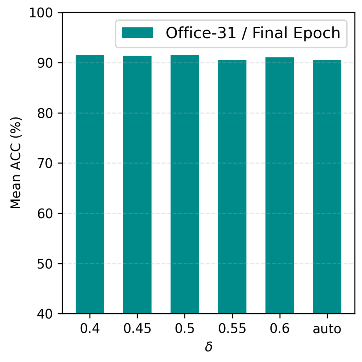
(d) Office-Home: α



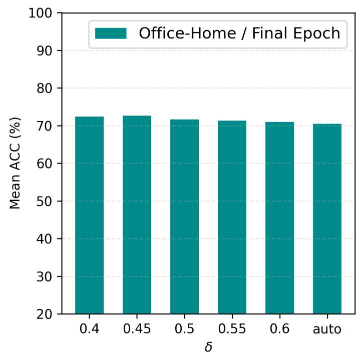
(e) Office-31: K_{nn}



(f) Office-Home: K_{nn}



(g) Office-31: δ



(h) Office-Home: δ

Figure A6: Hyperparameter sensitivity (mean accuracy) of λ_{loc}, α, K_{nn}, and δ on Office-31 and Office-Home under OPDA.