Unsupervised Learning under Latent Label Shift

Manley Roberts* **Pranav Mani*** **Saurabh Garg** **Zachary C. Lipton**
Carnegie Mellon University
{manleyroberts,zlipton}@cmu.edu; {pmani, sgarg2}@cs.cmu.edu

Abstract

What sorts of structure might enable a learner to discover classes from unlabeled data? Traditional approaches rely on feature-space similarity and heroic assumptions on the data. In this paper, we introduce unsupervised learning under *Latent Label Shift* (LLS), where we have access to unlabeled data from multiple domains such that the label marginals $p_d(y)$ can shift across domains but the class conditionals $p(\mathbf{x}|y)$ do not. This work instantiates a new principle for identifying classes: elements that shift together group together. For finite input spaces, we establish an isomorphism between LLS and topic modeling: inputs correspond to words, domains to documents, and labels to topics. Addressing continuous data, we prove that when each label's support contains a separable region, analogous to an anchor word, oracle access to $p(d|\mathbf{x})$ suffices to identify $p_d(y)$ and $p_d(y|\mathbf{x})$ up to permutation. Thus motivated, we introduce a practical algorithm that leverages domain-discriminative models as follows: (i) push examples through domain discriminator $p(d|\mathbf{x})$; (ii) discretize the data by clustering examples in $p(d|\mathbf{x})$ space; (iii) perform non-negative matrix factorization on the discrete data; (iv) combine the recovered $p(y|d)$ with the discriminator outputs $p(d|\mathbf{x})$ to compute $p_d(y|x)\ \forall d$. With semi-synthetic experiments, we show that our algorithm can leverage domain information to improve state of the art unsupervised classification methods. We reveal a failure mode of standard unsupervised classification methods when data-space similarity does not indicate true groupings, and show empirically that our method better handles this case. Our results establish a deep connection between distribution shift and topic modeling, opening promising lines for future work[2].

1 Introduction

Discovering systems of categories from unlabeled data is a fundamental but ill-posed challenge in machine learning. Typical unsupervised learning methods group instances together based on feature-space similarity. Accordingly, given a collection of photographs of animals, a practitioner might hope that, in some appropriate feature space, images of animals of the same species should be somehow similar to each other. But why should we expect a clustering algorithm to recognize that dogs viewed in sunlight and dogs viewed at night belong to the same category? Why should we expect that butterflies and caterpillars should lie close together in feature space?

In this paper, we offer an alternative principle according to which we might identify a set of classes: we exploit distribution shift across times and locations to reveal otherwise unrecognizable groupings among examples. For example, if we noticed that whenever we found ourselves in a location where butterflies are abundant, caterpillars were similarly abundant, and that whenever butterflies were scarce, caterpillars had a similar drop in prevalence, we might conclude that the two were tied to the same underlying concept, no matter how different they appear in feature space. In short, our principle suggests that latent classes might be uncovered whenever *instances that shift together group together*.

*Equal contribution.

[2]Code is available at https://github.com/acmi-lab/Latent-Label-Shift-DDFA.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

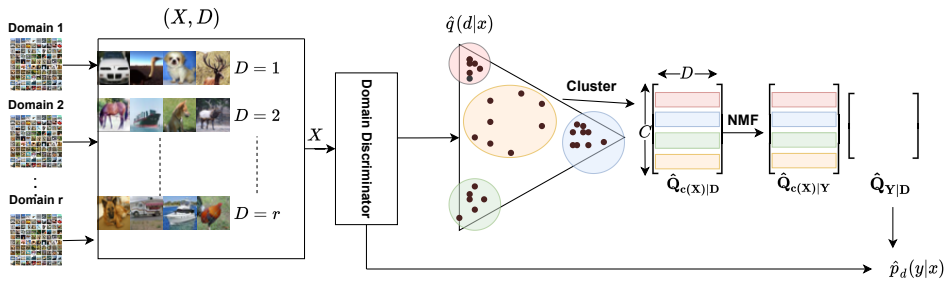

Figure 1: **Schematic of our DDFA algorithm**. After training a domain discriminator, we (i) push all data through the discriminator; (ii) cluster the data based on discriminator outputs; (iii) solve the resulting discrete topic modeling problem and then combine $\widehat{q}(d|x)$ and $\widehat{q}(y, d)$ to estimate $\widehat{p}_d(y|x)$.

Formalizing this intuition, we introduce the problem of unsupervised learning under *Latent Label Shift* (LLS). Here, we assume access to a collection of domains $d \in \{1, \ldots, r\}$, where the mixture proportions $p_d(y)$ vary across domains but the class conditional distribution $p(x|y)$ is domain-invariant. Our goals are to recover the underlying classes up to permutation, and thus to identify both the per-domain mixture proportions $p_d(y)$ and optimally adapted per-domain classifiers $p_d(y|x)$. The essential feature of our setup is that only the true y's, as characterized by their class-conditional distributions $p(x|y)$, could account for the observed shifts in $p_d(x)$. We prove that under mild assumptions, knowledge of this underlying structure is sufficient for inducing the full set of categories.

First, we focus on the *tabular setting*, demonstrating that when the input space is discrete and finite, LLS is isomorphic to topic modeling [8]. Here, each distinct input x maps to a *word* each *latent label* y maps to a *topic* and each domain d maps to a document. In this case, we can apply standard identification results for topic modeling [20, 4, 27, 33, 12] that rely only on the existence of anchor words within each topic (i.e., for each label y there is at least one x in the support of y, that is not in the support of any $y' \neq y$). Here, standard methods based on Non-negative Matrix Factorization (NMF) can recover each domain's underlying mixture proportion $p_d(y)$ and optimal predictor $p_d(y|x)$ [20, 33, 27]. However, the restriction to discrete inputs, while appropriate for topic modeling, proves restrictive when our interests extend to high-dimensional continuous input spaces.

Then, to handle high-dimensional inputs, we propose *Discriminate-Discretize-Factorize-Adjust (DDFA)*, a general framework that proceeds in the following steps: (i) pool data from all domains to produce a mixture distribution $q(x, d)$; (ii) train a domain discriminative model f to predict $q(d|x)$; (iii) push all data through f, cluster examples in the pushforward distribution, and tabularize the data based on cluster membership; (iv) solve the resulting discrete topic modeling problem (e.g., via NMF), estimating $q(y, d)$ up to permutation of the latent labels; (v) combine the predicted $q(d|x)$ and $q(y, d)$ to estimate $p_d(y)$ and $p_d(y|x)$. In developing this approach, we draw inspiration from recent works on distribution shift and learning from positive and unlabeled data that (i) leverage black box predictors to perform dimensionality reduction [40, 23, 24]; and (ii) work with *anchor sets*, separable subsets of continuous input spaces that belong to only one class's support [52, 41, 21, 6, 24].

Our key theoretical result shows that domain discrimination ($q(d|x)$) provides a sufficient representation for identifying all parameters of interest. Given oracle access to $q(d|x)$ (which is identified without labels), our procedure is consistent. Our analysis reveals that the true $q(d|x)$ maps all points in the same anchor set to a single point mass in the push-forward distribution. This motivates our practical approach of discretizing data by hunting for tight clusters in $q(d|x)$ space.

In semi-synthetic experiments, we adapt existing image classification benchmarks to the LLS setting, sampling without replacement to construct collections of label-shifted domains. We note that training a domain discriminative classifier is a difficult task, and find that warm starting the initial layers of our model with pretrained weights from unsupervised approaches can significantly boost performance. We show that warm-started DDFA outperforms state-of-the-art (SOTA) unsupervised approaches when domain marginals $p_d(y)$ are sufficiently sparse. In particular, we observe improvements of as much as 30% accuracy over unsupervised SOTA on CIFAR-20. Further, on subsets of FieldGuide dataset, where similarity between species and diversity within a species leads to failure of unsupervised learning, we show that DDFA recovers the true distinctions. To be clear, these are not apples-to-

apples comparisons: our methods are specifically tailored to the LLS setting. The takeaway is that the structure of the *LLS* setting can be exploited to outperform the best unsupervised learning heuristics.

2 Related Work

Unsupervised Learning Standard unsupervised learning approaches for discovering labels often rely on similarity in the original data space [42, 50]. While distances in feature space become meaningless for high-dimensional data, deep learning researchers have turned to similarity in a representation space learned via self-supervised contrastive tasks [44, 19, 26, 11], or similarity in a feature space learned end-to-end for a clustering task [9, 10, 47, 57]. Our problem setup closely resembles independent component analysis (ICA), where one seeks to identify statistically independent signal components from mixtures [34]. However, ICA's assumption of statistical independence among the components does not generally hold in our setup. In topic modeling [8, 4, 33, 12, 46], documents are modeled as mixtures of topics, and topics as categorical distributions over a finite vocabulary. Early topic models include the well-known Latent Dirichlet Allocation (LDA) [8], which assumes that topic mixing coefficients are drawn from a Dirichlet distribution, along with papers with more relaxed assumptions on the distribution of topic mixing coefficients (e.g., pLSI) [32, 46]. The topic modeling literature often draws on Non-negative Matrix Factorization (NMF) methods [45, 53], which decompose a given matrix into a product of two matrices with non-negative elements [18, 16, 25, 28]. In both Topic Modeling and NMF, a fundamental problem has been to characterize the precise conditions under which the system is uniquely identifiable [20, 4, 33, 12]. The anchor condition (also referred to as separability) is known to be instrumental for identifying topic models [4, 12, 33, 20]. In this work, we extend these ideas, leveraging separable subsets of each label's support (the anchor sets) to produce anchor words in the discretized problem. Existing methods have attempted to extend latent variable modeling to continuous input domains by making assumptions about the functional forms of the class-conditional densities, e.g., restricting to Gaussian mixtures [50, 49]. A second line of approach involves finding an appropriate discretization of the continuous space [56].

Distribution Shift under the Label Shift Assumption The label shift assumption, where $p_d(y)$ can vary but $p(x|y)$ cannot, has been extensively studied in the domain adaptation literature [51, 54, 60, 40, 29, 23] and also features in the problem of learning from positive and unlabeled data [22, 7, 24]. For both problems, many classical approaches suffer from the curse of dimensionality, failing in the settings where deep learning prevails. Our solution strategy draws inspiration from recent work on label shift [40, 1, 5, 23] and PU learning [7, 41, 52, 24] that leverage black-box predictors to produce sufficient low-dimensional representations for identifying target distributions of interest (other works leverage black box predictors heuristically [36]). **Key differences:** While PU learning requires identifying *one* new class for which we lack labeled examples provided that the positive class contains an anchor set [24], LLS can identify an arbitrary number of classes (up to permutation) from completely unlabeled data, provided a sufficient number of domains.

Domain Generalization The related problem of Domain Generalization (DG) also addresses learning with data drawn from multiple distributions and where the domain identifiers play a key role [43, 2]. However in DG, we are given *labeled* data from multiple domains, and our goal is to learn a classifier that can generalize to new domains. By contrast, in LLS, we work with unlabeled data only, leveraging the problem structure to identify the underlying labels.

3 Latent Label Shift Setup

Notation For a vector $v \in \mathbb{R}^p$, we use v_j to denote its jth entry, and for an event E, we let $\mathbb{I}[E]$ denote the binary indicator of the event. By $|A|$, we denote the cardinality of set A. With $[n]$, we denote the set $\{1, 2, \ldots, n\}$. We use $[A]_{i,j}$ to access the element at (i, j) in A. Let $\mathcal{X}$ be the input space and $\mathcal{Y} = \{1, 2, \ldots, k\}$ be the output space for multiclass classification. We assume throughout this work that the number of true classes k is known. Throughout this paper, we use capital letters to denote random variables and small case letters to denote the corresponding values they take. For example, by X we denote the input random variable and by x, we denote a value that X may take.

We now formally introduce the problem of unsupervised learning under LLS. In LLS, we assume that we observe unlabeled data from r domains. Let $\mathcal{R} = \{1, 2, \ldots, r\}$ be the set of domains. By p_d, we denote the probability density (or mass) function for each domain $d \in \mathcal{R}$.

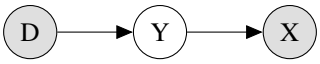

Figure 2: Relationship under Q between observed D, observed X, and latent Y.

Definition 1 (Latent label shift)**.** *We observe data from r domains. While the label distribution can differ across the domains, for all $d, d' \in \mathcal{R}$ and for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$, we have $p_d(x|y) = p_{d'}(x|y)$.*

Simply put, Definition 1 states that the conditional distribution $p_d(x|y)$ remains invariant across domains, i.e., they satisfy the label shift assumption. Thus, we can drop the subscript on this factor, denoting all $p_d(x|y)$ by $p(x|y)$. Crucially, under LLS, $p_d(y)$ can vary across different domains. Under LLS, we observe unlabeled data with domain label $\{(x_1, d_1), (x_2, d_2), \ldots, (x_n, d_n)\}$. Our goal breaks down into two tasks. Up to permutation of labels, we aim to (i) estimate the label marginal in each domain $p_d(y)$; and (ii) estimate the optimal per-domain predictor $p_d(y|x)$.

Mixing distribution Q A key step in our algorithm will be to train a domain discriminative model. Towards this end we define Q, a distribution over $\mathcal{X} \times \mathcal{Y} \times \mathcal{R}$, constructed by taking a uniform mixture over all domains. By q, we denote the probability density (or mass) function of Q. Define Q such that $q(x, y|D = d) = p_d(x, y)$, i.e., when we condition on $D = d$ we recover the joint distribution over $\mathcal{X} \times \mathcal{Y}$ specific to that domain d. For all $d \in \mathcal{R}$, we define $\gamma_d = q(d)$, i.e., the prevalence of each domain in our distribution Q. Notice that $q(x, y)$ is a mixture over the distributions $\{p_d(x, y)\}_{d \in \mathcal{R}}$, with $\{\gamma_d\}_{d \in \mathcal{R}}$ as the corresponding mixture coefficients. Under LLS (Definition 1), X does not depend on D when conditioned on Y (Fig. 2).

Additional notation for the discrete case To begin, we setup notation for discrete input spaces with $|\mathcal{X}| = m$. Without loss of generality, we assume that $\mathcal{X} = \{1, 2, \ldots, m\}$. The label shift assumption allows us to formulate the label marginal estimation problem in matrix form. Let $\mathbf{Q}_{X|D}$ be an $m \times r$ matrix such that $[\mathbf{Q}_{X|D}]_{i,d} = p_d(X = i)$, i.e., the d-th column of $\mathbf{Q}_{X|D}$ is $p_d(x)$. Let $\mathbf{Q}_{X|Y}$ be an $m \times k$ matrix such that $[\mathbf{Q}_{X|Y}]_{i,j} = p(X = i|Y = j)$, the j-th column is a distribution over X given $Y = j$. Similarly, define $\mathbf{Q}_{Y|D}$ as a $k \times r$ matrix whose d-th column is the domain marginal $p_d(y)$. Now with Definition 1, we have $p_d(x) = \sum_y p_d(x, y) = \sum_y p_d(x|y)p_d(y) = \sum_y p(x|y)p_d(y)$. Since this is true $\forall d \in \mathcal{R}$, we can express this in a matrix form as $\mathbf{Q}_{X|D} = \mathbf{Q}_{X|Y}\mathbf{Q}_{Y|D}$.

Additional assumptions Before we present identifiability results for the LLS problem, we introduce four additional assumptions required throughout the paper:

- A.1 There are at least as many domains as classes, i.e., $|\mathcal{R}| \geqslant |\mathcal{Y}|$.
- A.2 The matrix formed by label marginals (as columns) across different domains is full-rank, i.e., $\text{rank}(\mathbf{Q}_{Y|D}) = k$.
- A.3 Equal representation of domains, i.e., for all $d \in \mathcal{R}, \gamma_d = 1/r$.
- A.4 Fix $\epsilon > 0$. For all $y \in \mathcal{Y}$, there exists a unique subdomain $A_y \subseteq \mathcal{X}$, such that $q(A_y) \geqslant \epsilon$ and $x \in A_y$ if and only if the following conditions are satisfied: $q(x|y) > 0$ and $q(x|y') = 0$ for all $y' \in \mathcal{Y} \backslash \{y\}$. We refer to this assumption as the ϵ-anchor sub-domain condition.

We now comment on the assumptions. A.1–A.2 are benign, these assumptions just imply that the matrix $\mathbf{Q}_{Y|D}$ is full row rank. Without loss of generality, A.3 can be assumed when dealing with data from a collection of domains. When this condition is not satisfied, one could just re-sample data points uniformly at random from each domain d. Intuitively, A.4 states that for each label $y \in \mathcal{Y}$, we have some subset of inputs that only belong to that class y. To avoid vanishing probability of this subset, we ensure at least ϵ probability mass in our mixing distribution Q. The anchor word condition is related to the positive sub-domain in PU learning, which requires that there exists a subset of $\mathcal{X}$ in which all examples only belong to the positive class [52, 41, 21, 6].

4 Theoretical Analysis

In this section, we establish identifiability of LLS problem. We begin by considering the case where the input space is discrete and formalize the isomorphism to topic modeling. Then we establish the identifiability of the system in this discrete setting by appealing to existing results in topic

modeling [33]. Finally, extending results from discrete case, we provide novel analysis to establish our identifiability result for the continuous setting.

Isomorphism to topic modeling Recall that for the discrete input setting, we have the matrix formulation: $\mathbf{Q}_{X|D} = \mathbf{Q}_{X|Y}\mathbf{Q}_{Y|D}$. Consider a corpus of r documents, consisting of terms from a vocabulary of size m. Let $\mathbf{D}$ be an $\mathbb{R}^{m\times r}$ matrix representing the underlying corpus. Each column of $\mathbf{D}$ represents a document, and each row represents a term in the vocabulary. Each element $[\mathbf{D}]_{i,j}$ represents the frequency of term i in document j. Topic modeling [8, 32, 33, 4] considers each document to be composed as a mixture of k topics. Each topic prescribes a frequency with which the terms in the vocabulary occur given that topic. Further, the proportion of each topic varies across documents with the frequency of terms given topic remaining invariant.

We can state the topic modeling problem as: $\mathbf{D} = \mathbf{CW}$, where $\mathbf{C}$ is an $\mathbb{R}^{m\times k}$ matrix, $[\mathbf{C}]_{i,j}$ represents the frequency of term i given topic j, and $\mathbf{W}$ is an $\mathbb{R}^{k\times r}$ matrix, where $[\mathbf{W}]_{i,j}$ represents the proportion of topic i in document j. Note that all three matrices are column normalized. The isomorphism is then between document and domain, topic and label, term and input sample, i.e., $\mathbf{D} = \mathbf{CW} \equiv \mathbf{Q}_{X|D} = \mathbf{Q}_{X|Y}\mathbf{Q}_{Y|D}$. In both the cases, we are interested in decomposing a known matrix into two unknown matrices. This formulation is examined as a non-negative matrix factorization problem with an added simplicial constraint on the columns (columns sum to 1) [3, 27].

Identifiability of the topic modeling problem is well-established [20, 4, 27, 33, 12]. We leverage the isomorphism to topic modeling to extend this identifiability condition to our LLS setting. We formalize the adaptation here:

Theorem 1. *(adapted from Proposition 1 in Huang et al. [33]) Assume A.1, A.2 and A.4 hold (A.4 in the discrete setting is referred to as the anchor word condition). Then the solution to $\mathbf{Q}_{X|D} = \mathbf{Q}_{X|Y}\mathbf{Q}_{Y|D}$ is uniquely identified up to permutation of class labels.*

We refer readers to Huang et al. [33] for a proof of this theorem. Intuitively, Theorem 1 states that if each label y has at least one token in the input space that has support only in y, and A.1, A.2 hold, then the solution to $\mathbf{Q}_{X|Y}$, $\mathbf{Q}_{Y|D}$ is unique. Furthermore, under this condition, there exist algorithms that can recover $\mathbf{Q}_{X|Y}$, $\mathbf{Q}_{Y|D}$ within some permutation of class labels [33, 27, 3, 4].

Extensions to the continuous case We will prove identifiability in the continuous setting, when $\mathcal{X} = \mathbb{R}^p$ for some $p \geqslant 1$. In addition to A.1–A.4, we make an additional assumption that we have oracle access to $q(d|x)$, i.e., the true domain discriminator for mixture distribution Q. This is implied by assuming access to the marginal $q(x, d)$ from which we observe our samples. Formally, we define a push forward function f such that $[f(x)]_d = q(d|x)$, then push the data forward through f to obtain outputs in Δ^{r-1}. In the proof of Theorem 2, we will show that these outputs can be discretized in a fashion that maps anchor subdomains to anchor words in a tabular, discrete setting. We separately remark that the anchor word outputs are in fact extreme corners of the convex polytope in Δ^{r-1} which encloses all $f(x)$ mass; we discuss this geometry further in App. F. After constructing the anchor word discretization, we appeal to Theorem 1 to recover $\mathbf{Q}_{Y|D}$. Given $\mathbf{Q}_{Y|D}$, we show that we can use Bayes' rule and the LLS condition (Definition 1) to identify the distribution $q(y|x, d) = p_d(y|x)$ over latent variable y. We formalize this in the following theorem:

Theorem 2. *Let the distribution Q over random variables X, Y, D satisfy Assumptions A.1–A.4. Assuming access to the joint distribution $q(x, d)$, and knowledge of the number of true classes k, we show that the following quantities are identifiable: (i) $\mathbf{Q}_{Y|D}$, (ii) $q(y|X = x)$, for all $x \in \mathcal{X}$ that lies in the support (i.e. $q(x) > 0$); and (iii) $q(y|X = x, D = d)$, for all $x \in \mathcal{X}$ and $d \in \mathcal{R}$ such that $q(x, d) > 0$.*

Before presenting a proof sketch for Theorem 2, we first present key lemmas (we include their proofs in App. A).

Lemma 1. *Under the same assumptions as Theorem 2, the matrix $\mathbf{Q}_{Y|D}$ and $f(x) = q(d|x)$ uniquely determine $q(y|x)$ for all $y \in \mathcal{Y}$ and $x \in \mathcal{X}$ such that $q(x) > 0$.*

Lemma 1 states that given matrix $\mathbf{Q}_{Y|D}$ and oracle domain discriminator, we can uniquely identify $q(y|x)$. In particular, we show that for any $x \in \mathcal{X}$, $q(d|x)$ can be expressed as a convex combination of the k columns of $\mathbf{Q}_{D|Y}$ (which is computed from $\mathbf{Q}_{Y|D}$ and is column rank k) and the coefficients of the combination are $q(y|x)$. Combining this with the linear independence of the columns of $\mathbf{Q}_{D|Y}$, we show that these coefficients are unique. In the following lemma, we show how the identified $q(y|x)$ can then be used to identify $q(y|x, d)$:

Lemma 2. *Under the same assumptions as Theorem 2, for all $y \in \mathcal{Y}$, and $x \in \mathcal{X}$ such that $q(x, d) > 0$. the matrix $\mathbf{Q}_{Y|D}$ and $q(y|x)$ uniquely determine $q(y|x, d)$.*

To prove Lemma 2, we show that we can combine the conditional distribution over the labels given a sample $x \in \mathcal{X}$ with the prior distribution of the labels in each domain to determine the posterior distribution over labels given the sample x and the domain of interest. Next, we introduce a key property of the domain discriminator classifier f:

Lemma 3. *Under the same assumptions as Theorem 2, for all x, x' in anchor sub-domain, i.e., $x, x' \in A_y$ for a given label $y \in \mathcal{Y}$, we have $f(x) = f(x')$. Further, for any $y \in \mathcal{Y}$, if $x \in A_y, x' \notin A_y$, then $f(x) \neq f(x')$.*

Lemma 3 implies that the oracle domain discriminator f maps all points in an anchor subdomain, and only those points in that anchor subdomain to the same point in $f(x) = q(d|x)$ space. We can now present a proof sketch for Theorem 2 (full proof in App. B):

Proof sketch of Theorem 2. The key idea of the proof lies in proposing a discretization such that some subset of anchor subdomains for each label y in the continuous space map to distinct anchor words in discrete space. In particular, if there exists a discretization of the continuous space $\mathcal{X}$ that for any $y \in \mathcal{Y}$, maps all $x \in A_y$ to the same point in the discrete space, but no $x \notin A_y$ maps to this point, then this point serves as an anchor word. From Lemma 3, we know that all the $x \in A_y$ and only the $x \in A_y$ get mapped to specific points in the $f(x)$ space. Pushing all the $x \in \mathcal{X}$ through f, we know from A.4 that there exists k point masses of size ϵ, one for each $f(A_y)$ in the $f(x) = q(d|x)$ space. We can now inspect this space for point masses of size at least ϵ to find at most $\mathcal{O}(1/\epsilon)$ such point masses among which are contained the k point masses corresponding to the anchor subdomains. Discretizing this space by assigning each point mass to a group (and non-point masses to a single additional group), we have k groups that have support only in one y each. Thus, we have achieved a discretization with anchor words. Further, since the discrete space arises from a pushforward of the continuous space through f, the discrete space also satisfies the latent label shift assumption A.1. We now use Theorem 1 to claim identifiability of $\mathbf{Q}_{Y|D}$. We then use Lemmas 1 and 2 to prove parts (ii) and (iii).

5 DDFA Framework

Motivated by our identifiability analysis, in this section, we present an algorithm to estimate $\mathbf{Q}_{Y|D}, q(y|x)$, and $q(y|x, d)$ when X is continuous by exploiting domain structure and approximating the true domain discriminator f. Intuitively, $q(y|x, d)$ is the domain specific classifier $p_d(y|x)$ and $q(y|x)$ is the classifier for data from aggregated domains. $\mathbf{Q}_{Y|D}$ captures label marginal for individual domains. A naive approach would be to aggregate data from different domains and exploit recent advancements in unsupervised learning [57, 47, 9, 10]. However, aggregating data from multiple domains loses the domain structure that we hope to leverage. We highlight this failure mode of the traditional unsupervised clustering method in Sec. 6. We remark that DDFA draws heavy inspiration from the proof of Theorem 2, but we do not present a guarantee that the DDFA solution will converge to the identifiable solution. This is primarily due to the K-means clustering heuristic we rely on, which empirically offers effective noise tolerance, but theoretically has no guarantee of correct convergence.

Discriminate We begin Algorithm 1 by creating a split of the unlabeled samples into the training and validation sets. Using the unlabeled data samples and the domain that each sample originated from, we first train a domain discriminative classifier $\widehat{f}$. The domain discriminative classifier outputs a distribution over domains for a given input. This classifier is trained with cross-entropy loss to predict the domain label of each sample on the training set. With unlimited data, the minimizer of this loss is the true f, as we prove in App. C. To avoid overfitting, we stop training $\widehat{f}$ when the cross-entropy loss on the validation set stops decreasing. Note that here the validation set also only contains domain indices (and no information about true class labels).

Discretize We now push forward all the samples from the training and validation sets through the domain discriminator to get vector $\widehat{f}(x_i)$ for each sample x_i. In the proof of Theorem 2, we argue that when working with true f, and the entire marginal $q(x, d)$, we can choose a discretization satisfying the anchor word assumption by identifying point masses in the distribution of $f(x)$ and filtering to include those of at least ϵ mass. In the practical setting, because we have only a finite set of data points and a noisy $\widehat{f}$, we use clustering to approximately find point masses. We choose $m \geqslant k$ and

Algorithm 1 DDFA Training

input $k \geqslant 1, r \geqslant k, \{(x_i, d_i)\}_{i\in[n]} \sim q(x, d),$ A class of functions $\mathcal{F}$ from $\mathbb{R}^p \to \mathbb{R}^r$

1: Split into train set T and validation set V

2: Train $\widehat{f} \in \mathcal{F}$ to minimize cross entropy loss for predicting $d|x$ on T with early stopping on V

3: Push all $\{x_i\}_{i\in[n]}$ through $\widehat{f}$

4: Train clustering algorithm on the n points $\{\widehat{f}(x_i)\}_{i\in[n]},$ obtain m clusters.

5: $c(x_i) \leftarrow$ Cluster id of $\widehat{f}(x_i)$

6: $\widehat{q}(c(X) = a|D = b) \leftarrow \frac{\sum_{i\in[n]} \mathbb{I}[c(x_i)=a,\ d_i=b]}{\sum_{j\in[n]} \mathbb{I}[d_j=b]}$

7: Populate $\widehat{\mathbf{Q}}_{c(X)|D}$ as $[\widehat{\mathbf{Q}}_{c(X)|D}]_{a,b} \leftarrow \widehat{q}(c(X) = a|D = b)$

8: $\widehat{\mathbf{Q}}_{c(X)|Y}, \widehat{\mathbf{Q}}_{Y|D} \leftarrow$ NMF $(\widehat{\mathbf{Q}}_{c(X)|D})$

output $\widehat{\mathbf{Q}}_{Y|D}, \widehat{f}$

recover m clusters with any standard clustering procedure (e.g. K-means). This clustering procedure is effectively a useful, but imperfect heuristic: if the noise in $\widehat{f}$ is sufficiently small and the clustering sufficiently granular, we hope that our m discovered clusters will include k *pure* clusters, each of which only contains data points from a different anchor subdomain which are tightly packed around the true $f(A_y)$ for the corresponding label y. Clustering in this space is superior to a naive clustering on the input space because close proximity in this space indicates similarity in $q(d|x)$.

Let us denote the learned clustering function as c, where $c(x)$ is the cluster assigned to a datapoint x. We now leverage the cluster id $c(x_i)$ of each sample x_i to discretize samples into a finite discrete space $[m]$. Combining the cluster id with the domain source d_i for each sample, we estimate $\widehat{\mathbf{Q}}_{c(X)|D}$ by simply computing, for each domain, the fraction of its samples assigned to each cluster.

Factorize We apply an NMF algorithm to $\widehat{\mathbf{Q}}_{c(X)|D}$ to obtain estimates of $\widehat{\mathbf{Q}}_{c(X)|Y}$ and $\widehat{\mathbf{Q}}_{Y|D}$.

Adjust We begin Algorithm 2 by considering a test point (x', d'). To make a prediction, if we had access to oracle f and true $\mathbf{Q}_{Y|D}$, we could precisely compute $q(y|x')$ (Lemma 1). However, in place of these true quantities, we plug in the estimates $\widehat{f}$ and $\widehat{\mathbf{Q}}_{Y|D}$. Since our estimates contain noise, the estimate $\widehat{q}(y|x')$ is found by left-multiplying $\widehat{f}(x')$ with the pseudo-inverse of $\widehat{\mathbf{Q}}_{D|Y}$, as opposed to solving a sufficient system of equations. As our estimates $\widehat{f}$ and $\widehat{\mathbf{Q}}_{D|Y}$ approach the true values, the projection of $\widehat{f}(x')$ into the column space of $\widehat{\mathbf{Q}}_{D|Y}$ tends to $\widehat{f}(x')$ itself, so the pseudo-inverse approaches the true solution. Now we can use the constructive procedure introduced in the proof of Lemma 2 to compute the plug-in estimate $\widehat{q}(y|x', d') = \widehat{p}_{d'}(y|x')$.

Algorithm 2 DDFA Prediction

input $\widehat{\mathbf{Q}}_{Y|D}, \widehat{f}, (x', d') \sim q(x, d)$

1: Populate $\widehat{\mathbf{Q}}_{D|Y}$ as $[\widehat{\mathbf{Q}}_{D|Y}]_{d,y} \leftarrow \frac{[\widehat{\mathbf{Q}}_{Y|D}]_{y,d}}{\sum_{d''=1}^{d''=r} [\widehat{\mathbf{Q}}_{Y|D}]_{y,d''}}$

2: Assign $\widehat{q}(y|X = x') \leftarrow \left[\left(\widehat{\mathbf{Q}}_{D|Y}\right)^{\dagger} \widehat{f}(x')\right]_y$

3: Assign $\widehat{q}(y|X = x', D = d') \leftarrow \dfrac{[\widehat{\mathbf{Q}}_{D|Y}]_{d',y} \widehat{q}(y|X = x')}{\sum\limits_{y''\in[k]} [\widehat{\mathbf{Q}}_{D|Y}]_{d',y''} \widehat{q}(y''|X = x')}$

4: $y_{\text{pred}} \leftarrow \arg\max_{y\in[k]} \widehat{q}(y|X = x', D = d')$

output : $\widehat{q}(y|X = x', D = d') = \widehat{p}_{d'}(y|x'), \widehat{q}(y|X = x'), y_{\text{pred}}$

6 Experiments

Experiment code is available at `https://github.com/acmi-lab/Latent-Label-Shift-DDFA`.

Baselines We select the unsupervised classification method SCAN as a state-of-the-art baseline [57]. SCAN pretrains a ResNet [30] backbone using SimCLR [11] and MoCo [31] setups (pretext tasks).

SCAN then trains a clustering head to minimize the SCAN loss (refer [57] for more details) [3]. We make sure to evaluate SCAN on the same potentially class-imbalanced test subset we create for each experiment. Since SCAN is fit on a superset of the data DDFA sees, we believe this gives a slight data advantage to the SCAN baseline (although we acknowledge that the class balance for SCAN training is also potentially different from its evaluation class balance). To evaluate SCAN, we use the public pretrained weights available for CIFAR-10, CIFAR-20, and ImageNet-50. We also train SCAN ourselves on the train and validation portions of the FieldGuide2 and FieldGuide28 datasets with a ResNet18 backbone and SimCLR pretext task. We replicate the hyperparameters used for CIFAR training.

Datasets First we examine standard multiclass image datasets CIFAR-10, CIFAR-20 [38], and ImageNet-50 [17] containing images from 10, 20, and 50 classes respectively. Images in these datasets typically focus on a single large object which dominates the center of the frame, so unsupervised classification methods which respond strongly to similarity in visual space are well-suited to recover true classes up to permutation. These datasets are often believed to be separable (i.e., single true label applies to each image), so every example falls in an anchor subdomain (satisfying A.4).

Motivated by the application of LLS problem, we consider the FieldGuide dataset [4], which contains images of moths and butterflies. The true classes in this dataset are species, but each class contains images taken in immature (caterpillar) and adult stages of life. Based on the intuition that butterflies from a given species look more like butterflies from other species than caterpillars from their own species, we hypothesize that unsupervised classification will learn incorrect class boundaries which distinguish caterpillars from butterflies, as opposed to recovering the true class boundaries. Due to high visual similarity between members of different classes, this dataset may indeed have slight overlap between classes. However, we hypothesize that anchor subdomain still holds, i.e., there exist some images from each class that could only come from that class. Additionally, if we have access to data from multiple domains, it is natural to assume that within each domain the relative distribution of caterpillar to adult stages of each species stay relatively constant as compared to prevalence of different species. We create two subsets of this dataset: FieldGuide2, with two species, and FieldGuide28, with 28 species.

LLS Setup The full sampling procedure for semisynthetic experiments is described in App. D. Roughly, we sample $p_d(y)$ from a symmetric Dirichlet distribution with concentration α/k, where k is the number of classes and α is a generation parameter that adjusts the difficulty of the synthetic problem, and enforce maximum condition number κ on $\mathbf{Q}_{\mathbf{Y}|\mathbf{D}}$. Small α and small κ encourages sparsity in $\mathbf{Q}_{\mathbf{Y}|\mathbf{D}}$, so each label tends to only appear in a few domains. Larger parameters encourages $p_d(y)$ to tend toward uniform. We draw from test, train, and valid datasets without replacement to match these distributions, but discard some examples due to class imbalance.

Training and Evaluation The algorithm uses train and validation data consisting of pairs of images and domain indices. We train ResNet50 [30] (with added dropout) on images x_i with domain indices d_i as the label, choose best iteration by valid loss, pass all training and validation data through $\widehat{f}$, and cluster pushforward predictions $\widehat{f}(x_i)$ into $m \geqslant k$ clusters with Faiss K-Means [37]. We compute the $\widehat{\mathbf{Q}}_{c(X)|D}$ matrix and run NMF to obtain $\widehat{\mathbf{Q}}_{c(X)|Y}$, $\widehat{\mathbf{Q}}_{Y|D}$. To make columns sum to 1, we normalize columns of $\widehat{\mathbf{Q}}_{c(X)|Y}$, multiply each column's normalization coefficient over the corresponding row of $\widehat{\mathbf{Q}}_{Y|D}$ (to preserve correctness of the decomposition), and then normalize columns of $\widehat{\mathbf{Q}}_{Y|D}$. Some NMF algorithms only output solutions satisfying the anchor word property [3, 39, 27]. We found the strict requirement of an exact anchor word solution to lead to low noise tolerance. We therefore use the Sklearn implementation of standard NMF [13, 55, 48].

We instantiate the domain discriminator as ResNet18, and preseed its backbone with SCAN [57] pre-trained weights or [57] contrastive pre-text weights. We denote these models DDFA (SI) and DDFA (SPI) respectively. We predict class labels with Algorithm 2. With the Hungarian algorithm, implemented in [14, 58], we compute the highest true accuracy among any permutation of these labels (denoted "Test acc"). With the same permutation, we reorder rows of $\widehat{P}_{Y|D}$, then compute the average absolute difference between corresponding entries of $\widehat{\mathbf{Q}}_{Y|D}$ and $\mathbf{Q}_{Y|D}$ (denoted "$\mathbf{Q}_{Y|D}$ err").

[3] SCAN code: `https://github.com/wvangansbeke/Unsupervised-Classification`

[4] FieldGuide: `https://sites.google.com/view/fgvc6/competitions/butterflies-moths-2019`

Table 1: *Results on CIFAR-20*. Each entry is produced with the averaged result of 3 different random seeds. With DDFA (RI) we refer to DDFA with randomly initialized backbone. With DDFA (SI) we refer to DDFA's backbone initialized with SCAN. Note that in DDFA (SI), we do not leverage SCAN for clustering. α is the Dirichlet parameter used for generating label marginals in each domain, κ is the maximum allowed condition number of the generated $\mathbf{Q}_{Y|D}$ matrix, r is number of domains.

r	Approaches	$\alpha : 0.5,\ \kappa : 8$		$\alpha : 3,\ \kappa : 12$		$\alpha : 10,\ \kappa : 20$	
		Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err
20	SCAN	0.439	0.092	0.446	0.079	**0.434**	0.060
	DDFA (RI)	0.517	0.042	0.336	0.045	0.163	0.057
	DDFA (SI)	**0.784**	**0.023**	**0.593**	**0.027**	0.390	**0.034**
25	SCAN	0.438	0.090	0.441	0.078	0.438	0.060
	DDFA (RI)	0.489	0.049	0.292	0.049	0.075	0.081
	DDFA (SI)	**0.837**	**0.020**	**0.669**	**0.025**	**0.487**	**0.030**
30	SCAN	0.432	0.094	0.457	0.077	0.431	0.059
	DDFA (RI)	0.512	0.046	0.299	0.048	0.087	0.077
	DDFA (SI)	**0.820**	**0.022**	**0.743**	**0.021**	**0.543**	**0.028**

Results On CIFAR-10, we observe that DDFA alone is incapable of matching highly competitive state-of-the-art baseline SCAN performance—however, in suitably sparse problem settings (small α), it comes substantially close, indicating good recovery of true classes. Due to space constraints, we include CIFAR-10 results in App. E. DDFA (SI) combines SCAN's strong pretrain with domain discrimination fine-tuning to outperform SCAN in the easiest, sparsest setting and *certain* denser settings. On CIFAR-20, baseline SCAN is much less competitive, so our DDFA(SI) dominates baseline SCAN in all settings except the densest (Table 1). These results demonstrate how adding domain information can dramatically boost unsupervised baselines.

On FieldGuide-2, DDFA (SPI) outperforms SCAN baselines across all problem settings and domain counts (Table 5); in sparser settings, the accuracy gap is 20-30%. In this dataset, SCAN performs only slightly above chance, reflecting perhaps a total misalignment of learned class distinctions with true species boundaries. We do not believe that SCAN is too weak to effectively detect image groupings on this data; instead we acknowledge that the domain information available to DDFA (SPI) (and not to SCAN) is informative for ensuring recovery of the true class distinction between species as opposed to the visually striking distinction between adult and immature life stages. Results from more domains are available in App. E. On FieldGuide-28 (Table 6), DDFA outperforms SCAN when $\mathbf{Q}_{Y|D}$ is sufficiently sparse (sampled with $\alpha : 0.5$ or $\alpha : 3$), with the highest observed accuracy difference ranging above 30-40%.

7 Conclusion

Our theoretical results demonstrate that under LLS, we can leverage shifts among previously seen domains to recover labels in a purely unsupervised manner, and our practical instantiation of the DDFA framework demonstrates both (i) the practical efficacy of our approach; and (ii) that generic unsupervised methods can play a key role both in clustering discriminator outputs, and providing weights for initializing the discriminator. We believe that this work is just the first step in a new direction for leveraging structural assumptions together with distribution shift to perform unsupervised learning.

Within the LLS setup, several components of the DDFA framework warrant further investigation: (i) the deep domain discriminator can be enhanced in myriad ways; (ii) for clustering discriminator outputs, we might develop methods specially tailored to our setting to replace the current generic clustering heuristic; (iii) clustering might be replaced altogether with geometrically informed methods that directly identify the corners of the polytope; (iv) the theory of LLS can be extended beyond identification to provide statistical results that might hold when $q(d|x)$ can only be noisily estimated, and when only finite samples are available for the induced topic modeling problem; (v) when the number of true classes k is unknown, we may develop approaches to estimate this k.

Table 2: *Results on FieldGuide-2*. Each entry is produced with the averaged result of 3 different random seeds. With DDFA (RI) we refer to DDFA with randomly initialized backbone. With DDFA (SPI) we refer to DDFA initialized with pretext training adopted by SCAN. Note that in DDFA (SPI), we do not leverage SCAN for clustering. α is the Dirichlet parameter used for generating label marginals in each domain, κ is the maximum allowed condition number of the generated $\mathbf{Q}_{Y|D}$ matrix, r is number of domains. Full results available in App. E

r	Approaches	$\alpha : 0.5,\ \kappa : 3$		$\alpha : 3,\ \kappa : 5$		$\alpha : 10,\ \kappa : 7$	
		Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err
2	SCAN	0.583	0.508	0.564	0.524	0.577	0.281
	DDFA (RI)	0.715	**0.222**	0.602	0.500	0.622	**0.200**
	DDFA (SPI)	**0.776**	0.241	**0.773**	**0.150**	**0.658**	0.264
5	SCAN	0.586	0.881	0.576	0.422	0.580	0.234
	DDFA (RI)	0.653	0.126	0.596	0.253	0.576	**0.159**
	DDFA (SPI)	**0.953**	**0.093**	**0.784**	**0.141**	**0.617**	0.258
10	SCAN	0.582	0.828	0.589	0.374	**0.582**	0.186
	DDFA (RI)	0.692	0.237	0.619	**0.149**	0.534	0.271
	DDFA (SPI)	**0.907**	**0.155**	**0.714**	0.170	**0.582**	**0.164**

Limitations

Assumptions Our approach is limited by the set of assumptions needed (label shift, as many data domains as true latent classes, known true number of classes k, and other assumptions established in A.1-A.4). Future work should aim to relax some or all of these assumptions.

Theory The work does not include finite sample bounds for the DDFA algorithm. In addition, we cannot include a formal guarantee that the clustering heuristic in the Discretize step of DDFA will retrieve pure anchor sub-domain clusters under potentially noisy black-box prediction of $q(d|x)$. This problem is complicated by the difficulty of reasoning about the noise that may be produced by a neural network or other complex non-linear model (acting as the black-box domain discriminator), and by the lack of concrete guarantees that K-means will recover the anchor subdomains among its recovered clusters. In particular, in the case in which the anchor subdomains do not contain all of the mass (equivalently, there are some x which could belong to more than one y), the arbitrary distribution of mass outside of the anchor subdomains makes it difficult to reason about the behavior of K-means.

Semi-synthetic Experiments Semi-synthetic experiments present an ideal environment for evaluating under the precise label shift assumption. However, we do not evaluate on datasets in which the separation into domains is organic, and the label shift is inherent.

Acknowledgements

SG acknowledges Amazon Graduate Fellowship and JP Morgan PhD Fellowship for their support. ZL acknowledges Amazon AI, Salesforce Research, Facebook, UPMC, Abridge, the PwC Center, the Block Center, the Center for Machine Learning and Health, and the CMU Software Engineering Institute (SEI) via Department of Defense contract FA8702-15-D-0002, for their generous support of ACMI Lab's research on machine learning under distribution shift.

References

[1] Amr Alexandari, Anshul Kundaje, and Avanti Shrikumar. Adapting to label shift with bias-corrected calibration. In *International Conference on Machine Learning (ICML)*, 2019.

[2] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[3] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization – provably. In *Symposium on Theory of Computing (STOC)*, 2012. doi: 10.1145/2213977.2213994. URL `https://doi.org/10.1145/2213977.2213994`.

[4] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models–going beyond svd. In *Foundations of Computer Science (FOCS)*. IEEE, 2012.

[5] Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. Regularized learning for domain adaptation under label shifts. In *International Conference on Learning Representations (ICLR)*, 2019.

[6] Jessa Bekker and Jesse Davis. Estimating the class prior in positive and unlabeled data through decision tree induction. In *Association for the Advancement of Artificial Intelligence (AAAI)*, volume 32, 2018. URL `https://ojs.aaai.org/index.php/AAAI/article/view/11715`.

[7] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: a survey. *Machine Learning*, 109(4):719–760, apr 2020. doi: 10.1007/s10994-020-05877-5. URL `https://doi.org/10.1007%2Fs10994-020-05877-5`.

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning research*, 3(Jan):993–1022, 2003.

[9] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[10] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data, 2019. URL `https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Doersch_Unsupervised_Visual_Representation_ICCV_2015_paper.pdf`.

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning (ICML)*, pages 1597–1607. PMLR, 2020.

[12] Yinyin Chen, Shishuang He, Yun Yang, and Feng Liang. Learning topic models: Identifiability and finite-sample analysis. *arXiv preprint arXiv:2110.04232*, 2021.

[13] Andrzej Cichocki and Anh-Huy Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions*, 92-A:708–721, 03 2009. doi: 10.1587/transfun.E92.A.708.

[14] David F Crouse. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1679–1696, 2016.

[15] Luke N. Darlow, Elliot J. Crowley, Antreas Antoniou, and Amos J. Storkey. Cinic-10 is not imagenet or cifar-10, 2018. URL `https://arxiv.org/abs/1810.03505`.

[16] Thiago de Paulo Faleiros and Alneu de Andrade Lopes. On the equivalence between algorithms for non-negative matrix factorization and latent dirichlet allocation. In *European Symposium on Artificial Neural Networks (ESANN)*, 2016.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ieee, 2009.

[18] Chris Ding, Tao Li, and Wei Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8): 3913–3927, 2008.

[19] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision (ICCV)*, 2015. URL `https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Doersch_Unsupervised_Visual_Representation_ICCV_2015_paper.pdf`.

[20] David Donoho and Victoria Stodden. When does non-negative matrix factorization give a correct decomposition into parts? *Advances in Neural Information Processing Systems (NeurIPS)*, 16, 2003.

[21] Marthinus C. du Plessis, Gang Niu, and Masashi Sugiyama. Class-prior estimation for learning from positive and unlabeled data. *Machine Learning*, 106(4):463–492, nov 2016. doi: 10.1007/s10994-016-5604-6. URL `https://doi.org/10.1007%2Fs10994-016-5604-6`.

[22] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 213–220, 2008.

[23] Saurabh Garg, Yifan Wu, Sivaraman Balakrishnan, and Zachary Lipton. A unified view of label shift estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL `https://arxiv.org/abs/2003.07554`.

[24] Saurabh Garg, Yifan Wu, Alex Smola, Sivaraman Balakrishnan, and Zachary Lipton. Mixture proportion estimation and PU learning: A modern approach. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL `https://arxiv.org/abs/2111.00980`.

[25] Eric Gaussier and Cyril Goutte. Relation between plsa and nmf and implications. In *Conference on Research and Development in Information Retrieval (SIGIR)*, 2005.

[26] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018. URL `https://arxiv.org/abs/1803.07728`.

[27] Nicolas Gillis and Stephen A. Vavasis. Fast and robust recursive algorithms for separable non-negative matrix factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):698–714, apr 2014. doi: 10.1109/tpami.2013.226. URL `https://doi.org/10.1137%2F130946782`.

[28] Mark Girolami and Ata Kabán. On an equivalence between plsi and lda. In *Conference on Research and Development in Information Retrieval (SIGIR)*, 2003.

[29] Jiaxian Guo, Mingming Gong, Tongliang Liu, Kun Zhang, and Dacheng Tao. Ltf: A label transformation framework for correcting label shift. In *International Conference on Machine Learning*, pages 3843–3853. PMLR, 2020.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[31] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[32] Thomas Hofmann. Probabilistic latent semantic indexing. In *Conference on Research and Development in Information Retrieval (SIGIR)*, 1999.

[33] Kejun Huang, Xiao Fu, and Nikolaos D Sidiropoulos. Anchor-free correlated topic modeling: Identifiability and algorithm. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[34] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[35] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[36] Dmitry Ivanov. DEDPUL: Difference-of-estimated-densities-based positive-unlabeled learning. *arXiv preprint arXiv:1902.06965*, 2019.

[37] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[38] Alex Krizhevsky and Geoffrey Hinton. Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer, 2009.

[39] Abhishek Kumar, Vikas Sindhwani, and Prabhanjan Kambadur. Fast conical hull algorithms for near-separable non-negative matrix factorization. In *International Conference on Machine Learning*, pages 231–239. PMLR, 2013.

[40] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International Conference on Machine Learning (ICML)*. PMLR, 2018.

[41] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 38(3):447–461, mar 2016. doi: 10.1109/tpami.2015.2456899. URL `https://doi.org/10.1109%2Ftpami.2015.2456899`.

[42] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, pages 281–297. Oakland, CA, USA, 1967.

[43] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning (ICML)*. PMLR, 2013.

[44] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[45] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[46] Christos H Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences*, 61(2): 217–235, 2000.

[47] Sungwon Park, Sungwon Han, Sundong Kim, Danu Kim, Sungkyu Park, Seunghoon Hong, and Meeyoung Cha. Improving unsupervised image clustering with robust learning. *CoRR*, abs/2012.11150, 2020. URL `https://arxiv.org/abs/2012.11150`.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 12, 2011.

[49] Kedar S Prabhudesai, Boyla O Mainsah, Leslie M Collins, and Chandra S Throckmorton. Augmented latent dirichlet allocation (lda) topic model with gaussian mixture topics. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2451–2455. IEEE, 2018.

[50] Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

[51] Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the Outputs of a Classifier to New a Priori Probabilities: A Simple Procedure. *Neural Computation*, 2002.

[52] Clayton Scott. A Rate of Convergence for Mixture Proportion Estimation, with Application to Learning from Noisy Labels. In *Artificial Intelligence and Statistics (AISTATS)*, 2015. URL `https://proceedings.mlr.press/v38/scott15.html`.

[53] D Seung and L Lee. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems (NeurIPS)*, 2001.

[54] Amos Storkey. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, 30:3–28, 2009.

[55] Vincent YF Tan and Cédric Févotte. Automatic relevance determination in nonnegative matrix factorization with the/spl beta/-divergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(7), 2012.

[56] Dongping Tian. Research on plsa model based semantic image analysis: A systematic review. *J. Inf. Hiding Multim. Signal Process.*, 9(5):1099–1113, 2018.

[57] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels, 2020. URL `https://arxiv.org/abs/2005.12320`.

[58] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[59] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[60] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning (ICML)*. PMLR, 2013.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We believe that this work, which proposes a novel unsupervised learning problem does not present a significant societal concern. While this could potentially guide practitioners to improve classification, we do not believe that it will fundamentally impact how machine learning is used in a way that could conceivably be socially salient.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Sec. 3
   (b) Did you include complete proofs of all theoretical results? [Yes] See Sec. 4 and appendices
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include all the necessary details to replicate our experiments in appendices. Code URL is given in the paper.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Yes, we describe crucial details in Sec. 6 and defer precise details to appendices.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We include results with multiple seeds in appendices
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Refer to experimental setup in App. D.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Refer to experimental setup in App. D.
   (b) Did you mention the license of the assets? [Yes] Refer to experimental setup in App. D.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Refer to experimental setup in App. D.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Supplementary Material

A Proofs of Lemmas

In this section, we present several new lemmas which are required to prove Theorem 2, and provide proofs. We also provide proofs for Lemmas 1, 2, and 3.

Lemma 4. *Let distribution Q over random variables X, Y, D satisfy A.1–A.4. Then for all $y \in \mathcal{Y}$, $q(y) > 0$. That is, all labels have nonzero probability under Q.*

Proof of Lemma 4. Proof by contradiction. Let $y \in \mathcal{Y}$ with $q(y) = 0$.

$$\begin{aligned} q(y) &= \sum_{d \in \mathcal{R}} q(d) q(y|D = d) \\ &= \sum_{d \in \mathcal{R}} \gamma_y q(y|D = d) \\ &= \sum_{d \in \mathcal{R}} \frac{1}{r} q(y|D = d) \\ &= \frac{1}{r} \sum_{d \in \mathcal{R}} q(y|D = d)\,. \end{aligned}$$

Since $q(y|D = d) \geqslant 0$ for all $d \in \mathcal{R}$, we see that if $q(y) = 0$, then $q(y|D = d) = 0$ for all $d \in \mathcal{R}$.

Then $[\mathbf{Q}_{Y|D}]_{y,d} = 0$ for all $d \in \mathcal{R}$. Then there is a row (row y) in the matrix $\mathbf{Q}_{Y|D}$ in which every entry is 0, so $\mathbf{Q}_{Y|D}$ cannot be full row rank k. This violates assumption A.2. Then by contradiction we have shown $q(y) > 0$. □

Lemma 5. *Let distribution Q over random variables X, Y, D satisfy Assumptions A.1–A.4. Let $x \in \mathcal{X}$ such that $q(x) > 0$. Then if $x \in A_y$ for some $y \in \mathcal{Y}$, we have that $q(y|X = x) = 1$, and for all $y' \in \mathcal{Y} \backslash \{y\}$, $q(y'|X = x) = 0$. The converse is also true: if $q(y|X = x) = 1$ for some $y \in \mathcal{Y}$ and $q(y'|X = x) = 0\ \forall y' \in \mathcal{Y} \backslash \{y\}$, then we know that $x \in A_y$.*

Proof of Lemma 5. We prove directions one at a time.

Forward direction. Assume $x \in A_y$.

$$\begin{aligned} q(x) &= \sum_{y'' \in \mathcal{Y}} q(y'') q(x|Y = y'') \\ &= q(y) q(x|Y = y) + \sum_{y' \in \mathcal{Y} \backslash \{y\}} q(y') q(x|Y = y') \\ &= q(y) q(x|Y = y) + \sum_{y' \in \mathcal{Y} \backslash \{y\}} q(y')\,(0) \\ &= q(y) q(x|Y = y). \end{aligned}$$

Recalling $q(x|y) > 0$ (by A.4) and $q(y) > 0$ (by Lemma 4), we know that $q(x) = q(y)q(x|Y = y) > 0$. Then $q(y|X = x) = \dfrac{q(y)q(x|Y = y)}{q(x)} = \dfrac{q(x)}{q(x)} = 1$. Because probabilities sum to 1, $q(y|X = x) + \sum_{y' \in \mathcal{Y} \backslash \{y\}} q(y'|X = x) = 1$. Then because $q(y|X = x) = 1$, we have : $\sum_{y' \in \mathcal{Y} \backslash \{y\}} q(y'|X = x) = 0$. Then for all $y' \in \mathcal{Y} \backslash \{y\}$, it must be that $q(y'|X = x) = 0$. Then we have shown $q(y|X = x) = 1$, and for all $y' \in \mathcal{Y} \backslash \{y\},\ q(y'|X = x) = 0$.

Converse. Assume $q(y|X = x) = 1$ and for all $y' \in \mathcal{Y} \backslash \{y\}, q(y'|X = x) = 0$. We recall that $q(x) > 0$. Also, $q(y) > 0$ by Lemma 4. Then $q(x|Y = y) = \dfrac{q(y|X = x)q(x)}{q(y)} = \dfrac{(1)q(x)}{q(y)} > 0$. Let $y' \in \mathcal{Y} \backslash \{y\}$. Then $q(x|Y = y') = \dfrac{q(y'|X = x)q(x)}{q(y')} = \dfrac{(0)q(x)}{q(y')} = 0$. Then because $q(x|Y = y) > 0$ and $\forall y' \in \mathcal{Y} \backslash \{y\},\ q(x|Y = y') = 0$, we see that $x \in A_y$. □

Lemma 6. *Let random variables X, Y, D and distribution Q satisfy Assumptions A.1–A.4. Then, the matrix $\mathbf{Q}_{D|Y}$, defined as an $r \times k$ matrix whose elements are $[\mathbf{Q}_{D|Y}]_{i,j} = Q(D = i|Y = j)$, and in which each column is a conditional distribution over the domains given a label, has linearly independent columns. Furthermore, $\mathbf{Q}_{D|Y}$ can be computed directly from only $\mathbf{Q}_{Y|D}$.*

Proof of Lemma 6. Let random variables X, Y, D and distribution Q satisfy Assumptions A.1–A.4.

Each $[\mathbf{Q}_{D|Y}]_{d,y} = q(d|Y = y) = \dfrac{q(y|D = d)q(d)}{q(y)} = \dfrac{q(y|D = d)\gamma_d}{q(y)} = \dfrac{q(y|D = d)}{rq(y)}$.

Since each yth column of $\mathbf{Q}_{D|Y}$ is a probability distribution that sums to 1, and $rq(y)$ is constant down each yth column, we can obtain $\mathbf{Q}_{D|Y}$ by simply taking $\mathbf{Q}_{Y|D}^\top$, in which each $[\mathbf{Q}_{Y|D}^\top]_{d,y} = [\mathbf{Q}_{Y|D}]_{y,d} = q(y|D = d)$, and normalizing the columns so they sum to 1.

The matrix $\mathbf{Q}_{Y|D}$ has linearly independent rows by Assumption A.2. Then $\mathbf{Q}_{Y|D}^\top$ has linearly independent columns. Scaling these columns by a nonzero value does not change their linear independence, so the columns of $\mathbf{Q}_{D|Y}$ are also linearly independent.

Then matrix $\mathbf{Q}_{D|Y}$ has linearly independent columns, and can be computed by taking $\mathbf{Q}_{Y|D}^\top$ and normalizing its columns.

□

Lemma 7. *Let random variables X, Y, D and distribution Q satisfy Assumptions A.1–A.4. Let $d \in \mathcal{R}, x \in \mathcal{X}, y \in \mathcal{Y}$. Then $q(d|X = x, Y = y) = q(d|Y = y)$.*

Proof of Lemma 7.

$$\begin{aligned}
q(d|X = x, Y = y) &= \frac{q(x|D = d, Y = y)q(d|Y = y)}{q(x|Y = y)} \\
&= \frac{p_d(x|Y = y)q(d|Y = y)}{q(x|Y = y)} \\
&= \frac{p(x|Y = y)q(d|Y = y)}{q(x|Y = y)} \\
&= \frac{q(x|Y = y)q(d|Y = y)}{q(x|Y = y)} \\
&= q(d|Y = y).
\end{aligned}$$

□

Lemma 1. *We restate this lemma, first presented in Sec. 4, for convenience. Let the distribution Q over random variables X, Y, D satisfy Assumptions A.1–A.4. Then the matrix $\mathbf{Q}_{Y|D}$ and $f(x) = q(d|x)$ uniquely determine $q(y|x)$ for all $y \in \mathcal{Y}$ and $x \in \mathcal{X}$ such that $q(x) > 0$.*

Proof of Lemma 1. Let distribution Q over random variables X, Y, D satisfy Assumptions A.1-A.4. Let $x \in \mathcal{X}$ with $q(x) > 0$, and $y \in \mathcal{Y}$. Assume we know $\mathbf{Q}_{Y|D}$ and $[f(x)]_d = q(d|X = x)$. Notice that, for all $x \in \mathcal{X}, d \in \mathcal{R}$,

$$q(d|X = x) = \sum_{y' \in \mathcal{Y}} q(d|X = x, Y = y')q(y'|X = x).$$

Now using Lemma 7,

$$q(d|X = x) = \sum_{y' \in \mathcal{Y}} q(d|Y = y')q(y'|X = x).$$

Define the vector-valued function $g : \mathcal{X} \to \mathbb{R}^k$ such that $[g(x)]_y = q(y|X = x)$ for all $x \in \text{supp}_Q(X)$. $\mathbf{Q}_{D|Y}$ is a matrix of shape $r \times k$, with $[\mathbf{Q}_{D|Y}]_{i,j} = Q(D = i|Y = j)$. It can be computed from $\mathbf{Q}_{Y|D}$ and has linearly independent columns—both facts shown in Lemma 6.

Then $[f(x)]_d = q(d|X = x) = \mathbf{Q}_{D|Y}[d, :]g(x)$, a product between the dth row vector of $\mathbf{Q}_{D|Y}$ and the column vector $g(x)$. Then $f(x) = \mathbf{Q}_{D|Y}g(x)$.

This system is a linear system with $r \geqslant k$ equations. Recalling that $\mathbf{Q}_{D|Y}$ has k linearly independent columns, we can select any k linearly independent rows of $\mathbf{Q}_{D|Y}$ to solve the equation for the true, unique solution for the unknown vector $g(x)$. Another way to describe this is with the pseudo-inverse: $g(x) = (\mathbf{Q}_{D|Y})^{\dagger} f(x)$. Then we have $[g(x)]_y = q(y|X = x)$ for all $y \in \mathcal{Y}$.

□

Lemma 2. *We restate this lemma, first presented in Sec. 4, for convenience. Let the distribution Q over random variables X, Y, D satisfy Assumptions A.1–A.4. Then for all $y \in \mathcal{Y}$, and $x \in \mathcal{X}$ such that $q(x, d) > 0$. the matrix $\mathbf{Q}_{Y|D}$ and $q(y|x)$ uniquely determine $q(y|x, d)$.*

Proof of Lemma 2. Let distribution Q over random variables X, Y, D satisfy Assumptions A.1-A.4. Let $x \in \mathcal{X}, d \in \mathcal{R}$ with $q(x, d) > 0$, and $y \in \mathcal{Y}$.

Assume we know matrix $\mathbf{Q}_{Y|D}$ and $q(y'|X = x),\ \forall y' \in \mathcal{Y}$. We can compute $\mathbf{Q}_{D|Y}$ from $\mathbf{Q}_{Y|D}$ via Lemma 6.

$$\begin{aligned} q(y|X = x, D = d) &= \frac{q(y, x, d)}{q(x, d)} \\ &= \frac{q(d|X = x, Y = y)q(y|X = x)q(x)}{q(d|X = x)q(x)}. \end{aligned}$$

Using Lemma 7, $q(d|X = x, Y = y) = q(d|Y = y)$. We apply this property.

$$\begin{aligned} q(y|X = x, D = d) &= \frac{q(d|Y = y)q(y|X = x)q(x)}{q(d|X = x)q(x)} \\ &= \frac{q(d|Y = y)q(y|X = x)}{q(d|X = x)}. \end{aligned}$$

The denominator $q(d|X = x)$ is constant across all values of y, so we can write that $q(y|X = x, D = d) \propto q(d|Y = y)q(y|X = x)$ and normalize to find the probability:

$$q(y|X = x, D = d) = \frac{q(d|Y = y)q(y|X = x)}{\sum_{y' \in \mathcal{Y}} q(d|Y = y')q(y'|X = x)}.$$

We know $q(d|Y = y)$ as $[\mathbf{Q}_{D|Y}]_{d,y}$, and every $q(d|Y = y')$, where $y' \in \mathcal{Y}\backslash\{y\}$, as $[\mathbf{Q}_{D|Y}]_{d,y'}$. We also know $q(y|X = x)$ and every $q(y'|X = x)$ where $y' \in \mathcal{Y}\backslash\{y\}$, by the precondition assumptions. Then we can compute $q(y|X = x, D = d)$. □

Lemma 3. *We restate this lemma, first presented in Sec. 4, for convenience. Let the distribution Q over random variables X, Y, D satisfy Assumptions A.1–A.4. Then for all x, x' in anchor subdomain, i.e., $x, x' \in A_y$ for a given label $y \in \mathcal{Y}$, we have $f(x) = f(x')$. Further, for any $y \in \mathcal{Y}$, if $x \in A_y, x' \notin A_y$, then $f(x) \neq f(x')$.*

Proof of Lemma 3. Let distribution Q over random variables X, Y, D satisfy Assumptions A.1-A.4. Recall $f : \mathbb{R}^p \to \mathbb{R}^r$ is a vector-valued oracle function such that $[f(x)]_d = q(d|X = x)$ for all $x \in \text{supp}_Q(X)$. Also let us recall that $\mathbf{Q}_{D|Y}$ is defined as an $r \times k$ matrix whose elements $[\mathbf{Q}_{D|Y}]_{i,j} = Q(D = i|Y = j)$, and each column is a conditional distribution over the domains given a label. It has linearly independent columns by Lemma 6.

First recognize that for all $d \in \mathcal{R}, x \in \mathcal{X}$ such that $q(x) > 0$,

$$\begin{aligned} [f(x)]_d = q(d|X = x) &= \sum_{y'' \in \mathcal{Y}} q(d, y''|X = x). \\ &= \sum_{y'' \in \mathcal{Y}} q(d|Y = y'', X = x)q(y''|X = x). \end{aligned}$$

Using Lemma 7, $q(d|X = x, Y = y) = q(d|Y = y)$. We apply this property.

$$[f(x)]_d = q(d|X = x) = \sum_{y'' \in \mathcal{Y}} q(d|Y = y'')q(y''|X = x).$$

Then we can write $f(x) = \sum_{y'' \in \mathcal{Y}} q(y''|X = x)\mathbf{Q}_{D|Y}[:, y'']$, where $\mathbf{Q}_{D|Y}[:, y'']$ is the y''th column of $\mathbf{Q}_{D|Y}$. Now we could also rewrite $f(x) = \mathbf{Q}_{D|Y}\left[Q(Y = 1|X = x) \ldots Q(Y = k|X = x)\right]^\top$.

We now prove two key components of the lemma. Let $y \in \mathcal{Y}$. Let $x \in A_y$ such that $q(x) > 0$.

Points in same anchor sub-domain map together. Let $x' \in A_y$ such that $q(x') > 0$. We now seek to show that $f(x) = f(x')$. Recall that $x, x' \in A_y$. By Lemma 5, $q(y|X = x) = q(y|X = x') = 1$. Also by lemma 5, $\forall y'' \in \mathcal{Y}\backslash\{y\}, q(y''|X = x) = q(y''|X = x') = 0$. Then for all $y'' \in \mathcal{Y}$, $q(y''|X = x) = q(y''|X = x')$.

Therefore, $\forall d \in \mathcal{R}$,

$$\begin{aligned}
[f(x)]_d = q(d|X = x) &= \sum_{y'' \in \mathcal{Y}} q(d|Y = y'')q(y''|X = x) \\
&= \sum_{y'' \in \mathcal{Y}} q(d|Y = y'')q(y''|X = x') \\
&= q(d|X = x') = [f(x')]_d.
\end{aligned}$$

Then $f(x) = f(x')$.

Point outside of the anchor sub-domain does not map with points in the anchor sub-domain . Let $x_0 \notin A_y$ such that $q(x_0) > 0$. We now seek to show that $f(x) \neq f(x_0)$. Because $x_0 \notin A_y$ with $q(x_0) > 0$, and because A_y contains all x such that $q(x) > 0$, $q(y|X = x) = 1$, and $q(y'|X = x) = 0$ for all $y' \in \mathcal{Y}\backslash\{y\}$, then by Lemma 5, it must be that one of the following cases is true:

- **Case 1:** $q(y|X = x_0) \neq 1$
- **Case 2:** $q(y'|X = x_0) > 0$ for some $y' \in \mathcal{Y}\backslash\{y\}$.

In all circumstances, there exists some $y'' \in \mathcal{Y} : q(y''|x_0) \neq q(y''|x)$. Then,

$$\left[Q(Y = 1|X = x) \ldots Q(Y = k|X = x)\right]^\top \neq \left[Q(Y = 1|X = x_0) \ldots Q(Y = k|X = x_0)\right]^\top.$$

Because $\mathbf{Q}_{D|Y}$ has linearly independent columns (shown in Lemma 6), we now know that

$$\begin{aligned}
f(x) &= \mathbf{Q}_{D|Y}\left[Q(Y = 1|X = x) \ldots Q(Y = k|X = x)\right]^\top \\
&\neq \mathbf{Q}_{D|Y}\left[Q(Y = 1|X = x_0) \ldots Q(Y = k|X = x_0)\right]^\top = f(x_0).
\end{aligned}$$

So $f(x) \neq f(x_0)$.

□

B Proof of Theorem 2

Theorem 2. *We restate this theorem, first presented in Sec. 4, for convenience. Let the distribution Q over random variables X, Y, D satisfy Assumptions A.1–A.4. Assuming access to the joint distribution $q(x, d)$, and knowledge of the number of true classes k, we show that the following quantities are identifiable: (i) $\mathbf{Q}_{Y|D}$, (ii) $q(y|X = x)$, for all $x \in \mathcal{X}$ that lies in the support (i.e. $q(x) > 0$); and (iii) $q(y|X = x, D = d)$, for all $x \in \mathcal{X}$ and $d \in \mathcal{R}$ such that $q(x, d) > 0$.*

Proof of Theorem 2. Let distribution Q over random variables X, Y, D satisfy Assumptions A.1-A.4.

Recall $f : \mathcal{X} \to \mathbb{R}^r$ is a vector-valued oracle function such that $[f(x)]_d = q(d|X = x)$ for all $x \in \text{supp}_Q(X)$. It is known because we know the marginal $q(x, d)$. Let $y \in \mathcal{Y}$. Then by Lemma 3, f sends every $x \in A_y$ (and no other $x \notin A_y$) to the same value. We overload notation to denote this as $f(A_y)$. Then $Q(f(X) = f(A_y)) = Q(X \in A_y) \geqslant \epsilon$. Then in the marginal distribution of $f(X)$ with respect to distribution Q, there is a distinct point mass on each $f(A_y)$, with mass at least ϵ.

Because we know the marginal $q(x, d)$, we know the marginal $q(x)$, so we can obtain the distribution of $f(X)$ with respect to distribution Q. If we analyze the marginal distribution of $f(X)$ with respect to distribution Q, and recover all point masses with mass at least ϵ, we can recover no more than $\mathcal{O}\left(1/\epsilon\right)$ such points. We set $m \in \mathbb{Z}^+$ so that the number of points we recovered is $m - 1$.

We denote a mapping $\psi : \mathbb{R}^r \to [m]$. This mapping sends each value of $f(x)$ corresponding to a point mass with mass at least ϵ to a unique index in $\{1, ..., m - 1\}$. It sends any other value in $\mathbb{R}^p$ to m. We note that the ordering of the point masses might have $(m - 1)!$ permutations.

Notice that the point mass on each $f(A_y)$ must be recovered among these $m - 1$ masses. Recall that for all $y \in \mathcal{Y}$, $f(x) = f(A_y)$ if and only if $x \in A_y$. Then for all $y \in \mathcal{Y}$, $\psi(f(x)) = \psi(f(A_y))$ if and only if $x \in A_y$, because ψ does not send any other value in $\mathbb{R}^r$ besides $f(A_y)$ to $\psi(f(A_y))$.

For convenience, we now define a mapping $c : \mathcal{X} \to [m]$ such that $c = \psi \circ f$. We will also abuse notation here to denote $c(A_y) = \psi(f(A_y))$. Then $c(X)$ is a discrete, finite random variable that takes values in $[m]$. As c is a pushforward function on X, $c(X)$ satisfies the label shift assumption because X does (i.e., when conditioning on Y, the distribution of $c(X)$ is domain-invariant).

We might now define a matrix $\mathbf{Q}_{c(X)|D}$ in which each entry $[\mathbf{Q}_{c(X)|D}]_{i,d} = Q(c(X) = i|D = d)$. We recall that we know the number of true classes k. Then we know that there is a (possibly unique) unknown decomposition of the following form:

$$q(c(X)|d) = \sum_{y \in \mathcal{Y}} q(c(X)|Y = y, D = d) q(y|D = d)$$

Using the label shift property,

$$q(c(X)|d) = \sum_{y \in \mathcal{Y}} q(c(X)|Y = y) q(y|D = d).$$

To express this decomposition in matrix form, we write $\mathbf{Q}_{c(X)|D} = \mathbf{Q}_{c(X)|Y} \mathbf{Q}_{Y|D}$. Now we make observations about the unknown $\mathbf{Q}_{c(X)|Y}$. For all $y \in \mathcal{Y}$,

$$Q(c(x) = c(A_y)|Y = y) = Q(X \in A_y|Y = y) > 0\,.$$
$$Q(c(x) = c(A_y)|Y \neq y) = Q(X \in A_y|Y \neq y) = 0\,.$$

Then for each $y \in \mathcal{Y}$, the row of $\mathbf{Q}_{c(X)|Y}$ with row index $c(A_y)$ is positive in the yth column, and zero everywhere else. Restated, for each $y \in \mathcal{Y}$, there is some row with positive entry exactly in yth column. This is precisely the anchor word assumption for a discrete, finite random variable. We already know that $\mathbf{Q}_{Y|D}$ is full row-rank, so because $\mathbf{Q}_{c(X)|Y}$ satisfies the anchor word assumption, we can identify $\mathbf{Q}_{Y|D}$ up to permutation of rows by Theorem 1. In other words, when we set the constraint that the recovered $\mathbf{Q}_{c(X)|Y}$ must have k columns and satisfy anchor word and the recovered $\mathbf{Q}_{Y|D}$ must have k rows and be full row-rank, any solution to the decomposition $\mathbf{Q}_{c(X)|D} = \mathbf{Q}_{c(X)|Y} \mathbf{Q}_{Y|D}$ must identify the ground truth $\mathbf{Q}_{Y|D}$, up to permutation of its rows.

□

C Minimizing Cross-Entropy Loss yields Domain Discriminator

In this section we reason about our choice of cross entropy loss to estimate the domain discriminator. We show that in population, when optimizing over a sufficiently powerful class of functions, the minimizer of the cross entropy loss is the conditional distribution over domains given input X.

Define the vector-valued function $z : \mathcal{R} \to \mathbb{R}^r$ such that $z(D)$ is a one-hot random vector of length r, such that $[z(D)]_i = 1,$ iff $D = i$. Then we write the cross-entropy objective in expectation over the input random variable X and target random variable D as:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{(X,D)\sim Q}\left[-\sum_{i=1}^{i=r}[z(D)]_i \log([f(X)]_i)\right] \\ &= \mathbb{E}_X \mathbb{E}_{D|X}\left[-\sum_{i=1}^{i=r}[z(D)]_i \log([f(X)]_i)\right]\end{aligned}$$

where the second line follows by splitting the expectations using the law of iterative expectations. For ease of notation we denote conditioning on $X = x$ as just conditioning on X. We now move the inner expectation into the summation over the domain random variable and pull the terms that do not depend on D outside this inner expectation:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_X\left[-\sum_{i=1}^{i=r}\mathbb{E}_{D|X}\left[[z(D)]_i \log([f(X)]_i)\right]\right] \\ &= \mathbb{E}_X\left[-\sum_{i=1}^{i=r}\log([f(X)]_i)\mathbb{E}_{D|X}[[z(D)]_i]\right]\end{aligned}$$

Since $[Z(D)]_i$ is a random variable which is 1 when $D = i$ conditioned on X, and 0 otherwise, the inner expectation can be simplified as follows: $\mathbb{E}_{D|X}[[z(D)]_i] = 1 \times Q(D = i|X) + 0 \times (1 - Q(D = i|X))$, giving

$$\mathcal{L} = \mathbb{E}_X\left[-\sum_{i=1}^{i=r}\log([f(X)]_i)Q(D = i|X)\right]$$

In order to learn a distribution, we constrain our search space to the subset of functions from $\mathbb{R}^p \to [0, 1]^r$ that project to a simplex Δ^{r-1}. We now look to find the minimizer of the above equation from within this subset for each value of X. This corresponds to minimizing each term that contributes to the outer expectation over X, leading to the overall minimizer of the cross-entropy loss.

We formulate this objective along with the Lagrange constraint modeling the sum of components of $f(X)$ adding to 1. Note that we do not add the constraint of each component of $f(X)$ lying between 0 and 1, but it is easy to see that the resulting solution satisfies this constraint.

$$J = \min_{[f(X)]_1 \ldots [f(X)]_r} -\sum_{i=1}^{i=r}\log([f(X)]_i)Q(D = i|X) + \lambda\left(\sum_{i=1}^{i=r}[f(X)]_i - 1\right)$$

Setting partial derivative with respect to $[f(X)]_i$ to 0, we get $-\dfrac{Q(D = i|X)}{[f^\star(X)]_i} + \lambda = 0$ and $[f^\star(X)]_i = \frac{1}{\lambda}Q(D = i|X)$, where we use $f^\star$ to denote the minimizer.

The last piece is to solve for the Lagrange multiplier λ. This can be achieved by applying Karush-Kuhn-Tucker (KKT) conditions which suppose that the optimal solution lies on the constraint surface. This gives, $\sum_{i=1}^{i=r}[f^\star(X)]_i = 1$ which implies $\sum_{i=1}^{i=r}\frac{1}{\lambda}Q(D = i|X) = 1$. Since sum over domains $Q(D|X) = 1$, we get $\lambda = 1$.

Plugging this in, we finally get $[f^\star(X)]_i = Q(D = i|X)$. Thus, the optimal f^* obtained by minimizing the cross entropy objective will in fact recover the oracle domain discriminator. As mentioned, it is clear that the resulting solution satisfies the required conditions of valid probability distributions.

$$\begin{bmatrix} 0.17 & 0.65 \\ 0.83 & 0.35 \end{bmatrix}$$

(a) $\alpha : 0.5, \kappa : 3$

$$\begin{bmatrix} 0.37 & 0.06 \\ 0.63 & 0.94 \end{bmatrix}$$

(b) $\alpha : 3, \kappa : 5$

$$\begin{bmatrix} 0.42 & 0.25 \\ 0.58 & 0.75 \end{bmatrix}$$

(c) $\alpha : 10, \kappa : 7$

Figure 3: Example $\mathbf{Q}_{Y|D}$ matrices sampled for FieldGuide-2 with 2 classes and 2 domains. Each column represents the distribution across classes $p_d(y)$ for a given domain. At small α, each $p_d(y)$ is likelier to be "sparse" (our definition is an informal one meaning not that there are many zero entries, but instead that the distribution is heavily concentrated in a few classes). At large α, $p_d(y)$ tends toward a uniform distribution in which classes are represented evenly.

D Additional Experimental Details

Our code is available at `https://github.com/acmi-lab/Latent-Label-Shift-DDFA`. Here we present the full generation procedure for semisynthetic example problems, and discuss the parameters.

1. Choose a Dirichlet concentration parameter $\alpha > 0$, maximum condition number $\kappa \geqslant 1$ (with respect to 2-norm), and domain count $r \geqslant k$.
2. For each $y \in [k]$, sample $p_d(y) \sim \text{Dir}(\frac{\alpha}{k}\mathbf{1}_k)$.
3. Populate the matrix $\mathbf{Q}_{Y|D}$ with the computed $p_d(y)$s. If $\text{cond}(\mathbf{Q}_{Y|D}) > k$, return to step 2 and re-sample.
4. Distribute examples across domains according to $\mathbf{Q}_{Y|D}$, for each of train, test, and valid sets. This procedure entails creating a quota number of examples for each (class, domain) pair, and drawing datapoints without replacement to fill each quota. We must discard excess examples from some classes in the dataset due to class imbalance in the $\mathbf{Q}_{Y|D}$ matrix. Due to integral rounding, domains may be *slightly* imbalanced.
5. Conceal true class information and return (x_i, d_i) pairs.

It is important to note the role of κ and α in the above formulation. Although they are unknown parameters to the classification algorithm, they affect the sparsity of the $\mathbf{Q}_{Y|D}$ and difficulty of the problem. Small α encourages high sparsity in $p_d(y)$, and large α causes $p_d(y)$ to tend towards a uniform distribution. We observe an example of the effects of α in Fig. 3. κ has a strong effect on the difficulty of the problem. Consider the case when $k = 2$. When $\kappa = 1$, the only potential $\mathbf{Q}_{Y|D}$ matrices are $\mathbf{I}_2$ up to row permutation (which means that domains and classes are exactly correlated, so the domain indicates the class and the problem is supervised). In the other limit, if $\kappa \to +\infty$, we may generate $\mathbf{Q}_{Y|D}$ matrices that are nearly singular, breaking needed assumptions for domain discriminator output to uniquely identify true class of anchor subdomains. κ also helps control the class imbalance (if a row of $\mathbf{Q}_{Y|D}$ is small, indicating that the class is heavily under-represented across all domains, the condition number will increase).

D.1 FieldGuide-2 and FieldGuide-28 Datasets

The dataset and description is available at `https://sites.google.com/view/fgvc6/competitions/butterflies-moths-2019`. From this data we create two datasets FieldGuide-2 and FieldGuide-28. For FieldGuide-28 we select the 28 classes which have 1000 datapoints in the training file. Since the test set provided in the website does not have annotations, we manually create a test set by sampling 200 datapoints from training file of each of the 28 classes. Therefore, we finally have 22400 training points and 5600 testing points. The FieldGuide-2 dataset is created by considering two classes from the created FieldGuide-28 dataset.

D.2 Hyperparameters and Implementation Details: SCAN baseline

In all cases, we initialize the SCAN [57] network with the clustering head attached, sample data according to the $\mathbf{Q}_{Y|D}$ matrix, and predict classes. With the Hungarian algorithm, implemented in

[14, 58], we compute the highest true accuracy among any permutation of these labels (denoted "Test acc").

- **CIFAR-10 and CIFAR-20 Datasets [38]** We use ResNet-18 [30] backbone with weights trained by SCAN-loss and obtained from the SCAN repo `https://github.com/wvangansbeke/Unsupervised-Classification`. We use the same transforms present in the repo for test data.
- **ImageNet-50 Dataset [17]** We use ResNet-50 backbone with weights trained by SCAN-loss and obtained from the SCAN repo. We use the same transforms present in the repo for test data.
- **FieldGuide-2 and FieldGuide-28 Datasets** For each of the two datasets, we pretrain a different SCAN baseline network (including pretext and SCAN-loss steps) on all available data from the dataset. The backbone for each is ResNet-18. For training the pretext task, we use the same transform strategy used in the repo for CIFAR-10 data (with mean and std values as computed on the FieldGuide-28 dataset, and crop size 224). For training SCAN, we resize the smallest image dimension to 256, perform a random horizontal flip and random crop to size 224. We also normalize. For validation we resize smallest image dimension to 256, center crop to 224, and normalize.

D.3 Hyperparameters and Implementation Details: DDFA (RI)

This is the DDFA procedure with random initialization of the domain discriminator. The bulk of this procedure is described in Section 6, but for completeness we reiterate here.

We train ResNet-50 [30] (with random initialization and added dropout) based on the implementation from `https://github.com/kuangliu/pytorch-cifar` on images x_i with domain indices d_i as the label, choose best iteration by valid loss, pass all training and validation data through $\widehat{f}$, and cluster pushforward predictions $\widehat{f}(x_i)$ into $m \geqslant k$ clusters with Faiss K-Means [37]. We compute the $\widehat{\mathbf{Q}}_{c(X)|D}$ matrix and run NMF to obtain $\widehat{\mathbf{Q}}_{c(X)|Y}$, $\widehat{\mathbf{Q}}_{Y|D}$. To make columns sum to 1, we normalize columns of $\widehat{\mathbf{Q}}_{c(X)|Y}$, multiply each column's normalization coefficient over the corresponding row of $\widehat{\mathbf{Q}}_{Y|D}$ (to preserve correctness of the decomposition), and then normalize columns of $\widehat{\mathbf{Q}}_{Y|D}$.

Some NMF algorithms only output solutions satisfying the anchor word property [3, 39, 27]. We found the strict requirement of an exact anchor word solution to lead to low noise tolerance. We therefore use the Sklearn implementation of standard NMF [13, 55, 48].

We predict class labels with Algorithm 2. With the Hungarian algorithm, implemented in [14, 58], we compute the highest true accuracy among any permutation of these labels (denoted "Test acc"). With the same permutation, we reorder rows of $\widehat{Q}_{Y|D}$, then compute the average absolute difference between corresponding entries of $\widehat{\mathbf{Q}}_{Y|D}$ and $\mathbf{Q}_{Y|D}$ (denoted "$\mathbf{Q}_{Y|D}$ err").

In order to make hyperparameter choices for final experiments, such as the choice of the NMF solver, clustering algorithm, and learning rate, we consulted CIFAR-10 and CINIC-10 (similar to an extension of CIFAR-10) [15] final test task accuracy. We were also able to confirm our intuition that minimum validation loss on domain discriminator training corresponds to good final task performance. Final evaluation runs on CIFAR-10 were made with the best hyperparameters found here. We acknowledge that this may lead to test-set overfitting on CIFAR-10, but point out that on all other datasets, we consulted only validation loss on domain discriminator training when adjusting the hyperparameters. Final evaluation runs used the following fixed hyperparameters:

Common Hyperparameters

- Hardware: A single NVIDIA RTX A6000 GPU was used for each experiment (on this hardware, trial length varies < 1 hour to 24 hours, depending on the dataset).
- Architecture: ResNet-50 with added dropout
- Faiss KMeans number of iterations (niter): 100
- Faiss Kmeans number of clustering redos (nredo): 5
- Learning Rate: 0.001

- Learning Rate Decay: Exponential, parameter 0.97
- SKlearn NMF initialization: random

Dataset-Specific Hyperparameters

- **CIFAR-10 Dataset** Training Epochs: 100. Number of Clusters (m): 30
- **CIFAR-20 Dataset** Training Epochs: 100. Number of Clusters (m): 60
- **ImageNet-50 Dataset** DDFA (RI) was not evaluated for this dataset due to poor performance of the domain discriminator without an appropriate pre-seed in early trials.
- **FieldGuide-2 Dataset** Training Epochs: 100. Number of Clusters (m): 10
- **FieldGuide-28 Dataset** DDFA (RI) was not evaluated for this dataset due to poor performance of the domain discriminator without an appropriate pre-seed in early trials.

D.4 Hyperparameters and Implementation Details: DDFA (SI) and DDFA (SPI)

This is the DDFA procedure with SCAN initialization of the domain discriminator. DDFA (SI) uses the SCAN pretext + SCAN loss pretraining steps, while DDFA (SPI) uses only the SCAN pretext step.

The procedure is identical to the standard DDFA procedure, except that SCAN [57] pre-trained weights or SCAN [57] contrastive pre-text weights are used to initialize the domain discriminator before it is fine-tuned on the domain discrimination task. Hyperparameters used also differ. When SCAN pretrained weights are available, we use those. When they are not, we train SCAN ourselves.

Much like SCAN (RI), we used CIFAR-10 final test accuracy to choose hyperparameters and make algorithm decisions. For all other datasets, we consulted only validation domain discrimination loss. Final evaluation runs used the following fixed hyperparameters:

Common Hyperparameters

- Hardware: A single NVIDIA RTX A6000 GPU was used for each experiment (on this hardware, trial length varies < 1 hour to 24 hours, depending on the dataset).
- Faiss KMeans number of iterations (niter): 100
- Faiss Kmeans number of clustering redos (nredo): 5
- Learning Rate: 0.00001
- Learning Rate Decay: Exponential, parameter 0.97
- SKlearn NMF initialization: random

Dataset-Specific Hyperparameters

- **CIFAR-10 Dataset**
  Architecture: ResNet-18
  Pre-seed: Weights trained with SCAN pretext and SCAN-loss on entirety of CIFAR-10 (from SCAN repo).
  Training Epochs: 25
  Number of Clusters (m): 10
  Transforms used: Same as SCAN repo.
- **CIFAR-20 Dataset**
  Architecture: ResNet-18
  Pre-seed: Weights trained with SCAN pretext and SCAN-loss on entirety of CIFAR-20 (from SCAN repo).
  Training Epochs: 25
  Number of Clusters (m): 20
  Transforms used: Same as SCAN repo.

- **ImageNet-50 Dataset**
  Architecture: ResNet-50
  Pre-seed: Weights trained with SCAN pretext and SCAN-loss on entirety of ImageNet-50 (from SCAN repo).
  Training Epochs: 25
  Number of Clusters (m): 50
  Transforms used: Same as SCAN repo.
- **FieldGuide-2 Dataset**
  Architecture: ResNet-18
  Pre-seed: Weights trained with SCAN pretext on entirety of FieldGuide-2 (trained by us).
  Training Epochs: 30
  Number of Clusters (m): 2
  Transforms used for pretext: Same strategy as CIFAR-10 in SCAN repo with appropriate mean, std, and crop size 224.
  Transform used for SCAN: Resize to 256, Random horizontal flip, Random crop to 224, normalize
  Learning rate used for SCAN: 0.001 (other hyperparameters were same as in SCAN repo for CIFAR-10)
- **FieldGuide-28 Dataset**
  Architecture: ResNet-18
  Pre-seed: Weights trained with SCAN pretext on entirety of FieldGuide-28 (trained by us).
  Training Epochs: 60
  Number of Clusters (m): 28
  Transforms used for pretext: Same strategy as CIFAR-10 in SCAN repo with appropriate mean, std, and crop size 224.
  Transform used for SCAN: Resize to 256, Random horizontal flip, Random crop to 224, normalize
  Learning rate used for SCAN: 0.01 (other hyperparameters were same as in SCAN repo for CIFAR-10)

E Additional Experimental Results

Here we present additional experimental results. We also investigated evaluations on the Waterbirds dataset, but although DDFA showed some reasonable results, we did not find SCAN hyperparameters that lead to a successful SCAN baseline. Accordingly, we do not include these results as they do not present a reasonable comparison.

Table 3: *Results on CIFAR-10*. Each entry is produced with the averaged result of 3 different random seeds. With DDFA (RI) we refer to DDFA with randomly initialized backbone. With DDFA (SI) we refer to DDFA's backbone initialized with SCAN. Note that in DDFA (SI), we do not leverage SCAN for clustering. α is the Dirichlet parameter used for generating label marginals in each domain, κ is the maximum allowed condition number of the generated $\mathbf{Q}_{Y|D}$ matrix, r is number of domains. "Test acc" is classification accuracy, under the best permutation of the recovered classes, and "$\mathbf{Q}_{Y|D}$ err" is the average entry-wise absolute error in the recovered $\mathbf{Q}_{Y|D}$.

r	Approaches	$\alpha : 0.5,\ \kappa : 4$		$\alpha : 3,\ \kappa : 4$		$\alpha : 10,\ \kappa : 8$	
		Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err
10	SCAN	0.823	0.146	**0.826**	0.126	**0.804**	0.082
	DDFA (RI)	0.736	0.035	0.539	0.048	0.314	0.074
	DDFA (SI)	**0.899**	**0.023**	0.757	**0.040**	0.536	**0.054**
15	SCAN	0.822	0.154	0.817	0.116	**0.812**	0.080
	DDFA (RI)	0.773	0.033	0.532	0.046	0.275	0.074
	DDFA (SI)	**0.961**	**0.016**	**0.844**	**0.026**	0.733	**0.038**
20	SCAN	0.802	0.143	0.809	0.117	**0.818**	0.087
	DDFA (RI)	0.688	0.047	0.565	0.046	0.270	0.071
	DDFA (SI)	**0.966**	**0.016**	**0.904**	**0.019**	0.798	**0.030**
25	SCAN	0.801	0.155	0.811	0.114	0.811	0.085
	DDFA (RI)	0.724	0.039	0.562	0.044	0.280	0.086
	DDFA (SI)	**0.970**	**0.013**	**0.917**	**0.017**	**0.820**	**0.027**

Table 4: *Results on ImageNet-50*. Each entry is produced with the averaged result of 3 different random seeds. With DDFA (SI) we refer to DDFA's backbone initialized with SCAN. Note that in DDFA (SI), we do not leverage SCAN for clustering. α is the Dirichlet parameter used for generating label marginals in each domain, κ is the maximum allowed condition number of the generated $\mathbf{Q}_{Y|D}$ matrix, r is number of domains. "Test acc" is classification accuracy, under the best permutation of the recovered classes, and "$\mathbf{Q}_{Y|D}$ err" is the average entry-wise absolute error in the recovered $\mathbf{Q}_{Y|D}$.

r	Approaches	$\alpha : 0.5,\ \kappa : 200$		$\alpha : 3,\ \kappa : 205$		$\alpha : 10,\ \kappa : 210$	
		Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err
50	SCAN	**0.726**	0.039	**0.745**	0.037	**0.741**	0.032
	DDFA (SI)	0.720	**0.013**	0.632	**0.015**	0.343	**0.022**
60	SCAN	0.755	0.039	0.730	0.037	**0.748**	0.032
	DDFA (SI)	**0.818**	**0.010**	**0.743**	**0.012**	0.578	**0.018**

Table 5: *Full results on FieldGuide-2*. Each entry is produced with the averaged result of 3 different random seeds. With DDFA (RI) we refer to DDFA with randomly initialized backbone. With DDFA (SPI) we refer to DDFA initialized with pretext training adopted by SCAN. Note that in DDFA (SPI), we do not leverage SCAN for clustering. α is the Dirichlet parameter used for generating label marginals in each domain, κ is the maximum allowed condition number of the generated $\mathbf{Q}_{Y|D}$ matrix, r is number of domains.

r	Approaches	$\alpha : 0.5,\ \kappa : 3$		$\alpha : 3,\ \kappa : 5$		$\alpha : 10,\ \kappa : 7$	
		Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err
2	SCAN	0.583	0.508	0.564	0.524	0.577	0.281
	DDFA (RI)	0.715	**0.222**	0.602	0.500	0.622	**0.200**
	DDFA (SPI)	**0.776**	0.241	**0.773**	**0.150**	**0.658**	0.264
3	SCAN	0.589	0.858	0.590	0.458	0.590	0.273
	DDFA (RI)	0.840	0.180	0.660	0.218	0.611	**0.209**
	DDFA (SPI)	**0.960**	**0.055**	**0.830**	**0.148**	**0.693**	0.224
5	SCAN	0.586	0.881	0.576	0.422	0.580	0.234
	DDFA (RI)	0.653	0.126	0.596	0.253	0.576	**0.159**
	DDFA (SPI)	**0.953**	**0.093**	**0.784**	**0.141**	**0.617**	0.258
7	SCAN	0.580	0.777	0.586	0.449	0.587	0.275
	DDFA (RI)	0.757	0.269	0.623	0.177	0.581	**0.161**
	DDFA (SPI)	**0.904**	**0.115**	**0.816**	**0.145**	**0.661**	0.198
10	SCAN	0.582	0.828	0.589	0.374	**0.582**	0.186
	DDFA (RI)	0.692	0.237	0.619	**0.149**	0.534	0.271
	DDFA (SPI)	**0.907**	**0.155**	**0.714**	0.170	**0.582**	**0.164**

Table 6: *Results on FieldGuide-28*. Each entry is produced with the result of a single trial. With DDFA (SPI) we refer to DDFA initialized with pretext training adopted by SCAN. Note that in DDFA (SPI), we do not leverage SCAN for clustering. α is the Dirichlet parameter used for generating label marginals in each domain, κ is the maximum allowed condition number of the generated $\mathbf{Q}_{Y|D}$ matrix, r is number of domains. "Test acc" is classification accuracy, under the best permutation of the recovered classes, and "$\mathbf{Q}_{Y|D}$ err" is the average entry-wise absolute error in the recovered $\mathbf{Q}_{Y|D}$.

r	Approaches	$\alpha : 0.5,\ \kappa : 12$		$\alpha : 3,\ \kappa : 20$		$\alpha : 10,\ \kappa : 28$	
		Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err
28	SCAN	0.281	0.064	0.276	0.059	0.310	0.048
	DDFA (SPI)	**0.547**	**0.036**	**0.310**	**0.034**	**0.314**	**0.036**
37	SCAN	0.300	0.066	0.316	0.059	0.309	0.049
	DDFA (SPI)	**0.760**	**0.028**	**0.521**	**0.032**	**0.326**	**0.041**
42	SCAN	0.279	0.065	0.332	0.059	0.295	0.047
	DDFA (SPI)	**0.670**	**0.032**	**0.471**	**0.037**	**0.408**	**0.031**
47	SCAN	0.285	0.066	0.314	0.062	**0.307**	0.049
	DDFA (SPI)	**0.709**	**0.035**	**0.473**	**0.035**	0.299	**0.039**

F Discussion of Convex Polytope Geometry

The geometric properties of topic modeling for finite, discrete random variables has been explored in depth in related works ([33, 20, 12]). The observation that columns in $\mathbf{Q}_{X|D}$ are convex combinations of columns in $\mathbf{Q}_{X|Y}$ leads to a perspective on identification of the matrix decomposition as identification of the convex polytope in $\mathbb{R}^m$ which encloses all of the columns of $\mathbf{Q}_{X|D}$ (the corners of which correspond to columns of $\mathbf{Q}_{X|Y}$ under certain identifiability conditions).

Here, we briefly discuss an interesting but somewhat different application of convex polytope geometry. Instead of a convex polytope in $\mathbb{R}^m$ with corners as columns of $\mathbf{Q}_{X|Y}$, we concern

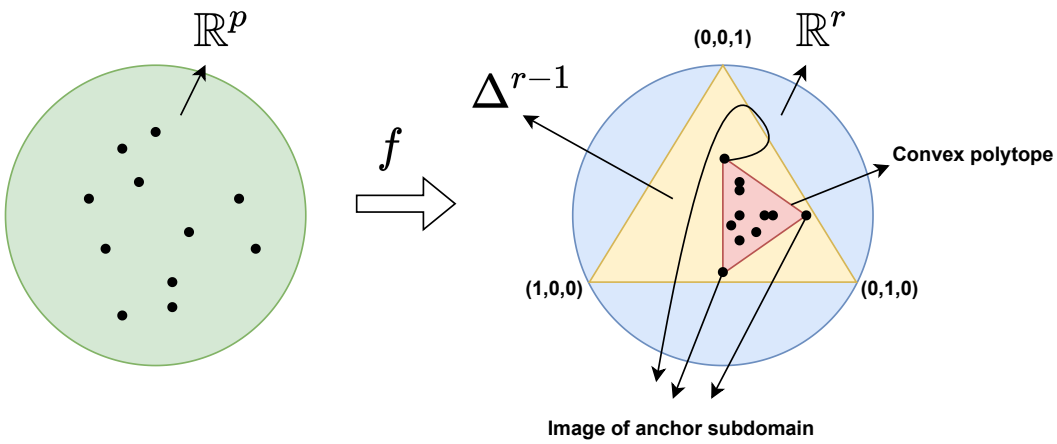

Figure 4: This figure illustrates the case with 3 domains and 3 classes. The oracle domain discriminator maps points from a high-dimensional input space to a $k = 3$ vertex convex polytope (shaded red) embedded in $\Delta^{r-1}, r = 3$ (shaded yellow). The anchor subdomains map to the vertices of this polytope.

ourselves with the convex polytope in $\mathbb{R}^r$ with corners as columns in $\mathbf{Q}_{D|Y}$, which must enclose all values taken by the oracle domain discriminator $f(x)$ for $x \in \mathcal{X}, q(x) > 0$.

Let us assume that Assumptions A.1–A.4 are satisfied. We recall the oracle domain discriminator f which is defined such that $[f(x)]_d = q(d|X = x)$. Let $x \in \mathcal{X} = \mathbb{R}^p$. Now, since the r values $q(d|X = x)$ for $d \in \{1, 2, ..., r\}$ together constitute a categorical distribution, each of these r values lie between 0 and 1, and also their sum adds to 1. Therefore the vector $f(x)$ lies on the simplex Δ^{r-1}. We now express $f(x)$ as a convex combination of the k columns of $\mathbf{Q}_{D|Y}$. We denote these column vectors $\mathbf{Q}_{D|Y}[:, y]$ for each $y \in \mathcal{Y} = [k]$. Note that each such vector also lies in the Δ^{r-1} simplex.

As an intermediate step in the proof of Lemma 3 given in App. A, we showed that each $f(x)$ is a linear combination of these columns of $\mathbf{Q}_{D|Y}$ with coefficients $q(y|X = x)$ for all $y \in \mathcal{Y}$. That is, we can rewrite $f(x) = \mathbf{Q}_{D|Y} \left[Q(Y = 1|X = x) \ldots Q(Y = k|X = x)\right]^\top$

Since the coefficients in the linear combination are probabilities which, taken together, form a categorical distribution, they lie between 0 and 1 and sum to 1. Thus, for all $x \in \mathcal{X}$ with $q(x) > 0$, $f(x)$ can be expressed as a *convex* combination of the columns of $\mathbf{Q}_{D|Y}$. Therefore, for any x with $q(x) > 0$, $f(x)$ lies inside the $k-$vertex convex polytope with corners as the columns of $\mathbf{Q}_{D|Y}$ (which are linearly independent by Lemma 6). This polytope is embedded in Δ^{r-1}.

Now consider x in an anchor sub-domain, that is $x \in A_y$ for some $y \in \mathcal{Y}$. We know that if $q(x) > 0$, $q(y|X = x) = 1$, $q(y'|X = x) = 0$ for all $y' \neq y$ (Lemma 5). Since the $q(y|X = x)$ are now one-hot, we have that $f(x) = \mathbf{Q}_{D|Y}[:, y]$ for $x \in A_y$. In words, this means that $f(A_y)$ is precisely the yth column of $\mathbf{Q}_{D|Y}$. It follows that the domain discriminator maps each of the k anchor sub-domains exactly to a unique vertex of the polytope. The situation is described in Fig. 4.

We could now recover the columns of $\mathbf{Q}_{D|Y}$, up to permutation, with the following procedure:

1. Push all $x \in \mathcal{X}$ through f.
2. Find the minimum volume convex polytope that contains the resulting density of points on the simplex. The vectors that compose the vertices of this polytope are the columns of $\mathbf{Q}_{D|Y}$, up to permutation.

Note that from Assumption A.4, we are guaranteed to have a region of the input space with at least $\epsilon > 0$ mass that gets mapped to each of the vertices when carrying out step (i). Therefore, our discovered minimum volume polytope must enclose all of these vertices. Since no mass will exist outside of the true polytope, requiring a minimum volume polytope will ensure that the recovered polytope fits the true polytope's vertices precisely (as any extraneous volume outside of the true

polytope must be eliminated). Then step (ii) recovers $\mathbf{Q}_{D|Y}$, up to permutation of columns. Having recovered $\mathbf{Q}_{D|Y}$, we can use Lemmas 1 and 2 to recover $q(y|x, d)$.

This procedure is a geometric alternative to the clustering approach outlined in Algorithm 1. In practice, fitting a convex hull around the outputs of a noisy, non-oracle estimated domain discriminator may be computationally expensive, and noise may lead this sensitive procedure to fail to recover the true vertices.

G Ablation Study on Number of Clusters

We conduct an ablation on the choice of m, the parameter indicating how many clusters to find in the $q(d|x)$ space. We use the CIFAR-20 dataset with 20 domains and employ DDFA (SI) and SCAN models, following the same hyperparameters as outlined in App. D, except for modifying the choice of m for DDFA (SI). Results are obtained as the average of three random seeds.

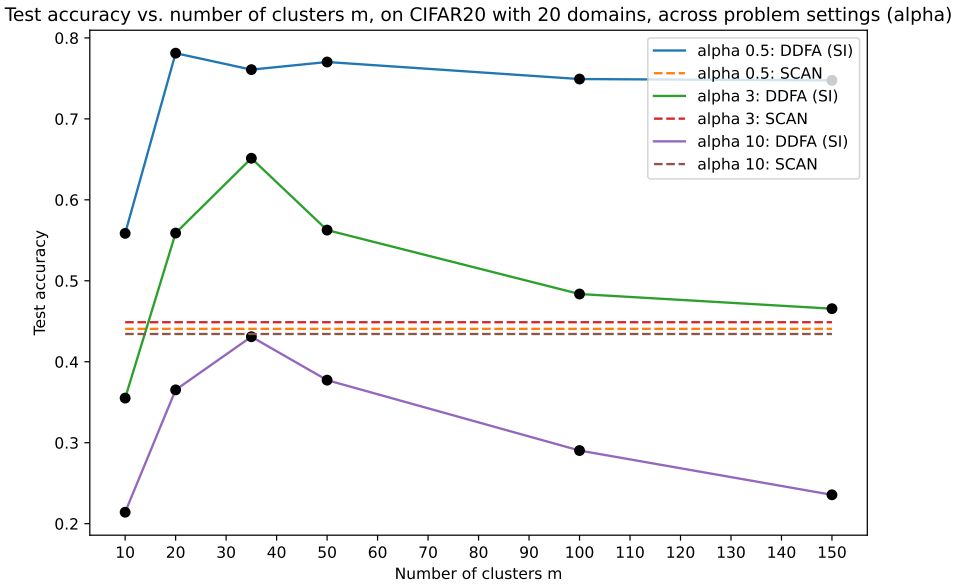

Figure 5: Test accuracy of DDFA (SI) approach and SCAN baseline on CIFAR20 with 20 domains, while modifying the number of clusters m for DDFA (SI). The choice of α roughly modifies the difficulty of the problem, where small α is easier. We note that typically we require choice of $m \geqslant k$. We portray one datapoint where this constraint is violated, and $m = 10$. Black dots indicate tested values of m, and lines are plotted only to show the trend. Larger accuracy is better.

The number of true classes is 20 in CIFAR-20. As seen in Fig. 5, when m is chosen to be 10, violating the typical constraint that $m \geqslant k$, we can still solve for the solution, but we get poor performance, seeing a drop in accuracy as much as 20% from a better-chosen value of m. Choosing m directly equal to or slightly larger than k provide the best performance, with a slope-off in performance at very large m.

The trend is roughly mirrored in Fig. 6, which shows how the reconstruction error changes over the same variation in m. Under all settings, using $m = 10 < k$ clusters provides a poor reconstruction, while the best reconstruction is found with m roughly equal to k or slightly larger. Performance degrades as m grows very large, although the effect is very slight for the alpha = 0.5 setting.

Intuitively, these results show that breaking the $m \geqslant k$ condition not only violates the theoretical identifiability, but also leads to poorer empirical performance; choosing very large m can also lead to degraded performance, likely due to propagation of inaccuracies in the finite-sample estimation of the $\widehat{Q}_{c(X)|D}$ matrix.

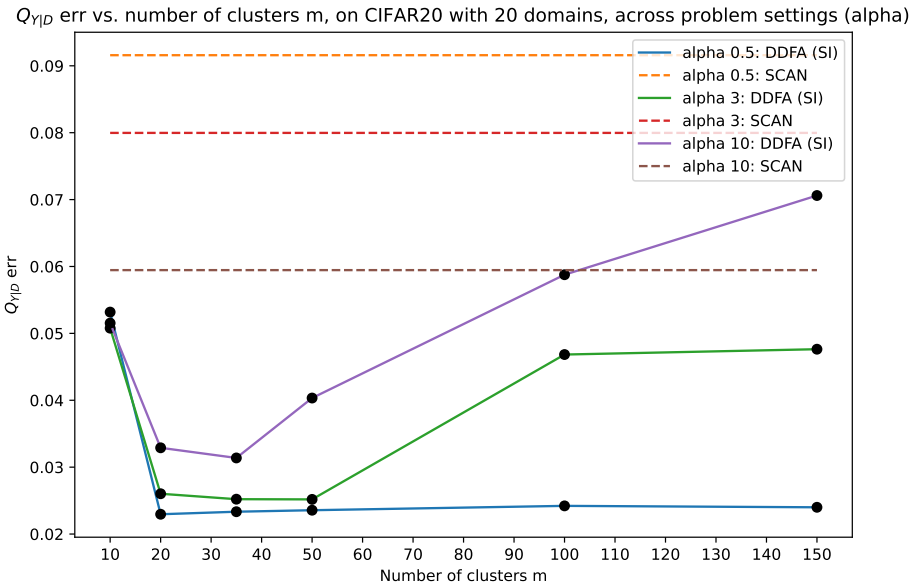

Figure 6: Element-wise average absolute $Q_{Y|D}$ reconstruction error of DDFA (SI) approach and SCAN baseline on CIFAR-20 with 20 domains, while modifying the number of clusters m for DDFA (SI). The choice of α roughly modifies the difficulty of the problem, where small α is easier. We note that typically we require choice of $m \geqslant k$. We portray one datapoint where this constraint is violated, and $m = 10$. Black dots indicate tested values of m, and lines are plotted only to show the trend. Smaller error is better.

H Ablation Study with a Naïve Feature Space

One might ask whether the semantic meaning of the domain discrimination space is necessary in DDFA; might we exchange the domain discrimination step in Algorithm 1 for a naive step in which we simply pass the input through an arbitrary feature extractor and then proceed to clustering in this space?

The first remark we make is that the domain discriminator does not purely provide a clustering representation; its semantic meaning is also important for the computation of the final domain-adjusted $p_d(y|x)$ prediction, as a reliable estimate of $q(d|x)$, in conjunction with the estimate of the $Q_{Y|D}$ matrix, allow us to estimate $p_d(y|x)$ via Algorithm 2. Without this semantic meaning, our class prediction can be based only on a coarse prediction at the level of the *cluster*, not the individual datapoint.

However, if we are still determined to use an alternate representation, it is indeed possible to do so. We illustrate a variant of DDFA using a naïve representation in Algorithms 3 and 4, and then evaluate this procedure on CIFAR-20 as an ablative study on DDFA. We compare naïve results with standard DDFA results, and with a traditional SCAN baseline, in Table 7.

Naïve Representation Variant of DDFA The only major changes from the original DDFA for Algorithm 3 are the removal of the need to train any $\widehat{f}$ domain discriminator, the use of the arbitrary representation space ϕ before clustering, and the reliance on the output of the clustering discretization function c as well as $\widehat{\mathbf{Q}}_{c(X)|Y}$ (which are both discarded in the original procedure). We need c and $\widehat{\mathbf{Q}}_{c(X)|Y}$ because in Algorithm 4 we will use them for domain-adjusted class prediction.

Algorithm 4 includes significant changes from the DDFA procedure. Since we do not have the estimate of $q(d|x)$ to use, we cannot directly reason about how different locations in the representation space induced by ϕ correspond to different probabilities of class labels. However, because we have $\widehat{\mathbf{Q}}_{c(X)|Y}$ and $\widehat{\mathbf{Q}}_{Y|D}$, two outputs of the NMF decomposition in Algorithm 3, we can calculate a

Algorithm 3 DDFA (Naïve) Training

input $k \geqslant 1, r \geqslant k, \{(x_i, d_i)\}_{i\in[n]} \sim q(x, d),$ A naive representation function ϕ from $\mathbb{R}^p \to \mathbb{R}^r$

1: Push all $\{x_i\}_{i\in[n]}$ through ϕ.
2: Train clustering algorithm on the n points $\{\phi(x_i)\}_{i\in[n]},$ obtain m clusters.
3: $c(x_i) \leftarrow$ Cluster id of $\phi(x_i)$
4: $\widehat{q}(c(X) = a|D = b) \leftarrow \frac{\sum_{i\in[n]} \mathbb{I}[c(x_i)=a,\ d_i=b]}{\sum_{j\in[n]} \mathbb{I}[d_j=b]}$
5: Populate $\widehat{\mathbf{Q}}_{c(X)|D}$ as $[\widehat{\mathbf{Q}}_{c(X)|D}]_{a,b} \leftarrow \widehat{q}(c(X) = a|D = b)$
6: $\widehat{\mathbf{Q}}_{c(X)|Y}, \widehat{\mathbf{Q}}_{Y|D} \leftarrow$ NMF $(\widehat{\mathbf{Q}}_{c(X)|D})$

output $\widehat{\mathbf{Q}}_{c(X)|Y}, \widehat{\mathbf{Q}}_{Y|D},$ clustering discretization function c

Algorithm 4 DDFA (Naïve) Prediction

input $\widehat{\mathbf{Q}}_{c(X)|Y}, \widehat{\mathbf{Q}}_{Y|D},$ clustering discretization function $c, (x', d') \sim q(x, d)$

1: Pass x' through c to get cluster id $c(x')$.
2: $\widehat{q}(c(x')|Y = y'') \leftarrow [\widehat{\mathbf{Q}}_{c(X)|Y}]_{c(x'),y''}$ for all y''
3: $\widehat{q}(y''|D = d') \leftarrow [\widehat{\mathbf{Q}}_{Y|D}]_{y'',d'}$ for all y''
4: $\widehat{q}(y|c(X) = c(x'), D = d') \leftarrow \dfrac{\widehat{q}(c(x')|Y = y)\widehat{q}(y|D = d')}{\sum\limits_{y''\in\mathcal{Y}} \widehat{q}(c(x')|Y = y'')\widehat{q}(y''|D = d')}$
5: $y_{\text{pred}} \leftarrow \arg\max_{y\in[k]} \widehat{q}(y|c(X) = c(x'), D = d')$

output : $\widehat{q}(y|c(X) = c(x'), D = d') = \widehat{p}_{d'}(y|c(x')), y_{\text{pred}}$

coarse prediction over labels y for each cluster, and then assign the same prediction to each point in that cluster. To obtain the closed-form for this coarse prediction $\widehat{p}_d(y|c(x))$ used in Algorithm 4, we use the following derivation:

$$\begin{aligned}\widehat{p}_d(y|c(x)) = \widehat{q}(y|d, c(x)) &= \frac{\widehat{q}(c(x)|y, d)\widehat{q}(y, d)}{\widehat{q}(d, c(x))} \\ &= \frac{\widehat{q}(c(x)|y, d)\widehat{q}(y, d)}{\sum\limits_{y''\in\mathcal{Y}} \widehat{q}(c(x)|y'', d)\widehat{q}(y'', d)}\end{aligned}$$

By label shift, $q(c(x)|y, d) = q(c(x)|y)$, then

$$\begin{aligned}\widehat{p}_d(y|c(x)) = \widehat{q}(y|d, c(x)) &= \frac{\widehat{q}(c(x)|y)\widehat{q}(y, d)}{\sum\limits_{y''\in\mathcal{Y}} \widehat{q}(c(x)|y'')\widehat{q}(y'', d)} \\ &= \frac{\widehat{q}(c(x)|y)\widehat{q}(y|d)\widehat{q}(d)}{\sum\limits_{y''\in\mathcal{Y}} \widehat{q}(c(x)|y'')\widehat{q}(y''|d)\widehat{q}(d)} \\ &= \frac{\widehat{q}(c(x)|y)\widehat{q}(y|d)(1/r)}{\sum\limits_{y''\in\mathcal{Y}} \widehat{q}(c(x)|y'')\widehat{q}(y''|d)(1/r)} \\ &= \frac{\widehat{q}(c(x)|y)\widehat{q}(y|d)}{\sum\limits_{y''\in\mathcal{Y}} \widehat{q}(c(x)|y'')\widehat{q}(y''|d)}\end{aligned}$$

Combining Algorithms 3 and 4 allows us to empirically evaluate the behavior of an ablation on DDFA which does not use any domain discriminator. For a reasonable comparison, we need a meaningful naïve representation space. We use a SCAN pretrain backbone for ResNet-18, and remove the last linear layer in the ResNet-18 backbone in order to expose a 512-dimension representation space. Since clustering in high-dimensional spaces often performs poorly, we also map this 512-dimension representation down to only r (the number of domains) dimensions using two different common dimensionality reduction methods: Independent Component Analysis (ICA) [35] and Principal

Component Analysis (PCA) [59]. These smaller-dimension clustering problems provide a closer comparison to the dimensionality of the clustering problem in the DDFA (SI) procedure, for which we employ m clusters. Note: we use ICA and PCA implementations from scikit-learn [48].

SCAN, DDFA (RI), and DDFA (SI) experiment details are the same as explained in App. D; in fact these are the same results presented for CIFAR-20 in Sec. 6.

In general, we can see that all variants of the Naïve/ablated DDFA procedure perform worse than DDFA (SI) in all problem settings, over both metrics of interest. All variants of the Naïve procedure are also worse than DDFA (RI) in both metrics of interest when α is 0.5 or 3, although they match or outpace it slightly in some of the hardest settings ($\alpha = 10$). It is worth pointing out that this is predominantly due to the fact that DDFA (RI) performs notably poorly at harder settings.

The SCAN baseline outpaces these Naïve approaches, in terms of classification accuracy, in all problem settings. However, the Naïve approaches always achieve better $Q_{Y|D}$ reconstruction error.

Table 7: *Extended Results on CIFAR-20.* Each entry is produced with the averaged result of 3 different random seeds. With DDFA (RI) we refer to DDFA with randomly initialized backbone. With DDFA (SI) we refer to DDFA's backbone initialized with SCAN. Note that in DDFA (SI), we do not leverage SCAN for clustering. With Naïve we refer to an ablation in which DDFA's domain discriminator is replaced with the SCAN pretrained backbone, with its final linear layer removed so that its output is a 512-dimension unsupervised representation space. With Naïve (ICA) and Naïve (PCA) we refer to similar ablations in which the activations from the second-to-last layer of SCAN network are mapped to r-dimensional space with ICA and PCA respectively. α is the Dirichlet parameter used for generating label marginals in each domain, κ is the maximum allowed condition number of the generated $\mathbf{Q}_{Y|D}$ matrix, r is number of domains.

r	Approaches	$\alpha : 0.5,\ \kappa : 8$		$\alpha : 3,\ \kappa : 12$		$\alpha : 10,\ \kappa : 20$	
		Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err	Test acc	$\mathbf{Q}_{Y\|D}$ err
20	SCAN	0.439	0.092	0.446	0.079	**0.434**	0.060
	DDFA (RI)	0.517	0.042	0.336	0.045	0.163	0.057
	DDFA (SI)	**0.784**	**0.023**	**0.593**	**0.027**	0.390	**0.034**
	Naïve	0.231	0.075	0.156	0.065	0.116	0.054
	Naïve (ICA)	0.225	0.073	0.137	0.070	0.105	0.056
	Naïve (PCA)	0.204	0.076	0.141	0.065	0.108	0.055
25	SCAN	0.438	0.090	0.441	0.078	0.438	0.060
	DDFA (RI)	0.489	0.049	0.292	0.049	0.075	0.081
	DDFA (SI)	**0.837**	**0.020**	**0.669**	**0.025**	**0.487**	**0.030**
	Naïve	0.224	0.078	0.165	0.064	0.112	0.055
	Naïve (ICA)	0.204	0.076	0.146	0.063	0.100	0.057
	Naïve (PCA)	0.207	0.078	0.135	0.067	0.105	0.054
30	SCAN	0.432	0.094	0.457	0.077	0.431	0.059
	DDFA (RI)	0.512	0.046	0.299	0.048	0.087	0.077
	DDFA (SI)	**0.820**	**0.022**	**0.743**	**0.021**	**0.543**	**0.028**
	Naïve	0.208	0.077	0.152	0.066	0.122	0.051
	Naïve (ICA)	0.197	0.076	0.134	0.064	0.097	0.056
	Naïve (PCA)	0.200	0.078	0.148	0.065	0.114	0.051