Towards Inheritable Models for Open-Set Domain Adaptation

Jogendra Nath Kundu*  Naveen Venkat*  Ambareesh Revanur  Rahul M V  R. Venkatesh Babu
Video Analytics Lab, CDS, Indian Institute of Science, Bangalore

Abstract

There has been a tremendous progress in Domain Adaptation (DA) for visual recognition tasks. Particularly, open-set DA has gained considerable attention wherein the target domain contains additional unseen categories. Existing open-set DA approaches demand access to a labeled source dataset along with unlabeled target instances. However, this reliance on co-existing source and target data is highly impractical in scenarios where data-sharing is restricted due to its proprietary nature or privacy concerns. Addressing this, we introduce a practical DA paradigm where a source-trained model is used to facilitate adaptation in the absence of the source dataset in future. To this end, we formalize knowledge inheritability as a novel concept and propose a simple yet effective solution to realize inheritable models suitable for the above practical paradigm. Further, we present an objective way to quantify inheritability to enable the selection of the most suitable source model for a given target domain, even in the absence of the source data. We provide theoretical insights followed by a thorough empirical evaluation demonstrating state-of-the-art open-set domain adaptation performance. Our code is available at https://github.com/val-iisc/inheritune.

1. Introduction

Deep neural networks perform remarkably well when the training and the testing instances are drawn from the same distributions. However, they lack the capacity to generalize in the presence of a domain-shift [42] exhibiting alarming levels of dataset bias or domain bias [45]. As a result, a drop in performance is observed at test time if the training data (acquired from a source domain) is insufficient to reliably characterize the test environment (the target domain). This challenge arises in several Computer Vision tasks [32, 25, 18] where one is often confined to a limited array of available source datasets, which are practically inadequate to represent a wide range of target domains. This has motivated a line of Unsupervised Domain Adaptation (UDA) works that aim to generalize a model to an unlabeled target domain, in the presence of a labeled source domain.

In this work, we study UDA in the context of image recognition. Notably, a large body of UDA methods is inspired by the potential of deep CNN models to learn transferable representations [52]. This has formed the basis of several UDA works that learn domain-agnostic feature representations [26, 44, 48] by aligning the marginal distributions of the source and the target domains in the latent feature space. Several other works learn domain-specific representations via independent domain transformations [47, 5, 32] to a common latent space on which the classifier is learned. The latent space alignment of the two domains permits the reuse of the source classifier for the target domain. These methods however operate under the assumption of a shared label-set ($C_s = C_t$) between the two domains (closed-set). This restricts their real-world applicability where a target domain often contains additional unseen categories beyond those found in the source domain.

Recently, open-set DA [35, 39] has gained much attention, wherein the target domain is assumed to have unshared categories ($C_s \subset C_t$), a.k.a category-shift. Target instances from the unshared categories are assigned a single unknown label [35] (see Fig. 1B). Open-set DA is more challenging,
since a direct application of distribution alignment (e.g. as in closed-set DA [20, 44]) reduces the model’s performance due to the interference from the unshared categories (an effect known as negative-transfer [34]). The success of open-set DA relies not only on the alignment of shared classes, but also on the ability to mitigate negative-transfer. State-of-the-art methods such as [53] train a domain discriminator using the source and the target data to detect and reject target instances that are out of the source distribution, thereby minimizing the effect of negative-transfer.

In summary, the existing UDA methods assume access to a labeled source dataset to obliquely receive a task-specific supervision during adaptation. However, this assumption of co-existing source and target datasets poses a significant constraint in the modern world, where coping up with strict digital privacy and copyright laws is of prime importance [33]. This is becoming increasingly evident in modern corporate dealings, especially in the medical and biometric industries, where a source organization (the model vendor) is often restricted to share its proprietary or sensitive data, alongside a pre-trained model to satisfy the client’s specific deployment requirements [7, 14]. Likewise, the client is prohibited to share private data to the model vendor [17]. Certainly, the collection of existing open-set DA solutions is inadequate to address such scenarios.

Thus, there is a strong motivation to develop practical UDA algorithms which make no assumption about data-exchange between the vendor and the client. One solution is to design self-adaptive models that effectively capture the task-specific knowledge from the vendor’s source domain and transfer this knowledge to the client’s target domain. We call such models as inheritable models, referring to their ability to inherit and transfer knowledge across domains without accessing the source domain data. It is also essential to quantify the knowledge inheritability of such models. Given an array of inheritable models, this quantification will allow a client to flexibly choose the most suitable model for the client’s specific target domain.

Addressing these concerns, in this work we demonstrate how a vendor can develop an inheritable model, which can be effectively utilized by the client to perform unsupervised adaptation to the target domain, without any data-exchange. To summarize, our prime contributions are:

- We propose a practical UDA scenario by relaxing the assumption of co-existing source and target domains, called as the vendor-client paradigm.
- We propose inheritable models to realize vendor-client paradigm in practice and present an objective measure of inheritability, which is crucial for model selection.
- We provide theoretical insights and extensive empirical evaluation to demonstrate state-of-the-art open-set DA performance using inheritable models.

2. Related Work

Closed-set DA. Assuming a shared label space ($\mathcal{C}_s = \mathcal{C}_t$), the central theme of these methods is to minimize the distribution discrepancy. Statistical measures such as MMD [51, 27, 28], CORAL [44] and adversarial feature matching techniques [10, 48, 46, 47, 40] are widely used. Recently, domain specific normalization techniques [23, 5, 4, 37] has started gaining attention. However, due to the shared labelset assumption these methods are highly prone to negative-transfer in the presence of new target categories.

Open-set DA. ATI-λ [35] assigns a pseudo class label, or an unknown label, to each target instance based on its distance to each source cluster in the latent space. OSVM [15] uses a class-wise confidence threshold to classify target instances into the source classes, or reject them as unknown. OSBP [39] and STA [24] align the source and target features through adversarial feature matching. However, both OSBP and ATI-λ are hyperparameter sensitive and are prone to negative-transfer. In contrast, STA [24] learns a separate network to obtain instance-level weights for target samples to avoid negative-transfer and achieves state-of-the-art results. All these methods assume the co-existence of source and target data, while our method makes no such assumption and hence has a greater practical significance.

Domain Generalization. Methods such as [9, 21, 8, 22, 31, 16] largely rely on an arbitrary number of co-existing source domains with shared label sets, to generalize across unseen target domains. This renders them impractical when there is an inherent category-shift among the data available with each vendor. In contrast, we tackle the challenging open-set scenario by learning on a single source domain.

Data-free Knowledge Distillation (KD). In a typical KD setup [13], a student model is learned to match the teacher model’s output. Recently, DFKD [29] and ZSKD [33] demonstrated knowledge transfer to the student when the teacher’s training data is not available. Our work is partly inspired by their data-free ideology. However, our work differs from KD in two substantial ways; 1) by nature of the KD algorithm, it does not alleviate the problem of domain-shift, since any domain bias exhibited by the teacher will be passed on to the student, and 2) KD can only be performed for the task which the teacher is trained on, and is not designed for recognizing new (unknown) target categories in the absence of labeled data. Handling domain-shift and category-shift simultaneously is necessary for any open-set DA algorithm, which is not supported by these methods.

Our formulation of an inheritable model for open-set DA is much different from prior arts - not only is it robust to negative-transfer but also facilitates domain adaptation in the absence of data-exchange.
3. Unsupervised Open-Set Domain Adaptation

In this section, we formally define the vendor-client paradigm and inheritability in the context of unsupervised open-set domain adaptation (UODA).

3.1. Preliminaries

Notation. Given an input space $\mathcal{X}$ and output space $\mathcal{Y}$, the source and target domains are characterized by the distributions $p$ and $q$ on $\mathcal{X} \times \mathcal{Y}$ respectively. Let $p_s$, $q_x$ denote the marginal input distributions and $p_{y|x}$, $q_{y|x}$ denote the conditional output distribution of the two domains. Let $C_s$, $C_t \subset \mathcal{Y}$ denote the respective label sets for the classification tasks ($C_s \subset C_t$). In the UODA problem, a labeled source dataset $D_s = \{(x_s, y_s) : x_s \sim p_x, y_s \sim p_{y|x}\}$ and an unlabeled target dataset $D_t = \{x_t : x_t \sim q_x\}$ are considered. The goal is to assign a label for each target instance $x_t$, by predicting the class for those in shared classes ($C_t^s = C_s \cap C_t$), and an ‘unknown’ label for those in unshared classes ($C_t^{uk} = C_t \setminus C_s$). For simplicity, we denote the distributions of target-shared and target-unknown instances as $q^{sh}$ and $q^{uk}$ respectively. We denote the model trained on the source domain as $h_s$ (source predictor) and the model adapted to the target domain as $h_t$ (target predictor).

Performance Measure. The primary goal of UODA is to improve the performance on the target domain. Hence, the performance of any UODA algorithm is measured by the error rate of target predictor $h_t$, i.e., $\xi_q(h_t)$ which is empirically estimated as $\hat{\xi}_q(h_t) = \mathbb{P}_{(x_t, y_t) \sim q}\{h_t(x_t) \neq y_t\}$, where $\mathbb{P}$ is the probability estimated over the instances $D_t$.

3.2. The vendor-client paradigm

The central focus of our work is to realize a practical DA paradigm which is fundamentally viable in the absence of the co-existence of the source and target domains. With this intent, we formalize our DA paradigm.

Definition 1 (vendor-client paradigm). Consider a vendor with access to a labeled source dataset $D_s$ and a client having unlabeled instances $D_t$ sampled from the target domain. In the vendor-client paradigm, the vendor learns a source predictor $h_s$ using $D_s$ to model the conditional $p_{y|x}$, and shares $h_s$ to the client. Using $h_s$ and $D_t$, the client learns a target predictor $h_t$ to model the conditional $q_{y|x}$.

This paradigm satisfies the two important properties; 1) it does not assume data-exchange between the vendor and the client which is fundamental to cope up with the dynamically reforming digital privacy and copyright regulations and, 2) a single vendor model can be shared with multiple clients thereby minimizing the effort spent on source training. Thus, this paradigm has a greater practical significance than the traditional UDA setup where each adaptation step requires an additional supervision from the source data [24, 39]. Following this paradigm, our goal is to realize the conditions on which one can successfully learn a target predictor. To this end, we formalize the inheritability of task-specific knowledge of the source-trained model.

3.3. Inheritability

We define an inheritable model from the perspective of learning a predictor ($h_t$) for the target task. Intuitively, given a hypothesis class $\mathcal{H} \subseteq \{h : \mathcal{X} \rightarrow \mathcal{Y}\}$, an inheritable model $h_s$ should be sufficient (i.e., in the absence of source domain data) to learn a target predictor $h_t$ whose performance is close to that of the best predictor in $\mathcal{H}$.

Definition 2 (Inheritability criterion). Let $\mathcal{H} \subseteq \{h : \mathcal{X} \rightarrow \mathcal{Y}\}$ be a hypothesis class, $\epsilon > 0$, and $\delta \in (0, 1)$. A source predictor $h_s : \mathcal{X} \rightarrow \mathcal{Y}$ can be learned using an unlabeled target sample $D_t = \{x_t \sim q_x\}$ when given access to the parameters of $h_s$, such that, with probability at least $(1 – \delta)$ the target error of $h_t$ does not exceed that of the best predictor in $\mathcal{H}$ by more than $\epsilon$. Formally,

$$\mathbb{P}[\xi_q(h_t) \leq \xi_q(\mathcal{H}) + \epsilon \mid h_s, D_t] \geq 1 – \delta$$

where, $\xi_q(\mathcal{H}) = \min_{h \in \mathcal{H}} \xi_q(h)$ and $\mathbb{P}$ is computed over the choice of sample $D_t$. This definition suggests that an inheritable model is capable of reliably transferring the task-specific knowledge to the target domain in the absence of the source data, which is necessary for the vendor-client paradigm. Given this definition, a natural question is, how to quantify inheritability of a vendor model for the target task. In the next Section, we address this question by demonstrating the design of inheritable models for UODA.

4. Approach

How to design inheritable models? There can be several ways, depending upon the task-specific knowledge required by the client. For instance, in UODA, the client must effectively learn a classifier in the presence of both domain-shift and category-shift. Here, not only is the knowledge of class-separability essential, but also the ability to detect new target categories as unknown is vital to avoid negative-transfer. By effectively identifying such challenges, one can develop inheritable models for tasks that require vendor’s dataset. Here, we demonstrate UODA using an inheritable model.

4.1. Vendor trains an inheritable model

In UODA, the primary challenge is to tackle negative-transfer. This challenge arises due to the overconfidence issue [19] in deep models, where unknown target instances are confidently predicted into the shared classes, and thus get aligned with the source domain. Methods such as [53] tend to avoid negative-transfer by leveraging a domain discriminator to assign a low instance-level weight for potentially unknown target instances during adaptation. However,
solutions such as a domain discriminator are infeasible in the absence of data-exchange between the vendor and the client. Thus, an inheritable model should have the ability to characterize the source distribution, which will facilitate the detection of unknown target instances during adaptation. Following this intuition, we design the architecture.

a) Architecture. As shown in Fig. 2A, the feature extractor \( F_s \) comprises of a backbone CNN model \( M_s \) and fully connected layers \( E_s \). The classifier \( G \) contains two sub-modules, a source classifier \( G_s \) with \([C_s] \) classes, and an auxiliary out-of-distribution (OOD) classifier \( G_n \) with \( K \) classes accounting for the ‘negative’ region not covered by the source distribution (Fig. 3C). The output \( \hat{y}_s \) for each input \( x_s \) is obtained by concatenating the outputs of \( G_s \) and \( G_n \) (i.e. concatenating \( G_s(F_s(x_s)) \) and \( G_n(F_s(x_s)) \)) followed by softmax activation. This equips the model with the ability to capture the class-separability knowledge (in \( G_s \)) and to detect OOD instances (via \( G_n \)). This setup is motivated by the fact that the overconfidence issue can be addressed by minimizing the classifier’s confidence for OOD instances [19]. Accordingly, the confidence of \( G_s \) is maximized for in-distribution (source) instances, and minimized for OOD instances (by maximizing the confidence of \( G_n \)).

b) Dataset preparation. To effectively learn OOD detection, we augment the source dataset with synthetically generated negative instances, i.e. \( D_n = \{(u_n, y_n) : u_n \sim r_u, y_n \sim r_{y|u} \} \), where \( r_u \) and \( r_{y|u} \) are the marginal latent space distribution and the conditional output distribution of the negative instances respectively. We use \( D_n \) to model the low source-density region as out-of-distribution (see Fig. 3C). To obtain \( D_n \), a possible approach explored by [19] could be to use a GAN framework to generate ‘boundary’ samples. However, this is computationally intensive and introduces additional parameters for training. Further, we require these negative samples to cover a large portion of the OOD region. This eliminates a direct use of linear interpolation techniques such as mixup [55, 50] which result in features generated within a restricted region (see Fig. 3A). Indeed, we propose an efficient way to generate OOD samples, which we call the feature-splicing technique.

Feature-splicing. It is widely known that in deep CNNs, higher convolutional layers specialize in capturing class-discriminative properties [54]. For instance, [56] assigns each filter in a high conv-layer with an object part, demonstrating that each filter learns a different class-specific trait. As a result of this specificity, especially when a rectified activation function (e.g. ReLU) is used, feature maps receive a high activation whenever the learned class-specific trait is observed in the input [6]. Consequently, we argue that, by suppressing such high activations, we obtain features devoid of the properties specific to the source classes and hence would more accurately represent the OOD samples. Then, enforcing a low classifier confidence for these samples can mitigate the overconfidence issue.

Feature-splicing is performed by replacing the top-\( d \) percentile activations, at a particular feature layer, with the corresponding activations pertaining to an instance belonging to a different class (see Fig. 3B). Formally,

\[
u_n = \phi_d(u_s^{c_i}, u_s^{c_j}) \quad \text{for} \quad c_i, c_j \in C_s, c_i \neq c_j
\]

where, \( u_s^{c_i} = M_s(x_s^{c_i}) \) for a source image \( x_s^{c_i} \) belonging to class \( c_i \), and \( \phi_d \) is the feature-splicing operator which replaces the top-\( d \) percentile activations in the feature \( u_s^{c_i} \) with the corresponding activations in \( u_s^{c_j} \) as shown in Fig. 3B (see Suppl. for algorithm). This process results in a feature which is devoid of the class-specific traits, but lies near the source distribution. To label these negative instances, we perform a \( K \)-means clustering and assign a unique negative class label to each cluster of samples. By training the auxiliary classifier \( G_n \) to discriminate these samples into these \( K \) negative classes, we mitigate the overconfidence issue as stated earlier. We found feature-splicing to be effective in practice. See Suppl. for other techniques that we explored.

c) Training procedure. We train the model in two steps. First, we pre-train \( \{F_s, G_s\} \) using source data \( D_s \) by employing the standard cross-entropy loss,

\[
\mathcal{L}_b = \mathcal{L}_{CE}(\sigma(G_s(F_s(x_s))), y_s)
\]

where, \( \sigma \) is the softmax activation function. Next, we freeze the backbone model \( M_s \), and generate negative instances \( D_n = \{(u_n, y_n)\} \) by performing feature-splicing using source features at the last layer of \( M_s \). We then continue the training of the modules \( \{E_s, G_s, G_n\} \) using supervision from both \( D_s \) and \( D_n \),

\[
\mathcal{L}_s = \mathcal{L}_{CE}(\hat{y}_s, y_s) + \mathcal{L}_{CE}(\hat{y}_n, y_n)
\]

where, \( \hat{y}_s = \sigma(G(F_s(x_s))) \) and \( \hat{y}_n = \sigma(G(E_s(u_n))) \), and the output of \( G \) is obtained as described in Sec. 4.1a (and depicted in Fig. 2). The joint training of \( G_s \) and \( G_n \) allows the model to capture the class-separability knowledge (in \( G_s \)) while characterizing the negative region (in \( G_n \)).
which renders a superior knowledge inheritability. Once the inheritable model $h_s = \{F_s, G\}$ is trained, it is shared to the client for performing UODA.

### 4.2. Client adapts to the target domain

With a trained inheritable model $h_s$ in hand, the first task is to measure the degree of domain-shift to determine the inheritability of the vendor’s model. This is followed by a selective adaptation procedure which encourages shared classes to align while avoiding negative-transfer.

**a) Quantifying inheritability.** In presence of a small domain-shift, most of the target-shared instances (pertaining to classes in $C_s^{th}$) will lie close to the high source-density regions in the latent space (e.g. Fig. 3E). Thus, one can rely on the class-separability knowledge of $h_s$ to predict target labels. However, this knowledge becomes less reliable with increasing domain-shift as the concentration of target-shared instances near the high density regions decreases (e.g. Fig. 3D). Thus, the inheritability of $h_s$ for the target task would decrease with increasing domain-shift. Moreover, target-unknown instances (pertaining to classes in $C_s^{th}$) are more likely to lie in the low source-density region than target-shared instances. With this intuition, we define an inheritability metric $w$ which satisfies,

$$
\mathbb{E}_{x_s \sim p_s} w(x_s) \geq \mathbb{E}_{x_t \sim q_s^{th}} w(x_t) \geq \mathbb{E}_{x_t \sim q_s^{th}} w(x_t) \quad (5)
$$

We leverage the classifier confidence to realize an instance-level measure of inheritability as follows,

$$
w(x) = \max_{c_i \in C_s} \sigma(G(F_s(x)))_{c_i} \quad (6)
$$

where $\sigma$ is the softmax activation function. Note that although softmax is applied over the entire output of $G$, $\max$ is evaluated over those corresponding to $G_s$ (shaded in blue in Fig. 2). We hypothesize that this measure follows Eq. 5, since, the source instances (in the high density region) receive the highest $G_s$ confidence, followed by target-shared instances (some of which are away from the high density region), while the target-unknown instances receive the least confidence (many of which lie away from the high density regions). Extending the instance-level inheritability, we define a model inheritability over the entire target dataset as,

$$
\mathcal{I}(h_s, D_s, D_t) = \frac{\text{mean}_{x_s \in D_s} w(x_s)}{\text{mean}_{x_t \in D_t} w(x_t)} \quad (7)
$$

A higher $\mathcal{I}$ arises from a smaller domain-shift implying a greater inheritability of task-specific knowledge (e.g. class-separability for UODA) to the target domain. Note that $\mathcal{I}$ is a constant for a given triplet $\{h_s, D_s, D_t\}$ and the value of the denominator in Eq. 7 can be obtained from the vendor.

**b) Adaptation procedure.** For performing adaptation to the target domain, we learn a target-specific feature extractor $F_t = \{M_t, E_t\}$ as shown in Fig. 2B (similar in architecture to $F_s$). $F_t$ is initialized from the source feature extractor $F_s = \{M_s, E_s\}$, and is gradually trained to selectively align the shared classes in the pre-classifier space (input to $G$) to avoid negative-transfer. The adaptation involves two processes - inherit (to acquire the class-separability knowledge) and tune (to avoid negative-transfer).

**Inherit.** As described in Sec. 4.2a, the class-separability knowledge of $h_s$ is reliable for target samples with high $w$. Subsequently, we choose top-$k$ percentile target instances based on $w(x_t)$ and obtain pseudo-labels using the source model, $y_p = \arg\max_{c_i \in C_s} [\sigma(G(F_s(x_t)))]_{c_i}$. Using the cross-entropy loss we enforce the target predictions to match the pseudo-labels for these instances, thereby inheriting the class-separability knowledge,

$$
\mathcal{L}_{inh} = \mathcal{L}_{CE}([\sigma(G(F_t(x_t))]), y_p) \quad (8)
$$

**Tune.** In the absence of label information, entropy minimization [27, 11] is popularly employed to move the features of unlabeled instances towards the high confidence regions. However, to avoid negative-transfer, instead of a direct application of entropy minimization, we use $w$ as a soft instance weight in our loss formulation. Target instances with higher $w$ are guided towards the high source density regions, while those with lower $w$ are pushed into the negative regions (see Fig. 3D→E). This separation is a key to minimize the effect of negative-transfer.
On a fine level, we separately calculate probability vectors for $G_s$ as $z_{i}^{sh} = \sigma(G_s(F_i(x_i)))$, and for $G_n$ as, $z_{i}^{uk} = \sigma(G_n(F_i(x_i)))$, and minimize the following loss,

$$L_{t_2} = w(x_t)H(z_{i}^{sh}) + (1 - w(x_t))H(z_{i}^{uk})$$

where, $H$ is the Shannon’s entropy. The total loss $L_{tune} = L_{t_1} + L_{t_2}$ selectively aligns the shared classes, while avoiding negative-transfer. Thus, the final adaptation loss is,

$$L_a = L_{inh} + L_{tune}$$

We now present a discussion on the success of this adaptation procedure from the theoretical perspective.

### 4.3. Theoretical Insights

We defined the inheritability criterion in Eq. 1 for transferring the task-specific knowledge to the target domain. To show that the knowledge of class-separability is indeed inheritable, it is sufficient to demonstrate that the inheritability criterion holds for the shared classes. Extending Theorem 3 in [1], we obtain the following result.

**Result 1.** Let $\mathcal{H}$ be a hypothesis class of VC dimension $d$. Let $S$ be a labeled sample set of $m$ points drawn from $q^{sh}$. If $h_t \in \mathcal{H}$ is the empirical minimizer of $\xi_{q^{sh}}$ on $S$, and $h_t^{*} \in \mathcal{H}$ is the optimal hypothesis for $q^{sh}$, then for any $\delta \in (0, 1)$, we have with probability of at least $1 - \delta$ (over the choice of samples),

$$\xi_{q^{sh}}(h_t) \leq \xi_{q^{sh}}(h_t^{*}) + 4\sqrt{\frac{2d\log(2(m + 1)) + 2\log(8/\delta)}{m}}$$

See Supplementary for the derivation of this result. Essentially, using $m$ labeled target-shared instances, one can train a predictor (here, $h_t^{*}$) which satisfies Eq. 12. However, in a completely unsupervised setting, the only way to obtain target labels is to exploit the knowledge of the vendor’s model. This is precisely what the pseudo-labeling process achieves. Using an inheritable model ($h_t^{*}$), we pseudo-label the top-$k$ percentile target instances with high precision and enforce $L_{inh}$. In doing so, we condition the target model to satisfy Eq. 12, which is the inheritability criterion for shared categories (given unlabeled instances $D_t$ and source model $h_t$). Thus, the knowledge of class-separability is transferred to the target model during the adaptation process.

Note that, with increasing number of labeled target instances (increasing $m$), the last term in Eq. 12 decreases. In our formulation, this is achieved by enforcing $L_{tune}$, which can be regarded as a way to self-supervise the target model.

In Sec. 5 we verify that, during adaptation the precision of target predictions improves over time. This self-supervision with an increasing number of correct labels is, in effect, similar to having a larger sample size $m$ in Eq. 12. Thus, adaptation tightens the bound in Eq. 12 (see Suppl.).

### 5. Experiments

In this section, we evaluate the performance of unsupervised open-set domain adaptation using inheritable models.

#### 5.1. Experimental Details

**a) Datasets.** Office-31 [38] consists of 31 categories of images in three different domains: Amazon (A), Webcam (W) and DSLR (D). Office-Home [49] is a more challenging dataset containing 65 classes from four domains: Real World (Re), Art (Ar), Clipart (Cl) and Product (Pr). VisDA [36] comprises of 12 categories of images from two domains: Real (R), Synthetic (S). The label sets $C_s$, $C_t$ are in line with [24] and [39] for all our comparisons. See Suppl. for sample images and further details.

**b) Implementation.** We implement the framework in PyTorch and use ResNet-50 [12] (till the last pooling layer) as the backbone models $M_s$ and $M_t$ for Office-31 and Office-Home, and VGG-16 [43] for VisDA. For inheritable model training, we use a batch size of 64 (32 source and negative instances each), and use the hyperparameters $d = 15$ and $K = 4|C_s|$. During adaptation, we use a batch size of 32 and set the hyperparameter $k = 15$. We normalize the instance weights $w(x_t)$ with the max weight of each batch $B$, i.e, $w(x_t) / \max_{x_t \in B} w(x_t)$. During inference, an unknown label is assigned if $\hat{y}_t = \arg\max_{c} \sigma(G(F_t(x_t)))$ is one of the $K$ negative classes, otherwise, a shared class label is predicted. See Supplementary for more details.

**c) Metrics.** In line with [39], we compute the open-set accuracy (OS) by averaging the class-wise target accuracy for $|C_s| + 1$ classes (considering target-unknown as a single class). Likewise, the shared accuracy (OS*) is computed as the class-wise average of target-shared classes ($C_t^{sh} = C_s$).

#### 5.2. Results

**a) State-of-the-art comparison.** In Tables 1-3, we compare against the state-of-the-art UODA method STA [24]. The results for other methods are taken from [24]. Particularly, in Table 1, we report the mean and std. deviation of OS and OS* over 3 separate runs. Due to space constraints, we report only OS in Table 2. It is evident that adaptation using an inheritable model outperforms prior arts that assume access to both vendor’s data (source domain) and client’s data (target domain) simultaneously. The superior performance of our method over STA is described as follows. STA learns a domain-agnostic feature extractor by aligning the two domains using an adversarial discriminator. This restricts the model’s flexibility to capture the di-
Table 1. Results on Office-31 (ResNet-50). \(|C_s| = 10, |C_t| = 20\). Our denotes adaptation using an inheritable model following the vendor-client paradigm, while all other methods use source domain data during adaptation.

| Method | A→W | A→D | D→W | D→A | W→A | Avg |
|--------|------|------|------|------|------|-----|
| ResNet | 82.3±1.2 | 82.7±0.9 | 85.2±0.3 | 85.5±0.9 | 94.1±0.3 | 94.3±0.7 |
| RTN [27] | 85.6±1.2 | 88.1±1.0 | 89.5±1.4 | 90.9±1.4 | 94.8±0.3 | 96.2±0.7 |
| DANN [10] | 85.3±0.7 | 87.7±1.1 | 86.5±0.6 | 87.7±1.0 | 97.5±1.0 | 98.3±0.5 |
| OpenMax [3] | 87.4±0.5 | 87.5±0.3 | 87.1±0.9 | 88.4±1.0 | 96.2±0.3 | 98.4±1.3 |
| ATI-OSBP [39] | 87.4±1.5 | 88.9±1.4 | 84.3±1.2 | 86.6±1.0 | 93.6±1.0 | 95.3±1.0 |
| OSBP [59] | 86.5±2.0 | 87.6±2.1 | 88.6±1.4 | 89.2±1.3 | 97.0±1.0 | 96.5±0.4 |
| STA [24] | 89.8±1.0 | 92.0±1.5 | 93.7±1.5 | 96.1±1.4 | 97.5±1.0 | 98.5±0.5 |
| Ours | 91.3±0.7 | 93.2±1.2 | 94.2±1.1 | 97.1±0.8 | 96.5±0.5 | 97.4±0.7 |

Table 2. Results on Office-Home (ResNet-50). \(|C_s| = 25, |C_t| = 65\). Our denotes adaptation using an inheritable model.

| Method | Ar→Cl | Pr→Cl | Rw→Cl | Ar→Pr | Cl→Ar | Pr→Ar | Rw→Pr | Ar→Rw | Cl→Rw | Pr→Rw | Avg |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----|
| ResNet | 55.4±0.4 | 52.7±0.6 | 51.9±0.5 | 69.3±0.7 | 61.8±0.5 | 74.1±0.4 | 61.4±0.6 | 64.0±0.3 | 70.0±0.3 | 78.7±0.6 | 71.0±0.6 | 74.9±0.9 |
| ATI-OSBP [39] | 56.5±1.2 | 52.6±1.6 | 53.5±1.4 | 69.1±1.1 | 63.5±1.5 | 74.1±1.5 | 61.7±1.2 | 64.8±0.9 | 70.7±0.5 | 79.2±0.7 | 72.9±0.7 | 75.8±1.6 |
| DANN [10] | 54.6±0.7 | 49.7±1.6 | 51.9±1.6 | 69.5±1.1 | 63.5±1.0 | 72.9±1.0 | 61.9±1.2 | 63.3±1.0 | 71.3±1.0 | 80.2±0.8 | 71.7±1.4 | 74.2±0.4 |
| OSBP [39] | 56.7±1.9 | 51.5±2.1 | 49.2±2.4 | 67.5±1.5 | 65.5±1.5 | 74.0±1.5 | 62.5±2.0 | 64.8±1.1 | 69.3±1.1 | 80.6±1.0 | 74.7±2.2 | 71.5±1.9 |
| STA [24] | 55.1±1.0 | 49.4±1.0 | 61.7±1.2 | 69.3±1.0 | 81.5±0.9 | 63.4±0.5 | 65.2±0.8 | 74.9±1.0 | 85.0±2.2 | 78.5±0.8 | 80.4±0.3 | 69.5 |
| Ours | 66.1±0.7 | 54.2±1.0 | 56.2±1.7 | 70.3±1.4 | 70.6±1.4 | 84.5±0.6 | 86.4±1.3 | 74.8±0.9 | 83.2±0.9 | 78.7±1.3 | 81.3±1.4 | 80.6 |

Figure 4. Sensitivity to hyperparam. \(K, d\). Best viewed in color.

at the pre-classifier space for both target-shared and target-unknown instances in Fig. 6B following the procedure laid out in [10]. The \(PAD\) value evaluated for target-shared instances using our model is much lower than a source-trained ResNet-50 model, while that for target-unknown is higher than a source-trained ResNet-50 model. This suggests that adaptation aligns the source and the target-shared distributions, while separating out the target-unknown instances.

5.3. Discussion

a) Model inheritability (I). Following the intuition in Sec. 4.2a, we evaluate the model inheritability (I) for the tasks \(D\→W\) and \(A\→W\) on Office-31. In Fig. 6C we observe that for the target \(W\), an inheritable model trained on the source \(D\) exhibits a higher \(I\) value than that trained on the source \(A\). Consequently, the adaptation task \(D\→W\) achieves a better performance than \(A\→W\), suggesting that a vendor model with a higher model inheritability is a better candidate to perform adaptation to a given target domain. Thus, given an array of inheritable vendor models, a client can reliably choose the most suitable model for the target domain by measuring \(I\). The ability to choose a vendor model without requiring the vendor’s source data enables the application of the vendor-client paradigm in practice.

b) Instance-level inheritability (w). In Fig. 5D, we show the histogram of \(w(x_i)\) values plotted separately for target-
shared and target-unknown instances, for the task A → D in Office-31 dataset. This empirically validates our intuition that the classifier confidence of an inheritable model follows the inequality in Eq. 5, at least for the extent of domain-shift in the available standard datasets.

c) Reliability of $w$. Due to the mitigation of overconfidence issue, we find the classifier confidence to be a good candidate for selecting target sample for pseudo-labeling. In Fig. 5B, we plot the prediction accuracy of the top-$k$ percentile target instances based on target predictor confidence ($\max_{c \in C} \sigma(G(F_i(x_i)))_{c_i}$). Particularly, the plot for epoch-0 shows the pseudo-labeling precision, since the target predictor is initialized with the parameters of the source predictor. It can be seen that the top-15 percentile samples are predicted with a precision close to 1. As adaptation proceeds, $L_{tune}$ improves the prediction performance of the target model, which can be seen as a rise in the plot in Fig. 5B. Therefore, the bound in Eq. 12 is tightened during adaptation. This verifies our intuition in Sec. 4.3

d) Qualitative results. In Fig. 6A we plot the t-SNE [30] embeddings of the last hidden layer (pre-classifier) features of a target predictor trained using STA [24] and our method, on the task A → D. Clearly, our method performs equally well in spite of the unavailability of source data during adaptation, suggesting that inheritable models can indeed facilitate adaptation in the absence of a source dataset.

e) Training time analysis. We show the benefit of using inheritable models, over a source dataset. Consider a vendor with a labeled source domain A, and two clients with the target domains D and W respectively. Using the state-of-the-art method STA [24] (which requires labeled source dataset), the time spent by each client for adaptation using source data is 575s on an average (1150s in total). In contrast, our method (a single vendor model is shared with both the clients) results in 250s of vendor’s source training time (feature-splicing: 77s, K-means: 66s, training: 154s), and an average of 69s for adaptation by each client (138s in total). Thus, inheritable models provide a much more efficient pipeline by reducing the cost on source training in the case of multiple clients (STA: 1150s, ours: 435s). See Supplementary for experiment details.

6. Conclusion

In this paper we introduced a practical vendor-client paradigm, and proposed inheritable models to address open-set DA in the absence of co-existing source and target domains. Further, we presented an objective way to measure inheritability which enables the selection of a suitable source model for a given target domain without the need to access source data. Through extensive empirical evaluation, we demonstrated state-of-the-art open-set DA performance using inheritable models. As a future work, inheritable models can be extended to problems involving multiple vendors and multiple clients.

Acknowledgements. This work is supported by a Wipro PhD Fellowship (Jogendra) and a grant from Uchhatar Avishkar Yojana (UAY, IISc_010), MHRD, Govt. of India.
[36] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. In CVPRW, 2018.

[37] Subhankar Roy, Aliaksandr Siarohin, Enver Sangineto, Samuel Rota Bulo, Nicu Sebe, and Elisa Ricci. Unsupervised domain adaptation using feature-whitening and consensus loss. In CVPR, 2019.

[38] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In ECCV, 2010.

[39] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In ECCV, 2018.

[40] Swami Sankaranarayanan, Yogesh Balaji, Carlos D. Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In CVPR, June 2018.

[41] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. IEEE transactions on pattern analysis and machine intelligence, 35(7):1757–1772, 2012.

[42] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. Journal of statistical planning and inference, 90(2):227–244, 2000.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[44] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In ECCV, 2016.

[45] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In CVPR, 2011.

[46] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In ICCV, 2015.

[47] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In CVPR, 2017.

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

[49] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In CVPR, 2017.

[50] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In ICML, 2019.

[51] Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In CVPR, 2017.

[52] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In NeurIPS, 2014.
Supplementary: Towards Inheritable Models for Open-Set Domain Adaptation

This supplementary is organized as follows:

- **Sec. 1**: Derivation of Result 1
- **Sec. 2**: Algorithm and implementation details
  - Architecture (Sec. 2.1, Table 1)
  - Inheritable model training (Sec. 2.2, Algo. 1)
  - Feature-splicing algorithm (Algo. 2)
  - Target domain adaptation (Sec. 2.3, Algo. 3)
- **Sec. 3**: Alternate methods to generate OOD samples
- **Sec. 4**: Miscellaneous
  - Training time comparison (Sec. 4.1)
  - Dataset description (Sec. 4.2)

### 1. Derivation of Result 1

We derive the result using Theorem 3 in [1], which provides a generalized learning bound for a given pair of source and target domains. We begin by quoting the theorem, following which we apply the theorem to our setting. For quoting the theorem, we use the notations given in [1].

**Theorem 3** [1]. Let $\mathcal{H}$ be a hypothesis space of VC dimension $d$. Let $\mathcal{U}_S$ and $\mathcal{U}_T$ be unlabeled samples of size $m'$ each, drawn from $\mathcal{D}_S$ and $\mathcal{D}_T$ respectively. Let $S$ be a labeled sample of size $m$ generated by drawing $\beta m$ points from $\mathcal{D}_T$ and $(1 - \beta)m$ points from $\mathcal{D}_S$ and labeling them according to $f_S$ and $f_T$, respectively. If $h \in \mathcal{H}$ is the empirical minimizer of $\hat{\epsilon}_\alpha(h)$ on $S$ and $h^*_T = \min_{h \in \mathcal{H}} \epsilon_T(h)$ is the target error minimizer, then for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ (over the choice of samples),

$$
\epsilon_T(h) \leq \epsilon_T(h^*_T) + A + 2(1 - \alpha)B \tag{1}
$$

where,

$$
A = 4\sqrt{\frac{\alpha^2}{\beta} + \frac{(1 - \alpha)^2}{1 - \beta}} \sqrt{\frac{2d \log(2(m + 1)) + 2 \log(8/\delta)}{m}} \tag{2}
$$

$$
B = \frac{1}{2} d_{H \Delta \mathcal{H}}(\mathcal{U}_S, \mathcal{U}_T) + 4\sqrt{\frac{2d \log(2m') + \log(8/\delta)}{m'}} + \lambda \tag{3}
$$

and the error $\hat{\epsilon}_\alpha(h) = \alpha \hat{\epsilon}_T(h) + (1 - \alpha)\hat{\epsilon}_S(h)$, with $\alpha$ being the relative importance given to the empirical target error. Here, $f_S$ and $f_T$ are the ground-truth labeling functions for the source and the target domains. Further note that, the notations used to describe the errors $\epsilon_T$ and $\hat{\epsilon}_\alpha$ are different from the notations we use in our paper (for instance, we refer to the target error as $\xi_q$). See [1] for more details.

In our formulation, we make two observations. Firstly, the objective of our adaptation step is to improve the performance on the target domain. This entails a choice of $\alpha = 1$ for defining the empirical error. Secondly, we do not have any data from the source domain, which implies $\beta = 1$.

Now, with $\alpha = 1$, the last term in Eq. 1 vanishes. Further, to evaluate Eq. 2 with $\alpha = 1, \beta = 1$ which obtains an indeterminate form, we take the limit as $\alpha \to 1, \beta \to 1$,

$$
limit_{\alpha \to 1, \beta \to 1} A = 4\sqrt{\frac{2d \log(2(m + 1)) + 2 \log(8/\delta)}{m}} \tag{4}
$$

This reduces Eq. 1 to,

$$
\epsilon_T(h) \leq \epsilon_T(h^*_T) + 4\sqrt{\frac{2d \log(2(m + 1)) + 2 \log(8/\delta)}{m}} \tag{5}
$$

Now, we describe the result. We argue that the knowledge of class-separability (i.e. the knowledge of how the classes are distinguished) is *inheritable*, by demonstrating that the inheritability criterion holds for shared classes.

During adaptation, we select the top-$k$ percentile target instances based on the value of $w$. In Sec. 5.3c of the paper, we empirically verify that the pseudo-labeling precision for the top-$k$ target instances is close to 1 (see Fig. 5B of the paper for epoch-0 at $k = 15$). Therefore, the pseudo-labeling process can be considered as obtaining target-shared instances with a small noise in the labels. For these instances, by minimizing $\mathcal{L}_{inh}$, we search the hypothesis space $\mathcal{H}$ for the empirical minimizer of $\xi_{q^{th}}$, i.e. $\tilde{h}_t = \arg\min_{h \in \mathcal{H}} \xi_{q^{th}}(h)$. Thus, considering the correctly pseudo-labeled target instances as the sample $S$ of size $m$ in Eq. 5, $\tilde{h}_t$ as the empirical minimizer of $\xi_{q^{th}}$, and $h^*_t$ as the optimal hypothesis for $q^{th}$ we can obtain the relation in Eq. 5 for the target-shared distribution ($q^{sh}$) as,
Table 1. Architecture of vendor and client models. ‘FC(Inp, Out)’ denotes Fully-Connected Layers with ‘Inp’ input nodes and ‘Out’ output nodes. BN denotes BatchNorm layer

| Component | Layers |
|-----------|--------|
| $E_s, E_t$ | FC(2048 × 1024) → ELU → FC(1024 × 1024) → BN → ELU → FC(1024 × 256) → ELU → FC(256 × 256) → BN → ELU |
| $G_s$ | FC(256 × $|C_s|)$ |
| $G_n$ | FC(256 × $K$) |

\[ \xi_{q^*}(\hat{h}_t) \leq \xi_{q^*}(h^*_t) + 4 \sqrt{\frac{2d \log(2(m + 1)) + 2 \log(8/\delta)}{m}} \]  

Hence, we obtain Result 1 of the paper, which is stated as,

**Result 1.** Let $\mathcal{H}$ be a hypothesis class of VC dimension $d$. Let $S$ be a labeled sample set of $m$ points drawn from $q^{th}$. If $\hat{h}_t \in \mathcal{H}$ be the empirical minimizer of $\xi_{q^*}$ on $S$, and $h^*_t = \text{argmin}_{h \in \mathcal{H}} \xi_{q^*}(h)$ be the optimal hypothesis for $q^{th}$, then for any $\delta \in (0, 1)$, we have with probability of at least $1 - \delta$ (over the choice of samples),

\[ \xi_{q^*}(\hat{h}_t) \leq \xi_{q^*}(h^*_t) + 4 \sqrt{\frac{2d \log(2(m + 1)) + 2 \log(8/\delta)}{m}} \]  

Algorithm 1 Pseudo-Code for inheritable model training

1: **require**: labeled source dataset $\mathcal{D}_s$, parameters $\theta_{M_s}, \theta_{E_s}, \theta_{G_s}, \theta_{G_n}$ of $M_s, E_s, G_s, G_n$, respectively, hyperparameters $K, d,$ and no. of negative instances $\eta_n$

**Step 1:** Pre-training on the source dataset
2: while $\mathcal{L}_b$ has not converged do
3: $(X_s, Y_s) \leftarrow$ batch sampled from $\mathcal{D}_s$
4: $\hat{Y}_s \leftarrow \sigma(G_s(F_s(X_s)))$
5: compute mean $\mathcal{L}_b$ for the batch using $\hat{Y}_s$ and $Y_s$
6: update $\theta_{M_s}, \theta_{E_s}, \theta_{G_s}$ by minimizing $\mathcal{L}_b$ using the Adam optimizer
7: **end while**

**Step 2:** Training the inheritable model
8: $\mathcal{D}_n \leftarrow$ FeatureSplicingAlgorithm($\mathcal{D}_s, M_s, K, d, \eta_n$)
9: while $\mathcal{L}_s$ has not converged do
10: $(X_s, Y_s) \leftarrow$ batch sampled from $\mathcal{D}_s$
11: $(U_n, Y_n) \leftarrow$ batch sampled from $\mathcal{D}_n$
12: $\hat{Y}_s \leftarrow \sigma(G_s(F_s(X_s)))$
13: $\hat{Y}_n \leftarrow \sigma(G(E_s(U_n)))$
14: compute mean $\mathcal{L}_s$ for the batch using $\hat{Y}_s, \hat{Y}_n, Y_s, Y_n$
15: update $\theta_{E_s}, \theta_{G_s}, \theta_{G_n}$ by minimizing $\mathcal{L}_s$ using the Adam optimizer
16: **end while**

2.1. Architecture

For experiments on Office-31 and Office-Home, we choose ResNet-50 up to the last AvgPool layer (2048 dimensions) as the backbone network for $M_s$. For experiments on VisDA dataset, we use the backbone as VGG-16 up to the last pooling layer (of shape $7 \times 7 \times 512$), followed by a global AvgPool along each channel to obtain an output of 512 dimensions. Modules $E_s, E_t, G_s, G_n$ are composed of fully connected layers, batch norm layers and non-linearity (ELU) as shown in Table 1.

2.2. Vendor trains an inheritable model

The vendor has access to an annotated source dataset $\mathcal{D}_s$ using which the vendor trains an inheritable model $h_s$. See Algorithm 1 for the pseudo-code. During training, we have the following image augmentations: random rotations, flip, color jitter and random crop. We pre-train the parameters of the components $\{M_s, E_s, G_s\}$ on the source domain by minimizing $\mathcal{L}_b$ (L2-L7 in Algo. 1). We then freeze the backbone $M_s$ and perform feature splicing (L8 in Algo. 1). See Algo. 2 for the pseudo-code for the feature splicing operation. We apply feature-splicing at the last layer of $M_s$ to obtain negative instances $s_n$. Specifically, L8-L10 in Algo. 2 shows the feature splicing operation ($\phi_{d}$), where the top-$d$ percentile activations are replaced. In this manner, we
Algorithm 2 FeatureSplicingAlgorithm

1: require: labeled source dataset $D_s$, parameters $\theta_M$ of $M_s$, hyperparameter $K, d$, no. of negative instances $\eta_u$

\[ \text{let } |D| \text{ denote the cardinality of a set } D, [\cdot] \text{ denote the indexing operation, } |\cdot| \text{ denote the append operation and } M \text{ denote the output dimensionality of } M_s \text{ (i.e. } M = 2048 \text{ for ResNet-50 and } M = 512 \text{ for VGG-16}) \]

Step 1: Generating negative instances

1: Number of dimensions to splice $\eta_d = M \times d / 100$
2: $U_n \leftarrow \emptyset \quad \triangleright \text{Empty list}$
3: while $|U_n| \leq \eta_u$ do
4: \[ (x_s^u, x_s^{u'}) \leftarrow \text{sample 2 instances from } D_s \text{ belonging to different classes } i.e. c_i, c_j \in C_s \text{ where } i \neq j \]
5: \[ u_s^c \leftarrow M_s(x_s^c) \]
6: \[ u_s^{c'} \leftarrow M_s(x_s^{c'}) \]
7: \[ \text{idx} \leftarrow \text{top } \eta_d \text{ entries in argsort}(u_s^c) \]
8: \[ u_n \leftarrow u_s^{c'}[\text{idx}] \]
9: \[ U_n \leftarrow U_n \cup u_n \]
10: end while

Step 2: Assigning labels to negative instances

11: Perform $K$-means clustering on $U_n$ and assign a unique negative class to each cluster
12: \[ D_n \leftarrow \{(u_n, y_n) : u_n \in U_n, y_n = \text{negative class label of } u_n \text{ obtained from the previous step} \}
13: \text{return } D_n

generate $\eta_u = 20000$ negative instances for Office-31 and $\eta_u = 50000$ for Office-Home and VisDA. We label these instances by performing a $K$-means clustering on the obtained features (L13-L14 in Alg. 2). Using $D_s$ and $D_n$ we train $\{E_s, G_s, G_n\}$ by minimizing $L_s$ (L9-L16 in Alg. 1).

2.3. Client adapts to the target domain

The client has access to unlabeled target dataset $D_t$ and vendor’s inheritable model $h_s = \{F_s, G\}$, using which the client performs adaptation to the target domain. See Algo. 3 for the pseudo-code. We use image augmentations as mentioned in Sec. 2.2. We obtain pseudo-labeled target instances (top-$k$ percentile, based on the instance-level inheritability value $w$) into a collection $D_t^p$ (L2-L15). Thereafter, during adaptation, each batch contains pseudo-labeled target instances (L18) and unlabeled target instances (L21).

We normalize the weights obtained for the unlabeled target instances (as mentioned in Sec. 5.1b of the paper), with the maximum weight in a batch (L28-L30). We then train $F_t$ to adapt to the target domain by minimizing $L_{inh} + L_{tune}$. 

Algorithm 3 Pseudo-Code for target domain adaptation

1: require: unlabeled target dataset $D_t$, parameters $\theta_{F_t}$, $\theta_{F_s}$, $\theta_{G_s}$, $\theta_{G_n}$ of $\{F_s, F_t, G_s, G_n\}$, hyperparameter $k$.

\[ \text{let } [\cdot] \text{ denote the indexing operation, and } \cdot \text{ denote the append operation} \]

Step 1: Pseudo-labeling process

1: Number of instances to pseudo-label $\eta_p = |D_t| \times k / 100$
2: $W \leftarrow \emptyset \quad \triangleright \text{Empty list}$
3: $Y_p \leftarrow \emptyset \quad \triangleright \text{Empty list}$
4: $X_p \leftarrow \emptyset \quad \triangleright \text{Empty list}$
5: for $x_t$ in $D_t$ do
6: \[ \hat{y}_p \leftarrow \sigma(G(F_s(x_t))) \]
7: \[ y_p \leftarrow \text{argmax}_{c_i \in C_s} \hat{y}_p[c_i] \]
8: \[ w \leftarrow \text{max}_{c_i \in C_s} \hat{y}_p[c_i] \]
9: \[ W \leftarrow W \cup w \]
10: \[ Y_p \leftarrow Y_p \cup y_p \]
11: \[ X_p \leftarrow X_p \cup x_t \]
12: end for
13: $D_t^p \leftarrow (X_p[|\text{idx}]], Y_p[|\text{idx}])$

Step 2: Adaptation process

15: Initialize $\theta_{F_t}$ from $\theta_{F_s}$.
16: while $L_{inh} + L_{tune}$ has not converged do
17: \[ \text{compute mean } L_{inh} \text{ for the batch using } \hat{Y}_p, Y_p \]
18: \[ (X_p, Y_p) \leftarrow \text{batch sampled from } D_t^p \]
19: \[ \hat{y}_p \leftarrow \sigma(G(F_t(X_p))) \]
20: \[ w' \leftarrow \text{max}_{x_t \in X, w(x_t)} \]
21: for each $x_t$ in $X_t$ do
22: \[ \hat{w} \leftarrow \Sigma_{c_i \in C_s} \sigma(G(F_t(x_t)))[c_i] \]
23: \[ w \leftarrow \text{max}_{c_i \in C_s} \hat{w}[c_i] \]
24: \[ z_t^{sh} \leftarrow \sigma(G_s(F_t(x_t))) \]
25: \[ z_t^{sk} \leftarrow \sigma(G_n(F_t(x_t))) \]
26: end for
27: for each $x_t$ in $X_t$ do \quad \triangleright \text{Normalize weights}
28: \[ w'(x_t) \leftarrow w(x_t) / (\max_{x_t \in X, w(x_t)}) \]
29: end for
30: compute mean $L_{tune}$ using $w', \hat{w}, z_t^{sh}, z_t^{sk}$
31: update $\theta_{F_t}$ by minimizing $L_{inh} + L_{tune}$ using the Adam optimizer
32: end while
3. Alternate methods to generate OOD samples

We argue in Sec. 4.1 of the paper that an inheritable model for the task of open-set DA should have the ability to mitigate the overconfidence issue. We achieve this by training an out-of-distribution (OOD) classifier ($G_u$). We explored potential techniques to generate OOD samples, and as mentioned in Sec. 4.1b in the paper, we found that the feature-splicing technique works well in practice for training an inheritable model for open-set DA. In Table 2, we report the adaptation performance on Office-31 using inheritable models trained with different OOD generation strategies. Here, we discuss the two other strategies we explored.

a) Linear interpolation between classes. We randomly choose a pair of source instances corresponding to different classes, and obtain a negative feature by linearly interpolating between the features of the two instances as,

$$u_n = \gamma \times u_s^i + (1 - \gamma) \times u_s^j$$

(8)

where $\gamma \sim \text{Beta}(2,2)$ as proposed in [6], $u_s^i = M_s(x_s^i)$, $u_s^j = M_s(x_s^j)$, and $x_s^i, x_s^j$ are two instances sampled from different source classes, i.e., $c_i, c_j \in \mathcal{C}_s$ ($i \neq j$). This is inspired by techniques such as mixup [7, 6] which encourage less confident predictions on the interpolations of latent features. As reported in Table 2, we find that linear interpolation performs worse than feature-splicing. This is because linear interpolation is less effective in producing OOD samples as it yields features from a constrained region between the source classes (as discussed in Sec. 4.1b of the paper, and shown in Fig. 3A of the paper). In contrast, feature-splicing is able to mimic plausible OOD samples through the suppression of class-specific traits.

b) Random suppression of the most active features.

Given a latent-space feature, we randomly scale down the values of top-15 percentile activations as,

$$u_n = \Gamma \odot u_s$$

(9)

where $u_s = M_s(x_s)$ and $\Gamma$ is a weight vector containing $\gamma \sim \text{uniform}(0.2, 0.3)$ at the indices corresponding to the top-15 percentile activations, and 1 elsewhere, and $\odot$ denotes element-wise product. In Table 2, we observe that this performs similar to feature-splicing. Essentially, by suppressing the top activations, we obtain a feature which is devoid of the class-specific traits (Sec. 4.1b of the paper) thereby resulting in a plausible OOD sample. However, this method of random suppression requires a hyperparameter search to identify an optimal range for the scale of suppression (in this case, 0.2 to 0.3). The feature-splicing technique is meant to avoid this hyperparameter search, by choosing the appropriate replacement for the class-discriminative activations for a given instance.

4. Miscellaneous

In this section, we provide the details of the machine and the datasets used for experiments.

4.1. Training time comparison

The machine used to run all our experiments has the following hardware specifications. CPU: Intel Core i7-7700K, GPU: NVIDIA GeForce GTX 1080 with CUDA v8.0.61.

The training time reported in Sec. 5.3e in the paper is obtained on the machine with the above specifications. For a fair evaluation, we compare the training time required for our method and the previous state-of-the-art method STA [2] on identical settings. We use a batch size of 32, with identical optimizers and learning rates for adaptation.

a) STA. The STA method adapts to target domain in two steps. The first step trains a multi-binary classifier and a domain discriminator using both source and target samples (16 instances each, in a batch) and the second step involves adaptation to the target domain using both source and target samples (again, 16 instances from each domain). For both A→D and A→W, on an average, Step 1 took 177s while Step 2 took 398s.

b) Ours. For our method, we fixed all hyperparameters to the values as mentioned in Sec. 5.1b of the paper. Overall, our method is much more efficient in the case of multiple clients. This is clearly because in our approach, the source training is done only once (by the vendor) requiring about 250s. Following this, both the clients use the same vendor model to adapt to their respective target domains requiring 69s on an average. This is contrast to STA where each client has to train on the vendor’s source dataset, requiring additional computation during adaptation.

4.2. Dataset description

In our experiments, we follow STA [2] to choose the label sets. See Fig. 1 for sample images from each dataset.

The Office-31 [4] dataset contains 4652 images from 3 domains: Amazon (A), DSLR (D) and Webcam (W). In alphabetical order, the first 10 classes are used as shared classes and, the classes 21-31 are chosen as target-unknown. The Office-Home [5] dataset was curated by crawling the web, and thus exhibits a higher domain-shift as compared
Figure 1. Sample images from the benchmark datasets: **Office-31** showing images belonging to class “Speaker”, **Office-Home** showing images belonging to class “Flower” and **VisDA** showing images belonging to class “Truck”.

to the **Office-31** dataset. It contains 65 classes of objects with about 15,500 images split into four domains: Art (Ar), Clipart (Cl), Product (Pr) and Real-World (Rw). The first 25 classes in alphabetical order are chosen as shared classes and classes 26-65 are chosen as the target-unknown classes.

The **VisDA**[3] dataset exhibits a significant amount of domain-shift between its two domains: Synthetic (S) and Real World (R) having about 150k images and 56k images respectively. The Synthetic domain was created by rendering 3D models. The following classes are selected as the shared classes, $C_s = \{\text{bicycle, bus, car, motorcycle, train, truck}\}$, while the target-unknown classes are chosen as, $C_{uk} = \{\text{aeroplane, horse, knife, person, plant, skateboard}\}$.

**References**

[1] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. Machine learning, 79(1-2):151–175, 2010.

[2] Hong Liu, Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Qiang Yang. Separate to adapt: Open set domain adaptation via progressive separation. In CVPR, 2019.

[3] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. In CVPRW, 2018.

[4] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In ECCV, 2010.

[5] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In CVPR, 2017.

[6] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In ICML, 2019.

[7] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In ICLR, 2018.
References

[1] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. Machine learning, 79(1-2):151–175, 2010.

[2] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In NeurIPS, 2007.

[3] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In CVPR, 2016.

[4] Fabio Maria Cariucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulò. Autodial: Automatic domain alignment layers. In ICCV, 2017.

[5] Woong-Gi Chang, Tackgeun You, Seunguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In CVPR, 2019.

[6] Hangting Chen, Yunhe Wang, Chang Xu, Zhaoxui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In ICCV, 2019.

[7] Boris Chidlovskii, Stéphane Clinchant, and Gabriela Csurka. Domain adaptation in the absence of source domain data. In ACM SIGKDD. ACM, 2016.

[8] Zhengming Ding and Yun Fu. Deep domain generalization with structured low-rank constraint. IEEE Transactions on Image Processing, 27(1):304–313, 2017.

[9] Antonio D’Innocente and Barbara Caputo. Domain generalization with domain-specific aggregation modules. In GCPR, 2018.

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

[11] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In NeurIPS, 2005.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

[14] Nick Hynes, Raymond Cheng, and Dawn Song. Efficient deep learning on multi-source private data. arXiv preprint arXiv:1807.06689, 2018.

[15] Lalit P Jain, Walter J Scheirer, and Terrance E Boult. Multi-class open set recognition using probability of inclusion. In ECCV, 2014.

[16] Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei A Efros, and Antonio Torralba. Undoung the damage of dataset bias. In ECCV, 2012.

[17] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In NeurIPS Workshop on Private Multi Party Machine Learning, 2016.

[18] Jogendra Nath Kundu, Nishank Lakkakula, and R Venkatesh Babu. Um-adapt: Unsupervised multi-task adaptation using adversarial cross-task distillation. In ICCV, 2019.

[19] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In ICLR, 2018.

[20] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. In NeurIPS, 2017.

[21] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and fuller domain generalization. In ICCV, 2017.

[22] Da Li, Jiashu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In ICCV, 2019.

[23] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. Pattern Recognition, 80:109–117, 2018.

[24] Hong Liu, Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Qiang Yang. Separate to adapt: Open set domain adaptation via progressive selection. In CVPR, 2019.

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

[26] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In ICML, 2015.

[27] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In NeurIPS, 2016.

[28] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In ICML, 2017.

[29] Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. In NeurIPS, 2017.

[30] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of machine learning research, 9(Nov):2579–2605, 2008.

[31] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In ICML, 2013.

[32] Jogendra Nath Kundu, Phani Krishna Uppala, Anuj Pahuj, and R Venkatesh Babu. Adadepth: Unsupervised content congruent adaptation for depth estimation. In CVPR, 2018.

[33] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaizakh Shaj, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. In ICML, 2019.

[34] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10):1345–1359, 2009.

[35] Pan Panareda Busto and Juergen Gall. Open set domain adaptation. In ICCV, 2017.
[36] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. In CVPRW, 2018.

[37] Subhankar Roy, Aliaksandr Siarohin, Enver Sangineto, Samuel Rota Bulo, Nicu Sebe, and Elisa Ricci. Unsupervised domain adaptation using feature-whitening and consensus loss. In CVPR, 2019.

[38] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In ECCV, 2010.

[39] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In ECCV, 2018.

[40] Swami Sankaranarayanan, Yogesh Balaji, Carlos D. Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In CVPR, June 2018.

[41] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. IEEE transactions on pattern analysis and machine intelligence, 35(7):1757–1772, 2012.

[42] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. Journal of statistical planning and inference, 90(2):227–244, 2000.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[44] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In ECCV, 2016.

[45] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In CVPR, 2011.

[46] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In ICCV, 2015.

[47] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In CVPR, 2017.

[48] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. arXiv preprint arXiv:1412.5474, 2014.

[49] Hemanth Venkateswaran, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In CVPR, 2017.

[50] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In ICML, 2019.

[51] Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In CVPR, 2017.

[52] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In NeurIPS, 2014.

[53] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Universal domain adaptation. In CVPR, June 2019.

[54] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In ECCV, 2014.

[55] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In ICLR, 2018.

[56] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In CVPR, 2018.