

d-SNE: Domain Adaptation using Stochastic Neighborhood Embedding

Xiang Xu[†], Xiong Zhou^{‡*}, Ragav Venkatesan[‡], Gurumurthy Swaminathan[‡], Orchid Majumder[‡]

[†]Computational Biomedicine Lab, University of Houston, Houston, USA

[‡] AWS AI, Seattle, USA

[†]xxu18@central.uh.edu, [‡]{xiongzh, ragavven, gurumurs, orchid}@amazon.com

Abstract

Deep neural networks often require copious amount of labeled-data to train their scads of parameters. Training larger and deeper networks is hard without appropriate regularization, particularly while using a small dataset. Laterally, collecting well-annotated data is expensive, time-consuming and often infeasible. A popular way to regularize these networks is to simply train the network with more data from an alternate representative dataset. This can lead to adverse effects if the statistics of the representative dataset are dissimilar to our target. This predicament is due to the problem of domain shift. Data from a shifted domain might not produce bespoke features when a feature extractor from the representative domain is used. In this paper, we propose a new technique (d-SNE) of domain adaptation that cleverly uses stochastic neighborhood embedding techniques and a novel modified-Hausdorff distance. The proposed technique is learnable end-to-end and is therefore, ideally suited to train neural networks. Extensive experiments demonstrate that d-SNE outperforms the current states-of-the-art and is robust to the variances in different datasets, even in the one-shot and semi-supervised learning settings. d-SNE also demonstrates the ability to generalize to multiple domains concurrently.

1. Introduction

The use of pre-trained models and transfer-learning have become commonplace in today's deep learning-centric computer vision. Consider a pre-trained model $\mathcal{M}_{\mathcal{D}^s}$ trained using a large dataset $\mathcal{D}^s = \{(x_i^s, y_i^s)\}_{i=1}^{N^s}$, where x_i^s is the i^{th} sample of the s^{th} domain and N^s is the number of samples in the s^{th} domain. Suppose that a typical user has a smaller dataset $\mathcal{D}^t = \{(x_j^t, y_j^t)\}_{j=1}^{N^t}$, with $N^t \ll N^s$, on which they want to train their model. Also consider that the label spaces are the same, i.e., $\{y^s, y^t\} \in [0, 1, \dots, c - 1]$.

*Equal contribution. This work was done during Xiang's internship at AWS AI.

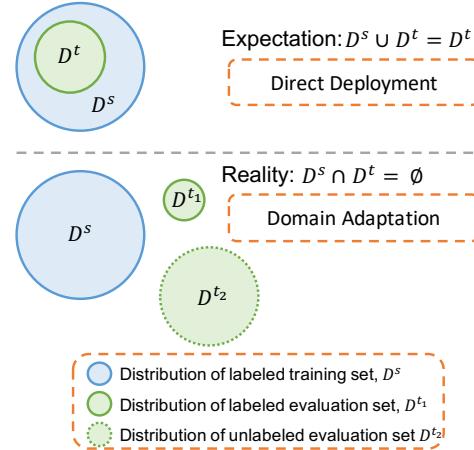


Figure 1: Domain adaptation in the true data space: Expectation vs. Reality.

The user should be able to repurpose the model $\mathcal{M}_{\mathcal{D}^s}$ to work with dataset \mathcal{D}^t . Unless the user is extremely lucky as shown in the top case of figure 1, such a deployment will not work. This is due to domain-shift. Features become meaningless and their spaces get transformed, therefore classifier boundaries have to be redrawn. The class of such problems where the knowledge from another domain is recycled to work to a new target domain is called *domain adaptation*. If the solution can perform equally-well in both domains, it is called as *domain generalization*.

Typical choices of dataset for source are large-scale datasets such as ImageNet [3]. Donahue *et al.* popularized the idea of repurposing networks trained on this dataset to be used as generic feature extractors [5]. They hypothesized and successfully demonstrated that in many cases, when there is limited labeled data available in the target domain, as long as it contains only *natural images*, the feature extractors learnt from ImageNet are general enough to produce discriminative features. Follow-up studies have analysed the transferability of neural networks and the generality of datasets in-detail [33, 29]. In all these cases, the

label-space is considered independently for both domains and the classifier layer of the networks are sanitized. Domain adaptation improves the performance of an existing model $\mathcal{M}_{\mathcal{D}^s}$ for \mathcal{D}^t by *adapting* the knowledge of the model learned from \mathcal{D}^s to \mathcal{D}^t with the assumption that the label spaces are same, and therefore not needing to sanitize the classifier layers [19, 31]. There are two different philosophies in which domain adaptation is typically attacked: (i) **Domain Transformation:** To build a transformation from target data to source domain and reuse the source feature extractor and classifier ($x^t \rightarrow x^s$). Consider the GAN-based methods [1, 21]. These work on the input-level and transform samples from the target domain to mimic distributions of source domains. (ii) **Latent-Space Transformation:** To build a transformation of features extracted from source and features extracted from target into each other or into a common latent space. Since these are working on the conditional feature spaces, these methods are typically supervised.

Figure 2 illustrates the major branches and types of domain adaptation. *d*-SNE falls under the latent-space transformation philosophy, where we create a joint-latent embedding space that is agnostic and invariant to domain-shift. We also focus on the tougher problem where $N^t \ll N^s$, or few-shot supervised domain adaptation. This imposes a constraint that only a few labeled-target samples are available.

To create this embedding space, we use a strategy that is very similar to the popular stochastic neighborhood embedding technique (SNE) [12]. To modify SNE for domain adaptation, we use a novel modified-Hausdorff distance metric in a min – max formulation. *d*-SNE minimizes the distance between the samples from \mathcal{D}^s and \mathcal{D}^t so as to maximize the margin of inter-class distance for discrimination and minimize the intra-class distance from both domains to achieve domain-invariance. This discrimination is learnt as a max-margin nearest-neighbor form to make the network optimization easy. Our proposed idea is still learnable in an end-to-end fashion, therefore making it ideal for training neural networks.

Extensive experimental results in different scenarios indicate that our algorithm is robust and outperforms the state-of-the-art algorithms with only a few labeled data samples. In several cases, *d*-SNE outperforms even unsupervised methods that have access to all samples in the target domain. We generalize *d*-SNE such that it can work on a semi-supervised setting that further pushes the states-of-the-art. Furthermore, *d*-SNE also demonstrates good capabilities in domain generalization without additional training required, which is typically not the case in any state-of-the-art.

The key contributions in this paper include the following:

1. Use of stochastic neighborhood embedding and large-

margin nearest neighborhood to learn a domain-agnostic latent-space.

2. Use of a modified-Hausdorff distance and a novel min – max formulation in this space to help few-shot supervised learning.
3. Demonstration of domain generalization and achieving states-of-the-art results on common benchmark datasets.
4. Extension to semi-supervised settings pushing the states-of-the-art further.

The rest of this article is organized as follows: section 2 surveys related literature in all categories described in figure 2, section 3 dives deep into the proposed idea and provides a theoretical exposition, section 4 presents validation of this idea through experimental evidence and section 5 provides concluding remarks.

2. Related Works

In this section we will survey recent related works in each category of domain adaptation.

Domain transformation: In this category, methods learn a generative model that can transform either the source to the target domain (which is more common) or vice-versa. To learn this generative model itself, no supervision is required. Since unlabeled data is available in plenty, this model can be learnt easily leveraging a plethora of unsupervised data. They use this generative model to prepare a joint dataset in either one of the domains and learn a feature extractor and classifier in that *common* domain [14, 1, 13, 21].

With the advent of Generative Adversarial Networks (GANs) [9] these transformations have become easier Liu and Tuzel proposed a pair of GANs, each of which was responsible for synthesizing the images in the source and target domain, respectively [14]. With an innovative weight-sharing constraint as a regularizer, the generative models were used for generating a pair of images in two domains. Rather than generating the images from random variables in two domains, the generator in PixelDA, proposed by Bousmalis *et al.* transformed the images from the source domain and forced them to map into the distribution of the target domain [1]. CyCADA, proposed by Hoffman *et al.* used cycle-consistent loss and semantic loss along with the GAN losses and bettered the state-of-the-art [13].

While all the previous methods transformed the source domain data to target domain, Russo *et al.* proposed SBADA-GAN, which also considered the transformation from target to source domain [21]. They defined class consistency loss, which learned to obtain the same label used when mapping from source to target and back to the source domain. Since they generated images in both domains, they

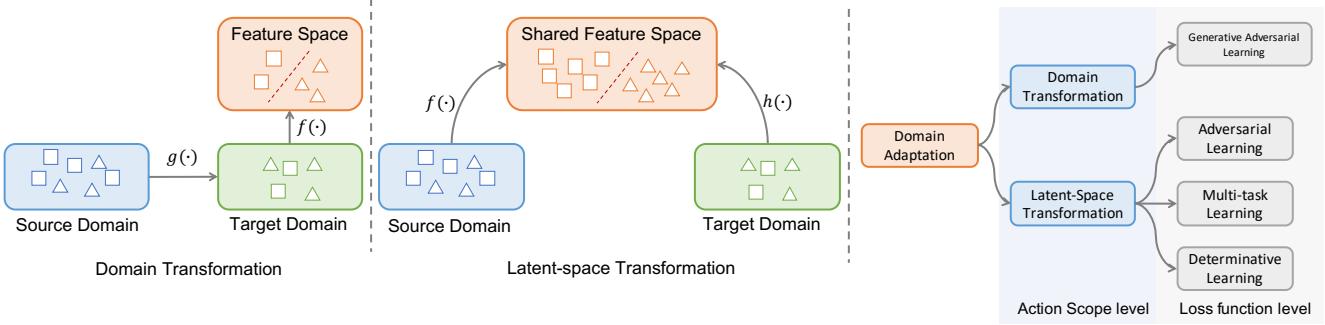


Figure 2: Various types of domain adaptation.

learnt two independent classifiers in each. This implied that they were able to make a prediction using the linear combination or predictions from both domains.

Latent-space transformation: Latent-space transformations can be further divided into two major categories: domain adversarial learning (DAL) and domain multi-task learning (DMTL).

Domain adversarial learning: Perhaps the most popular of DAL techniques is the Domain Adversarial Neural Networks (DANN) introduced by Ganin *et al.* [7]. This work introduced a gradient reversal layer to flip the gradients when the network was back-propagating. Using this gradient flipping, they were able to learn both a discriminative and a domain-invariant feature space. The network was optimized to simultaneously minimize the label error and maximize the loss of the domain classifier. Tzeng *et al.* generalized the architecture of adversarial domain adaptation for unsupervised learning in their work, Adversarial Discriminative Domain Adaptation (ADDA) [28]. ADDA used two independent discriminators from source and target domain to map features in the shared feature space. A label-relaxed version of domain adversarial learning was proposed in [2].

Domain multi-task learning: In order to improve the discriminative capabilities of feature representations, Tzeng *et al.* introduced a shared feature extractor for both source and target domain with three different losses in a multi-task learning manner [27]. Ding *et al.* uses a knowledge graph model to jointly optimize target labels with domain-free features in a unified framework [4]. These losses also acted as a strong regularizer. Rozantsev *et al.* argued that the weights of the network learnt from different domain should be related, yet different for each other [20]. To this end, they added linear transformations between the weights to regularize the networks to behave thusly. Associative Domain Adaptation is another technique in the DMTL regime proposed by Haeusser *et al.* which enforced association between the source and target domains [10]. CCSA and FADA furthered the contrastive loss techniques by creating a uni-

fied framework for supervised domain adaptation and generalization [16, 15]. A decision-boundary iterative refinement training strategy (DIRT-T) was proposed by Shu *et al.* which required an initialization using virtual adversarial training [25]. They refined the model’s weights with a KL divergence loss. *Self-ensembling* extended the mean teacher model in the domain adaptation setting and introduced some tricks such as confidence thresholding, data augmentation, and class imbalance loss [6, 26]. Others learn a shared feature space from the images in the source and target domain [30, 23].

3. d-SNE

Consider the distance between a sample from the source domain and one from a target domain in the latent-space,

$$d(x_i^s, x_j^t) = \|\Phi_{\mathcal{D}^s}(x_i^s) - \Phi_{\mathcal{D}^t}(x_j^t)\|_2^2, \quad (1)$$

where $\Phi_{\mathcal{D}^s}(\cdot) \rightarrow \mathbb{R}^d$ and $\Phi_{\mathcal{D}^t}(\cdot) \rightarrow \mathbb{R}^d$ are neural networks that transform the samples to a common latent-space of d -dimensions from the source and target domains, parameterized by w_s and w_t respectively. In this latent-space,

$$p_{ij} = \frac{\exp(-d(x_i^s, x_j^t))}{\sum_{x \in \mathcal{D}^s} \exp(-d(x, x_j^t))}. \quad (2)$$

is the probability that the target sample $x_j^t \in \mathcal{D}^t$ has the same label as the source sample $x_i^s \in \mathcal{D}^s$. Since we are working under the supervised regime, we actually have the label for both x_i^s and x_j^t , which are y_i^s and y_j^t , respectively. If $y_i^s = y_j^t$, we want p_{ij} to be maximized. If otherwise, we want p_{ij} to be minimized. Notice that in this framework, the training samples in the source domain are chosen from a probability distribution that favors nearby points over far-away ones. In other words, the larger the distance between x and x_j^t , the smaller probability for selecting x as the neighbor of x_j^t for any sample $x \in \mathcal{D}^s$.

Consider that $y_j^t = k$ and that $\mathcal{D}_k^s = \{\forall x_l^s | y_l^s = k\}$. The

probability p_j of making the correct prediction of x_j^t is:

$$p_j = \frac{\sum_{x \in \mathcal{D}_k^s} \exp(-d(x, x_j^t))}{\sum_{x \in \mathcal{D}^s} \exp(-d(x, x_j^t))} = \sum_{i=0}^{N_k^s} p_{ij}, \quad (3)$$

where, $N_k^s = |\mathcal{D}_k^s|$. Notice that given a target sample and label $(x_j^t, y_j^t = k)$, the source domain \mathcal{D}^s is split into two parts as a *same-class set* \mathcal{D}_k^s and a different-class set, $\mathcal{D}_{\neq k}^s$.

The denominator in equation (3) can now be decomposed as $\sum_{x \in \mathcal{D}_k^s} \exp(-d(x, x_j^t)) + \sum_{x \in \mathcal{D}_{\neq k}^s} \exp(-d(x, x_j^t))$. Given p_j for one sample, the objective function for the domain adaptation problem can be derived as,

$$\sum_{x_j \in \mathcal{D}^t} \frac{1}{p_j} = \sum_{x_j \in \mathcal{D}^t} \left(\frac{\sum_{x \in \mathcal{D}_{\neq k}^s} \exp(-d(x, x_j))}{\sum_{x \in \mathcal{D}_k^s} \exp(-d(x, x_j))}, \text{ for } k = y_j \right). \quad (4)$$

Since we want to maximize the probability p_j of making the correct prediction of x_j , we minimize the log-likelihood of $\frac{1}{p_j}$, which is equivalent to minimizing the ratio of intra-class distances to inter-class distances in the latent space.

$$\mathcal{L} = \log \left(\frac{\sum_{x \in \mathcal{D}_{\neq k}^s} \exp(-d(x, x_j))}{\sum_{x \in \mathcal{D}_k^s} \exp(-d(x, x_j))}, \text{ for } k = y_j \right). \quad (5)$$

Relaxation: Since we have sum of exponentials in the likelihood formulation, the ratio in equation (5) may have a scaling issue. This leads to adverse effects in stochastic optimization techniques such as stochastic gradient descent. Since our feature extractors $\Phi_{\mathcal{D}^s}$ and $\Phi_{\mathcal{D}^t}$ are neural networks, this is essential. Therefore, we relax this likelihood with the use of a modified-Hausdorffian distance. Instead of optimizing the global distance as in equation (5), we only minimize the largest distance between the samples of the same class and maximize the smallest distance between the samples of different classes. The final loss is,

$$\tilde{\mathcal{L}} = \sup_{x \in \mathcal{D}_k^s} \{a | a \in d(x, x_j)\} - \inf_{x \in \mathcal{D}_{\neq k}^s} \{b | b \in d(x, x_j)\}, \quad \text{for } k = y_j. \quad (6)$$

End-to-End Learning: Our feature extractors are two independent neural networks $\Phi_{\mathcal{D}^s}$ and $\Phi_{\mathcal{D}^t}$. Pragmatically, a single network can be shared between the two domains ($\Phi_{\mathcal{D}^s} = \Phi_{\mathcal{D}^t}$) if the input data from the source and target domains have the same dimensionality. *d*-SNE allows the target points to select neighbors from the source domain, therefore, the supervision can be transferred from the source domain to the target domain. Since we have labeled data from both domains, standard cross-entropy losses can be used as regularization on top of the domain adaptation losses to train the networks. Since each domain gets its own

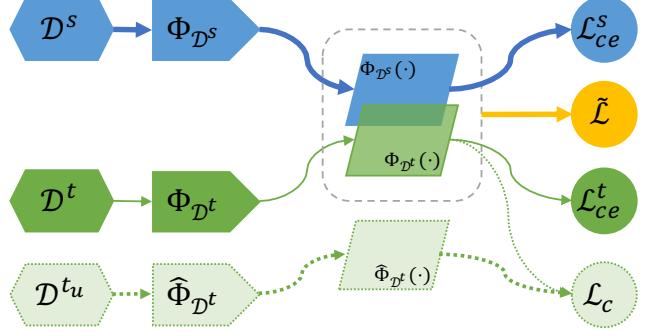


Figure 3: The learning setup. The segment in the bottom in lighter shade and dotted lines is the semi-supervised extension.

cross-entropy, we create a multi-task setup to learn these networks in parallel. Our learning formulation is therefore defined by,

$$\operatorname{argmin}_{w_s, w_d} \tilde{\mathcal{L}} + \alpha \mathcal{L}_{ce}^s + \beta \mathcal{L}_{ce}^t \quad (7)$$

Although we have one minimization form, we divide them for each network, since the weight updates can be conducted independently. Figure 3 illustrates our setup.

Semi-supervised Extension: As was already discussed in the introduction section, having access to unlabeled data helps boost performance. *d*-SNE can be extended easily to accommodate unlabeled data. This extends our proposal into a semi-supervised setting. This is illustrated in the bottom row of figure 3. Suppose that the unlabeled data from the target domain is represented as \mathcal{D}_u^t . We train an unsupervised network $\hat{\Phi}_{\mathcal{D}_u^t}$, parameterized by \hat{w}_t to produce an embedding for the unlabeled image in the latent space. Using a technique similar to the Mean-Teacher network technique proposed by Tarvainen et al., [26]. We use a consistency loss \mathcal{L}_c across $\hat{\Phi}_{\mathcal{D}_u^t}$ and $\Phi_{\mathcal{D}^t}$, by taking an L_2 error between the embeddings.

In particular, the source and target networks are first trained as equation (7). The unlabeled data \mathcal{D}_u^t from the target domain are then used to train the Mean-Teacher model, where new network networks are initialized with the trained target network $\hat{\Phi}_{\mathcal{D}^t} \rightarrow \Phi_{\mathcal{D}^t}$. To generate inputs for both networks, stochastic augmentations, such as flipping, cropping, color jittering, are used to create two sibling samples. Since these are two variants of the same sample and belong to the same class, the consistency loss is an error of the embedding. The weights of $\hat{\Phi}_{\mathcal{D}_u^t}$ network are updated by back-propagating the consistency loss. Instead of sharing weights, the weights of $\Phi_{\mathcal{D}^t}$ are updated with an exponential moving average of the network weights of $\hat{\Phi}_{\mathcal{D}_u^t}$.



Figure 4: Samples from the datasets used.

4. Experiments and Results

To demonstrate the efficiency of d -SNE, three sets of experiments were conducted using three kinds of datasets: (i) digits datasets [18]: four datasets are included as different domains in the digits datasets. MNIST contains 28×28 grayscale images with 70,000 images overall. MNIST-M is a synthetic dataset generated from MNIST by superimposing random backgrounds. USPS consists of 16×16 grayscale images, with 9,298 images overall. SVHN contains RGB photographs of house numbers, with 99,280 images. (ii) office datasets [22]: three sets are included in the office domains. These images are of the same objects but are collected from different sources. Specifically, Office-31 A has 2,817 images, which is collected from the Amazon website; 498 images in office-31 D are captured by DSLR camera and 795 images in office-31 W are captured by web camera. (iii) VisDA-C dataset [7]: two synthetic and real image domains are included in VisDA-C dataset. 152,397 synthetic images are rendered using 3D CAD models as the source domain while the target domain consists of real images. Figure 4 shows samples from the datasets used.

4.1. Digit Datasets

The first set of experiments adapts the domains of digit datasets. In the first experiment, the domains considered are MNIST and USPS. A total of 2,000 samples in MNIST are randomly selected for the source domain. A small number of samples per-class ranging from 1 to 7 were randomly selected from the target domain for training. The inputs

$ \mathcal{D}_k^t , \forall k$	0	1	3	5	7
CCSA [16]	65.40	85.00	90.10	92.40	92.90
FADA [15]	65.40	89.10	91.90	93.40	94.40
d -SNE	73.01	92.90	93.55	95.13	96.13

Table 1: MNIST \rightarrow USPS datasets. $|\mathcal{D}_k^t|, \forall k$ is essentially number of samples per-class from the target domain. As can be seen, d -SNE is clearly able to outperform the states-of-the-art in all scenarios. As the cardinality of the samples per-class increases, the performance across the algorithms converge.

from the source and target domains have the same dimensionalities, $\Phi_{\mathcal{D}}^s = \Phi_{\mathcal{D}}^t$. The states-of-the-art that we use as benchmarks for this experiments are CCSA [16] and FADA [15]. We use the same network architecture as them. Table 1 shows the overall classification accuracies for adaptation from MNIST to USPS datasets. As can be seen that the proposed method outperforms both CCSA and FADA in all the cases even in the one-shot learning case. For the non-adaptation baseline ($|\mathcal{D}_k^t| = 0$), it can be noticed that our implementation achieved a higher accuracy than CCSA and FADA. We attribute this to a better hyperparameter tuning. For the other four cases, we were unable to out-tune their parameters both with our and their own implementations. Therefore, we consider their reported numbers as the best for CCSA [16] and FADA [15].

In the second experiment, we used four datasets including MNIST, USPS, MNIST-M, and SVHN to create five domain adaptation experiments: MNIST \rightarrow MNIST-M, MNIST \leftrightarrow USPS, and MNIST \leftrightarrow SVHN. Several states-of-the-art algorithms including, the ones from before use this setup, therefore enabling us to do a lot of comparisons. It is to be noted that some of the benchmarks are unsupervised, wherein the algorithm uses all of the images in an unlabeled fashion, while the supervised algorithms only use 10 images per-class in the target domain. We use the same network architecture as Wen et al., [32]. The overall classification accuracies are shown in Table 2. Compared to the other supervised benchmarks, d -SNE outperforms the states-of-the-art in all experiments. We observed that all supervised methods in general achieved lower accuracies than unsupervised methods in domain pairs MNIST \rightarrow MNIST-M and SVHN \rightarrow MNIST. In experiments of MNIST \leftrightarrow USPS, d -SNE can achieve higher accuracies than even unsupervised methods. MNIST and USPS datasets has relatively lower intra-class variance compared to MNIST-M and SVHN, which we attribute to these results. Even though the comparison to unsupervised setting is unfair, we can clearly note that the semi-supervised setting of d -SNE pushed our supervised performance closer. The methods that outperform us are

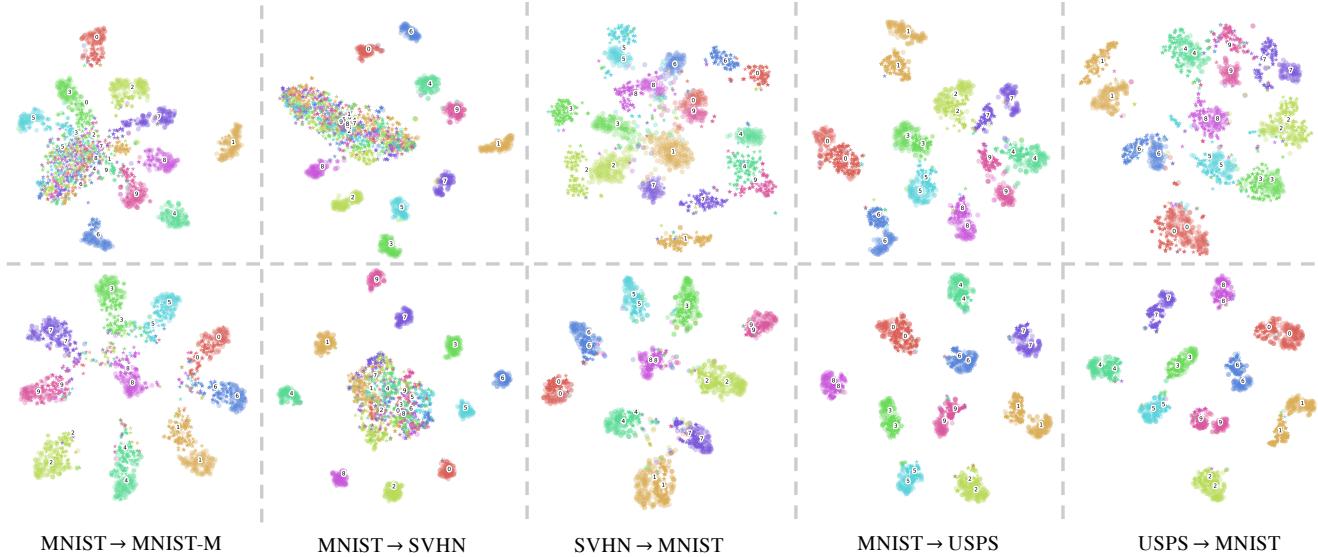


Figure 5: t-SNE visualizations without (top) and with (bottom) domain adaptations.

Method	$ \mathcal{D}_k^t , \forall k$	Setting	MNIST → MNIST-M	MNIST → USPS	USPS → MNIST	MNIST → SVHN	SVHN → MNIST
PixelDA [1]	\mathcal{U}		98.20	95.90	-	-	-
ADA [10]			87.47	-	-	-	97.60
I2I [17]			-	95.1	92.2	-	92.1
DIRT-T [25]			98.90	-	-	54.50	99.40
SE [6]			-	98.26 \pm 0.11	98.07 \pm 2.82	13.96 \pm 4.41	99.18 \pm 0.12
SBADA-GAN [21]			99.40	95.04	97.60	61.08	76.14
G2A [23]			-	95.30 \pm 0.70	90.80 \pm 1.30	-	92.40 \pm 0.90
FADA [15]	7	\mathcal{S}	-	94.40	91.50	47.00	87.20
CCSA [16]	10		78.29 \pm 2.00	97.27 \pm 0.19	95.71 \pm 0.42	37.63 \pm 3.62	94.57 \pm 0.40
<i>d</i> -SNE	0		50.98 \pm 1.64	93.16 \pm 0.71	83.37 \pm 0.93	26.22 \pm 2.02	66.02 \pm 0.72
<i>d</i> -SNE	7	\mathcal{S}	84.62 \pm 0.04	97.53 \pm 0.10	97.52 \pm 0.08	53.19 \pm 0.28	95.68 \pm 0.03
<i>d</i> -SNE	10		87.80 \pm 0.16	99.00 \pm 0.08	98.49 \pm 0.35	61.73 \pm 0.47	96.45 \pm 0.20
<i>d</i> -SNE	10	\mathcal{SS}	94.12	-	-	77.63 \pm 0.26	97.60

Table 2: Classification accuracy for domain adaptation methods on digits datasets. The unsupervised setting (\mathcal{U}) uses all the images in the target domain. The supervised setting (\mathcal{S}) uses 10 labeled samples per-class from the target domain. We reimplemented CCSA and FADA using the same network and settings as our method. The best results are marked in **bold**. If the best result is not in the supervised-only setting, we mark the best among the supervised-only methods in *italics*. The results are averaged over three runs and we report mean and standard deviations over the three runs.

	MNIST → MNIST-M	MNIST → SVHN	SVHN → MNIST
before	99.45%	99.51%	88.96%
after	99.51%	99.59%	94.94%

Table 3: Results of domain generalization.

typically good when using simple datasets. In Table 5 we can see that with more realistic and complicated datasets, our semi-supervised formulation is also on par and often better than the states-of-the-art. We suppose this difference in performance to the intuition that digit datasets are eas-

ily separable even in unsupervised setting. Figure 5 shows some t-SNE visualizations of adaptations.

Domain Generalization: As an added benefit, *d*-SNE shows good domain generalization. Here, we use the model artifacts that we get after the model is adapted to the target domain, and without re-training or fine-tuning, we measure the accuracy on the source dataset. Table 3 shows classification accuracies on the source domain before and after domain adaptation. We can notice from Table 3 that the network actually improves the original performance on the source dataset. This is further evidence of *d*-SNE and the strength of the latent-space it produces.

Method	$ \mathcal{D}_k^t , \forall k$	Setting	A → D	A → W	D → A	D → W	W → A	W → D	Avg.
DANN [7]	\mathcal{U}		-	73.00	-	96.40	-	99.20	-
DRCN [8]			67.10 ± 0.30	68.70 ± 0.30	56.00 ± 0.50	96.40 ± 0.30	54.09 ± 0.50	99.00 ± 0.2	73.60
kNN-Ad [24]			84.10	81.10	58.30	96.40	63.80	99.20	80.48
I2I [17]			71.10	75.30	50.10	96.50	52.10	99.60	74.12
G2A [23]			87.70 ± 0.50	89.50 ± 0.50	72.80 ± 0.30	97.90 ± 0.30	71.40 ± 0.40	99.8 ± 0.4	86.50
SDA [27]	3	\mathcal{S}	86.10 ± 1.20	82.70 ± 0.80	66.20 ± 0.30	95.70 ± 0.50	65.00 ± 0.5	97.60 ± 0.20	82.22
FADA [15]	3		88.20 ± 1.00	88.10 ± 1.20	68.10 ± 0.60	96.40 ± 0.80	71.10 ± 0.90	97.50 ± 0.90	84.90
CCSA [16]	0		61.20 ± 0.90	62.3 ± 0.80	58.5 ± 0.80	80.1 ± 0.60	51.6 ± 0.90	95.6 ± 0.70	68.20
CCSA [16]	3		89.00 ± 1.20	88.20 ± 1.00	71.80 ± 0.50	96.40 ± 0.80	72.10 ± 1.00	97.60 ± 0.40	85.80
d-SNE (VGG-16)	0	\mathcal{S}	62.40 ± 0.40	61.49 ± 0.75	48.92 ± 1.03	82.24 ± 1.42	47.52 ± 0.94	90.42 ± 1.00	65.49
d-SNE (VGG-16)	3		91.44 ± 0.23	90.13 ± 0.07	71.06 ± 0.18	97.10 ± 0.07	71.74 ± 0.42	97.46 ± 0.24	86.49
d-SNE (ResNet-101)	0	\mathcal{S}	80.41 ± 0.79	75.26 ± 1.32	67.39 ± 0.18	96.39 ± 0.41	65.55 ± 1.91	98.31 ± 1.87	80.55
d-SNE (ResNet-101)	3		94.65 ± 0.38	96.58 ± 0.14	75.51 ± 0.44	99.10 ± 0.24	74.20 ± 0.24	100.00 ± 0.00	90.01

Table 4: Results of office31 experiments. *d*-SNE with ResNet-101 base network achieves the best results with only 3 samples in the target domain while *d*-SNE with VGG-16 base network outperforms the baselines in the majority of cases.

4.2. Office31 Dataset

For our experiment involving the various domains of the office31 dataset, we followed the same protocol used by Tzeng et al., and Motian et al., [27, 15, 16]. For the source dataset, we randomly selected 20 samples per-class from A, 8 samples per-class from D and W domains. For the target dataset, 3 samples per-class were selected from all three domains. The rest of samples from the target domain were used for evaluation.

We used ResNet-101 as the base network and added two extra dense layers to obtain feature representations [11]. The dimensionality of the latent-space was 512. As is customary in literature, we used pre-trained weights from ImageNet as initialization [15, 16, 23, 6]. Table 4 reports the results of the experiments. Even with drastically less data samples, *d*-SNE significantly outperformed all the states-of-the-art benchmarks. When the domain shift is extremely large, which was the case with in A→W, W→A, A→D and D→A, *d*-SNE shows larger margins compared to baselines. *d*-SNE can handle domain shift better than other states-of-the-art, especially under limited data conditions. CCSA and FADA used VGG-16 as their base network. While ResNets are the preferred base networks contemporaneously, for fair comparisons we also report accuracies of *d*-SNE with VGG-16 as the base network. As can be seen from Table 4 that the proposed *d*-SNE outperformed both CCSA and FADA in the majority of cases. It is worth mentioning that our non-adaptation baseline ($(|\mathcal{D}_k^t| = 0)$) with VGG-16 actually achieved worse results than CCSA and FADA’s baselines. This highlights the strength of our domain adaptation loss.

4.3. VisDA-C dataset

VisDA-C is a new dataset, therefore we only have a few reported benchmarks: G2A, SE and CCSA [23, 6, 16]. G2A is unsupervised, SE is semi-supervised and CCSA is super-

Method	Setting	$ \mathcal{D}_k^t = 0, \forall k$	Adaptation
G2A [23]	\mathcal{U}	44.50	77.10
SE [6]	\mathcal{SS}	52.80	85.40
CCSA [16]	\mathcal{S}	52.80	76.89
<i>d</i> -SNE	\mathcal{S}	52.80	80.66
<i>d</i> -SNE	\mathcal{SS}	52.80	86.15

Table 5: Results on the VisDA-C dataset. Source domain was synthetic and target domain was real images with 10 images per-class used for training. The metrics for both G2A and SE were reported from the original source. The results for CCSA were obtained from our own implementation.

vised like *d*-SNE. With the settings being unique for each algorithm, the results may not be fair, but we include them for comparisons. We replicated the experimental protocol for the VisDA-C dataset as described in G2A [23].

Similar to the baselines, pre-trained models trained on ImageNet were used as initialization. Table 5 shows the results on VisDA-C dataset. With only 10 samples per-class from the target domain for training, *d*-SNE outperformed CCSA and G2A (in unsupervised setting which used all the unlabeled images in the target domain also). The supervised *d*-SNE cannot match the performance of the semi-supervised SE, albeit this is not a fair comparison to make. With the semi-supervised extension, *d*-SNE established itself as the clear state-of-the-art. Fig. 6 demonstrates that the proposed domain adaptation algorithm can align features from both domains while making them discriminative.

5. Conclusions

Domain adaptation has recently seen a massive boom, largely due to the availability of large quantities of data

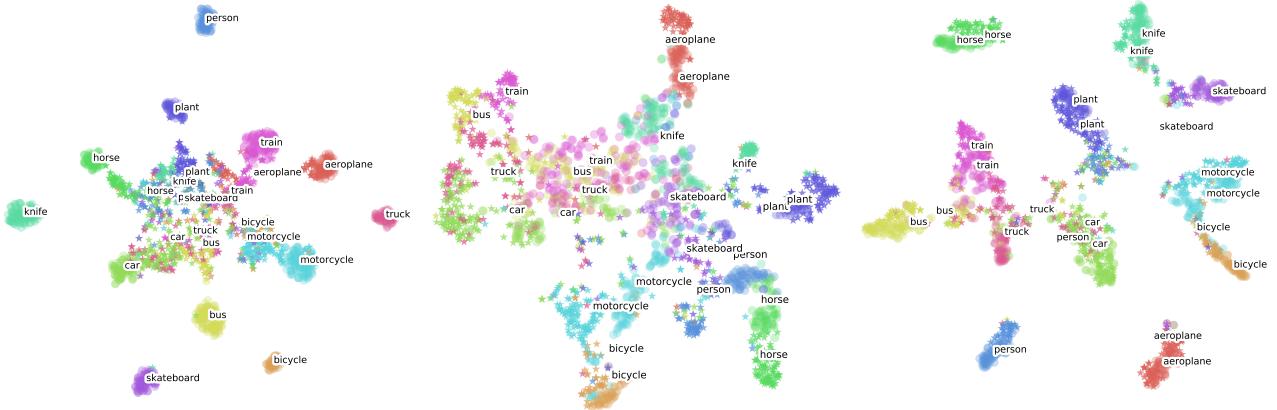


Figure 6: t-SNE visualization of d -SNE’s latent-embedding space for the VisDA-C dataset. (a) Embeddings produced by the model trained with source images only. (b) Embeddings produced by the model trained with target images only and (c) The joint latent-embedding space of d -SNE. Different colors represent different classes. Embeddings from the source and target domains are indicated by circles and stars, respectively.

but from varying domains. Deep-learning-based domain adaptation is a fairly new phenomenon. In this article, we propose a novel use of the stochastic neighborhood embedding technique-based supervised domain adaptation, that is capable of training neural networks end-to-end. Experiments across standard benchmarking datasets show that our method establishes a clear state-of-the-art in most datasets. We propose a semi-supervised extension that pushes the performance further.

References

- [1] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 95–104, Honolulu, Hawaii, jul 2017.
- [2] Z. Cao, L. Ma, M. Long, and J. Wang. Partial adversarial domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150, 2018.
- [3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. Computer Vision and Pattern Recognition*, 2009.
- [4] Z. Ding, S. Li, M. Shao, and Y. Fu. Graph adaptive knowledge transfer for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 37–52, 2018.
- [5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.
- [6] G. French, M. Mackiewicz, and M. Fisher. Self-Ensembling for Visual Domain Adaptation. In *Proc. International Conference on Learning Representations*, pages 1–15, Vancouver, Canada, apr 2018.
- [7] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17:1–35, 2016.
- [8] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *Proc. European Conference on Computer Vision*, pages 597–613, Amsterdam, The Netherlands, oct 2016.
- [9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [10] P. Haeusser, T. Frerix, A. Mordvintsev, and D. Cremers. Associative Domain Adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2784–2792, Venice, Italy, oct 2017.
- [11] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.
- [12] G. E. Hinton and S. T. Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.
- [13] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *ArXiv preprint arXiv: 1711.03213*, pages 1–12, 2017.
- [14] M.-Y. Liu and O. Tuzel. Coupled Generative Adversarial Networks. In *Proc. Advances in Neural Information Processing Systems*, pages 469–477, Barcelona, Spain, dec 2016.
- [15] S. Motiian, Q. Jones, S. M. Iranmanesh, and G. Doretto. Few-Shot Adversarial Domain Adaptation. In *Proc. Advances in Neural Information Processing Systems*, pages 1–11, Long Beach, CA, dec 2017.
- [16] S. Motiian, M. Piccirilli, D. A. Adjero, and G. Doretto. Unified Deep Supervised Domain Adaptation and Generaliza-

- tion. In *Proc. IEEE International Conference on Computer Vision*, pages 5715–5725, Venice, Italy, oct 2017.
- [17] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim. Image to image translation for domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4500–4509, 2018.
- [18] Y. Netzer and T. Wang. Reading digits in natural images with unsupervised feature learning. In *Proc. Advances in Neural Information Processing Systems*, pages 1–9, Granada, Spain, dec 2011.
- [19] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa. Visual Domain Adaptation: A Survey of Recent Advances. *IEEE Signal Processing*, 32(3):53–69, 2015.
- [20] A. Rozantsev, M. Salzmann, and P. Fua. Beyond Sharing Weights for Deep Domain Adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
- [21] P. Russo, F. M. Carlucci, T. Tommasi, and B. Caputo. From source to target and back: symmetric bi-directional adaptive GAN. In *Proc. IEEE Computer Vision and Pattern Recognition*, Salt Lake City, Utah, jun 2018.
- [22] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *Proc. European conference on computer vision*, pages 213 – 226, Grete, Greece, 2010.
- [23] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, and R. Chellappa. Generate To Adapt: Aligning Domains using Generative Adversarial Networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, Utah, jun 2018.
- [24] O. Sener, H. O. Song, A. Saxena, and S. Savarese. Learning Transferrable Representations for Unsupervised Domain Adaptation. In *Proc. Advances in Neural Information Processing Systems*, pages 2110–2118, Barcelona, Spain, dec 2016.
- [25] R. Shu, H. H. Bui, H. Narui, and S. Ermon. A DIRT-T Approach to unsupervised Domain Adaption. In *Proc. International Conference on Learning Representations*, pages 1–19, Vancouver, Canada, apr 2018.
- [26] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proc. Advances in Neural Information Processing Systems*, Long Beach, CA, dec 2017.
- [27] E. Tzeng, J. Hoffman, T. Darrell, K. Saenko, and U. Lowell. Simultaneous Deep Transfer Across Domains and Tasks. In *Proc. International Conference on Computer Vision*, Las Condes, Chile, dec 2015.
- [28] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2962–2971, Honolulu, Hawaii, jul 2017.
- [29] R. Venkatesan, V. Gatupalli, and B. Li. On the generality of neural image features. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 41–45. IEEE, 2016.
- [30] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017.
- [31] M. Wang and W. Deng. Deep Visual Domain Adaptation: A Survey. *ArXiv preprint arXiv: 1802.03601*, pages 1–17, 2018.
- [32] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016.
- [33] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.