Safe Self-Refinement for Transformer-based Domain Adaptation

Tao Sun[1], Cheng Lu[2], Tianshuo Zhang[2], Haibin Ling[1]

[1]Stony Brook University, [2]XPeng Motors

{tao,hling}@cs.stonybrook.edu, luc@xiaopeng.com, tonyzhang2035@gmail.com

Abstract

Unsupervised Domain Adaptation (UDA) aims to leverage a label-rich source domain to solve tasks on a related unlabeled target domain. It is a challenging problem especially when a large domain gap lies between the source and target domains. In this paper we propose a novel solution named SSRT (Safe Self-Refinement for Transformer-based domain adaptation), which brings improvement from two aspects. First, encouraged by the success of vision transformers in various vision tasks, we arm SSRT with a transformer backbone. We find that the combination of vision transformer with simple adversarial adaptation surpasses best reported Convolutional Neural Network (CNN)-based results on the challenging DomainNet benchmark, showing its strong transferable feature representation. Second, to reduce the risk of model collapse and improve the effectiveness of knowledge transfer between domains with large gaps, we propose a Safe Self-Refinement strategy. Specifically, SSRT utilizes predictions of perturbed target domain data to refine the model. Since the model capacity of vision transformer is large and predictions in such challenging tasks can be noisy, a safe training mechanism is designed to adaptively adjust learning configuration. Extensive evaluations are conducted on several widely tested UDA benchmarks and SSRT achieves consistently the best performances, including 85.43% on Office-Home, 88.76% on VisDA-2017 and 45.2% on DomainNet.

1. Introduction

Deep neural networks have achieved impressive performance in a variety of machine learning tasks. However, the success often relies on a large amount of labeled training data, which can be costly or impractical to obtain. Unsupervised Domain Adaptation (UDA) [36] handles this issue by transferring knowledge from a label-rich source domain to a different unlabeled target domain. Over the past years, many UDA methods have been proposed [4, 12, 14, 24, 44]. Among them, adversarial adaptation [4, 14, 44] that learns domain-invariant feature repre-

sentation using the idea of adversarial learning has been a prevailing paradigm. Deep UDA methods are usually applied in conjunction with a pretrained Convolutional Neural Network (CNN, *e.g.*, ResNet [8]) backbone in vision tasks. On medium-sized classification benchmarks such as Office-Home [33] and VisDA [20], the reported state-of-the-arts are very impressive [12]. However, on large-scale datasets like DomainNet [19], the most recent results in the literature by our submission report a best average accuracy of 33.3% [10], which is far from satisfactory.

With the above observations, we focus our investigation on challenging cases from two aspects:

- First, from the *representation aspect*, it is desirable to use a more powerful backbone network. This directs our attention to the recently popularized vision transformers, which have been successfully applied to various vision tasks [2, 3, 42]. Vision transformer processes an image as a sequence of tokens, and uses global self-attention to refine this representation. With its long-range dependencies and large-scale pre-training, vision transformer obtains strong feature representation that is ready for downstream tasks. Despite this, its application in UDA is still under-explored. Hence we propose to integrate vision transformer to UDA. We find that by simply combining ViT-B/16 [3] with adversarial adaptation, it can achieve 38.5% average accuracy on DomainNet, better than the current arts using ResNet-101 [8,10]. This shows that the feature representation of vision transformer is discriminative as well as transferable across domains.

- Second, from the *domain adaptation aspect*, a more reliable strategy is needed to protect the learning process from collapse due to large domain gaps. As strong backbones with large capacity like vision transformer increase the chance of overfitting to source domain data, a regularization from target domain data is desired. A common practice in UDA is to utilize model predictions for self-training or enforce clustering structure on target domain data [12, 24, 43]. While this helps generally, the supervisions can be noisy when the domain gap is large. Therefore, an adaptation method is expected to be *Safe* [11] enough to avoid model collapse.

Motivated by the above discussions, in this paper, we propose a novel UDA solution named SSRT (Safe Self-Refinement for Transformer-based domain adaptation). SSRT takes a vision transformer as the backbone network and utilizes predictions on perturbed target domain data to refine the adapted model. Specifically, we add random offsets to the latent token sequences of target domain data, and minimize the discrepancy of model's predicted probabilities between the original and perturbed versions using the Kullback Leibler (KL) divergence. This imposes a regularization on the corresponding transformer layers in effect. Moreover, SSRT has several important components that contribute to its excellent performance, including multi-layer perturbation and bi-directional supervision.

To protect the learning process from collapse, we propose a novel *Safe Training* mechanism. As UDA tasks vary widely even when they are drawn from the same dataset, a specific learning configuration (*e.g.*, hyper-parameters) that works on most tasks may fail on some particular ones. The learning configuration is thus desired to be automatically adjustable. For example, for perturbation-based methods [17, 25], a small perturbation may under-exploit their benefits while a large one may result in collapse. Recent works [1, 29] apply a manually defined ramp-up period at the beginning of training. However, this cannot solve the issue when its maximum value is improper for the current task. In contrast, we propose to monitor the whole training process and adjust learning configuration adaptively. We use a diversity measure of model predictions on the target domain data to detect model collapse. Once it occurs, the model is restored to a previously achieved state and the configuration is reset. With this safe training strategy, our SSRT avoids significant performance deterioration on adaptation tasks with large domain gaps. The code is available at https://github.com/tsun/SSRT.

In summary, we make the following contributions:

- We develop a novel UDA solution SSRT, which adopts a vision transformer backbone for its strong transferable feature representation, and utilizes the predictions on perturbed target domain data for model refinement.

- We propose a safe training strategy to protect the learning process from collapse due to large domain gaps. It adaptively adjusts learning configuration during the training process with a diversity measure of model predictions on target domain data.

- SSRT is among the first to explore vision transformer for domain adaptation. Vision transformer-based UDA has shown promising results, especially on large-scale datasets like DomainNet.

- Extensive experiments are conducted on widely tested benchmarks. Our SSRT achieves the best performances, including 85.43% on Office-Home, 88.76% on VisDA-2017 and 45.2% on DomainNet.

2. Related Work

Unsupervised Domain Adaptation. There are several prevailing categories of UDA methods. Discrepancy-based methods minimize the distribution divergence between source and target domains with discrepancy measures [15, 28, 32]. Adversarial adaptation methods learn domain-invariant representations by playing a two-player min-max game between the feature extractor and a domain discriminator [4, 28, 31, 32]. Recently, many works exploit self-training for domain adaptation [16, 45, 46]. They generate pseudo labels for target domain data and take them as labeled data to refine the model.

Transformer in Vision. Vision Transformer (ViT) [3] is a pioneering work that applies a convolution-free transformer structure for image classification. Following that, many ViT variants have been proposed [7, 13, 30, 41]. Transformer has been applied successfully to various vision tasks including image classification [3, 30], object detection [2], semantic segmentation [27], *etc*.

The application of vision transformer in domain adaptation, however, is still very scarce. Notably, two concurrent explorations [39, 40] have been recently reported on arXiv. Specifically, CDTrans [39] is a pure transformer solution for UDA, and it applies cross attention on source-target image pairs. TVT [40] proposes a transferable multi-head self-attention module and combines it with adversarial adaptation. Our method is different in that it uses pairs of target domain data and their perturbed version to refine the model. This guarantees the same semantic class. Besides, we delicately design the components of our model and the training strategy to avoid collapse on challenging tasks.

Consistency Regularization. Consistency regularization is an important technique in semi-supervised learning that achieves state-of-the-art results [25]. It leverages the idea that model predictions should be similar for semantically identical data. Some methods create perturbed inputs with adversarial training [17], while others use standard data augmentations [1, 25, 37]. These works mostly manipulate raw input images. In contrast, our study focuses on the latent token sequence representation of vision transformer.

3. Proposed Method

3.1. Problem Formulation

In Unsupervised Domain Adaptation, there is a source domain with labeled data $\mathcal{D}_s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ from $\mathcal{X} \times \mathcal{Y}$ and a target domain with unlabeled data $\mathcal{D}_t = \{(\boldsymbol{x}_i^t)\}_{i=1}^{n_t}$ from $\mathcal{X}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the label space. UDA aims to learn a classifier $h = g \circ f$, where $f(\cdot; \theta_f) : \mathcal{X} \to \mathcal{Z}$ denotes the feature extractor, $g(\cdot; \theta_g) : \mathcal{Z} \to \mathcal{Y}$ denotes the class predictor, and $\mathcal{Z}$ is the latent space. Adversarial adaptation learns domain-invariant feature via a bi-
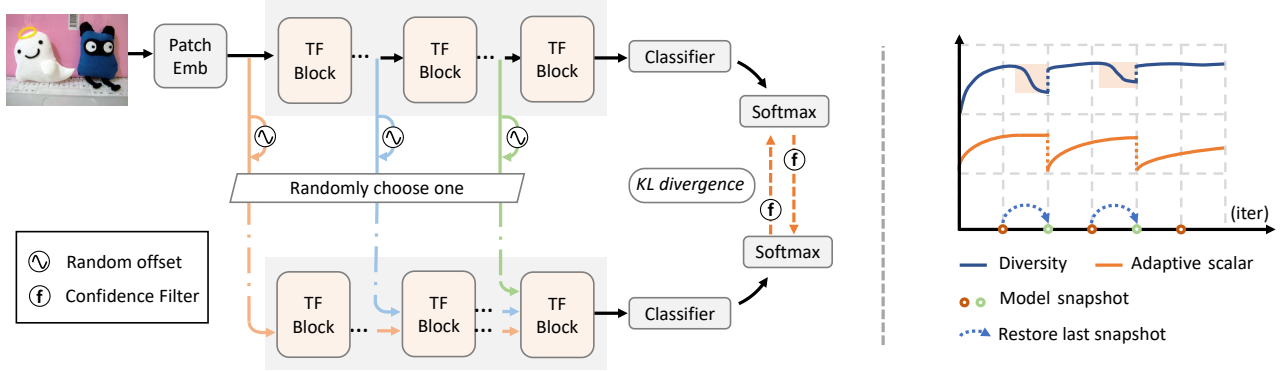
Figure 1. Overview of SSRT. (**Left**) Illustration of Self-Refinement for our transformer-based model. The two branches share parameters. Random offsets are added to the input token sequences of transformer (TF) blocks. The model is refined using its predictions of the original and perturbed versions supervised by KL divergence. (**Right**) Illustration of Safe Training mechanism. See text for details.

nary domain discrimination $d(\cdot; \theta_d) : \mathcal{Z} \rightarrow [0, 1]$ that maps features to domain labels.

The objective is formulated as

$$\min_{f,g} \max_{d} \mathcal{L} = \mathcal{L}_{\mathrm{CE}} - \mathcal{L}_{\mathrm{d}} + \beta \mathcal{L}_{\mathrm{tgt}}, \quad (1)$$

where $\mathcal{L}_{\mathrm{CE}}$ is the standard cross-entropy loss on source domain data, $\mathcal{L}_{\mathrm{d}}$ is domain adversarial loss, defined as

$$\mathcal{L}_{\mathrm{d}} = -\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_s} \big[\log d(f(\boldsymbol{x})) \big] - \mathbb{E}_{x \sim \mathcal{D}_t} \big[\log(1 - d(f(\boldsymbol{x}))) \big],$$

β is a trade-off parameter, and $\mathcal{L}_{\mathrm{tgt}}$ is a loss on target domain data. A common choice of $\mathcal{L}_{\mathrm{tgt}}$ is the Mutual Information Maximization loss [6, 23]. In our method, we instantiate it as the self-refinement loss $\mathcal{L}_{\mathrm{SR}}$, introduced in Sec. 3.4.

3.2. Method Framework

We aim to regularize the latent feature spaces of transformer backbone by refining the model with perturbed target domain data. Figure 1 illustrates the framework of our proposed SSRT. Only target domain data are shown here. The network consists of a vision transformer backbone and a classifier head. Domain discriminator is not plotted. For each target domain image, the *Patch Embedding* layer transforms it into a token sequence including a special class token and image tokens. Then the sequence is refined with a series of *Transformer Blocks*. The classifier head takes the class token and outputs label prediction. We randomly choose one transformer block and add a random offset to its input token sequence. Then the corresponding predicted class probabilities of original and perturbed versions are used for bi-directional self-refinement. To avoid noisy supervision, only reliable predictions are used via a *Confidence Filter*. To reduce the risk of model collapse, we use a safe training mechanism to learn the model.

3.3. Multi-layer Perturbation for Transformer

While many works manipulate the raw input images [1, 17, 25], it may be better to do that at hidden layers [34]. Vision transformer has some particular properties due to its special architecture. Since the *Patch Embedding* layer is merely a convolutional layer plus the position embedding, a linear operation on raw input can be shifted equivalently to the first transformer block. Besides, due to residual connections within transformer blocks, the token sequences at adjacent blocks are highly correlated. The best layer to add perturbation, however, varies across tasks. Empirically, perturbing relatively deep layers performs better but at a higher risk of model collapse. Therefore, we randomly choose one from multiple layers, which proves to be more robust than perturbing any single layer from them. In fact, it imposes a regularization on multiple layers simultaneously, making the learning process safer.

Given a target domain image $\boldsymbol{x}$, let b_x^l be its input token sequence of the l-th transformer block. b_x^l can be viewed as a latent representation of $\boldsymbol{x}$ in a hidden space. Since its dimension is high while the support of target domain data is limited in the space, it is inefficient to perturb b_x^l arbitrarily. Instead, we utilize the token sequence b_{xr}^l of another randomly chosen target domain image $\boldsymbol{x}_r$ to add an offset. The perturbed token sequence of b_x^l is obtained as

$$\tilde{b}_x^l = b_x^l + \alpha [b_{xr}^l - b_x^l]_\times, \quad (2)$$

where α is a scalar and $[\cdot]_\times$ means no gradient back-propagation. Note that although gradients cannot back-propagate through the offset, they can pass through b_x^l. The importance of this is elaborated in the following section.

In addition to the manually injected perturbation, the *Dropout* layer in the classifier head also works randomly for the two branches. This creates another source of discrepancy for the self-refinement loss.

3.4. Bi-directional Self-Refinement

Now we are ready to define the loss function used for self-refinement. Let $\boldsymbol{p}_x$ and $\tilde{\boldsymbol{p}}_x$ be the predicted probability vectors corresponding to b_x^l and $\tilde{b}_x^l$, respectively. To measure their distance, KL divergence is commonly used.

$$D_{\mathrm{KL}}(\boldsymbol{p}_t \| \boldsymbol{p}_s) = \sum_i \boldsymbol{p}_t[i] \log \frac{\boldsymbol{p}_t[i]}{\boldsymbol{p}_s[i]}, \qquad (3)$$

where $\boldsymbol{p}_t$ is the teacher probability (*a.k.a.* target probability) and $\boldsymbol{p}_s$ is the student probability. Note that KL divergence is asymmetric in $\boldsymbol{p}_t$ and $\boldsymbol{p}_s$. While it is natural to take $\boldsymbol{p}_x$ as the teacher probability since it corresponds to the original data, we find the reverse also works. Moreover, as shown in Sec. 4.3, it is more robust to combine them together. Our bi-directional self-refinement loss is defined as

$$\mathcal{L}_{\mathrm{SR}} = \mathbb{E}_{\mathcal{B}_t \sim \mathcal{D}_t} \Big\{ \omega \mathbb{E}_{\boldsymbol{x} \sim F[\mathcal{B}_t; \boldsymbol{p}]} D_{\mathrm{KL}}(\boldsymbol{p}_x \| \tilde{\boldsymbol{p}}_x) \\ + (1 - \omega) \mathbb{E}_{\boldsymbol{x} \sim F[\mathcal{B}_t; \tilde{\boldsymbol{p}}]} D_{\mathrm{KL}}(\tilde{\boldsymbol{p}}_x \| \boldsymbol{p}_x) \Big\}, \qquad (4)$$

where ω is a random variable drawn from a Bernoulli distribution $\mathcal{B}(0.5)$, F is a *Confidence Filter* defined as

$$F[\mathcal{D}; \boldsymbol{p}] = \{\boldsymbol{x} \in \mathcal{D} | \max(\boldsymbol{p}_x) > \epsilon\}, \qquad (5)$$

and ϵ is a predefined threshold. $\mathcal{L}_{\mathrm{SR}}$ refines the model with confident predictions and regularizes it to predict smoothly in the latent feature spaces.

Typically, the loss gradient is only back-propagated through the student probability (*i.e.*, $\boldsymbol{p}_s$ in Eq. 3) [1, 17, 18]. We find, however, it is better to back-propagate gradient through both teacher and student probabilities in our framework. Recall that $\partial \mathcal{L}_{\mathrm{SR}} / \partial \tilde{b}_x^l$ is propagated to b_x^l identically in Eq. 2. Each model parameter is therefore updated based on the joint effects from $\boldsymbol{p}_x$ and $\tilde{\boldsymbol{p}}_x$. This avoids excessively large gradients from any single probability. We observe degraded performance when either the gradients of teacher probabilities in KL divergence or that of b_x^l are blocked.

3.5. Safe Training via Adaptive Adjustment

In the proposed self-refinement strategy, setting a proper value of the perturbation scalar α and the self-refinement loss weight β is critical. Excessively large perturbations lead to a collapse of the predicted class distribution, while a small one may under-exploit its benefit. Since the target domain is fully unlabeled and domain adaptation tasks vary widely even for the same dataset, it is desired to adjust these values adaptively. Some works [1, 29] apply a ramp-up period at the beginning of training. While this alleviates the tendency to collapse during this period, it cannot solve the issue when the maximum value is improper for current adaptation tasks.

Algorithm 1 Safe Training Mechanism.

Initialization: $last\_restore = 0$, save snapshot of $\mathcal{M}$
1: **procedure** CHECKDIVDROP(div, L, T, $iter$)
2:    **for** $l = 1$ **to** L **do**   ▷ check at multi-scales
3:       $divs = \mathrm{div}(iter - T, \dots, iter)$ ▷ get diversity
4:       $divs = \mathrm{split}(divs, 2^l)$ ▷ to even sub-intervals
5:       **for** $i = 0$ **to** $\mathrm{len}(divs) - 1$ **do**
6:          **if** $\mathrm{avg}(divs[i+1]) < \mathrm{avg}(divs[i]) - 1$ **then**
7:             **return** True   ▷ significant dropping
8:          **end if**
9:       **end for**
10:    **end for**
11:    **return** False
12: **end procedure**
13:
14: **procedure** SAFETRAINING($\mathcal{M}$, div, T, L, $iter$)
15:    **if** $iter \% T == 0$ **and** $iter >= T$ **then**
16:       **if** CHECKDIVDROP(div, L, T, $iter$) **then**
17:          Restore $\mathcal{M}$ to last snapshot, $t_r = iter$
18:          **if** $iter - last\_restore \leq T_r$ **then**
19:             $T_r = T_r \times 2$ ▷ avoid oscillation
20:          **end if**
21:          $last\_restore = iter$
22:       **end if**
23:       Save snapshot of $\mathcal{M}$
24:    **end if**
25:    **return** $\mathcal{M}$, T_r, t_r
26: **end procedure**

Algorithm 2 SSRT algorithm.

Input: Model $\mathcal{M}$, source data $\mathcal{D}_s$, target data $\mathcal{D}_t$, confidence threshold ϵ, self-refinement loss weight β, perturbation scalar α, Safe Training parameters T and L, diversity measure $\mathrm{div}(\cdot)$.
Initialization: $T_r = T$, $t_r = 0$
1: **for** $iter = 0$ **to** max_Iter **do**
2:    Sample a batch from source data and target data
3:    Obtain r via Eq. 6, $\alpha_r = r\alpha$, $\beta_r = r\beta$
4:    Randomly choose $l \in \{0, 4, 8\}$, add perturbation via Eq. 2 using α_r, obtain $\mathcal{L}_{\mathrm{SR}}$ via Eq. 4
5:    Update model parameters via Eq. 1 using β_r
6:    $\mathcal{M}, T_r, t_r \leftarrow$ SAFETRAINING($\mathcal{M}$, div, T, L, $iter$)
7: **end for**

We propose a *Safe Training* mechanism. The observation is that whenever the model begins to collapse, the diversity of model predictions will decrease simultaneously. Our goal is to detect such events while monitoring the training process. Once it occurs, the learning configuration is reset and meanwhile the model is restored to a previously achieved state. Specifically, an adaptive scalar $r \in [0, 1]$ is

Table 1. Accuracies (%) on **Office-Home**. *CDTrans uses DeiT-base backbone. °TVT uses ViT-base backbone. "-S/B" indicates ViT-small/base backbones, respectively.

| Method	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg.
ResNet-50 [8]	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
CDAN+E [14]	50.7	70.6	76.0	57.6	70.0	70.0	57.4	50.9	77.3	70.9	56.7	81.6	65.8
SAFN [38]	52.0	71.7	76.3	64.2	69.9	71.9	63.7	51.4	77.1	70.9	57.1	81.5	67.3
CDAN+TN [35]	50.2	71.4	77.4	59.3	72.7	73.1	61.0	53.1	79.5	71.9	59.0	82.9	67.6
SHOT [12]	57.1	78.1	81.5	68.0	78.2	78.1	67.4	54.9	82.2	73.3	58.8	84.3	71.8
DCAN+SCDA [10]	60.7	76.4	82.8	69.8	77.5	78.4	68.9	59.0	82.7	74.9	61.8	84.5	73.1
CDTrans* [39]	68.8	85.0	86.9	81.5	87.1	87.3	79.6	63.3	88.2	82.0	66.0	90.6	80.5
TVT° [40]	74.89	86.82	89.47	82.78	87.95	88.27	79.81	71.94	90.13	85.46	74.62	90.56	83.56
ViT-S [3]	47.01	76.98	83.54	69.84	77.11	80.42	68.15	44.08	82.86	74.78	47.97	84.66	69.78
Baseline-S	59.59	80.11	84.67	73.84	78.49	81.36	74.41	59.82	86.27	80.10	62.59	87.23	75.71
SSRT-S (ours)	67.03	84.21	88.32	79.85	84.28	87.58	80.72	66.03	88.27	82.04	69.44	89.86	80.64
ViT-B [3]	54.68	83.04	87.15	77.30	83.42	85.54	74.41	50.90	87.22	79.56	53.79	88.80	75.48
Baseline-B	66.96	85.74	88.07	80.06	84.12	86.67	79.52	67.03	89.44	83.64	70.15	91.17	81.05
Baseline-B+MI	70.63	88.62	89.99	82.08	87.84	89.28	81.01	68.82	**91.26**	85.17	71.66	**92.45**	83.23
SSRT-B (ours)	**75.17**	**88.98**	**91.09**	**85.13**	**88.29**	**89.95**	**85.04**	**74.23**	**91.26**	**85.70**	**78.58**	91.78	**85.43**
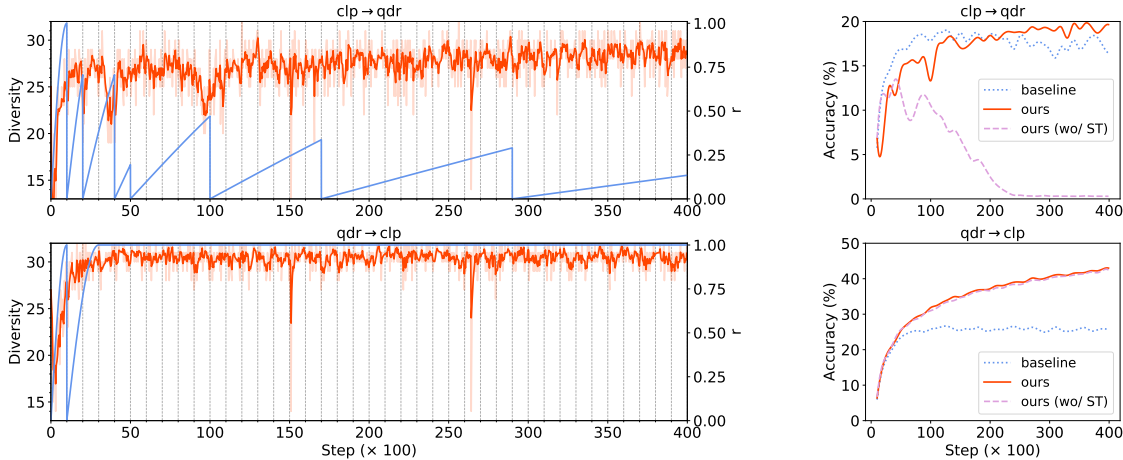
Figure 2. Representative training histories using Safe Training (ST) on DomainNet clp→qdr and qdr→clp. (**Left**) Plots of the diversity of model predictions on target domain data and the adaptive scalar r. For better visualization, both original values (light color) and smoothed values (dark color) of diversity are shown. (**Right**) Plots of comparison test accuracies on target domain data.

adopted to modulate α and β, *i.e.*, $\alpha_r = r\alpha$ and $\beta_r = r\beta$. We define a fixed period T and divide the training process into consecutive intervals. A model snapshot is saved at the end of each interval. Then r is defined as

$$r(t) = \begin{cases} \sin\left(\frac{\pi}{2T_r}(t - t_r)\right) & \text{if } t - t_r < T_r \\ 1.0 & \text{otherwise} \end{cases}, \quad (6)$$

where t is the current training step. Initially, $T_r = T$ and $t_r = 0$. It hence takes T steps for r to ramp up to 1.0. At the end of each interval, the diversity of model predictions within this interval is checked to find abrupt dropping. If not existed, the formulation of r remains unchanged. Otherwise, t_r is reset to current training step t, and the model is restored to last snapshot. To avoid oscillation between collapse and restoration, T_r is doubled if the last restoration occurs within T_r steps. Figure 1 illustrates the training

process with adaptive adjustment. Two events of diversity dropping are identified (marked with pink areas), leading to two model restorations and reset of r.

The remaining issue is which diversity measure to use and how to detect diversity dropping. We find that the number of unique model predicted labels on each target training batch $\mathcal{B}_t$ works well. We hence define the following diversity measure:

$$\text{div}(t; \mathcal{B}_t) = unique\_labels(h(\mathcal{B}_t)). \quad (7)$$

To detect diversity dropping, we split the interval into sub-intervals and check whether the average diversity value drops across each sub-interval. We implement this at multi-scales to improve the sensitivity of detection. Every consecutive sub-intervals of $T/2^1, \cdots, T/2^L$ steps are checked for a given integer L. Details are listed in Alg. 1 and Alg. 2.

Table 2. Accuracies (%) on **DomainNet**. In each sub-table, the column-wise means source domain and the row-wise means target domain.

ResNet-101 [8]

	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	19.3	37.5	11.1	52.2	41.0	32.2
inf	30.2	-	31.2	3.6	44.0	27.9	27.4
pnt	39.6	18.7	-	4.9	54.5	36.3	30.8
qdr	7.0	0.9	1.4	-	4.1	8.3	4.3
rel	48.4	22.2	49.4	6.4	-	38.8	33.0
skt	46.9	15.4	37.0	10.9	47.0	-	31.4
Avg.	34.4	15.3	31.3	7.4	40.4	30.5	26.6

MIMTFL [5]

	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	15.1	35.6	10.7	51.5	43.1	31.2
inf	32.1	-	31.0	2.9	48.5	31.0	29.1
pnt	40.1	14.7	-	4.2	55.4	36.8	30.2
qdr	18.8	3.1	5.0	-	16.0	13.8	11.3
rel	48.5	19.0	47.6	5.8	-	39.4	32.1
skt	51.7	16.5	40.3	12.3	53.5	-	34.9
Avg.	38.2	13.7	31.9	7.2	45.0	32.8	28.1

CDAN [14]

	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	20.4	36.6	9.0	50.7	42.3	31.8
inf	27.5	-	25.7	1.8	34.7	20.1	22.0
pnt	42.6	20.0	-	2.5	55.6	38.5	31.8
qdr	21.0	4.5	8.1	-	14.3	15.7	12.7
rel	51.9	23.3	50.4	5.4	-	41.4	34.5
skt	50.8	20.3	43.0	2.9	50.8	-	33.6
Avg.	38.8	17.7	32.8	4.3	41.2	31.6	27.7

MDD+SCDA [10]

	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	20.4	43.3	15.2	59.3	46.5	36.9
inf	32.7	-	34.5	6.3	47.6	29.2	30.1
pnt	46.4	19.9	-	8.1	58.8	42.9	35.2
qdr	31.1	6.6	18.0	-	28.8	22.0	21.3
rel	55.5	23.7	52.9	9.5	-	45.2	37.4
skt	55.8	20.1	46.5	15.0	56.7	-	38.8
Avg.	44.3	18.1	39.0	10.8	50.2	37.2	33.3

CD-Trans* [39]

	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	27.9	57.6	27.9	73.0	58.8	49.0
inf	58.6	-	53.4	9.6	71.1	47.6	48.1
pnt	60.7	24.0	-	13.0	69.8	49.6	43.4
qdr	2.9	0.4	0.3	-	0.7	4.7	1.8
rel	49.3	18.7	47.8	9.4	-	33.5	31.7
skt	66.8	23.7	54.6	27.5	68.0	-	48.1
Avg.	47.7	18.9	42.7	17.5	56.5	38.8	37.0

ViT-B [3]

	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	27.2	53.1	13.2	71.2	53.3	43.6
inf	51.4	-	49.3	4.0	66.3	41.1	42.4
pnt	53.1	25.6	-	4.8	70.0	41.8	39.1
qdr	30.5	4.5	16.0	-	27.0	19.3	19.5
rel	58.4	29.0	60.0	6.0	-	45.8	39.9
skt	63.9	23.8	52.3	14.4	67.4	-	44.4
Avg.	51.5	22.0	46.1	8.5	60.4	40.3	38.1

Baseline-B

	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	30.9	53.3	16.3	72.7	55.4	45.7
inf	43.0	-	40.8	7.8	56.4	35.9	36.8
pnt	55.7	28.6	-	7.4	70.5	48.3	42.1
qdr	25.5	5.2	9.7	-	15.5	17.1	14.6
rel	62.3	32.5	62.5	8.2	-	50.7	43.2
skt	66.4	30.6	58.0	18.1	70.1	-	48.6
Avg.	50.6	25.6	44.9	11.6	57.0	41.5	38.5

Baseline-B +MI

	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	30.5	55.8	18.1	74.7	57.5	47.3
inf	53.2	-	52.8	9.2	68.3	45.3	45.8
pnt	56.8	27.6	-	7.3	70.8	49.3	42.4
qdr	31.6	5.1	13.3	-	25.3	23.0	19.6
rel	65.7	32.4	63.9	6.9	-	51.7	44.1
skt	68.9	30.6	61.0	19.3	72.9	-	50.5
Avg.	55.2	25.2	49.4	12.2	62.4	45.3	41.6

SSRT-B (ours)

	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	33.8	60.2	19.4	75.8	59.8	49.8
inf	55.5	-	54.0	9.0	68.2	44.7	46.3
pnt	61.7	28.5	-	8.4	71.4	55.2	45.0
qdr	42.5	8.8	24.2	-	37.6	33.6	29.3
rel	69.9	37.1	66.0	10.1	-	58.9	48.4
skt	70.6	32.8	62.2	21.7	73.2	-	52.1
Avg.	60.0	28.2	53.3	13.7	65.3	50.4	45.2

Table 3. Accuracies (%) on **VisDA-2017**.

Method	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	Avg.
ResNet-101 [8]	55.1	53.3	61.9	59.1	80.6	17.9	79.7	31.2	81.0	26.5	73.5	8.5	52.4
DANN [4]	81.9	77.7	82.8	44.3	81.2	29.5	65.1	28.6	51.9	54.6	82.8	7.8	57.4
CDAN [14]	85.2	66.9	83.0	50.8	84.2	74.9	88.1	74.5	83.4	76.0	81.9	38.0	73.9
SAFN [38]	93.6	61.3	84.1	70.6	94.1	79.0	91.8	79.6	89.9	55.6	89.0	24.4	76.1
SWD [9]	90.8	82.5	81.7	70.5	91.7	69.5	86.3	77.5	87.4	63.6	85.6	29.2	76.4
SHOT [12]	94.3	88.5	80.1	57.3	93.1	94.9	80.7	80.3	91.5	89.1	86.3	58.2	82.9
CDTrans* [39]	97.1	90.5	82.4	77.5	96.6	96.1	93.6	88.6	97.9	86.9	90.3	62.8	88.4
TVT° [40]	92.92	85.58	77.51	60.48	93.60	98.17	89.35	76.40	93.56	92.02	91.69	55.73	83.92
ViT-B [3]	99.09	60.66	70.55	82.66	96.50	73.06	97.14	19.73	64.48	94.74	97.21	15.36	72.60
Baseline-B	98.55	82.59	85.97	57.07	94.93	97.20	94.58	76.68	92.11	96.54	94.31	52.24	85.23
Baseline-B+MI	98.63	90.79	81.83	47.28	96.29	98.36	84.68	70.70	93.30	97.54	94.55	55.03	84.08
SSRT-B (ours)	98.93	87.60	89.10	84.77	98.34	98.70	96.27	81.08	94.86	97.90	94.50	43.13	88.76

4. Experiments

We evaluate our method on four popular UDA benchmarks. **Office-31** [22] contains 4,652 images of 31 classes from three domains: Amazon (A), DSLR (D) and Webcam (W). **Office-Home** [33] consists of 15,500 images of 65 classes from four domains: Artistic (Ar), Clip Art (Cl), Product (Pr), and Real-world (Rw) images. **VisDA-2017** [20] is a Synthetic-to-Real dataset, with about 0.2 million images in 12 classes. **DomainNet** [19] is the largest DA dataset containing about 0.6 million images of 345 classes in 6 domains: Clipart (clp), Infograph (inf), Painting (pnt), Quickdraw (qdr), Real (rel), Sketch (skt).

We use the ViT-base and ViT-small with 16×16 patch size [3, 26], pre-trained on ImageNet [21], as the vision transformer backbones. For all tasks, we use an identical set of hyper-parameters ($\alpha = 0.3$, $\beta = 0.2$, $\epsilon = 0.4$, $T = 1000$, $L = 4$). Ablation studies on them are provided in Sec. 4.6. More details can be found in the supplementary material.

Our comparison methods include DANN [4], CDAN [14], CDAN+E [14], SAFN [38], SAFN+ENT [38], CDAN+TN [35], SHOT [12], DCAN+SCDA [10], MDD+SCDA [10], SWD [9], MIMTFEL [5], TVT [40] and CDTrans [39]. "Baseline" is ViT with adversarial adaptation (see Sec. 3.1). We also include its combination with Mutual Information (MI) loss [6, 23] in comparison.

4.1. Results on Benchmarks

Tables 1-4 present evaluation results on four benchmarks. We use "-S/B" to indicate results using ViT-small/base backbones, respectively. For Office-Home and Offce-31, CNN-based methods use ResNet-50 as their backbones; whereas for DomainNet and VisDA they use ResNet-101. Generally, the transformer-based results are much better. This is attributed to its strong transferable feature representations. ViT-base is better than ViT-small, due to higher model complexity. Apparently, Baselines improve over source-only training. Integrating Mutual Information

Table 4. Accuracies (%) on **Office-31**.

| Method	A→W	D→W	W→D	A→D	D→A	W→A	Avg.
ResNet-50 [8]	68.4	96.7	99.3	68.9	62.5	60.7	76.1
DANN [4]	82.0	96.9	99.1	79.7	68.2	67.4	82.2
SAFN+ENT [38]	90.1	98.6	99.8	90.7	73.0	70.2	87.1
CDAN+TN [35]	95.7	98.7	100.	94.0	73.4	74.2	89.3
SHOT [12]	90.1	98.4	99.9	94.0	74.7	74.3	88.6
MDD+SCDA [10]	95.3	99.0	100.	95.4	77.2	75.9	90.5
CDTrans* [39]	96.7	99.0	100.	97.0	81.1	81.9	92.6
TVT° [40]	96.4	99.4	100.	96.4	84.9	86.1	93.8
ViT-S [3]	86.9	98.6	100.	88.6	76.0	75.9	87.7
Baseline-S	91.9	99.1	100.	89.2	78.4	77.9	89.4
SSRT-S (ours)	95.7	99.2	100.	95.8	79.2	79.9	91.6
ViT-B [3]	91.2	99.2	100.	90.4	81.1	80.6	90.4
Baseline-B	92.5	99.2	100.	93.6	80.7	80.7	91.1
SSRT-B (ours)	97.7	99.2	100.	98.6	83.5	82.2	93.5

Table 5. Accuracies (%) compared with perturbing raw inputs. $X^\dagger$ means averaged over all 5 tasks with X being the target domain.

	OH	DN	clp†	inf†	pnt†	qdr†	rel†	skt†
Baseline-B	81.1	38.5	50.6	25.6	44.9	11.6	57.0	41.5
SSRT-B (raw)	85.0	44.2	58.6	26.7	51.7	13.7	63.9	50.8
SSRT-B	85.4	45.2	60.0	28.2	53.3	13.7	65.3	50.4

(MI) loss further improves. Compared with other methods, SSRT-B performs the best on Office-Home, Domain-Net and VisDA. It improves 4.38% on Office-Home, 3.53% on VisDA-2017 and 6.7% on DomainNet over Baseline-B despite that Baseline-B is already very strong. In particular, on the challenging DomainNet dataset, SSRT-B achieves an impressive 45.2% average accuracy. It is worth mentioning that in DomainNet some domains have large gaps from the others, such as *inf* and *qdr*. Transferring among these domains and others is very difficult. It is thus desired to transfer safely and not deteriorate the performance significantly. Looking at tasks with *qdr* being target domain, SSRT-B obtains 29.3% average accuracy, while many other methods perform poorly. We illustrate the effects of some important components that contribute to our excellent performance in the following sections.

4.2. Effects of Multi-layer Perturbation

Table 5 verifies that applying perturbation to the latent token sequences performs better than to the raw input images on Office-Home (OH) and DomainNet (DN). Fig. 5a compares performances when adding the same amount of perturbation to each layer while not using safe training. As can be seen, the best layer to apply perturbation varies across tasks. Besides, a layer that works for one task may fail on others. In our experiments, we uniformly choose one layer from {0,4,8}. As a comparison, perturbing any single layer from it decreases the average accuracy on DomainNet by -1.0%, -1.5% and -1.5%, respectively.
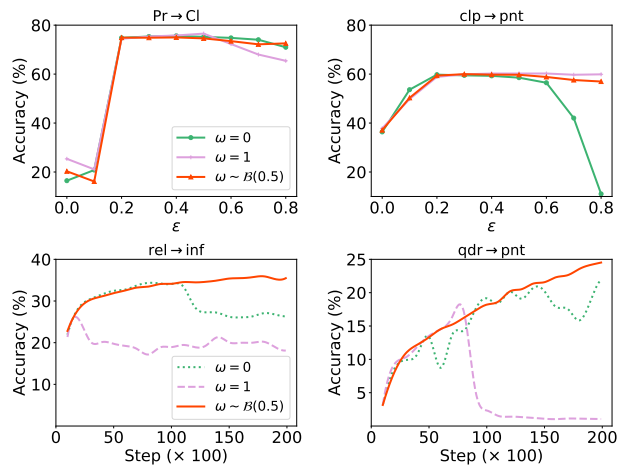
Figure 3. Comparison of self-refinement losses. (**Upper**) Varying confidence threshold ϵ. (**Lower**) Test accuracies on target domain data. (Safe Training not applied)

Table 6. Accuracies (%) using comparison losses. All results are reported at training step of 20k. $X^\dagger$ means averaged over all 5 tasks with X being target domain. ‡ Using Safe Training.

	OH	DN	clp†	inf†	pnt†	qdr†	rel†	skt†
Baseline-B	81.1	38.9	50.7	25.5	46.1	11.9	57.4	42.0
$\omega = 0$	85.5	41.1	57.3	22.0	52.2	1.8	63.4	49.9
$\omega = 1$	85.7	40.1	56.6	23.4	48.1	0.3	63.3	49.0
$\omega \sim \mathcal{B}(0.5)$	85.4	41.8	57.0	26.6	53.0	1.8	63.2	49.5
$\omega \sim \mathcal{B}(0.5)^\ddagger$	85.4	43.4	57.0	28.2	51.8	13.0	62.9	47.4

4.3. Effects of Bi-directional Self-Refinement

Our method adopts bi-directional supervision for self-refinement in Eq. 4. The main consideration is to improve method's safeness. Figure 3 compares with uni-directional self-refinement by fixing ω to be 0 or 1. In the upper two figures, their performance drops for relatively large confidence threshold ϵ. In the lower two figures, model collapse occurs after training for some steps. In contrast, bi-directional self-refinement is more robust as it combines the two losses, thus reduces the negative effect of either one. Table 6 presents some quantitative results. On Office-Home, all losses perform similarly well. On DomainNet, bi-directional self-refinement works better. However, they all fail on challenging tasks when target domain is *qdr*. This is solved with Safe Training.

Another important issue is when to back-propagate gradients. Table 7 shows that the performance degrades when either the gradient for b_x^l in Eq. 2 or the teacher probability of KL divergence in Eq. 4 are blocked. An interesting finding is that the bi-directional self-refinement appears to be more robust even when the gradients are blocked. We believe this is because the two losses are complementary.

Table 7. Blocking gradient back-propagation for different variables. Note that $\boldsymbol{p}_x$ and $\tilde{\boldsymbol{p}}_x$ in the table only refer to the teacher probability in KL divergence. (Safe Training not applied)

	b_x^l	$\boldsymbol{p}_x$	$\tilde{\boldsymbol{p}}_x$	Pr→Ar	Pr→Cl	Pr→Rw
$\omega = 0$			×	4.70	2.66	16.39
$\omega = 1$		×		79.15	44.38	89.14
$\omega \sim \mathcal{B}(0.5)$		×	×	84.10	71.32	90.75
$\omega \sim \mathcal{B}(0.5)$	×			84.38	72.60	90.87
$\omega \sim \mathcal{B}(0.5)$				85.74	74.98	91.16
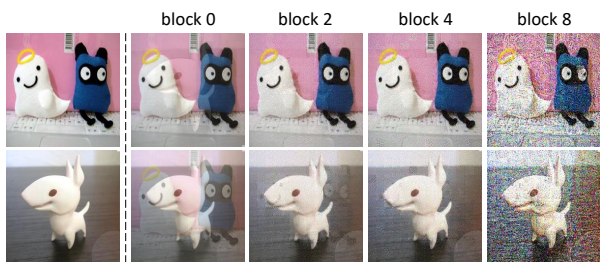
Figure 4. Visualization of perturbation at different layers.

(a) Perturbation at different layer[†]

(b) Safe Training parameters

(c) Perturbation scalar

(d) Self-Refinement loss weight

Figure 5. Plots of ablation studies. Horizontal dash lines indicate baseline accuracies. ([†]Safe Training not applied)

4.4. Effects of Safe Training

As observed previously, the vanilla training strategy may fail on some tasks. The reason is that the predicted class distribution on target domain data collapses due to excessive perturbation or too large loss weight, even if they work well on other tasks. Safe Training adjusts their values adaptively to avoid such situation. Figure 2 presents detailed training histories on two representative tasks to show how it works. For $qdr{\rightarrow}clp$, the adaptive scalar r quickly converges to 1.0 and the diversity stabilizes to a relatively high value. Training model with or without Safe Training performs similarly. For $clp{\rightarrow}qdr$, diversity drops after some steps, and r resets to smaller values. A clear correlation between diversity and accuracy can be observed. For example, at step of 10k, the accuracy drops abruptly and diversity drops concurrently. Without Safe Training, model collapses after about 10k iterations. With Safe Training, the model trains normally and surpasses the baseline finally. It should be noted that model collapse mainly affects target domain data. For $clp{\rightarrow}qdr$ without safe training, the final accuracy on source domain is 96.9% while that on target domain is only 0.3%.

4.5. Visualization of Perturbation

To visualize the perturbed version of a target domain image $\boldsymbol{x}$, we initialize a trainable variable $\boldsymbol{x}_{vis}$ as $\boldsymbol{x}$, and optimize $\boldsymbol{x}_{vis}$ to minimize $\|\tilde{b}_x^l - b_{xvis}^l\|^2$, where $\tilde{b}_x^l$ is the perturbed token sequence of $\boldsymbol{x}$ and b_{xvis}^l is the corresponding token sequence of $\boldsymbol{x}_{vis}$. Then $\boldsymbol{x}_{vis}$ gives us an idea on how the perturbation in the latent space reflects on the raw input images. Figure 4 visualizes perturbed version of two images when adding perturbation to different transformer blocks. For shallow layers, an effect of blending with the
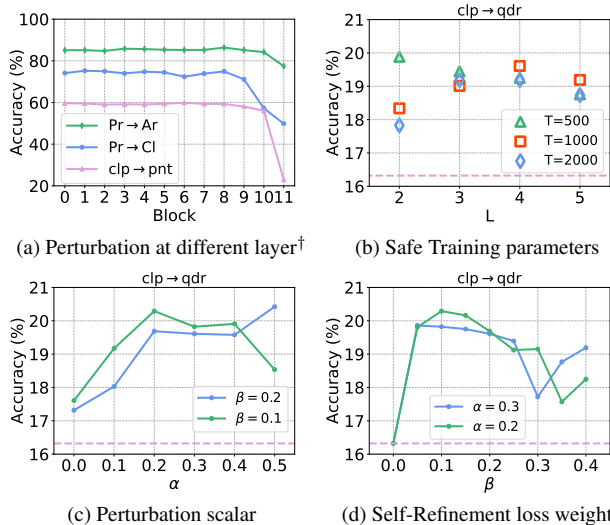
other image can be observed. However, for deep layers, this effect is less noticeable due to highly non-linear transformation of the network. This also indicates the complementary in using multi-layer perturbation.

4.6. Ablation Studies

Figure 5 presents ablation studies on hyper-parameters. Figure 5a plots results of perturbing different layers. Figure 5b plots Safe Training with different parameters. T and L affects its granularity. A smaller T implies more quick response. A larger L increases sensitivity but at the risk of more false-positive detections. Many combinations of T and L work well in our method. Figure 5c and 5d plots accuracy curves *vs.* the perturbation scalar α and the self-refinement loss weight β. Even for obviously unreasonable values like $\alpha = 0.5$, Safe Training can still adjust them adaptively to avoid model collapse. When $\alpha = 0$, our method still has some gain over baseline. This is due to random dropout operations in the classifier head.

5. Conclusion

In this paper, we propose a novel UDA method named SSRT. It leverages a vision transformer backbone, and uses perturbed target domain data to refine the model. A safe training strategy is developed to avoid model collapse. Experiments on benchmarks show its best performance.

Limitation. Although we advance the average accuracy on DomainNet to 45.2%, it is far from saturated. One way is to combine multiple source domains. Another way is to incorporate some meta knowledge about target domains. We plan to extend our study in these directions in the future.

References

[1] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019. 2, 3, 4

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229, 2020. 1, 2

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 1, 2, 5, 6, 7

[4] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189, 2015. 1, 2, 6, 7

[5] Jian Gao, Yang Hua, Guosheng Hu, Chi Wang, and Neil M Robertson. Reducing distributional uncertainty by mutual information maximisation and transferable feature learning. In *ECCV*, pages 587–605. Springer, 2020. 6

[6] Ryan Gomes, Andreas Krause, and Pietro Perona. Discriminative clustering by regularized information maximization. In *NeurIPS*, 2010. 3, 6

[7] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1, 5, 6, 7

[9] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *CVPR*, pages 10285–10295, 2019. 6

[10] Shuang Li, Mixue Xie, Fangrui Lv, Chi Harold Liu, Jian Liang, Chen Qin, and Wei Li. Semantic concentration for domain adaptation. In *ICCV*, pages 9102–9111, 2021. 1, 5, 6, 7

[11] Yu-Feng Li and Zhi-Hua Zhou. Towards making unlabeled data never hurt. In *TPAMI*, pages 175–188. IEEE, 2014. 1

[12] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, pages 6028–6039, 2020. 1, 5, 6, 7

[13] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 2

[14] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, pages 1645–1655, 2018. 1, 5, 6

[15] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, pages 2208–2217, 2017. 2

[16] Ke Mei, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang. Instance adaptive self-training for unsupervised domain adaptation. In *ECCV*, pages 415–430, 2020. 2

[17] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *TPAMI*, 41(8):1979–1993, 2018. 2, 3, 4

[18] Avital Oliver, Augustus Odena, Colin Raffel, Ekin D Cubuk, and Ian J Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. *NeurIPS*, 2018. 4

[19] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, pages 1406–1415, 2019. 1, 6

[20] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *arXiv preprint arXiv:1710.06924*, 2017. 1, 6

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 6

[22] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV*, pages 213–226, 2010. 6

[23] Yuan Shi and Fei Sha. Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. In *ICML*, 2012. 3, 6

[24] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *ICLR*, 2018. 1

[25] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020. 2, 3

[26] Andreas Steiner, Alexander Kolesnikov, , Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021. 6

[27] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021. 2

[28] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, pages 443–450, 2016. 2

[29] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017. 2, 4

[30] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357, 2021. 2

[31] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, pages 7167–7176, 2017. 2

[32] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 2

[33] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, pages 5018–5027, 2017. 1, 6

[34] 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*, pages 6438–6447, 2019. 3

[35] Ximei Wang, Ying Jin, Mingsheng Long, Jianmin Wang, and Michael Jordan. Transferable normalization: Towards improving transferability of deep neural networks. In *NeurIPS*, 2019. 5, 6, 7

[36] Garrett Wilson and Diane J Cook. A survey of unsupervised deep domain adaptation. *ACM TIST*, 11(5):1–46, 2020. 1

[37] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. In *NeurIPS*, 2020. 2

[38] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *ICCV*, pages 1426–1435, 2019. 5, 6, 7

[39] Tongkun Xu, Weihua Chen, Pichao Wang, Fan Wang, Hao Li, and Rong Jin. Cdtrans: Cross-domain transformer for unsupervised domain adaptation. *arXiv preprint arXiv:2109.06165v1*, 2021. 2, 5, 6, 7

[40] Jinyu Yang, Jingjing Liu, Ning Xu, and Junzhou Huang. Tvt: Transferable vision transformer for unsupervised domain adaptation. *arXiv preprint arXiv:2108.05988*, 2021. 2, 5, 6, 7

[41] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In *ECCV*, pages 191–207, 2020. 2

[42] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *ICCV*, 2021. 1

[43] Yabin Zhang, Bin Deng, Kui Jia, and Lei Zhang. Label propagation with augmented anchors: a simple semi-supervised learning baseline for unsupervised domain adaptation. In *ECCV*, pages 781–797, 2020. 1

[44] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael I Jordan. Bridging theory and algorithm for domain adaptation. In *ICML*, pages 7404–7413, 2019. 1

[45] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, pages 289–305, 2018. 2

[46] Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. Confidence regularized self-training. In *ICCV*, pages 5982–5991, 2019. 2