Domain Adaptation with Adversarial Training on Penultimate Activations

**Tao Sun** $^1$, **Cheng Lu** $^2$, **Haibin Ling** $^1$

$^1$ Stony Brook University, USA
$^2$ XPeng Motors, USA
{tao,hling}@cs.stonybrook.edu, luc@xiaopeng.com

**Abstract**
Enhancing model prediction confidence on target data is an important objective in Unsupervised Domain Adaptation (UDA). In this paper, we explore adversarial training on penultimate activations, i.e., input features of the final linear classification layer. We show that this strategy is more efficient and better correlated with the objective of boosting prediction confidence than adversarial training on input images or intermediate features, as used in previous works. Furthermore, with activation normalization commonly used in domain adaptation to reduce domain gap, we derive two variants and systematically analyze the effects of normalization on our adversarial training. This is illustrated both in theory and through empirical analysis on real adaptation tasks. Extensive experiments are conducted on popular UDA benchmarks under both standard setting and source-data free setting. The results validate that our method achieves the best scores against previous arts. Code is available at https://github.com/tsun/APA.

**Introduction**
Unsupervised Domain Adaptation (UDA) aims to transfer knowledge from a label rich source domain to an unlabeled target domain (Pan and Yang 2009; Wang and Deng 2018; Wilson and Cook 2020). The two domains are relevant, yet there is often a distribution shift between them. Due to the domain gap, models from the source domain often predict incorrectly or less confidently on the target domain. Thus, how to enhance model prediction confidence on target data becomes critical in UDA.

The mainstream paradigm for UDA is *feature adaptation* (Sun and Saenko 2016; Tzeng et al. 2014; Ganin and Lempitsky 2015; Tzeng et al. 2017), which learns domain-invariant features. Despite its popularity, this paradigm has some intrinsic limitations in the situation with label distribution shift (Li et al. 2020a; Prabhu et al. 2021). Recently, *self-training* (Zhu 2005) has drawn increasing attention in UDA, which applies consistency regularization or pseudo-labeling loss on the unlabeled target data. Self-training effectively enhances prediction confidence on target data, showing superior results (Liang, Hu, and Feng 2020; Prabhu et al. 2021; Liu, Wang, and Long 2021; Sun, Lu, and Ling 2022).

Adversarial Training (Goodfellow, Shlens, and Szegedy 2015; Jeddi et al. 2020; Shu et al. 2021) aims to improve model’s robustness by injecting adversarial examples. Originally, adversarial training is proposed for supervised learning and requires the ground-truth class labels. Miyato et al. (2018) generalize it to semi-supervised learning and propose Virtual Adversarial Training (VAT). VAT can be viewed as a consistency regularization, whose main idea is to force the model to make similar predictions on clean and perturbed images. VAT is later introduced into domain adaptation to improve local smoothness (Shu et al. 2018). Despite its popularity (Cicce and Soatto 2019; Kumar et al. 2018; Lee et al. 2019b), VAT is commonly used as an auxiliary loss. In contrast, training with VAT alone often leads to poor performance on domain adaptation benchmarks (Liu, Wang, and Long 2021; Lee et al. 2019b; Zhang et al. 2021). Another drawback is that VAT requires an additional back-propagation through the entire network to obtain gradients, which increases the computation cost.

The above observations motivate us to seek more effective ways of adversarial training for UDA problems. A common practice in UDA is to split the training process into a source-training stage and a target adaptation stage. Due to the relevance between the two domains, we find that weights of the linear classification layer change quite slowly in the second stage. In fact, some source-data free works (Liang, Hu, and Feng 2020) freeze the classifier during adaptation and focus on improving feature extractor. Since our goal is to enhance prediction confidence on unlabeled target samples, adversarial training on *penultimate activations*, i.e., the input features of the final linear classification layer, is more correlated with this goal than adversarial training on input images or intermediate features. Our analysis shows that adversarial perturbations on other layer representations can be mapped into perturbations on penultimate activations. The mapping maintains the value of adversarial loss but leads to much higher accuracies. This indicates that it is more effective to manipulate the penultimate activations. Moreover, approximating the optimal adversarial perturbation for penultimate activations is also much easier as it only involves the linear classifier in the optimization problem.

We propose a framework of domain adaptation with adversarial training on penultimate activations. The method is applicable to both standard and source-data free setting. As
activation normalization with $\ell_2$ norm is commonly used inUDA to reduce domain gap (Xu et al. 2019; Prabhu et al. 2021), we derive two variants and systematically analyze the effects of normalization on our adversarial training. The shrinking effect on adversarial loss gradients is discussed. We also analyze the correlations among adversarial perturbations, activation gradients and actual activation changes after model update on real adaptation tasks.

To summarize, our contributions include: 1) we propose a novel UDA framework of adversarial training on penultimate activations; 2) we systematically analyze its relations and advantages over previous adversarial training on input images or intermediate features; and 3) we conduct extensive experiments to validate the superior performance of our method under both standard and source-data free settings.

Related Work

Unsupervised Domain Adaptation. Many UDA methods have been proposed recently (Pan and Yang 2009; Wang and Deng 2018; Wilson and Cook 2020). A mainstream paradigm is feature adaptation that learns domain invariant feature representations. Some approaches minimize domain discrepancy statistics (Long et al. 2017; Sun and Saenko 2016), and others utilize the idea of adversarial learning by updating feature extractor to fool a domain discriminator (Tzeng et al. 2014; Ganin and Lempitsky 2015; Tzeng et al. 2017). Despite the popularity of this paradigm, recent works (Li et al. 2020a; Prabhu et al. 2021) show its intrinsic limitations when the label distribution changes across domains. On the other hand, self-training has become a promising paradigm, which adopts the ideas from semi-supervised learning to exploit unlabeled data. One line of pseudo-labeling uses pseudo-labels generated by the model as supervision to update itself (Liang, Hu, and Feng 2020; Liu, Wang, and Long 2021; Sun, Lu, and Ling 2022). Another line of consistency regularization improves local smoothness using pairs of semantically identical predictions (Sun et al. 2022). Self-training does not force to align source and target domains. Hence it alleviates negative transfer under a large domain shift.

Adversarial training. Adversarial training aims to produce robust models by injecting adversarial examples (Goodfellow, Shlens, and Szegedy 2015; Jeddi et al. 2020; Shu et al. 2021). Many adversarial attack methods have been designed to generate adversarial examples. Among them, gradient-based attack is a major category that obtains adversarial perturbations from adversarial loss gradients on clean data (Goodfellow, Shlens, and Szegedy 2015; Madry et al. 2017). Traditionally, the labeling information is required to generate perturbation directions. Miyato et al. (2018) propose Virtual Adversarial Training (VAT) for semisupervised learning. Adversarial training can be applied on input images (Goodfellow, Shlens, and Szegedy 2015; Miyato et al. 2018) or intermediate features (Jeddi et al. 2020; Shu et al. 2021). Recently, Chen et al. (2022) show that adversarial augmentation on intermediate features yields good performance across diverse visual recognition tasks.

Adversarial training in DA. Shu et al. (2018) first use VAT to incorporate the locally-Lipschitz constraint in conditional entropy minimization. VAT is also used as a smooth regularizer in (Cicek and Soatto 2019; Kumar et al. 2018; Lee et al. 2019b; Li et al. 2020b). Jiang et al. (2020a) devise a bidirectional adversarial training network for SSDA, and penalize VAT with entropy. Liu et al. (2019) generate transferable examples to fill in the domain gap in the feature space. Lee et al. (2019b) propose to learn discriminative features through element-wise and channel-wise adversarial dropout. Kim and Kim (2020) perturb target samples to reduce the intra-domain discrepancy in SSDA.

Approach

Notations and Preliminaries

In UDA, there is a source domain $P_s(X, Y)$ and a target domain $P_t(X, Y)$, where $X$ and $Y$ are the input image space and the label space respectively. We have access to labeled source domain samples $D_s = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$ and unlabeled target domain samples $D_t = \{(x_i^t)\}_{i=1}^{n_t}$. The goal is to learn a classifier $h = g \circ f$, where $f : X \rightarrow Z$ denotes the feature extractor, $g : Z \rightarrow Y$ denotes the class predictor, and $Z$ is the latent feature space. In particular, we assume that $g$ is a linear classifier with trainable weights $W$ and biases $B$, i.e., $g(z) = Wz + B$. $z = f(x)$ is then called the penultimate activation (Seo, Lee, and Kwak 2021).

VAT (Miyato et al. 2018) smooths the model in semi-supervised learning by applying adversarial training on unlabeled data. Its objective is

$$\ell(x) = D[p(y|x), p(y|x + r^{(\epsilon)})]$$

with $r^{(\epsilon)} = \arg \max_r \ell_r(r) = D[p(y|x), p(y|x + r)]$ (1)

where $D[\cdot, \cdot]$ is Kullback-Leibler divergence. Let overlines indicate $\ell_2$ normalization, e.g., $\overline{z} = z/\|z\|_2$ for a vector $z$. In (Miyato et al. 2018), $r^{(\epsilon)}$ is approximated by

$$r^{(\epsilon)} = \epsilon \overline{\nabla_r \ell_r(r)} + \epsilon d$$

in which $\epsilon$ is the perturbation magnitude, $\xi$ is a small constant and $d$ is a random unit vector. The perturbation is “virtually” adversarial because it does not come from the ground-truth labels.

Adversarial Training on Penultimate Activations

Instead of adversarial training on input images or intermediate features, we propose Adversarial training on Penultimate Activations (APA) for unsupervised domain adaptation. We first describe motivations and details of our method, along with two variants with activation normalization in this section. Then we will discuss its relation and advantages over adversarial training on other layer representations, and make an in-depth analysis in the next section.

Two-stage training pipeline. In consistent with previous works (Prabhu et al. 2021; Liang, Hu, and Feng 2020), we split the training process into a source-training stage and a target adaptation stage, shown in Fig. 1. During the first stage, models are trained on source data only with standard cross entropy loss. In the second stage, the obtained source
Figure 1: Framework of the proposed Adversarial Training on Penultimate Activations. APA\textsuperscript{u} and APA\textsuperscript{n} apply adversarial perturbations on the un-normalized and normalized activations, respectively.

models are adapted to the target domain using target data (and source data when available). The adversarial training is applied in the second stage to enhance prediction confidence on unlabeled target samples.

**Motivations.** Since the classifier is initialized with source data and the two domains are relevant, we first want to know how fast the classifier weights change during the second stage. In Fig. 2, we plot the averaged cosine similarity between initial classifier weights $W^{(0)}$ and the weights $W$ during training on 12 tasks of Office-Home. It shows that the weights change quite slowly. Therefore, it is reasonable to assume that the decision boundaries of $g$ change negligibly within a short period of training epochs. Some work (Liang, Hu, and Feng 2020) freeze the classifier during adaptation. We do not choose that, but find freezing the classifier does not make much difference to the performance (c.f. Tab. 7).

To improve prediction confidence on unlabeled target data, a natural way is to move their penultimate activations away from the decision boundaries. This can be realized through adversarial training. Alternatively, we can apply adversarial training on input images or intermediate features to update the penultimate activations in an indirect manner. But this would be less effective. We will discuss this in detail in the approach analysis section.

**Method formulation.** We adopt adversarial training on penultimate activations to improve prediction confidence of target data during the adaptation stage. The objective is

$$\ell^{(p)}(x) = D[g_\sigma (f(x)), g_\sigma (f(x) + r^{(p)})]$$

with $r^{(p)} = \arg\max_{\|r\|_2 \leq \varepsilon} \ell^{(p)}(r)$

$$= \arg\max_{\|r\|_2 \leq \varepsilon} D[g_\sigma (f(x)), g_\sigma (f(x) + r)]$$

where $g_\sigma (\cdot) = \sigma (g(\cdot))$ applies the softmax operator $\sigma$ after the $g$. This $\arg\max$ optimization problem only involves $g$ as $f(x)$ is fixed and $g$ is a linear function, thus can be solved efficiently. In contrast, approximating $r^{(v)}$ needs to back-propagate through the entire highly nonlinear deep neural network, which is often computationally expensive. $r^{(p)}$ can be approximated by

$$r^{(p)} = \varepsilon \cdot \nabla_x \ell^{(p)}(r)|_{r=\varepsilon d}$$

The objective of APA is

$$\min_{f,g} \mathcal{L} = \mathbb{E}_{(x_s,y_s) \sim P_{x}} \ell_{ce}(x_s, y_s) + \beta \mathbb{E}_{x_t \sim P_{x}} \ell^{(p)}(x_t)$$

where $\ell_{ce}$ is cross entropy loss and $\beta$ is a hyper-parameter. Following previous works (Jiang et al. 2020b; Prabhu et al. 2021) that tackle label shift, we use (pseudo) class-balanced sampling during training.

The method can be easily extended to source-data free setting. Since source samples are unavailable during the adaptation stage, we use pseudo labels of confident target samples as an additional supervision term. The objective is

$$\min_{f,g} \mathcal{L}_{sf} = \mathbb{E}_{(x_t, \hat{y}_t) \sim P_{x}} I^r_{\hat{y}_t} \ell_{ce}(x_t, \hat{y}_t) + \beta \ell^{(p)}(x_t)$$

where $\hat{y}_t$ is the pseudo label of $x_t$, $I^r_{\hat{y}_t}$ is 1 if $\max (\hat{y}_t) \geq \tau$ and 0 otherwise. $\tau$ is a confidence threshold.

**Variants with activation normalization.** Using $\ell_2$ normalization on penultimate activations is a common technique to reduce domain gap (Xu et al. 2019; Prabhu et al. 2021). It follows two variants of our method, APA\textsuperscript{u} and APA\textsuperscript{n}, that apply adversarial training on the un-normalized and normalized activations, respectively. Figure 1 illustrates the framework.

The adversarial loss for APA\textsuperscript{u} is

$$\ell^{(p_u)}(x) = D[g_\sigma (f(x)), g_\sigma (f(x) + r^{(p_u)})]$$

with $r^{(p_u)} = \arg\max D[g_\sigma (f(x)), g_\sigma (f(x) + r)]$ (7)

and the adversarial loss for APA\textsuperscript{n} is

$$\ell^{(p_n)}(x) = D[g_\sigma (f(x)), g_\sigma (f(x) + r^{(p_n)})]$$

with $r^{(p_n)} = \arg\max D[g_\sigma (f(x)), g_\sigma (f(x) + r)]$ (8)

where the optimal perturbation $r^{(p_u)}$ and $r^{(p_n)}$ can be approximated in a similar way as Eq. 4. $f(x)$ is also fixed in the $\arg\max$ optimization problems.
Table 1: Accuracies (%) on Office-Home (ResNet-50).

| Method | A→C | A→P | A→R | C→A | C→P | C→R | P→A | P→C | P→R | R→A | R→C | R→P | Avg. |
|--------|-----|-----|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| ResNet-50 | 34.9 | 50.0 | 58.0 | 37.4 | 41.9 | 46.2 | 38.5 | 31.2 | 60.4 | 53.9 | 41.2 | 39.9 | 46.1 |
| DANN    | 45.6 | 59.3 | 70.1 | 47.0 | 58.5 | 60.9 | 46.1 | 43.7 | 68.5 | 63.2 | 51.8 | 76.8 | 57.6 |
| CDAN    | 50.7 | 70.6 | 76.0 | 57.6 | 70.0 | 70.0 | 57.4 | 50.9 | 73.7 | 70.9 | 56.7 | 81.6 | 65.8 |
| SAFN    | 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 |
| MDD     | 54.9 | 73.7 | 77.8 | 60.0 | 71.4 | 71.8 | 61.2 | 53.6 | 78.1 | 72.5 | 60.2 | 82.3 | 68.1 |
| SENTRY  | 61.8 | 77.4 | 80.1 | 66.3 | 71.6 | 74.7 | 66.8 | 63.0 | 80.9 | 74.0 | 66.3 | 84.1 | 72.2 |
| CST     | 59.0 | 79.6 | 83.4 | 68.4 | 77.1 | 76.7 | 68.9 | 56.4 | 83.0 | 75.3 | 62.2 | 85.1 | 73.0 |
| VAT     | 49.1 | 75.0 | 78.6 | 58.4 | 71.4 | 72.4 | 57.0 | 46.4 | 78.2 | 69.4 | 54.0 | 82.8 | 66.1 |
| APA     | 61.2 | 80.0 | 82.4 | 69.8 | 78.3 | 77.4 | 70.5 | 57.9 | 78.9 | 73.9 | 63.6 | 86.1 | 73.8 |
| APA     | 62.0 | 81.2 | 82.6 | 71.5 | 80.6 | 79.3 | 71.9 | 60.1 | 83.4 | 76.9 | 64.2 | 86.1 | 75.0 |
| APA+FM  | 63.1 | 80.6 | 82.6 | 71.8 | 79.7 | 79.4 | 71.4 | 61.5 | 82.7 | 75.9 | 65.5 | 86.3 | 75.0 |
| APA+FM  | 64.0 | 81.6 | 83.7 | 70.9 | 80.3 | 80.3 | 72.8 | 62.2 | 83.0 | 76.8 | 65.5 | 86.8 | 75.7 |

Table 1: Accuracies (%) on Office-Home (ResNet-50).

![Figure 3: (Upper) Mapping input image perturbations of different magnitudes to penultimate activations. (Lower) Mapping intermediate perturbations at different layers to penultimate activations.](image)

We apply an additional perturbation projection step to ensure that $\tilde{f}(x) + r^{(p_n)}$ lies on the unit ball via

$$r^{(p_n)} \leftarrow \tilde{f}(x) + r^{(p_n)} - \tilde{f}(x)$$

(9)

**Approach Analysis**

**Advantages over Adversarial Training on Input Images or Intermediate Features**

Previous works apply adversarial training on input images or intermediate features in domain adaptation. Their performances when using adversarial training alone are unsatisfactory (Liu, Wang, and Long 2021; Lee et al. 2019b; Zhang et al. 2021). When the adversarial perturbation is applied far from the classifier, its impact on model prediction relies on many non-linear transformations in the remaining layers of the deep feature extractor. What is worse, the dimension of input images or intermediate features is usually large (e.g., inputs of $224 \times 224 \times 3$ for a typical computer vision task), making it unfavorable to obtain satisfactory approximations of optimal perturbations from the optimization view.

We show that the perturbation on other layer representations can be mapped into perturbation on the penultimate activations. The mapping maintains the adversarial loss value but leads to higher accuracies.

Let $f = f^0 \circ f^a$ be a decomposition of the feature extractor, in which both can be an identity mapping. We use underlines to highlight modules or variables that do not require gradient. When applying adversarial perturbation $r^{(i)}$ on the output of $f^a$, we have

$$\ell^{(i)}(x) = D[g_\sigma(f(x)), g_\sigma(f^a(x) + r^{(i)})]$$

(10)

The perturbation $r^{(i)}$ can be mapped onto penultimate activation as $r^{(i+\ell)} \triangleq f^\ell(f^a(x) + r^{(i)}) - f(x)$, and the adversarial loss becomes

$$\ell^{(i+\ell)}(x) = D[g_\sigma(f(x)), g_\sigma(f(x) + r^{(i+\ell)})]$$

(11)

Comparing Eq. 10 and Eq. 11, their loss values are identical. However, the backpropagation is different. Figure 3 shows that training with Eq. 10 boosts the accuracies by a large margin than Eq. 10. The explanation is that when using $\ell^{(i+\ell)}$, it directly refines the penultimate activation $f(x)$ of $x$. In contrast, when using $\ell^{(i)}$, it updates the activation of the perturbed sample. Given that many UDA tasks work in a transductive manner where the goal is to make correct prediction for $x^*$, adversarial training on representations other than penultimate activations is less correlated with this goal.

**Interpretation of APA**

APA applies adversarial training on penultimate activations to improve prediction confidence. Intuitively, $f(x)$ is expected to be moved away from the boundaries. Figure 4 illustrates this idea. $r^{(i)}$ is the adversarial perturbation, $\nabla^{(i)}$
In particular, $\nabla_r$ is the direction where the activation is the direction where the activation has the strongest correlation with $u_n$, followed by $\delta(p_n)$, $\delta(p_u)$, and $\delta(p_i)$. This validates that APA is more effective in improving model prediction confidence than image-based training.

**Shrinking Effect of Normalization**

APA$^u$ and APA$^n$ have different properties due to different placing of normalization. With a similar mapping trick, we can create adversarial losses with identical loss values but different gradients. Let $r^{(p_u)} \triangleq r(p_u) \cdot \|f(x)\|$ and $r^{(p_{u-n})} \triangleq f(x) + r(p_u) - f(x)$, it is easy to prove that $\ell(p_u)|_{r^{(p_{u-n})}}$ equals $\ell(p_u)$ and $\ell(p_{u-n})|_{r^{(p_{u-n})}}$ equals $\ell(p_u)$ in terms of objective value. Let $\zeta(v) = v/\|v\|_2$ be the $\ell_2$ normalization function, $x$ omitted, $\zeta_u \triangleq \zeta(f + r^{(p_u)})$, and $\zeta_n \triangleq \zeta(f + r^{(p_n)})$, then with some derivations

$$
\left( \frac{\partial \zeta(u)}{\partial f} \right)^\top = \left( \frac{\partial \zeta(p_u)}{\partial \zeta_u} \right)^\top J_{\zeta}(f + r^{(p_u)}) \quad (12)
$$

$$
\left( \frac{\partial \zeta(n)}{\partial f} \right)^\top = \left( \frac{\partial \zeta(p_n)}{\partial \zeta_n} \right)^\top J_{\zeta}(f) \quad (13)
$$

The Jacobian matrix of $\zeta(v)$ is $J_{\zeta}(v) = (I - vv^\top/(v^\top v))\|v\|$. Using the mapping trick, we can have $\zeta_n = \zeta_u$ and $\partial \zeta_n/\partial \zeta_u = \partial \zeta_n/\partial \zeta_u$. Suppose the second term of $J_{\zeta}$ is negligible, the gradient of $f$ is shrunk by $\|f + r^{(p_u)}\|$ in APA$^u$, while intact in APA$^n$. Figure 5 shows the effect of perturbation magnitude $\epsilon$ on Office-Home. (*) Compensated with activation norm ratio.

**Table 2: Accuracies (%) on VisDA (ResNet-101).**

<table>
<thead>
<tr>
<th>Method</th>
<th>plane</th>
<th>bcycl</th>
<th>bus</th>
<th>car</th>
<th>horse</th>
<th>knife</th>
<th>mcycl</th>
<th>person</th>
<th>plant</th>
<th>skbrd</th>
<th>train</th>
<th>truck</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResNet-101</td>
<td>55.1</td>
<td>53.3</td>
<td>61.9</td>
<td>59.1</td>
<td>80.6</td>
<td>17.9</td>
<td>79.7</td>
<td>31.2</td>
<td>81.0</td>
<td>26.5</td>
<td>73.5</td>
<td>8.5</td>
<td>52.4</td>
</tr>
<tr>
<td>DANN</td>
<td>81.9</td>
<td>77.7</td>
<td>82.8</td>
<td>44.3</td>
<td>81.2</td>
<td>29.5</td>
<td>65.1</td>
<td>28.6</td>
<td>51.9</td>
<td>54.6</td>
<td>82.8</td>
<td>7.8</td>
<td>57.4</td>
</tr>
<tr>
<td>CDAN</td>
<td>85.2</td>
<td>66.9</td>
<td>83.0</td>
<td>50.8</td>
<td>84.2</td>
<td>74.9</td>
<td>88.1</td>
<td>74.5</td>
<td>83.4</td>
<td>76.0</td>
<td>81.9</td>
<td>38.0</td>
<td>73.9</td>
</tr>
<tr>
<td>SAFN</td>
<td>93.6</td>
<td>61.3</td>
<td>84.1</td>
<td>70.6</td>
<td>94.1</td>
<td>79.0</td>
<td>91.8</td>
<td>79.6</td>
<td>89.9</td>
<td>55.6</td>
<td>89.0</td>
<td>24.4</td>
<td>76.1</td>
</tr>
<tr>
<td>SWD</td>
<td>90.8</td>
<td>82.5</td>
<td>81.7</td>
<td>70.5</td>
<td>91.7</td>
<td>69.5</td>
<td>86.3</td>
<td>77.5</td>
<td>87.4</td>
<td>63.6</td>
<td>85.6</td>
<td>29.2</td>
<td>76.4</td>
</tr>
<tr>
<td>VAT</td>
<td>93.5</td>
<td>67.0</td>
<td>86.7</td>
<td>72.1</td>
<td>92.4</td>
<td>63.7</td>
<td>88.2</td>
<td>60.4</td>
<td>90.2</td>
<td>68.9</td>
<td>84.9</td>
<td>12.4</td>
<td>71.9</td>
</tr>
<tr>
<td>APA$^u$</td>
<td>97.1</td>
<td>90.8</td>
<td>84.5</td>
<td>72.8</td>
<td>97.4</td>
<td>96.4</td>
<td>83.9</td>
<td>83.7</td>
<td>95.8</td>
<td>87.3</td>
<td>89.8</td>
<td>42.6</td>
<td>85.2</td>
</tr>
<tr>
<td>APA$^n$</td>
<td>96.9</td>
<td>88.3</td>
<td>83.1</td>
<td>65.9</td>
<td>95.8</td>
<td>95.3</td>
<td>85.7</td>
<td>78.9</td>
<td>94.5</td>
<td>90.1</td>
<td>88.5</td>
<td>46.4</td>
<td>84.1</td>
</tr>
<tr>
<td>APA$^u$+FM</td>
<td>97.2</td>
<td>92.4</td>
<td>89.1</td>
<td>77.5</td>
<td>97.6</td>
<td>97.3</td>
<td>85.2</td>
<td>85.2</td>
<td>95.0</td>
<td>91.9</td>
<td>90.3</td>
<td>39.3</td>
<td>86.5</td>
</tr>
<tr>
<td>APA$^n$+FM</td>
<td>97.0</td>
<td>90.0</td>
<td>86.4</td>
<td>76.3</td>
<td>97.2</td>
<td>97.3</td>
<td>89.4</td>
<td>82.6</td>
<td>94.5</td>
<td>94.6</td>
<td>50.8</td>
<td>52.2</td>
<td>87.4</td>
</tr>
</tbody>
</table>

Figure 4: (Left) Illustration of adversarial perturbation, activation gradient, and actual activation change. (Right) Averaged cosine similarity of these variables during the training process of two UDA tasks. (See text for details)

Figure 5: Comparing APA variants on Office-Home.

Figure 6: Effects of perturbation magnitude $\epsilon$ on Office-Home. (*) Compensated with activation norm ratio.
Table 3: Per-class average accuracies (%) on Office-Home (RS-UT) (ResNet-50).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>69.8</td>
<td>38.3</td>
<td>67.3</td>
<td>35.8</td>
<td>53.3</td>
<td>52.3</td>
<td>52.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DANN</td>
<td>71.6</td>
<td>46.5</td>
<td>68.4</td>
<td>38.8</td>
<td>58.8</td>
<td>58.0</td>
<td>56.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COAL</td>
<td>73.6</td>
<td>42.6</td>
<td>73.3</td>
<td>40.6</td>
<td>59.2</td>
<td>57.3</td>
<td>58.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InstaPBM</td>
<td>75.6</td>
<td>42.9</td>
<td>70.3</td>
<td>39.3</td>
<td>61.8</td>
<td>63.4</td>
<td>58.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MDD+IA</td>
<td>76.1</td>
<td>50.0</td>
<td>74.2</td>
<td>45.4</td>
<td>61.1</td>
<td>63.1</td>
<td>61.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SENTRY</td>
<td>76.1</td>
<td>56.8</td>
<td>73.6</td>
<td>54.7</td>
<td>65.9</td>
<td>64.3</td>
<td>65.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CST</td>
<td>79.6</td>
<td>55.6</td>
<td>77.4</td>
<td>47.1</td>
<td>61.6</td>
<td>62.0</td>
<td>63.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VAT</td>
<td>78.0</td>
<td>45.4</td>
<td>73.0</td>
<td>43.2</td>
<td>60.6</td>
<td>62.1</td>
<td>60.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APA^u</td>
<td>80.2</td>
<td></td>
<td>60.3</td>
<td></td>
<td>53.8</td>
<td>65.7</td>
<td>67.0</td>
<td>67.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APA^n</td>
<td>80.3</td>
<td></td>
<td>57.7</td>
<td></td>
<td>54.9</td>
<td>64.1</td>
<td>67.5</td>
<td>66.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Per-class average accuracies (%) on DomainNet (ResNet-50).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>58.8</td>
<td>67.9</td>
<td>53.1</td>
<td>76.7</td>
<td>53.5</td>
<td>84.4</td>
<td>55.5</td>
<td>60.2</td>
<td>74.6</td>
<td>54.6</td>
<td>57.8</td>
<td>62.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DANN</td>
<td>63.4</td>
<td>73.6</td>
<td>72.6</td>
<td>86.5</td>
<td>65.7</td>
<td>70.6</td>
<td>86.9</td>
<td>73.2</td>
<td>70.1</td>
<td>85.7</td>
<td>75.2</td>
<td>70.0</td>
<td>74.5</td>
<td></td>
</tr>
<tr>
<td>COAL</td>
<td>73.8</td>
<td>75.4</td>
<td>70.5</td>
<td>89.6</td>
<td>69.9</td>
<td>71.3</td>
<td>89.8</td>
<td>73.2</td>
<td>80.5</td>
<td>73.2</td>
<td>70.5</td>
<td>75.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>InstaPBM</td>
<td>80.1</td>
<td>75.9</td>
<td>70.8</td>
<td>89.7</td>
<td>70.2</td>
<td>72.8</td>
<td>89.6</td>
<td>74.4</td>
<td>72.2</td>
<td>87.0</td>
<td>79.7</td>
<td>71.7</td>
<td>77.8</td>
<td></td>
</tr>
<tr>
<td>SENTRY</td>
<td>83.9</td>
<td>76.7</td>
<td>74.4</td>
<td>90.6</td>
<td>76.0</td>
<td>79.5</td>
<td>90.3</td>
<td>82.9</td>
<td>75.6</td>
<td>90.4</td>
<td>82.4</td>
<td>74.0</td>
<td>81.4</td>
<td></td>
</tr>
<tr>
<td>CST</td>
<td>83.9</td>
<td>78.1</td>
<td>77.5</td>
<td>90.9</td>
<td>76.4</td>
<td>79.7</td>
<td>90.8</td>
<td>82.5</td>
<td>76.5</td>
<td>90.0</td>
<td>82.8</td>
<td>74.4</td>
<td>82.0</td>
<td></td>
</tr>
<tr>
<td>VAT</td>
<td>73.6</td>
<td>74.8</td>
<td>65.2</td>
<td>88.1</td>
<td>66.1</td>
<td>64.5</td>
<td>88.2</td>
<td>71.4</td>
<td>70.7</td>
<td>87.3</td>
<td>77.4</td>
<td>69.9</td>
<td>74.8</td>
<td></td>
</tr>
<tr>
<td>APA^u</td>
<td>85.0</td>
<td>79.7</td>
<td>79.0</td>
<td>91.2</td>
<td>77.9</td>
<td>79.3</td>
<td>91.3</td>
<td>82.4</td>
<td>78.5</td>
<td>90.7</td>
<td>84.3</td>
<td>77.7</td>
<td>83.1</td>
<td></td>
</tr>
<tr>
<td>APA^n</td>
<td>83.5</td>
<td>80.4</td>
<td>80.2</td>
<td>91.8</td>
<td>77.2</td>
<td>79.9</td>
<td>91.7</td>
<td>83.7</td>
<td>80.4</td>
<td>91.2</td>
<td>85.4</td>
<td>79.3</td>
<td>83.9</td>
<td></td>
</tr>
</tbody>
</table>

The table compares different APA variants on Office-Home, where APA^{n→u} means using perturbation r_{(p_n→u)} in APA^u. As can be seen, perturbing normalized activations performs slightly better. Figure 6 shows as \( \epsilon \) increases (consequently \( \|f + r_{(p_n)}\| \) increases), the accuracy of perturbing un-normalized activations drops due to shrinking gradients. This can be compensated by magnifying adversarial loss with activation norm ratio of \( \|f + r_{(p_n)}\|/\|f\| \).

**Experiments**

**Setup**

**Datasets.** **Office-Home (OH)** has 65 classes from four domains: Artistic (A), Clip Art (C), Product (P), and Real-world (R). We use both the original version and the RS-UT (Reverse-unbalanced Source and Unbalanced Target) version (Tan, Peng, and Saenko 2020) that is manually created to have a large label shift. **VisDA-2017** (Peng et al. 2017) is a synthetic-to-real dataset of 12 objects. **DomainNet** (Peng et al. 2019) (DN) is a large UDA benchmark. We use the 40-class version (Tan, Peng, and Saenko 2020) from four domains: Clipart (C), Painting (P), Real (R), Sketch (S).

**Baseline methods.** We compare our proposed APA with several lines of methods. Methods for vanilla UDA include DANN (Ganin and Lempitsky 2015), CDAN (Long et al. 2018), MDD (Zhang et al. 2019), SWD (Lee et al. 2019a), SAFN (Xu et al. 2019), CST (Liu, Wang, and Long 2021). Methods that handle label shift include COAL (Tan, Peng, and Saenko 2020), InstaPBM (Li et al. 2020a), MDD+IA (Jiang et al. 2020b), SENTRY (Prabhu et al. 2021), SHOT (Liang, Hu, and Feng 2020), A^2Net (Xia, Zhao, and Ding 2021), NRC (Yang et al. 2021), DIPE (Wang et al. 2022), HCL (Huang et al. 2021) are methods for source-data-free setting. We also compare with self-training based methods using Entropy Minimization, Mutual Information Maximization (MI) (Gomes, Krause, and Perona 2010; Shi and Sha 2012), VAT (Miyato et al. 2018), and FixMatch (FM) (Sohn et al. 2020) loss. The results summarized in Tab. 1–4 are taken from the corresponding papers whenever available.

**Implementation details.** We implement our methods with PyTorch. The pre-trained ResNet-50 or ResNet-101 (He et al. 2016) models are used as the backbone network of \( f \). Then \( f \) contains a bottleneck layer with Batch Normalization. The classification head \( g \) is a single Fully-Connected layer. For all tasks, we use batch size 16, \( \beta = 0.1, \tau = 0.75, \epsilon = 30 \) for APA^u, \( \epsilon = 1.0 \) for APA^n, with an only exception on VisDA where we use \( \beta = 0.04 \) for APA^n instead. More details can be found in the supp. material.

**Results**

**Results on standard benchmarks.** Tables 1,2 present evaluation results on three standard benchmarks. Our method achieves consistently the best scores against previous arts under vanilla setting. The performance is further boosted.
when combined with FixMatch (Sohn et al. 2020). For source-data free setting, A²Net (Xia, Zhao, and Ding 2021) is a strong comparison method employing adversarial inference, contrastive matching, and self-supervised rotation. APA⁺⁺FM improves +1.4% over it on VisDA.

**Results on label-shifted benchmarks.** Tables 3, 4 present evaluation results on two benchmarks with a large label distribution shift. Our methods improve over others tailored for domain gap is large (e.g., tasks with low accuracy). This validates that activation normalization can bring source and target distribution closer, thus reduce the impact of accumulated error in self-training.

**Analysis**

**Effects of activation normalization.** As shown in Fig. 7, using activation normalization consistently improves on all tasks. The performance gain is most significant when the domain gap is large (e.g., tasks with low accuracy). This validates that activation normalization can bring source and target distribution closer, thus reduce the impact of accumulated error in self-training.

![Figure 7: Comparison between using activation normalization (solid markers) and not (hollow markers).](image)

**Comparison with other self-training losses.** To fairly compare with other self-training methods, we implement them under the same framework and hyper-parameters as APA. The only difference is the loss used for target samples. Results are listed in Tab. 6. Among them, FixMatch and SENTRY require additional random augmented target samples. Our method achieves the best scores on all datasets.

<table>
<thead>
<tr>
<th>loss</th>
<th>DN</th>
<th>OH RS-UT</th>
<th>OH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENT</td>
<td>79.00</td>
<td>62.76</td>
<td>70.68</td>
</tr>
<tr>
<td>MI</td>
<td>78.38</td>
<td>64.55</td>
<td>69.81</td>
</tr>
<tr>
<td>FixMatch</td>
<td>76.35</td>
<td>62.12</td>
<td>69.46</td>
</tr>
<tr>
<td>SENTRY</td>
<td>80.31</td>
<td>66.64</td>
<td>73.04</td>
</tr>
<tr>
<td>VAT</td>
<td>74.76</td>
<td>60.37</td>
<td>66.05</td>
</tr>
<tr>
<td>APA⁺⁺</td>
<td>83.08</td>
<td>67.19</td>
<td>73.80</td>
</tr>
<tr>
<td>APA⁺</td>
<td>83.88</td>
<td>66.87</td>
<td>74.98</td>
</tr>
</tbody>
</table>

Table 6: Comparison with other self-training losses.

**Why projecting \( r^{(p_n)} \) back to unit sphere?** In Eq. 9, we post-process \( r^{(p_n)} \) to ensure \( f(x) + r^{(p_n)} \) of unit norm. An alternative way is to add another normalization operation without using perturbation projection by optimizing

\[
\ell'(r_{n}) (x) = D [g_s (f(x)), g_s (f(x) + r^{(p_n)})]
\]  

(14)

However, this will encounter similar shrinking effect of normalization as discussed. Table 5 shows that using perturbation projection consistently performs better than this strategy on all DomainNet tasks.

**Effects of freezing classifier \( g \).** In previous experiments, we allow parameters of the classifier \( g \) to update during the adaptation stage. Whereas \( g \) and the corresponding decision boundaries in the feature space usually change negligibly. Some previous work (Liang, Hu, and Feng 2020) freeze \( g \) during adaptation stage. Table 7 shows that the two strategies perform comparably well in our method.

<table>
<thead>
<tr>
<th>freeze</th>
<th>DN</th>
<th>OH RS-UT</th>
<th>OH</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>83.32</td>
<td>66.34</td>
<td>73.80</td>
</tr>
<tr>
<td>×</td>
<td>83.08</td>
<td>67.19</td>
<td>73.76</td>
</tr>
</tbody>
</table>

Table 7: Effects of freezing parameters of \( g \) during the adaptation stage in APA⁺⁺.

**Sensitivity of hyper-parameters.** APA mainly involves two hyper-parameters, the perturbation magnitude \( \epsilon \) and the adversarial loss weight \( \beta \). Note that the average norm of penultimate activations is about 30. In Fig. 6, We explicitly present results with large perturbations to show its robustness. Figure 8 plots the effects of \( \beta \). Our method is insensitive to the \( \epsilon \) and \( \beta \) within a wide range.

![Figure 8: Effects of \( \beta \) in APA⁺⁺ (solid lines) and APA⁺ (dashed lines).](image)

**Conclusion**

This paper explores adversarial training on penultimate activations in UDA. We show its advantage through comparison with adversarial training on input images and intermediate features. Two variants are derived with activation normalization. Extensive experiments on popular benchmarks are conducted to show the superiority of our method over previous arts under both standard and source-data free setting. Our work demonstrates that adversarial training is a strong strategy in UDA tasks.

9941


