Unsupervised Domain Adaptation without Source Data by Casting a BAIT

Shiqi Yang[1], Yaxing Wang[1], Joost van de Weijer[1], Luis Herranz[1], Shangling Jui[2]
[1] Computer Vision Center, Universitat Autonoma de Barcelona, Barcelona, Spain
[2] Huawei Kirin Solution, Shanghai, China
{syang,yaxing,joost,lherranz}@cvc.uab.es,
jui.shangling@huawei.com

Abstract

Unsupervised domain adaptation (UDA) aims to transfer the knowledge learned from a labeled source domain to an unlabeled target domain. Existing UDA methods require access to source data during adaptation, which may not be feasible in some real-world applications. In this paper, we address the source-free unsupervised domain adaptation (SFUDA) problem, where only the source model is available during the adaptation. We propose a method named BAIT to address SFUDA. Specifically, given only the source model, with the source classifier head fixed, we introduce a new learnable classifier. When adapting to the target domain, class prototypes of the new added classifier will act as a bait. They will first approach the target features which deviate from prototypes of the source classifier due to domain shift. Then those target features are pulled towards the corresponding prototypes of the source classifier, thus achieving feature alignment with the source classifier in the absence of source data. Experimental results show that the proposed method achieves state-of-the-art performance on several benchmark datasets compared with existing UDA and SFUDA methods.

1. Introduction

Though achieving great success, typically deep neural networks demand a huge amount of labeled data for training. However, collecting labeled data is often laborious and expensive. It would, therefore, be ideal if the knowledge obtained on label-rich datasets can be transferred to unlabeled data. For example, after training on synthetic images, it would be beneficial to transfer the obtained knowledge to the domain of real-world images. However, deep networks are weak at generalizing to unseen domains, even when the differences are only subtle between the datasets [25]. In real-world situations, a typical factor impairing the model generalization ability is the distribution shift between data from different domains.

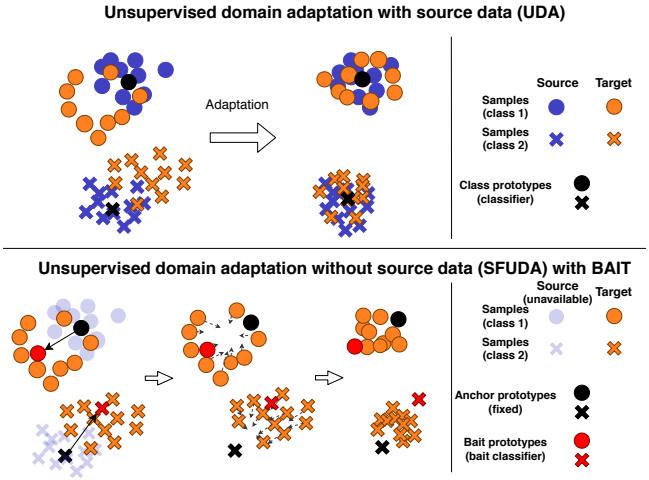


Figure 1: **Top**: Unsupervised domain adaptation with source data. Existing methods demand access to source data to align the feature distribution of target domain with source domain. **Bottom**: The proposed method BAIT only requires access to the source model. It deploys bait prototypes (from newly added bait classifier) to make target features cluster around source prototypes, thus eliminating domain shift.

Unsupervised Domain Adaptation (UDA) aims to reduce the domain shift between labeled and unlabeled target domains. Early works [8, 26] learn domain-invariant features to link the target domain to the source domain. Along with the growing popularity of deep learning, many works benefit from its powerful representation learning ability for domain adaptation [44, 20, 22, 36, 4, 23]. Those methods typically minimize the distribution discrepancy between two domains [19, 20, 22], or deploy adversarial training [36, 44, 4, 23].

However, a crucial requirement in the methodology of these methods is that they require access to the source domain data during the adaptation process to the target domain. Accessibility to the source data of a trained source model is often impossible in many real-world applications, for example deploying domain adaptation algorithms on mobile

Method	**Source-free**	**Avg**
CDAN [21]	×	65.8
CDAN	√	60.7 (5.1%↓)
BNM [5]	×	67.9
BNM	√	61.3 (6.6%↓)

Table 1: Accuracy on Office-Home under normal setting and source-free setting. If only provided source model without source data, the normal UDA method will get lower results.

devices where the computation capacity is limited, or in situations where data-privacy rules limit access to the source domain. Without access to the source domain data, the above methods suffer from inferior performance. For example, the state-of-the-art methods CDAN [21] and BNM [5] will get much lower performance (5.1% and 6.6% drop respectively, as shown in Tab. 1), if only provided with source model instead of source data during adaptation.

Because of its relevance and practical interest, the *source-free unsupervised domain adaptation* (SFUDA) setting where the model is first trained on the source domain and has no longer access to the source data afterwards, has started to get traction recently [13, 18, 17, 14]. Among theses methods, USFDA [13] addresses universal DA [42] and SF [14] addresses for open-set DA [32]. Both have the drawback of requiring to generate images or features of non-existing categories. SHOT [18] and MA [17] address close-set SFUDA. MA [17] is based on target-style image generation by a conditional GAN, which demands a large computation capacity and is time-consuming. Meanwhile, SHOT [18] proposes to transfer the source hypothesis, i.e. the fixed source classifier, to the target data. The main gain in SHOT is from its pseudo-labeling strategy. However, SHOT has two limitations. First, it needs to access all target data for computing pseudo label, only after this phase it can start the adaptation to the target domain. This might be infeasible in some online streaming applications, where target data cannot be revisited, for example in robotics applications. Secondly, it heavily depends on pseudo-labels being correct, but some wrong pseudo-labels could compromise the training procedure.

Existing UDA methods do not fully utilize the implicit information inside the classifier. These methods aim to apply the model to both labeled source and unlabeled target data, where a domain shift exists. This means that the classifier should correctly predict both source and target data. From the perspective of linear classifiers as class prototypes (represented by the weights of the classifier), the source and target features should cluster around their corresponding class prototype, which is dominated by labeled source data. Based on this analysis, instead of aligning features from two domains leveraging all source and target data (as UDA methods do), we propose to directly align the target features with the source classifier. Thus, the adaptation to the target domain can be achieved without accessing the source data. Although SHOT also proposes to make target features match source classifier, it achieves adaptation by utilizing source classifier for pseudo labeling which may result in noisy information.

As illustrated in Fig. 1, in our method, after getting the source model, we first propose to freeze the classifier head of source model during the whole adaptation process. And then we add an extra classifier (called *bait* classifier) initialized from the source classifier (referred to as *anchor* classifier). The class prototype (i.e. the weights of the classifier) of the new added bait classifier will move towards the target features, acting as an estimated prototype of target features. The feature extractor then will drive target features towards the prototype of the anchor classifier. Through this process, all target features are expected to cluster around the corresponding class prototype of the anchor classifier, thus achieving the adaptation. In the experiment, we show that our method, which is dubbed as BAIT, outperforms all existing normal UDA methods, even though these methods can access source data at all time. Moreover, we also show our proposed BAIT surpasses other SFUDA methods.

We summarize our contributions as follows:

- We propose a new method for the challenging source-free domain adaptation scenario. Our method does neither require image generation as in [17, 13, 14] and does not require the usage of pseudo-labeling [18].
- Our method prevents the need for source data by deploying an additional classifier to align target features with the corresponding class prototypes of source classifier.
- We demonstrate that the proposed BAIT approach obtains similar results or outperforms existing UDA and SFUDA methods on several datasets, even without access to source data. Notably, we obtain state-of-the-art performance on the large domain adaptation dataset VisDA.

2. Related Works

Domain adaptation with source data. Domain adaptation aims to reduce the shift between the source and target domains. Moment matching methods align feature distributions by minimizing the feature distribution discrepancy, including methods such as DAN [20] and DDC [37], which deploy Maximum Mean Discrepancy. CORAL [34] matches the second-order statistics of the source to target domain. Inspired by adversarial learning, DANN [6] formulates domain adaptation as an adversarial two-player game. CDAN [21] trains a deep networks conditioned on several sources of information. DIRT-T [33] performs domain adversarial training with an added term that penalizes violations of the cluster assumption. Domain adaptation has also been tackled from other perspectives. RCA [4] proposes a multi-classification discriminator. DAMN [1] introduces a framework where

each domain undergoes a different sequence of operations. AFN [40] shows that the erratic discrimination of target features stems from much smaller norms than those found in source features.

Another set of methods also deploy two or multiple classifiers for domain adaptation, such as MCD [31], ADR [30], SWD [15], CLAN [24] and STAR [23], but the motivations are totally different. All those methods adopt prediction diversity between multiple learnable classifiers to achieve local or category-level feature alignment between source and target domains, while our proposed BAIT uses the class prototype of second classifier to estimate the centroid of target features, aiming to align target features with the *fixed* source classifier in the absence of source data.

Domain adaptation without source data. All these methods, however, require access to source data during adaptation. Recently, USFDA [13] and FS [14] explore the source-free setting, but they focus on the universal DA task [42] and open-set DA [32], where the label spaces of source and target domain are not identical. And their proposed methods are based on generation of simulated negative labeled samples during source straining period, in order to increase the generalization ability for unknown class. Most relevant works are SHOT [18] and MA [17], both are for close-set DA. SHOT needs to compute and update pseudo labels before updating model, which has to access all target data and may also have negative impact on training from the noisy pseudo labels, and MA needs to generate target-style training images based on conditional GAN, which demands large computation capacity.

Unlike these methods, our method introduces an additional classifier to achieve feature alignment with the fixed source classifier. This idea is motivated from multiple classifier based UDA methods [31, 30, 15, 24, 23], however there the source data is crucial for the training of the second classifier. By using the entropy of the source classifier, we split the target data in two groups, thereby allowing us to train the second classifier without the need of source data. The additional classifier is used to cluster target features around the prototype of the fixed classifier. The idea of fixing the classifier is also proposed in [18], but other then us they utilize the fixed classifier to produce pseudo labels. This allows us to avoid the negative impact of training with noisy pseudo-labels.

3. BAIT for Source-Free Domain Adaptation

We denote the labeled source domain data with n_s samples as $\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$, where the y_i^s is the corresponding label of x_i^s, and the unlabeled target domain data with n_t samples as $\mathcal{D}_t = \{x_j^t\}_{j=1}^{n_t}$, and the number of classes is K. Usually UDA methods eliminate the domain shift by aligning the feature distribution between the source and target domains. Unlike the normal setting, we consider the more challenging SFUDA setting which during adaptation to the target data has no longer access to the source data, and has only access to the model trained on the source data. In the following sections, we elaborate our method under SFUDA setting.

3.1. Prototype of source classifier as anchor

We decompose the neural network into two parts: a feature extractor f, and a classifier head C_1 which only contains one fully connected layer (with weight normalization). We first train the baseline model on the labeled source data $\mathcal{D}_s$ with standard cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{n_s}\sum_{i=1}^{n_s}\sum_{k=1}^{K} I_{[k=y_i^s]} \log p_k(x_i^s) \tag{1}$$

where the p_k is the k-th element of the softmax output, and $I_{[z]}$ is the indicator function which is 1 if z is true, and 0 otherwise.

A closer look at the training process of UDA methods unveils that the feature extractor aims to learn a discriminative representation, and the classifier strives to distinguish the representations of the various classes. UDA methods tackle domain shift by aligning the feature distribution (from the feature extractor) of the source and target domains. A successful alignment of the features means that the features produced by the feature extractor f from both domains will be classified correctly by the classifier head.

Since the k-th class prediction is computed by $\mathbf{p_k}(\mathbf{x}) = \sigma(\|\mathbf{W_k}^{\mathrm{T}}\| f(\mathbf{x}))$ where the $\|\mathbf{W_k}\|$ is the L2 normalized weight of classifier, similar to [2, 29], intuitively we can regard the normalized weight of each class in the classifier as a class prototype, as shown a in Fig 2(top). Due to the domain shift, the cluster of target features generated by the source-training feature extractor will deviate from the source class prototype. So from the perspective of the class prototype, a good model after domain adaptation should have all source and target features gathering around their corresponding class prototype.

We freeze the source-trained classifier C_1. This implicitly allows us to store the relevant information from the source domain, *i.e.*, the fixed source class prototypes in the feature space, without actually accessing the source data. With the source class prototype fixed as an anchor in the feature space, target features should cluster around the corresponding prototype. Since the source classifier is fixed all the time and the features from target domain are expected to gather around those class prototypes, we refer to classifier C_1 as the *anchor classifier*, and its class prototype as *anchor prototype*.

3.2. Prototype of second classifier as bait

For the fixed anchor classifier to be successful for source-free domain adaptation we require to address two problems.

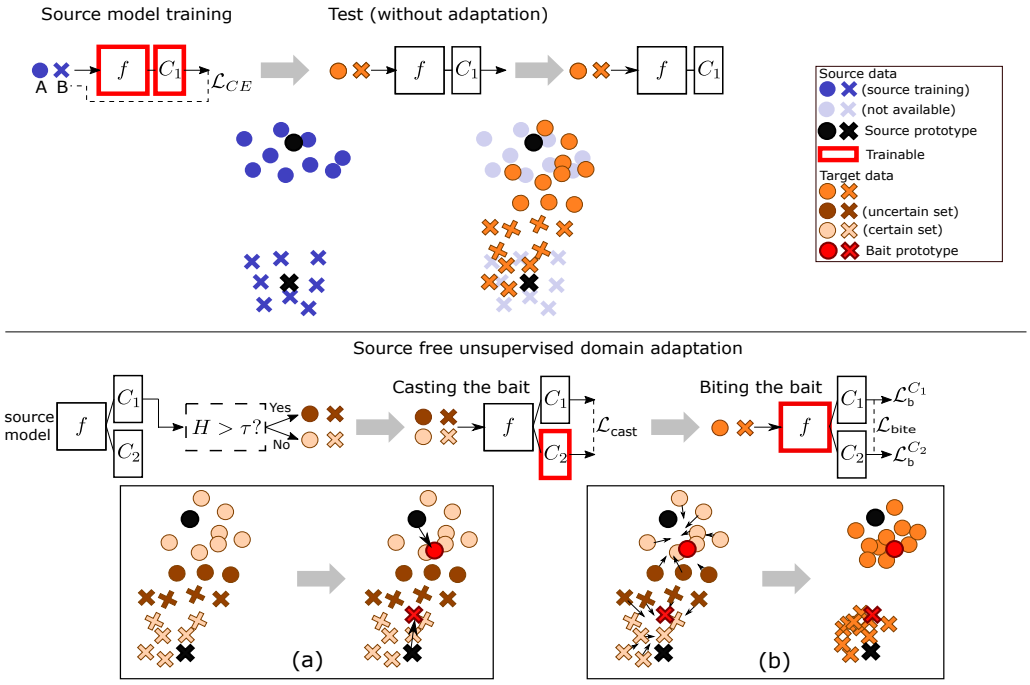


Figure 2: Illustration of training process. The top shows that the source-training model fails on target domain due to domain shift. The bottom illustrates our adaptation process. Bottom (a): splitting feature into certain set and uncertain set by whether the prediction entropy H is above the threshold τ, then casting the bait prototype towards uncertain set while also staying close to certain set. Bottom (b): training feature extractor push all features towards both prototypes of C_1 and C_2, thus achieving aligning target features with source classifier.

First, part of the target data will not be well classified (have uncertain predictions) due to the domain shift, and this data needs to be identified. Secondly, we have to adapt the feature extractor in such a way that this data can subsequently be classified correctly by the anchor classifier. Therefore, we propose the BAIT method that is a two-step algorithm which exactly addresses these two problems. Our method is shown in Fig. 2(bottom). After training the model on the source data, we get a feature extractor f, and an anchor classifier C_1. We fix C_1 in the subsequent training periods, and use it to initialize an extra classifier C_2. The prototypes of the extra classifier C_2 are optimised to approach target features which are not clearly classified by C_1. Next those target features are to be pulled towards the source class prototype. Hereafter we refer to the classifier C_2 as the *bait classifier*, and its class prototype as *bait prototype*.

In order to train the desired C_2, we propose a 2-step training policy which alternately trains bait classifier C_2 and feature extractor f.

Step 1: casting the bait. In step 1, we only train bait classifier C_2, and freeze feature extractor f. As shown in Fig. 2(top), due to domain shift, some target features will not locate around the source prototype, which is also referred to as misalignment [4, 10]. In order to pull target features towards the anchor prototype, *i.e.*, aligning target features

Algorithm 1 Unsupervised domain adaptation with BAIT

Require: $\mathcal{D}_t$ ▷ unlabeled target data
Require: f, C_1 ▷ network trained with source data $\mathcal{D}_s$
$C_2 \leftarrow C_1$
while not done **do**
  Sample batch $\mathcal{T}$ from $\mathcal{D}_t$
  Entropy based splitting: $\mathcal{U}$ and $\mathcal{C}$ ▷ Eq. 2
  $C_2 \leftarrow \text{argmin}_{C_2} \mathcal{L}_{\text{cast}}(C_2)$ ▷ Eq. 3
  $f \leftarrow \text{argmin}_f \mathcal{L}_{\text{bite}}(f) + \mathcal{L}_b(f)$ ▷ Eq. 4& 5
end while

with the source classifier, we deploy bait prototypes to find those features far away from the anchor prototypes. In order words, the bait classifier aims to estimate the centroid of target features for which the anchor classifier has uncertain predictions. Therefore before adaptation, we split the features of the current batch of data into two sets: the uncertain $\mathcal{U}$ and certain set $\mathcal{C}$, as shown in Fig. 2 (a), according to their prediction entropy:

$$\begin{aligned} \mathcal{U} &= \left\{ x | x \in \mathcal{D}_t, H\left(p^{(1)}(x)\right) > \tau \right\} \\ \mathcal{C} &= \left\{ x | x \in \mathcal{D}_t, H\left(p^{(1)}(x)\right) \leq \tau \right\} \end{aligned} \tag{2}$$

where $p^{(1)}(x) = \sigma(C_1(f(x))$ is the prediction of the an-

chor classifier (σ represents the softmax operation) and $H(p(x)) = -\sum_{i=1}^{K} p_i \log p_i$. The threshold τ is estimated as a percentile of the entropy of $p_1(x)$ in $\mathcal{T}$, set to 50% (i.e. the median).

Then, we make the bait prototype move toward those higher entropy features, but still stay nearby target features with lower entropy. We achieve this by increasing symmetric KL-divergence between predictions on $\mathcal{U}$ from C_1 and C_2, while decreasing it on $\mathcal{C}$:

$$\begin{aligned}\mathcal{L}_{\text{cast}}(C_2) = &\sum_{x\in\mathcal{C}} D_{SKL}\left(p^{(1)}(x), p^{(2)}(x)\right) \\ &- \sum_{x\in\mathcal{U}} D_{SKL}\left(p^{(1)}(x), p^{(2)}(x)\right)\end{aligned} \tag{3}$$

where D_{SKL} is the symmetric KL divergence: $D_{SKL}(a,b) = \frac{1}{2}\left(D_{KL}(a|b) + D_{KL}(b|a)\right)$.

As shown in Fig. 2 (a), given that C_2 is initialized from C_1, increasing the KL-divergence on the uncertain set between two classifiers will drive the prototype of C_2 to those features with higher entropy. Decreasing it on the certain set encourages the two classifiers have similar prediction for those features. This will ensure that the bait prototype will not go too far.

Step 2: biting the bait. In this stage, we only train the feature extractor f, aiming to pull target features towards anchor prototypes. While it is hard to directly drive all target features to the correct anchor prototype, we seek to make target features cluster around both anchor and bait prototypes.

Specifically, we update the feature extractor f to move target featurses towards both the bait and anchor prototype by minimizing the proposed bite loss:

$$\mathcal{L}_{\text{bite}}(f) = \sum_{i=1}^{n_t}\sum_{k=1}^{K}[-p_{i,k}^{(2)} \log p_{i,k}^{(1)} - p_{i,k}^{(1)} \log p_{i,k}^{(2)}] \tag{4}$$

By minimizing this loss, the prediction distribution of the bait classifier should be similar to that of the anchor classifier and vice verse, which means target features are excepted to get closer to both bait and anchor prototypes.

Intuitively, as shown in Fig. 2 (b), minimizing the bite loss $\mathcal{L}_{\text{bite}}$ will push target features towards prototypes of the two classifiers. Metaphorically, in this stage target features bite the bait (prototype) and be pulled towards anchor (prototype), indicating they are clustering around both bait and anchor prototypes.

Additionally, in order to avoid the degenerate solutions [7], which allocate all uncertain features to a few anchor class prototype, we adopt the class balance loss (CB loss) $\mathcal{L}_b$ to regularize the feature extractor:

$$\mathcal{L}_{\text{b}}(f) = \sum_{k=1}^{K}[KL(\bar{p}_k^{(1)}(x))||\boldsymbol{q}_k) + KL(\bar{p}_k^{(2)}(x))||\boldsymbol{q}_k)] \tag{5}$$

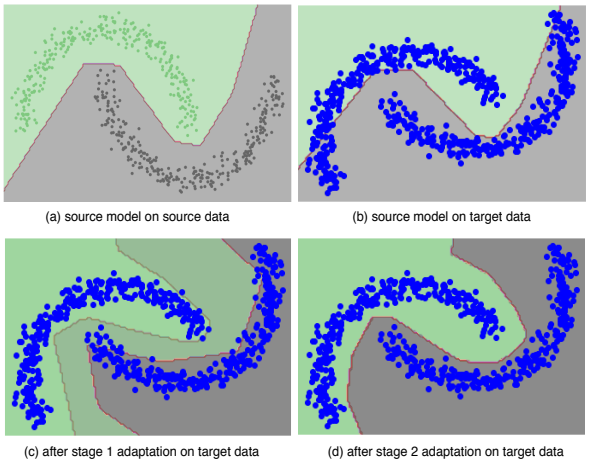


Figure 3: Toy experiment on the twinning moon 2D dataset. The blue points refer to target data. The green and grey refer to source data. Decision boundaries after training model only on the source data and testing on source (a) and target (b) data. (c) After stage 1 training in the middle of adaptation with only target data. The two borderlines denote two decision boundaries (with C_1 in red). (d) After stage 2 training, the two decision boundaries almost coincide.

where $\bar{p}_k = \frac{1}{n_t}\sum_{x\in\mathcal{D}_t} p_k(x)$ is the empirical label distribution, and $\boldsymbol{q}$ is uniform distribution $\boldsymbol{q}_k = \frac{1}{K}, \sum_{k=1}^{K}\boldsymbol{q}_k = 1$. With the class balance loss $\mathcal{L}_b$, the model is expected to have more balanced prediction.

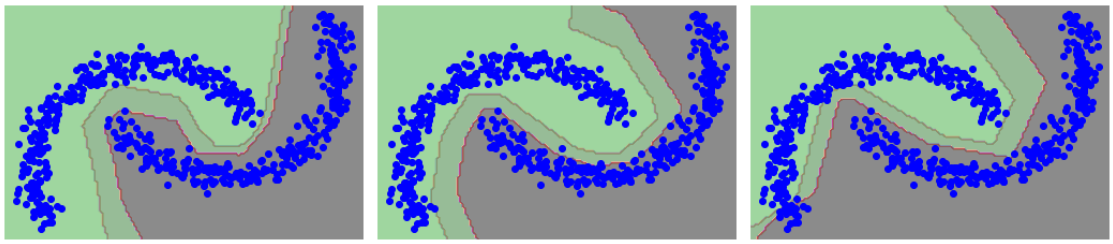
Overall, the whole adaptation process is illustrated in Algorithm 1. Note that this 2-step training happens in every mini-batch iteration during adaptation.

4. Experiments

4.1. Experiment on Twinning moon dataset

We carry out our experiment on the twinning moon dataset. For this data set, the data samples from the source domain are represented by two inter-twinning moons, which contain 300 samples each. We generate the data in the target domain by rotating the source data by 30°. Here the rotation degree can be regarded as the domain shift. First we train the model only on the source domain, and test the model on all domains. As shown in Fig. 3(a) and (b), due to the domain shift the model performs worse on the target data. Then we adapt the model to the target domain with the anchor and bait classifiers, without access to any source data. As shown in Fig 3(c) during adaptation the bait loss moves the decision boundary[1] of the bait classifier to different regions than the anchor classifier. After adaptation the two decision boundaries almost coincide and both classifiers give the correct prediction, as shown in Fig. 3(d), indicating that target

[1]Note here the decision boundary is from the whole model, since the input are data instead of features.



(a) (b) (c)

Figure 4: Visualization on twinning-moon after adaptation on target data, but without weight normalization on classifier. All data are target data. (a) Using τ (threshold for splitting) with the median of prediction entropy, as in the paper. (b) Decaying the τ (from the median to 0) during adaptation. (c) Not using splitting.

Method (Synthesis → Real)	**Source-free**	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	Per-class
ADR [30]	×	94.2	48.5	84.0	72.9	90.1	74.2	**92.6**	72.5	80.8	61.8	82.2	28.8	73.5
CDAN [21]	×	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
CDAN+BSP [3]	×	92.4	61.0	81.0	57.5	89.0	80.6	90.1	77.0	84.2	77.9	82.1	38.4	75.9
SWD [16]	×	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
MDD [43]	×	-	-	-	-	-	-	-	-	-	-	-	-	74.6
IA [10]	×	-	-	-	-	-	-	-	-	-	-	-	-	75.8
DMRL [39]	×	-	-	-	-	-	-	-	-	-	-	-	-	75.5
MCC [11]	×	88.7	80.3	80.5	71.5	90.1	93.2	85.0	71.6	89.4	73.8	85.0	36.9	78.8
SHOT [18]	√	92.6	81.1	80.1	58.5	89.7	86.1	81.5	77.8	89.5	84.9	84.3	**49.3**	79.6
SFDA [12]	√	86.9	81.7	**84.6**	63.9	93.1	91.4	86.6	71.9	84.5	58.2	74.5	42.7	76.7
*MA [17]	√	**94.8**	73.4	68.8	**74.8**	93.1	**95.4**	88.6	**84.7**	89.1	84.7	83.5	48.1	<u>81.6</u>
BAIT (ours)	√	93.7	**83.2**	84.5	65.0	92.9	**95.4**	88.1	80.8	**90.0**	**89.0**	84.0	45.3	**82.7**

Table 2: Accuracies (%) on VisDA-C for ResNet101-based unsupervised domain adaptation methods. **Source-free** means setting without access to source data during adaptation. Underlined results are second highest result. * means method needs to generate extra images.

Method	**Source-free**	A→D	A→W	D→W	W→W	D→D	A→W	A→A	Avg
MCD [31]	×	92.2	88.6	98.5	**100.0**	69.5	69.7	86.5	
CDAN [21]	×	92.9	94.1	98.6	**100.0**	71.0	69.3	87.7	
MDD [43]	×	90.4	90.4	98.7	99.9	75.0	73.7	88.0	
MDD+IA [10]	×	92.1	90.3	98.7	99.8	**75.3**	74.9	88.8	
BNM [5]	×	90.3	91.5	98.5	**100.0**	70.9	71.6	87.1	
DMRL [39]	×	93.4	90.8	99.0	100.0	73.0	71.2	87.9	
BDG [41]	×	93.6	93.6	99.0	100.0	73.2	72.0	88.5	
MCC [11]	×	95.6	95.4	98.6	100.0	72.6	73.9	89.4	
SRDC [35]	×	**95.8**	**95.7**	**99.2**	100.0	76.7	77.1	**90.8**	
*USFDA [13]	√	-	-	-	-	-	-	85.4	
SHOT [18]	√	93.1	90.9	98.8	99.9	74.5	74.8	88.7	
SFDA [12]	√	92.2	91.1	98.2	99.5	71.0	71.2	87.2	
*MA [17]	√	92.7	93.7	98.5	99.8	**75.3**	**77.8**	<u>89.6</u>	
BAIT (ours)	√	92.0	94.6	98.1	**100.0**	74.6	75.2	89.1	

Table 3: Accuracies (%) on Office-31 for ResNet50-based unsupervised domain adaptation methods. **Source-free** means setting without access to source data during adaptation. Underline means the second highest result. * means method needs to generate extra images.

features cluster around both the anchor and bait prototypes.

Importance of entropy based splitting. In the paper, we propose to split the data in a certain/uncertain set (see Eq. 2) to avoid the trivial solution that the bait prototypes move to a position faraway from the anchor prototypes and target features, which may lead to failure of the model. Although in the following experiment, we find splitting the batch into certain/uncertain set only improves the results a bit, we conjecture the reasons are twofold: (a) the $\mathcal{L}_{cast}$ is only applied on the current batch data instead of all data, (b) we also apply weight normalization on the classifier, which will ensure that the prototypes will remain close to each other.

To prove importance of this splitting, we conduct additional experiments on the twining-moon dataset, where *one batch contains all target data and there is no weight normalization inside the classifier*. As shown in Fig. 4, we visualize the decision boundaries after adaptation with different settings for the certain/uncertain splitting: (a) the same splitting as proposed in our paper, (b) expanding the uncertain set during the adaptation (τ decaying from the median entropy) and (c) no splitting. First, we find that expanding the uncertain set can also work as shown in Fig 4(b), since features from the certain set are already pulled quite close to the anchor prototypes in the earlier stages of training. And as shown in Fig. 4 (c), when not splitting the data, this will damage the training, as the decision boundary crosses the data.

4.2. Experiments on recognition benchmarks

Datasets. We use three benchmark datasets. **Office-31** [28] contains 3 domains (Amazon, Webcam, DSLR) with 31

Method	**Source-free**	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
MCD [31]	×	48.9	68.3	74.6	61.3	67.6	68.8	57.0	47.1	75.1	69.1	52.2	79.6	64.1
CDAN [21]	×	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
MDD [43]	×	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
MDD+IA [10]	×	56.0	77.9	79.2	64.4	73.1	74.4	64.2	54.2	79.9	71.2	58.1	83.1	69.5
BNM [5]	×	52.3	73.9	80.0	63.3	72.9	74.9	61.7	49.5	79.7	70.5	53.6	82.2	67.9
BDG [41]	×	51.5	73.4	78.7	65.3	71.5	73.7	65.1	49.7	81.1	74.6	55.1	84.8	68.7
SRDC [35]	×	52.3	76.3	81.0	**69.5**	76.2	78.0	**68.7**	53.8	81.7	**76.3**	57.1	**85.0**	71.3
SHOT [18]	√	56.9	**78.1**	81.0	67.9	**78.4**	**78.1**	67.0	54.6	81.8	73.4	58.1	84.5	**71.6**
SFDA [12]	√	48.4	73.4	76.9	64.3	69.8	71.7	62.7	45.3	76.6	69.8	50.5	79.0	65.7
BAIT (ours)	√	**57.4**	77.5	**82.4**	68.0	77.2	75.1	67.1	**55.5**	**81.9**	73.9	**59.5**	84.2	**71.6**

Table 4: Accuracies (%) on Office-Home for ResNet50-based unsupervised domain adaptation methods. **Source-free** means source-free setting without access to source data during adaptation. Underline means the second highest result.

classes and 4,652 images. **Office-Home** [38] contains 4 domains (Real, Clipart, Art, Product) with 65 classes and a total of 15,500 images. **VisDA** [27] is a more challenging datasets, with 12-class synthesis-to-real object recognition tasks, its source domain contains 152k synthetic images while the target domain has 55k real object images.

Baseline methods On all these three datasets, we compare our BAIT with normal UDA methods which have access to source data, including MCD [31], CDAN [21], Implicit Alignment [10], BNM [5], MDD [43], DMRL [39], BDG [41], MCC [11] and SRDC [35]. We will compare with the recent SFUDA methods, including USFDA [13], SHOT [18], SFDA [12] and MA [17].

Model details We adopt the backbone of ResNet-50 [9] (for office datasets) or ResNet-101 (for VisDA) along with an extra fully connected (fc) layer as feature extractor, and a fc layer as classifier head. We adopt SGD with momentum 0.9 and batch size of 128 on all datasets. On the source domain, the learning of the ImageNet pretrained backbone and the newly added layers are 1e-3 and 1e-2 respectively, except for the ones on VisDA, which are 1e-4 and 1e-3 respectively. We further reduce the learning rate 10 times training on the target domain. We train 20 epochs on the source domain, and 30 epochs on the target domain. All experiments are conducted on a single RTX 6000 GPU. All results are reported from the classifier C_1. *We attach our code in supplemental material.*

Quantitative Results. The results on the VisDA, Office-31 and Office-Home dataset are shown in Tab. 2-4. In these tables, the top part (denoted by × on *source-free* column) shows results for the normal setting with access to source data during adaptation. The bottom one (denoted by √ on *source-free* column) shows results for the source-free setting. As reported in Tab. 2 and Tab. 4, our method outperforms state-of-the-art methods which have access to source data on VisDA and Office-Home, and achieves competitive results on Office-31 (Tab. 3).

The proposed method still obtains the best performance when comparing with current source-free methods. In particular, our method surpasses SFDA [12] by 6.0%, SHOT [18] by 3.1%, and MA [17] by 1.1% on the more challenging VisDA dataset (Tab. 2). Note that MA highly relies on the extra synthesized data. On Office-31 (Tab. 3), the proposed BAIT achieves better result than SFDA and SHOT, and competitive result to MA. Our BAIT surpasses SFDA by a large margin, and is on par with SHOT on Office-Home (Tab. 4). The reported results clearly demonstrate the efficacy of the proposed method without access to the source data during adaptation.

Method	Avg.
Source only	46.1
Single classifier (w/ $\mathcal{L}_b$)	52.4
BAIT (w/o $\mathcal{L}_b$, w/ splitting)	64.5
BAIT (w/ $\mathcal{L}_b$, w/o splitting)	70.6
BAIT	**71.6**

Table 5: Ablation study on Office-Home dataset in the source-free setting. *Single classifier* (w-$\mathcal{L}_b$) is to adapt the source model to target domain by optimizing $\mathcal{L}_b$. Splitting is performed according to Eq. 2.

Ablation Study. We conduct an ablation study to isolate the validity of the key components of our method: the bait classifier and the splitting mechanism. As reported in the second row of Tab. 5 on the Office-Home dataset, directly testing the model on target domain shows the worst performance. Adapting the model trained on the source domain to the target domain with one classifier (the third row of Tab. 5), we are able to improve the result, but still obtain relatively low accuracy. Adding the second classifier and applying the splitting mechanism results in much better results. Also leveraging the $\mathcal{L}_b$ loss improves results significantly. Finally, combining both the $\mathcal{L}_b$ loss and the splitting mechanism, we achieve the best score.

Evolution of the accuracy over time. Fig. 5 shows the evolution of the accuracy during adaptation to the target domain of Office-Home for a single classifier (only minimizing the class balance loss $\mathcal{L}_b$), and the two classifiers of BAIT (C_1 and C_2). The starting point is the accuracy after training on the source data. The single classifier has an ephemeral im-

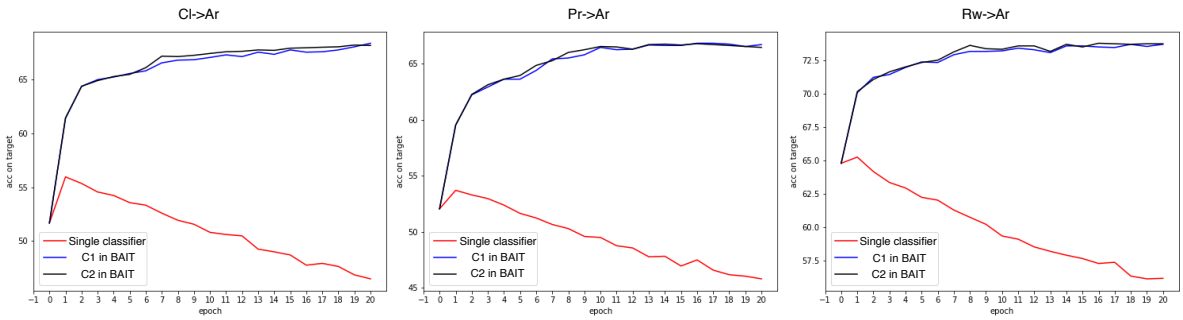


Figure 5: Accuracy curves when training on target data, on three subtasks of Office-Home: Cl→Ar, Pr→Ar and Rw→Ar. The three curves correspond to the accuracy with C_1, C_2 in BAIT and only one classifier with class balance loss respectively.

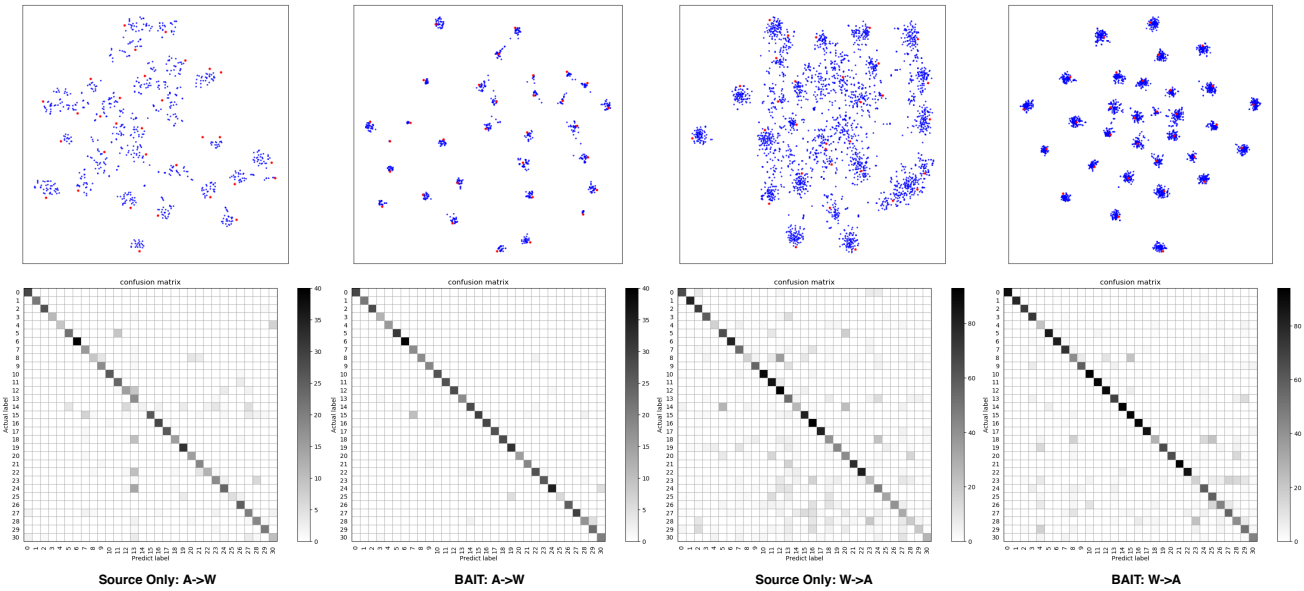


Figure 6: **Top**: t-SNE visualization for features from the target domain. The red points are the class prototype from C_1. Zoom in for better inspection. **Bottom**: Confusion matrix for A→W and W→A (Office-31) of the source model and BAIT. The Y-axis shows the ground truth labels while the X-axis shows the predicted labels.

provement (about 2 epochs), followed by an abrupt change towards a progressive decline. This may imply that target features are randomly grouped around a few anchor prototypes. The two classifiers of BAIT, however, have faster and more stable convergence, thanks to the bait classifier trying to find and push target features towards the corresponding anchor prototypes.

Embedding visualization. Fig. 6 (top) shows the t-SNE visualization of target features obtained with the source model and after adaptation with BAIT. Target features form more compact and clear clusters after BAIT than in the source model, indicating that BAIT produces more discriminative features. We also show the class prototype (red points), which shows target features indeed cluster around the corresponding prototypes.

Confusion matrices. Fig. 6 (bottom) shows the confusion matrices of both the source model and BAIT for the two subtasks A→W and W→A on Office-31. They show that BAIT results in significantly fewer misclassifications, further verifying the effectiveness of our method.

5. Conclusion

There are many practical scenarios where source data may not be available (e.g. due to privacy or availability restrictions) or may be expensive to process. In this paper we study this challenging yet promising unsupervised domain adaptation setting (i.e. SFUDA), and propose BAIT, a fast and effective approach. BAIT considers the source classifier as a (fixed) collection of anchor class prototypes, and aligns target features with them via an extra bait classifier that locates uncertain target features and drags them towards those anchor prototypes. The experimental results show that BAIT achieves state-of-the-art SFUDA performance on

several datasets, even outperforming other UDA methods with access to source data.

References

[1] Roger Bermudez Chacon, Mathieu Salzmann, and Pascal Fua. Domain-adaptive multibranch networks. In *8th International Conference on Learning Representations*, 2020. 2

[2] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019. 3

[3] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *International Conference on Machine Learning*, pages 1081–1090, 2019. 6

[4] Safa Cicek and Stefano Soatto. Unsupervised domain adaptation via regularized conditional alignment. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1416–1425, 2019. 1, 2, 4

[5] Shuhao Cui, Shuhui Wang, Junbao Zhuo, Liang Li, Qingming Huang, and Qi Tian. Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations. *CVPR*, 2020. 2, 6, 7

[6] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 2

[7] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision*, pages 5736–5745, 2017. 5

[8] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073. IEEE, 2012. 1

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[10] Xiang Jiang, Qicheng Lao, Stan Matwin, and Mohammad Havaei. Implicit class-conditioned domain alignment for unsupervised domain adaptation. *arXiv preprint arXiv:2006.04996*, 2020. 4, 6, 7

[11] Ying Jin, Ximei Wang, Mingsheng Long, and Jianmin Wang. Minimum class confusion for versatile domain adaptation. *ECCV*, 2020. 6, 7

[12] Youngeun Kim, Sungeun Hong, Donghyeon Cho, Hyoungseob Park, and Priyadarshini Panda. Domain adaptation without source data. *arXiv preprint arXiv:2007.01524*, 2020. 6, 7

[13] Jogendra Nath Kundu, Naveen Venkat, and R Venkatesh Babu. Universal source-free domain adaptation. *CVPR*, 2020. 2, 3, 6, 7

[14] Jogendra Nath Kundu, Naveen Venkat, Ambareesh Revanur, R Venkatesh Babu, et al. Towards inheritable models for open-set domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12376–12385, 2020. 2, 3

[15] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3

[16] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10285–10295, 2019. 6

[17] Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9641–9650, 2020. 2, 3, 6, 7

[18] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. *ICML*, 2020. 2, 3, 6, 7

[19] Mingsheng Long, Yue Cao, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Transferable representation learning with deep adaptation networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(12):3071–3085, 2018. 1

[20] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. *ICML*, 2015. 1, 2

[21] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1647–1657, 2018. 2, 6, 7

[22] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in neural information processing systems*, pages 136–144, 2016. 1

[23] Zhihe Lu, Yongxin Yang, Xiatian Zhu, Cong Liu, Yi-Zhe Song, and Tao Xiang. Stochastic classifiers for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9111–9120, 2020. 1, 3

[24] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2507–2516, 2019. 3

[25] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014. 1

[26] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009. 1

[27] 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. 7

[28] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010. 6

[29] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8050–8058, 2019. 3

[30] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Adversarial dropout regularization. *ICLR*, 2018. 3, 6

[31] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2018. 3, 6, 7

[32] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 153–168, 2018. 2, 3

[33] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *ICLR*, 2018. 2

[34] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 2

[35] Hui Tang, Ke Chen, and Kui Jia. Unsupervised domain adaptation via structurally regularized deep clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8725–8735, 2020. 6, 7

[36] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017. 1

[37] 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

[38] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017. 7

[39] Yuan Wu, Diana Inkpen, and Ahmed El-Roby. Dual mixup regularized learning for adversarial domain adaptation. *ECCV*, 2020. 6, 7

[40] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 3

[41] Guanglei Yang, Haifeng Xia, Mingli Ding, and Zhengming Ding. Bi-directional generation for unsupervised domain adaptation. In *AAAI*, pages 6615–6622, 2020. 6, 7

[42] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Universal domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2720–2729, 2019. 2, 3

[43] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *International Conference on Machine Learning*, pages 7404–7413, 2019. 6, 7

[44] Yabin Zhang, Hui Tang, Kui Jia, and Mingkui Tan. Domain-symmetric networks for adversarial domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5031–5040, 2019. 1

Supplementary Material for "Unsupervised Domain Adaptation without Source Data by Casting a BAIT"

1 Visualization on bite loss $\mathcal{L}_{bite}$

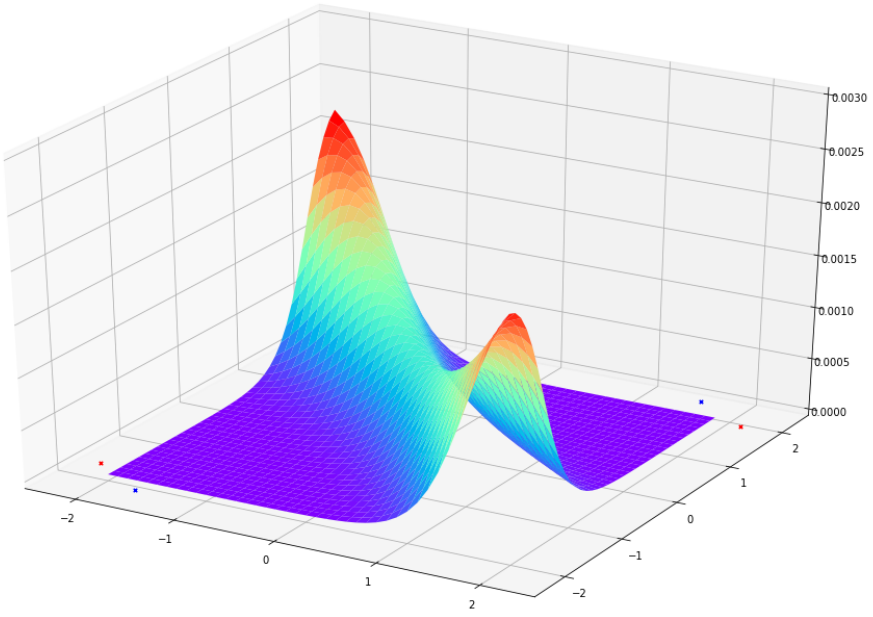


Fig. 1. Visualization on bite loss. The red and blue points are two class prototypes from two classifiers. The surface denotes the value of $\mathcal{L}_{bite}$ (higher altitude means higher values). Lower $\mathcal{L}_{bite}$ means features are clustering around the prototypes.