This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Semi-supervised Domain Adaptation via Minimax Entropy

Kuniaki Saito[1], Donghyun Kim[1], Stan Sclaroff[1], Trevor Darrell[2] and Kate Saenko[1]
[1]Boston University, [2]University of California, Berkeley
[1]{keisaito, donhk, sclaroff, saenko}@bu.edu, [2]trevor@eecs.berkeley.edu

Abstract

Contemporary domain adaptation methods are very effective at aligning feature distributions of source and target domains without any target supervision. However, we show that these techniques perform poorly when even a few labeled examples are available in the target domain. To address this semi-supervised domain adaptation (SSDA) setting, we propose a novel Minimax Entropy (MME) approach that adversarially optimizes an adaptive few-shot model. Our base model consists of a feature encoding network, followed by a classification layer that computes the features' similarity to estimated prototypes (representatives of each class). Adaptation is achieved by alternately maximizing the conditional entropy of unlabeled target data with respect to the classifier and minimizing it with respect to the feature encoder. We empirically demonstrate the superiority of our method over many baselines, including conventional feature alignment and few-shot methods, setting a new state of the art for SSDA. Our code is available at `http://cs-people.bu.edu/keisaito/research/MME.html`.

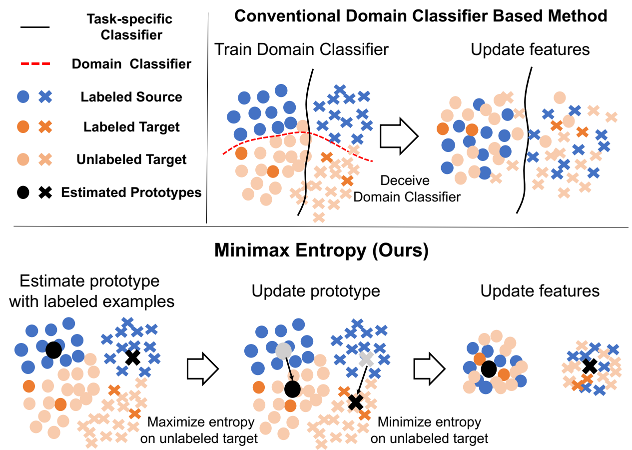


Figure 1: We address the task of semi-supervised domain adaptation. Top: Existing domain-classifier based methods align source and target distributions but can fail by generating ambiguous features near the task decision boundary. Bottom: Our method estimates a representative point of each class (prototype) and extracts discriminative features using a novel minimax entropy technique.

1. Introduction

Deep convolutional neural networks [16] have significantly improved image classification accuracy with the help of large quantities of labeled training data, but often generalize poorly to new domains. Recent unsupervised domain adaptation (UDA) methods [11, 19, 20, 28, 37] improve generalization on unlabeled target data by aligning distributions, but can fail to learn discriminative class boundaries on target domains (see Fig. 1.) We show that in the Semi-Supervised Domain Adaptation (SSDA) setting where a few target labels are available, such methods often do not improve performance relative to just training on labeled source and target examples, and can even make it worse.

We propose a novel approach for SSDA that overcomes the limitations of previous methods and significantly improves the accuracy of deep classifiers on novel domains with only a few labels per class. Our approach, which we call Minimax Entropy (MME), is based on optimizing a minimax loss on the conditional entropy of unlabeled data, as well as the task loss; this reduces the distribution gap while learning discriminative features for the task.

We exploit a cosine similarity-based classifier architecture recently proposed for few-shot learning [12, 5]. The classifier (top layer) predicts a K-way class probability vector by computing cosine similarity between K class-specific weight vectors and the output of a feature extractor (lower layers), followed by a softmax. Each class weight vector is an estimated "prototype" that can be regarded as a representative point of that class. While this approach outperformed more advanced methods in few-shot learning and we confirmed its effectiveness in our setting, as we show below it is still quite limited. In particular, it does not leverage unlabeled data in the target domain.

Our key idea is to minimize the distance between the class prototypes and neighboring unlabeled target samples, thereby extracting discriminative features. The problem is how to estimate domain-invariant prototypes without many

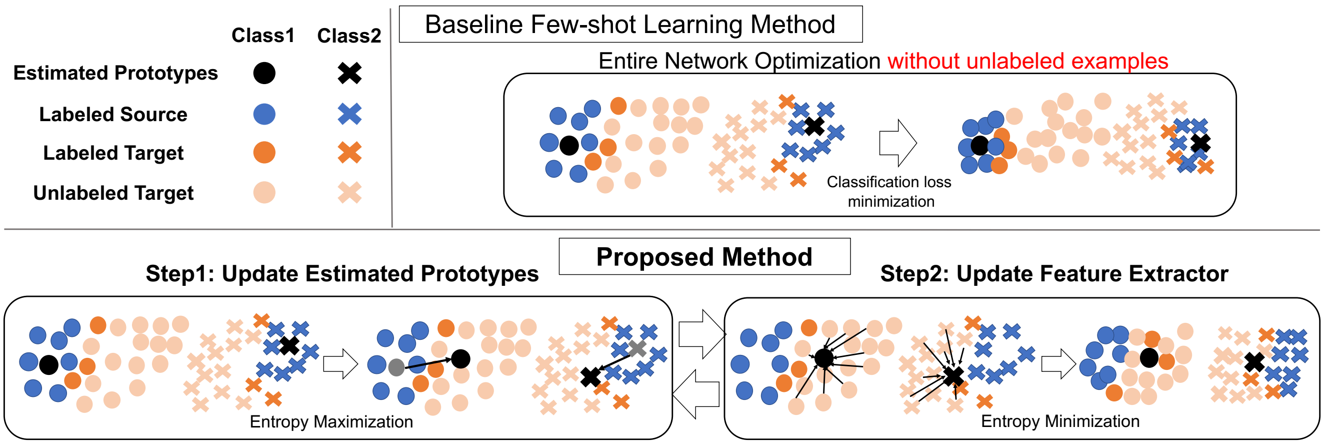


Figure 2: Top: baseline few-shot learning method, which estimates class prototypes by weight vectors, yet does not consider unlabeled data. Bottom: our model extracts discriminative and domain-invariant features using unlabeled data through a domain-invariant prototype estimation. Step 1: we update the estimated prototypes in the classifier to maximize the entropy on the unlabeled target domain. Step 2: we minimize the entropy with respect to the feature extractor to cluster features around the estimated prototype.

labeled target examples. The prototypes are dominated by the source domain, as shown in the leftmost side of Fig. 2 (bottom), as the vast majority of labeled examples come from the source. To estimate domain-invariant prototypes, we move weight vectors toward the target feature distribution. Entropy on target examples represents the similarity between the estimated prototypes and target features. A uniform output distribution with high entropy indicates that the examples are similar to all prototype weight vectors. Therefore, we move the weight vectors towards target by maximizing the entropy of unlabeled target examples in the first adversarial step. Second, we update the feature extractor to minimize the entropy of the unlabeled examples, to make them better clustered around the prototypes. This process is formulated as a mini-max game between the weight vectors and the feature extractor and applied over the unlabeled target examples.

Our method offers a new state-of-the-art in performance on SSDA; as reported below, we reduce the error relative to baseline few-shot methods which ignore unlabeled data by 8.5%, relative to current best-performing alignment methods by 8.8%, and relative to a simple model jointly trained on source and target by 11.3% in one adaptation scenario. Our contributions are summarized as follows:

- We highlight the limitations of state-of-the-art domain adaptation methods in the SSDA setting;
- We propose a novel adversarial method, Minimax Entropy (MME), designed for the SSDA task;
- We show our method's superiority to existing methods on benchmark datasets for domain adaptation.

2. Related Work

Domain Adaptation. Semi-supervised domain adaptation (SSDA) is a very important task [8, 40, 1], however it has not been fully explored, especially with regard to deep learning based methods. We revisit this task and compare our approach to recent semi-supervised learning or unsupervised domain adaptation methods. The main challenge in domain adaptation (DA) is the gap in feature distributions between domains, which degrades the source classifier's performance. Most recent work has focused on unsupervised domain adaptation (UDA) and, in particular, feature distribution alignment. The basic approach measures the distance between feature distributions in source and target, then trains a model to minimize this distance. Many UDA methods utilize a domain classifier to measure the distance [11, 37, 19, 20, 33]. The domain-classifier is trained to discriminate whether input features come from the source or target, whereas the feature extractor is trained to deceive the domain classifier to match feature distributions. UDA has been applied to various applications such as image classification [27], semantic segmentation [32], and object detection [6, 29]. Some methods minimize task-specific decision boundaries' disagreement on target examples [30, 28] to push target features far from decision boundaries. In this respect, they increase between-class variance of target features; on the other hand, we propose to make target features well-clustered around estimated prototypes. Our MME approach can reduce within-class variance as well as increasing between-class variance, which results in more discriminative features. Interestingly, we empirically observe that UDA methods [11, 20, 28] often fail in improving accuracy in SSDA.

Semi-supervised learning (SSL). Generative [7, 31],

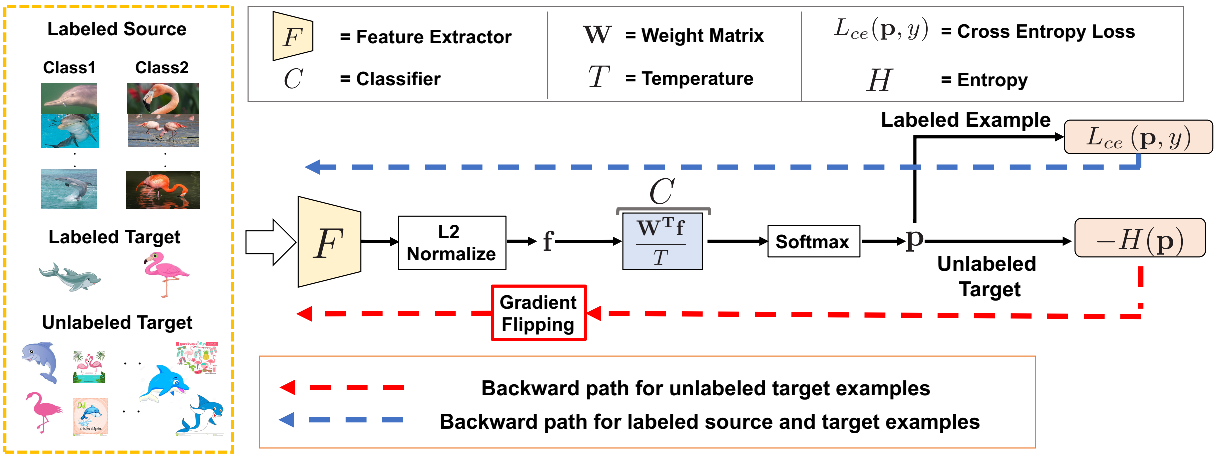


Figure 3: An overview of the model architecture and MME. The inputs to the network are labeled source examples (y=label), a few labeled target examples, and unlabeled target examples. Our model consists of the feature extractor F and the classifier C which has weight vectors ($\mathbf{W}$) and temperature T. $\mathbf{W}$ is trained to maximize entropy on unlabeled target (Step 1 in Fig. 2) whereas F is trained to minimize it (Step 2 in Fig. 2). To achieve the adversarial learning, the sign of gradients for entropy loss on unlabeled target examples is flipped by a gradient reversal layer [11, 37].

model-ensemble [17], and adversarial approaches [22] have boosted performance in semi-supervised learning, but do not address domain shift. Conditional entropy minimization (CEM) is a widely used method in SSL [13, 10]. However, we found that CEM fails to improve performance when there is a large domain gap between the source and target domains (see experimental section.) MME can be regarded as a variant of entropy minimization which overcomes the limitation of CEM in domain adaptation.

Few-shot learning (FSL). Few shot learning [35, 39, 26] aims to learn novel classes given a few labeled examples and labeled "base" classes. SSDA and FSL make different assumptions: FSL does not use unlabeled examples and aims to acquire knowledge of *novel* classes, while SSDA aims to adapt to the *same* classes in a new domain. However both tasks aim to extract discriminative features given a few labeled examples from a novel domain or novel classes. We employ a network architecture with ℓ_2 normalization on features before the last linear layer and a temperature parameter T, which was proposed for face verification [25] and applied to few-shot learning [12, 5]. Generally, classification of a feature vector with a large norm results in confident output. To make the output more confident, networks can try to increase the norm of features. However, this does not necessarily increase the between-class variance because increasing the norm does not change the direction of vectors. ℓ_2 normalization on feature vectors can solve this issue. To make the output more confident, the network focuses on making the direction of the features from the same class closer to each other and separating different classes. This simple architecture was shown to be very effective for few-shot learning [5] and we build our method on it in our work.

3. Minimax Entropy Domain Adaptation

In semi-supervised domain adaptation, we are given source images and the corresponding labels in the source domain $\mathcal{D}_s = \{(\mathbf{x}_i^s, {y_i}^s)\}_{i=1}^{m_s}$. In the target domain, we are also given a limited number of labeled target images $\mathcal{D}_t = \{(\mathbf{x}_i^t, {y_i}^t)\}_{i=1}^{m_t}$, as well as unlabeled target images $\mathcal{D}_u = \{(\mathbf{x}_i^u)\}_{i=1}^{m_u}$. Our goal is to train the model on $\mathcal{D}_s, \mathcal{D}_t$, and $\mathcal{D}_u$ and evaluate on $\mathcal{D}_u$.

3.1. Similarity based Network Architecture

Inspired by [5], our base model consists of a feature extractor F and a classifier C. For the feature extractor F, we employ a deep convolutional neural network and perform ℓ_2 normalization on the output of the network. Then, the normalized feature vector is used as an input to C which consists of weight vectors $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K]$ where K represents the number of classes and a temperature parameter T. C takes $\frac{F(\mathbf{x})}{\|F(\mathbf{x})\|}$ as an input and outputs $\frac{1}{T}\frac{\mathbf{W}^T F(\mathbf{x})}{\|F(\mathbf{x})\|}$. The output of C is fed into a softmax-layer to obtain the probabilistic output $\mathbf{p} \in R^n$. We denote $\mathbf{p}(\mathbf{x}) = \sigma(\frac{1}{T}\frac{\mathbf{W}^T F(\mathbf{x})}{\|F(\mathbf{x})\|})$, where σ indicates a softmax function. In order to classify examples correctly, the direction of a weight vector has to be representative to the normalized features of the corresponding class. In this respect, the weight vectors can be regarded as estimated prototypes for each class. An architecture of our method is shown in Fig. 3.

3.2. Training Objectives

We estimate domain-invariant prototypes by performing entropy maximization with respect to the estimated prototype. Then, we extract discriminative features by performing entropy minimization with respect to feature extractor. Entropy maximization prevents overfitting that can reduce the expressive power of the representations. Therefore, entropy maximization can be considered as the step of selecting prototypes that will not cause overfitting to the source examples. In our method, the prototypes are parameterized by the weight vectors of the last linear layer. First, we train F and C to classify labeled source and target examples correctly and utilize an entropy minimization objective to extract discriminative features for the target domain. We use a standard cross-entropy loss to train F and C for classification:

$$\mathcal{L} = \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}_s,\mathcal{D}_t}\mathcal{L}_{ce}\left(\mathbf{p}(\mathbf{x}), y\right). \tag{1}$$

With this classification loss, we ensure that the feature extractor generates discriminative features with respect to the source and a few target labeled examples. However, the model is trained on the source domain and a small fraction of target examples for classification. This does not learn discriminative features for the entire target domain. Therefore, we propose minimax entropy training using unlabeled target examples.

A conceptual overview of our proposed adversarial learning is illustrated in Fig. 2. We assume that there exists a single domain-invariant prototype for each class, which can be a representative point for both domains. The estimated prototype will be near source distributions because source labels are dominant. Then, we propose to estimate the position of the prototype by moving each $\mathbf{w_i}$ toward target features using unlabeled data in the target domain. To achieve this, we increase the entropy measured by the similarity between $\mathbf{W}$ and unlabeled target features. Entropy is calculated as follows,

$$H = -\mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}_u}\sum_{i=1}^{K} p(y=i|\mathbf{x})\log p(y=i|\mathbf{x}) \tag{2}$$

where K is the number of classes and $p(y=i|\mathbf{x})$ represents the probability of prediction to class i, namely i th dimension of $\mathbf{p}(\mathbf{x}) = \sigma(\frac{1}{T}\frac{\mathbf{W}^{\mathrm{T}}F(\mathbf{x})}{\|F(\mathbf{x})\|})$. To have higher entropy, that is, to have uniform output probability, each $\mathbf{w_i}$ should be similar to all target features. Thus, increasing the entropy encourages the model to estimate the domain-invariant prototypes as shown in Fig. 2.

To obtain discriminative features on unlabeled target examples, we need to cluster unlabeled target features around the estimated prototypes. We propose to decrease the entropy on unlabeled target examples by the feature extractor F. The features should be assigned to one of the prototypes to decrease the entropy, resulting in the desired discriminative features. Repeating this prototype estimation (entropy maximization) and entropy minimization process yields discriminative features.

To summarize, our method can be formulated as adversarial learning between C and F. The task classifier C is trained to maximize the entropy, whereas the feature extractor F is trained to minimize it. Both C and F are also trained to classify labeled examples correctly. The overall adversarial learning objective functions are:

$$\begin{aligned}\hat{\theta}_F &= \operatorname*{argmin}_{\theta_F} \mathcal{L} + \lambda H \\ \hat{\theta}_C &= \operatorname*{argmin}_{\theta_C} \mathcal{L} - \lambda H\end{aligned} \tag{3}$$

where λ is a hyper-parameter to control a trade-off between minimax entropy training and classification on labeled examples. Our method can be formulated as the iterative minimax training. To simplify training process, we use a gradient reversal layer [11] to flip the gradient between C and F with respect to H. With this layer, we can perform the minimax training with one forward and back-propagation, which is illustrated in Fig. 3.

3.3. Theoretical Insights

As shown in [2], we can measure domain-divergence by using a domain classifier. Let $h \in \mathcal{H}$ be a hypothesis, $\epsilon_s(h)$ and $\epsilon_t(h)$ be the expected risk of source and target respectively, then $\epsilon_t(h) \leqslant \epsilon_s(h) + d_{\mathcal{H}}(p,q) + C_0$ where C_0 is a constant for the complexity of hypothesis space and the risk of an ideal hypothesis for both domains and $d_{\mathcal{H}}(p,q)$ is the $\mathcal{H}$-divergence between p and q.

$$d_{\mathcal{H}}(p,q) \triangleq 2\sup_{h\in\mathcal{H}}\left|\Pr_{\mathbf{x}^s\sim p}\left[h(\mathbf{f}^s)=1\right] - \Pr_{\mathbf{x}^t\sim q}\left[h(\mathbf{f}^t)=1\right]\right|. \tag{4}$$

where $\mathbf{f}^s$ and $\mathbf{f}^t$ denote the features in the source and target domain respectively. In our case the features are outputs of the feature extractor. The $\mathcal{H}$-divergence relies on the capacity of the hypothesis space $\mathcal{H}$ to distinguish distributions p and q. This theory states that the divergence between domains can be measured by training a domain classifier and features with low divergence are the key to having a well-performing task-specific classifier. Inspired by this, many methods [11, 3, 37, 36] train a domain classifier to discriminate different domains while also optimizing the feature extractor to minimize the divergence.

Our proposed method is also connected to Eq. 4. Although we do not have a domain classifier or a domain classification loss, our method can be considered as minimizing domain-divergence through minimax training on unlabeled target examples. We choose h to be a classifier that decides a binary domain label of a feature by the value of the entropy, namely,

$$h(\mathbf{f}) = \begin{cases} 1, & \text{if } H(C(\mathbf{f})) \geq \gamma, \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where C denotes our classifier, H denotes entropy, and γ is a threshold to determine a domain label. Here, we assume C outputs the probability of the class prediction for simplicity. Eq. 4 can be rewritten as follows,

$$
\begin{aligned}
d_{\mathcal{H}}(p, q) &\triangleq 2 \sup_{h \in \mathcal{H}} \left| \Pr_{\mathbf{f}^s \sim p} [h(\mathbf{f}^s) = 1] - \Pr_{\mathbf{f}^t \sim q} \left[h(\mathbf{f}^t) = 1\right] \right| \\
&= 2 \sup_{C \in \mathcal{C}} \left| \Pr_{\mathbf{f}^s \sim p} [H(C(\mathbf{f}^s)) \geq \gamma] - \Pr_{\mathbf{f}^t \sim q} [H(C(\mathbf{f}^t)) \geq \gamma] \right| \\
&\leq 2 \sup_{C \in \mathcal{C}} \Pr_{\mathbf{f}^t \sim q} \left[H(C(\mathbf{f}^t)) \geq \gamma\right].
\end{aligned}
$$

In the last inequality, we assume that $\Pr_{\mathbf{f}^s \sim p} [H(C(\mathbf{f}^s)) \geq \gamma] \leq \Pr_{\mathbf{f}^t \sim p} \left[H(C(\mathbf{f}^t)) \geq \gamma\right]$. This assumption should be realistic because we have access to many labeled source examples and train entire networks to minimize the classification loss. Minimizing the cross-entropy loss (Eq. 1) on source examples ensures that the entropy on a source example is very small. Intuitively, this inequality states that the divergence can be bounded by the ratio of target examples having entropy greater than γ. Therefore, we can have the upper bound by finding the C that achieves maximum entropy for all target features. Our objective is finding features that achieve lowest divergence. We suppose there exists a C that achieves the maximum in the inequality above, then the objective can be rewritten as,

$$\min_{\mathbf{f}^t} \max_{C \in \mathcal{C}} \Pr_{\mathbf{f}^t \sim q} \left[H(C(\mathbf{f}^t)) \geq \gamma\right] \tag{6}$$

Finding the minimum with respect to $\mathbf{f}^t$ is equivalent to find a feature extractor F that achieves that minimum. Thus, we derive the minimax objective of our proposed learning method in Eq . 3. To sum up, our maximum entropy process can be regarded as measuring the divergence between domains, whereas our entropy minimization process can be regarded as minimizing the divergence. In our experimental section, we observe that our method actually reduces domain-divergence (Fig. 6c). In addition, target features produced by our method look aligned with source features and are just as discriminative. These come from the effect of the domain-divergence minimization.

4. Experiments

4.1. Setup

We randomly selected one or three labeled examples per class as the labeled training target examples (one-shot and three-shot setting, respectively.) We selected three other labeled examples as the validation set for the target domain. The validation examples are used for early stopping, choosing the hyper-parameter λ, and training scheduling. The other target examples are used for training without labels, their labels are only used to evaluate classification accuracy (%). All examples of the source are used for training.

Datasets. Most of our experiments are done on a subset of **DomainNet** [24], a recent benchmark dataset for large-scale domain adaptation that has many classes (345) and six domains. As labels of some domains and classes are very noisy, we pick 4 domains (Real, Clipart, Painting, Sketch) and 126 classes. We focus on the adaptation scenarios where the target domain is not real images, and construct 7 scenarios from the four domains. See our supplemental material for more details. **Office-Home** [38] contains 4 domains (Real, Clipart, Art, Product) with 65 classes. This dataset is one of the benchmark datasets for unsupervised domain adaptation. We evaluated our method on 12 scenarios in total. **Office** [27] contains 3 domains (Amazon, Webcam, DSLR) with 31 classes. Webcam and DSLR are small domains and some classes do not have a lot of examples while Amazon has many examples. To evaluate on the domain with enough examples, we have 2 scenarios where we set Amazon as the target domain and DSLR and Webcam as the source domain.

Implementation Details. All experiments are implemented in Pytorch [23]. We employ AlexNet [16] and VGG16 [34] pre-trained on ImageNet. To investigate the effect of deeper architectures, we use ResNet34 [14] in experiments on DomainNet. We remove the last linear layer of these networks to build F, and add a K-way linear classification layer C with a randomly initialized weight matrix W. The value of temperature T is set 0.05 following the results of [25] in all settings. Every iteration, we prepared two mini-batches, one consisting of labeled examples and the other of unlabeled target examples. Half of the labeled examples comes from source and half from labeled target. Using the two mini-batches, we calculated the objective in Eq. 3. To implement the adversarial learning in Eq. 3, we use a gradient reversal layer [11, 37] to flip the gradient with respect to entropy loss. The sign of the gradient is flipped between C and F during backpropagation. We adopt SGD with momentum of 0.9. In all experiments, we set the trade-off parameter λ in Eq. 3 as 0.1. This is decided by the validation performance on Real to Clipart experiments. We show the performance sensitivity to this parameter in our supplemental material, as well as more details including learning rate scheduling.

Baselines. **S+T** [5, 25] is a model trained with the labeled source and labeled target examples without using unlabeled target examples. **DANN [11]** employs a domain classifier to match feature distributions. This is one of the most popular methods in UDA. For fair comparison, we modify this method so that it is trained with the labeled source, labeled target, and unlabeled target examples. **ADR [28]** utilizes a task-specific decision boundary to align features and ensure that they are discriminative on the target. **CDAN [20]** is one of the state-of-the art methods on UDA and performs domain alignment on features that are conditioned on the output of classifiers. In addition, it utilizes entropy minimization on target examples. CDAN integrates domain-classifier based alignment and entropy minimization. Comparison with these UDA methods (DANN, ADR, CDAN) reveals how much gain will be obtained compared to the

Net	Method	R to C		R to P		P to C		C to S		S to P		R to S		P to R		MEAN	
		1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot
AlexNet	S+T	43.3	47.1	42.4	45.0	40.1	44.9	33.6	36.4	35.7	38.4	29.1	33.3	55.8	58.7	40.0	43.4
	DANN	43.3	46.1	41.6	43.8	39.1	41.0	35.9	36.5	36.9	38.9	32.5	33.4	53.6	57.3	40.4	42.4
	ADR	43.1	46.2	41.4	44.4	39.3	43.6	32.8	36.4	33.1	38.9	29.1	32.4	55.9	57.3	39.2	42.7
	CDAN	46.3	46.8	45.7	45.0	38.3	42.3	27.5	29.5	30.2	33.7	28.8	31.3	56.7	58.7	39.1	41.0
	ENT	37.0	45.5	35.6	42.6	26.8	40.4	18.9	31.1	15.1	29.6	18.0	29.6	52.2	60.0	29.1	39.8
	MME	**48.9**	**55.6**	**48.0**	**49.0**	**46.7**	**51.7**	**36.3**	**39.4**	**39.4**	**43.0**	**33.3**	**37.9**	**56.8**	**60.7**	**44.2**	**48.2**
VGG	S+T	49.0	52.3	55.4	56.7	47.7	51.0	43.9	48.5	50.8	55.1	37.9	45.0	69.0	71.7	50.5	54.3
	DANN	43.9	56.8	42.0	57.5	37.3	49.2	46.7	48.2	51.9	55.6	30.2	45.6	65.8	70.1	45.4	54.7
	ADR	48.3	50.2	54.6	56.1	47.3	51.5	44.0	49.0	50.7	53.5	38.6	44.7	67.6	70.9	50.2	53.7
	CDAN	57.8	58.1	57.8	59.1	51.0	57.4	42.5	47.2	51.2	54.5	42.6	49.3	71.7	74.6	53.5	57.2
	ENT	39.6	50.3	43.9	54.6	26.4	47.4	27.0	41.9	29.1	51.0	19.3	39.7	68.2	72.5	36.2	51.1
	MME	**60.6**	**64.1**	**63.3**	**63.5**	**57.0**	**60.7**	**50.9**	**55.4**	**60.5**	**60.9**	**50.2**	**54.8**	**72.2**	**75.3**	**59.2**	**62.1**
ResNet	S+T	55.6	60.0	60.6	62.2	56.8	59.4	50.8	55.0	56.0	59.5	46.3	50.1	71.8	73.9	56.9	60.0
	DANN	58.2	59.8	61.4	62.8	56.3	59.6	52.8	55.4	57.4	59.9	52.2	54.9	70.3	72.2	58.4	60.7
	ADR	57.1	60.7	61.3	61.9	57.0	60.7	51.0	54.4	56.0	59.9	49.0	51.1	72.0	74.2	57.6	60.4
	CDAN	65.0	69.0	64.9	67.3	63.7	68.4	53.1	57.8	63.4	65.3	54.5	59.0	73.2	78.5	62.5	66.5
	ENT	65.2	71.0	65.9	69.2	65.4	71.1	54.6	60.0	59.7	62.1	52.1	61.1	75.0	**78.6**	62.6	67.6
	MME	**70.0**	**72.2**	**67.7**	**69.7**	**69.0**	**71.7**	**56.3**	**61.8**	**64.8**	**66.8**	**61.0**	**61.9**	**76.1**	78.5	**66.4**	**68.9**

Table 1: Accuracy on the DomainNet dataset (%) for one-shot and three-shot settings on 4 domains, R: Real, C: Clipart, P: Clipart, S: Sketch. Our MME method outperformed other baselines for all adaptation scenarios and for all three networks, except for only one case where it performs similarly to ENT.

Net	Method	Office-Home		Office	
		1-shot	3-shot	1-shot	3-shot
AlexNet	S+T	44.1	50.0	50.2	61.8
	DANN	45.1	50.3	55.8	64.8
	ADR	44.5	49.5	50.6	61.3
	CDAN	41.2	46.2	49.4	60.8
	ENT	38.8	50.9	48.1	65.1
	MME	**49.2**	**55.2**	**56.5**	**67.6**
VGG	S+T	57.4	62.9	68.7	73.3
	DANN	60.0	63.9	69.8	75.0
	ADR	57.4	63.0	69.4	73.7
	CDAN	55.8	61.8	65.9	72.9
	ENT	51.6	64.8	70.6	75.3
	MME	**62.7**	**67.6**	**73.4**	**77.0**

Table 2: Results on Office-Home and Office dataset (%). The value is the accuracy averaged over all adaptation scenarios. Performance on each setting is summarized in supplementary material.

existing domain alignment-based methods. **ENT [13]** is a model trained with labeled source and target and unlabeled target using standard entropy minimization. Entropy is calculated on unlabeled target examples and the entire network is trained to minimize it. The difference from MME is that ENT does not have a maximization process, thus comparison with this baseline clarifies its importance.

Note that all methods except for CDAN are trained with exactly the same architecture used in our method. In case of CDAN, we could not find any advantage of using our architecture. The details of baseline implementations are in our supplemental material.

4.2. Results

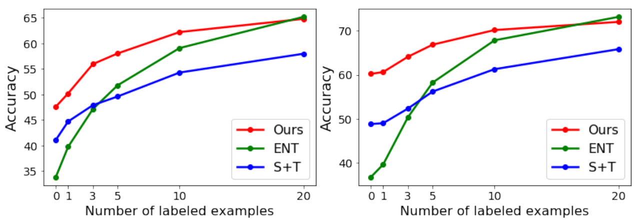
Overview. The main results on the DomainNet dataset are shown in Table 1. First, our method outperformed other baselines for all adaptation scenarios and all three networks except for one case. On average, our method outperformed S+T with 9.5% and 8.9% in ResNet one-shot and three-shot setting respectively. The results on Office-Home and Office are summarized in Table 2, where MME also outperforms all baselines. Due to the limited space, we show the results averaged on all adaptation scenarios.

Comparison with UDA Methods. Generally, baseline UDA methods need strong base networks such as VGG or ResNet to perform better than S+T. Interestingly, these methods cannot improve the performance in some cases. The superiority of MME over existing UDA methods is supported by Tables 1 and 2. Since CDAN uses entropy minimization and ENT significantly hurts the performance for AlexNet and VGG, CDAN does not consistently improve the performance for AlexNet and VGG.

Comparison with Entropy Minimization. ENT does not improve performance in some cases because it does not account for the domain gap. Comparing results on one-shot and three-shot, entropy minimization gains performance with the help of labeled examples. As we have more labeled

Method	R - C	R - P	P - C	C - S	S - P	R - S	P - R	Avg
Source	41.1	42.6	37.4	30.6	30.0	26.3	52.3	37.2
DANN	44.7	36.1	35.8	33.8	**35.9**	27.6	49.3	37.6
ADR	40.2	40.1	36.7	29.9	30.6	25.9	51.5	36.4
CDAN	44.2	39.1	37.8	26.2	24.8	24.3	**54.6**	35.9
ENT	33.8	43.0	23.0	22.9	13.9	12.0	51.2	28.5
MME	**47.6**	**44.7**	**39.9**	**34.0**	33.0	**29.0**	53.5	**40.2**

Table 3: Results on the DomainNet dataset in the unsupervised domain adaptation setting (%).



(a) AlexNet (b) VGG

Figure 4: Accuracy vs the number of labeled target examples. The ENT method needs more labeled examples to obtain similar performance to our method.

Method	R to C		R to S	
	1-shot	3-shot	1-shot	3-shot
S+T (Standard Linear)	41.4	44.3	26.5	28.7
S+T (Few-shot [5, 25])	**43.3**	**47.1**	**29.1**	**33.3**
MME (Standard Linear)	44.9	47.7	30.0	32.2
MME (Few-shot [5, 25])	**48.9**	**55.6**	**33.3**	**37.9**

Table 4: Comparison of classifier architectures on the DomainNet dataset using AlexNet, showing the effectiveness of the architecture proposed in [5, 25].

target examples, the estimation of prototypes will be more accurate without any adaptation. In case of ResNet, entropy minimization often improves accuracy. There are two potential reasons. First, ResNet pre-trained on ImageNet has a more discriminative representation than other networks. Therefore, given a few labeled target examples, the model can extract more discriminative features, which contributes to the performance gain in entropy minimization. Second, ResNet has batch-normalization (BN) layers [15]. It is reported that BN has the effect of aligning feature distributions [4, 18]. Hence, entropy minimization was done on aligned feature representations, which improved the performance. When there is a large domain gap such as C to S, S to P, and R to S in Table 1, BN is not enough to handle the domain gap. Therefore, our proposed method performs much better than entropy minimization in such cases. We show an analysis of BN in our supplemental material, revealing its effectiveness for entropy minimization.

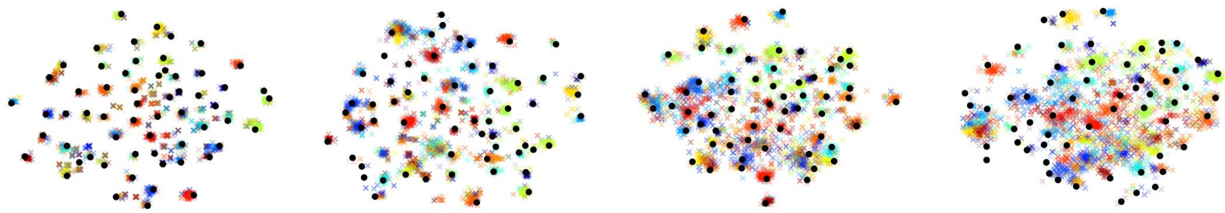
4.3. Analysis

Varying Number of Labeled Examples. First, we show the results on unsupervised domain adaptation setting in Table 3. Our method performed better than other methods on average. In addition, only our method improved performance compared to source only model in all settings. Furthermore, we observe the behavior of our method when the number of labeled examples in the target domain varies from 0 to 20 per class, which corresponds to 2520 labeled examples in total. The results are shown in Fig. 4. Our method works much better than S+T given a few labeled examples. On the other hand, ENT needs 5 labeled examples per class to improve performance. As we add more labeled examples, the performance gap between ENT and ours is reduced. This result is quite reasonable, because prototype estimation will become more accurate without any adaptation as we have more labeled target examples.

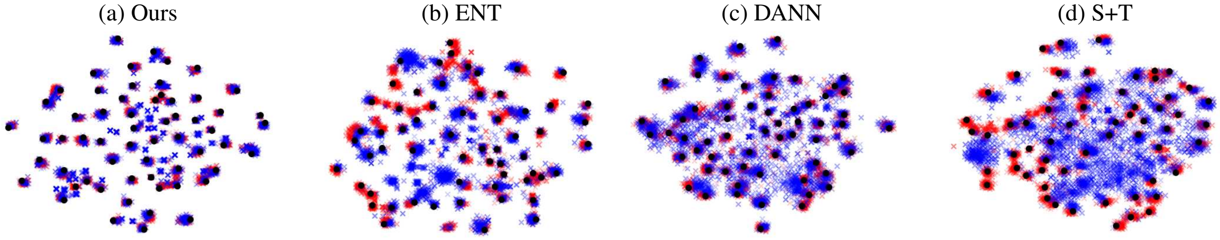
Effect of Classifier Architecture. We introduce an ablation study on the classifier network architecture proposed in [5, 25] with AlexNet on DomainNet. As shown in Fig. 3, we employ ℓ_2 normalization and temperature scaling. In this experiment, we compared it with a model having a standard linear layer without ℓ_2 normalization and temperature. The result is shown in Table 4. By using the network architecture proposed in [5, 25], we can improve the performance of both our method and the baseline S+T model (model trained only on source examples and a few labeled target examples.) Therefore, we can argue that the network architecture is an effective technique to improve performance when we are given a few labeled examples from the target domain.

Feature Visualization. In addition, we plot the learned features with t-SNE [21] in Fig. 5. We employ the scenario Real to Clipart of DomainNet using AlexNet as the pretrained backbone. Fig 5 (a-d) visualizes the target features and estimated prototypes. The color of the cross represents its class, black points are the prototypes. With our method, the target features are clustered to their prototypes and do not have a large variance within the class. We visualize features on the source domain (red cross) and target domain (blue cross) in Fig. 5 (e-h). As we discussed in the method section, our method aims to minimize domain-divergence. Indeed, target features are well-aligned with source features with our method. Judging from Fig. 5f, entropy minimization (ENT) also tries to extract discriminative features, but it fails to find domain-invariant prototypes.

Quantitative Feature Analysis. We quantitatively investigate the characteristics of the features we obtain using the same adaptation scenario. First, we perform the analysis on the eigenvalues of the covariance matrix of target features. We follow the analysis done in [9]. Eigenvectors represent the components of the features and eigenvalues represent their contributions. If the features are highly discrimina-

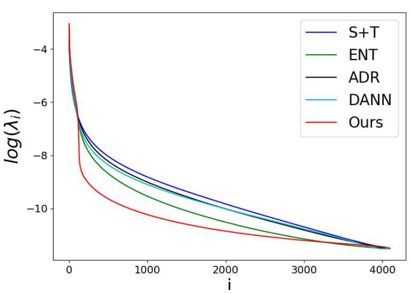


(a) Ours (b) ENT (c) DANN (d) S+T

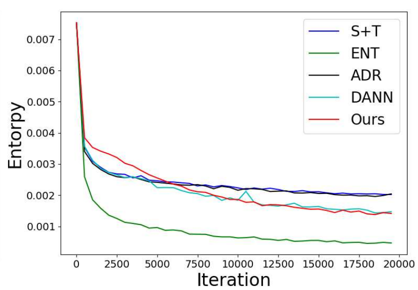


(e) Ours (f) ENT (g) DANN (h) S+T

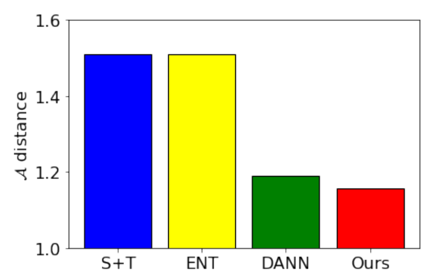
Figure 5: Feature visualization with t-SNE. (a-d) We plot the class prototypes (black circles) and features on the target domain (crosses). The color of a cross represents its class. We observed that features on our method show more discrimative features than other methods. (e-h) Red: Features of the source domain. Blue: Features of the target domain. Our method's features are well-aligned between domains compared to other methods.



(a) Eigenvalues



(b) Entropy



(c) $\mathcal{A}$-distance

Figure 6: (a) Eigenvalues of the covariance matrix of the features on the target domain. Eigenvalues reduce quickly in our method, which shows that features are more discriminative than other methods. (b) Our method achieves lower entropy than baselines except ENT. (c) Our method clearly reduces domain-divergence compared to S+T.

tive, only a few components are needed to summarize them. Therefore, in such a case, the first few eigenvalues are expected to be large, and the rest to be small. The features are clearly summarized by fewer components in our method as shown in Fig. 6a. Second, we show the change of entropy value on the target in Fig. 6b. ENT diminishes the entropy quickly, but results in poor performance. This indicates that the method increases the confidence of predictions incorrectly while our method achieves higher accuracy at the same time. Finally, in Fig. 6c, we calculated $\mathcal{A}$-distance by training a SVM as a domain classifier as proposed in [2]. Our method greatly reduces the distance compared to S+T. The claim that our method reduces a domain divergence is empirically supported with this result.

5. Conclusion

We proposed a novel Minimax Entropy (MME) approach that adversarially optimizes an adaptive few-shot model for semi-supervised domain adaptation (SSDA). Our model consists of a feature encoding network, followed by a classification layer that computes the features' similarity to a set of estimated prototypes (representatives of each class). Adaptation is achieved by alternately maximizing the conditional entropy of unlabeled target data with respect to the classifier and minimizing it with respect to the feature encoder. We empirically demonstrated the superiority of our method over many baselines, including conventional feature alignment and few-shot methods, setting a new state of the art for SSDA.

6. Acknowledgements

This work was supported by Honda, DARPA, BAIR, BDD, and NSF Award No. 1535797.

References

[1] Shuang Ao, Xiang Li, and Charles X Ling. Fast generalized distillation for semi-supervised domain adaptation. In *AAAI*, 2017.

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

[3] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *NIPS*, 2016.

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

[5] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv*, 2018.

[6] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain adaptive faster r-cnn for object detection in the wild. In *CVPR*, 2018.

[7] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *NIPS*, 2017.

[8] Jeff Donahue, Judy Hoffman, Erik Rodner, Kate Saenko, and Trevor Darrell. Semi-supervised domain adaptation with instance constraints. In *CVPR*, 2013.

[9] Abhimanyu Dubey, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Maximum-entropy fine grained classification. In *NIPS*, 2018.

[10] Ayse Erkan and Yasemin Altun. Semi-supervised learning via generalized maximum entropy. In *AISTATS*, 2010.

[11] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2014.

[12] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, 2018.

[13] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NIPS*, 2005.

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

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*, 2015.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[17] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv*, 2016.

[18] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv*, 2016.

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

[20] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NIPS*, 2018.

[21] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(11):2579–2605, 2008.

[22] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. *arXiv*, 2015.

[23] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[24] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. *ICCV*, 2019.

[25] Rajeev Ranjan, Carlos D Castillo, and Rama Chellappa. L2-constrained softmax loss for discriminative face verification. *arXiv*, 2017.

[26] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *arXiv*, 2016.

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

[28] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Adversarial dropout regularization. In *ICLR*, 2018.

[29] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Strong-weak distribution alignment for adaptive object detection. *arXiv*, 2018.

[30] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, 2018.

[31] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

[32] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. In *CVPR*, 2018.

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

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014.

[35] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.

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

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

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

[39] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NIPS*, 2016.

[40] Ting Yao, Yingwei Pan, Chong-Wah Ngo, Houqiang Li, and Tao Mei. Semi-supervised domain adaptation with subspace learning for visual recognition. In *CVPR*, 2015.