Cross-Modal Prediction Consistency based Self-Training for Unsupervised Domain Adaptation

Dongwan Kim[1] Geeho Kim[1] Seonguk Seo[1] Yumin Suh[1] Bohyung Han[1]
Taeho Lee[2] Jongwoo Han[2] Hyejeong Jeon[2]
[1]ECE & ASRI, Seoul National University, Korea [2]AI Laboratory, LG Electronics

Abstract

We propose an unsupervised domain adaptation method based on self-training. Our method collects target examples with confident pseudo labels and uses them as a new source domain. Specifically, we adopt a two-stage framework to train the network. At the first stage, we train several different models independently from each other while varying the backbone architecture and input modality. We select confident target training samples, which have the same label predictions from all models, and augment the source dataset with the selected confident samples and their corresponding pseudo labels. At the second stage, we retrain a network to adapt to the target domain using the augmented source and target domain datasets. In the experiments, our method consistently improves accuracy from the baseline for two test domains (clipart and painting) in the DomainNet dataset.

1. Introduction

Unsupervised domain adaptation aims to transfer knowledge learned from label-rich source domains to the target domain using only unlabeled examples. Recently, some approaches based on self-training [7, 9] or pseudo target labeling have shown promising results for unsupervised domain adaptation [10, 2]. They alternately iterate between training a classifier and labeling target training examples. Their common observation is that precise label prediction and its confidence estimation is crucial to achieve high accuracy by avoiding overfitting to noisy pseudo labels. In particular, a method which works for various conditions without adjusting hyper-parameters is desired.

To this end, we propose a framework for selecting confident target example which is robust against hyper-parameter selection. Our intuition is that if models trained from multiple different conditions agree on label prediction of a given target example, then the predicted label is likely to be correct. Specifically, we train multiple models independently from each other where each model is trained with input images represented in a certain modality, such as RGB and edge images. In addition, models with different backbone architectures are trained to further diversify the pool of models.

We adopt a two-stage framework to train a network. At the first stage, several different classifiers are independently trained from the source domains and adapted to the target domain. Based on the trained models, we select a confident subset of target training examples, whose predicted labels from all the models are same, and augment the source domain with their corresponding pseudo labels. At the second stage, we train a new model with the augmented source domains and the original target domains. Our experiments show that this second stage training consistently improves the accuracy over baseline models by introducing a pseudo-target classification loss for each of the two target domains (clipart and painting) from the DomainNet dataset.

2. Related Works

Self-training or noisy label learning has been widely investigated in unsupervised and semi-supervised learning. Reed *et al*. [7] adopts a bootstrapping method for noisy label learning. Sukhbaatar *et al*. [9] introduces a noise layer to match the network outputs with the noisy label distribution. Zou *et al*. [14] proposes a class-balanced self-training framework for semantic segmentation and uses spatial priors to refine pseudo labels. Yu *et al*. [11] introduces a probabilistic model to deal with edge misalignment by treating it as latent variable optimization.

Several recent domain adaptation approaches have adopted self-training; They estimate target pseudo labels and directly train a model for the target domain. They employ different strategies to reduce the adversarial effect of noisy pseudo labels. Moving Semantic Transfer Network (MSTN) [10] aligns source centroid and pseudo label target centroid for each class in feature space. Asymmetric Tri-training Network (ATN) [8] and Collaborative and Adversarial Network (CAN) [12] estimate pseudo labels based on

the consensus of multiple models. Domain-Specific Batch Normalization (DSBN) [2] reserves separate batch normalization for each domain and train the models with two classification losses—the source domain with ground-truth labels and the target domain with pseudo labels. Confidence-Regularized Self-Training (CRST) [13] treats pseudo labels as continuous latent variables and optimizes them by label and model regularization. Most of them needs to carefully set the hyper-parameters, whereas the proposed method is less sensitive to the number of models used to select pseudo labels.

3. Our Approach

For a given baseline domain adaptation method, we propose a two-stage training procedure to enhance its accuracy. At the first stage, we train several different baseline models while varying the input image representations and the backbone architectures. Based on the label predictions on target training examples from the trained models, we collect a set of target domain examples with confident labels, which is used to augment the original source domain dataset. At the second stage, We retrain a network using the baseline domain adaptation method while employing the augmented source dataset and the original target dataset. Since the confident subset of target domain dataset is treated as an additional source domain with their pseudo labels, the second stage of training improves accuracy from the baseline.

3.1. Problem Statement

Consider data from two distinctive domains, the source domain $\mathcal{S} = \{X_s, Y_s\}$ and the target domain $\mathcal{T} = \{X_t\}$. An example from the source domain $\mathbf{x}_s \in X_s$ has an associated label $y_s \in Y_s$, whereas one from the target domain $\mathbf{x}_t \in X_t$ has no paired ground-truth label. We employ a feature extractor $f(\mathbf{x}; \mathbf{m}_f)$ and a classifier $c(\cdot; \mathbf{m}_c)$, where $\mathbf{m}_f$ and $\mathbf{m}_c$ represent dropout masks that can be applied at an arbitrary layer of each sub network. The feature extractor creates a latent vector from a data point sampled from either of the two domains $\mathbf{x} \sim \mathcal{S} \cup \mathcal{T}$. The latent vector is used as an input to a classifier. As in [5], we denote the entire neural network as a composition of the feature extractor and the classifier: $h(\mathbf{x}; \mathbf{m}_f, \mathbf{m}_c) = c(f(\mathbf{x}; \mathbf{m}_f); \mathbf{m}_c)$. For given $\mathbf{x}$, the predicted label is denoted as $g(\mathbf{x}) = \arg\max c(f(\mathbf{x}))$.

3.2. Baseline Model: Drop to Adapt with Stochastic Weight Averaging

We adopt the single source domain adaptation method with Drop to Adapt (DTA) regularization loss [5] with SWA [4] as our baseline. (Please refere to [5] for more details.) Given the source dataset, $\mathcal{S}$, and the target dataset, $\mathcal{T}$, the complete DTA loss becomes

$$L(\mathcal{S}, \mathcal{T}) = L_C(\mathcal{S}) + \lambda_1 L_{DTA}(\mathcal{S}) + \lambda_2 L_{DTA}(\mathcal{T}) + \lambda_3 L_E(\mathcal{T}), \tag{1}$$

where L_C denotes the cross-entropy loss for classification, L_E denotes the entropy minimization term, and $\lambda_1, \lambda_2, \lambda_3$ are weights for each loss term. The DTA loss, L_{DTA}, can be decomposed into two parts

$$L_{DTA}(\mathcal{X}) = L_{fDTA}(\mathcal{X}) + L_{cDTA}(\mathcal{X}), \tag{2}$$

since it is applied to both the feature extractor and the classifier networks. This loss term minimizes the divergence between two predictions of a single input x: one with a random dropout mask m^s and another with an adversarial dropout mask m^{adv}. The adversarial dropout mask is specifically generated to maximize divergence between the two predictions, and the network aims to minimize this divergence. In general, L_{DTA} can be expressed as:

$$L_{DTA}(\mathcal{X}) = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \left[D[h(\mathbf{x}; m^s), h(\mathbf{x}; m^{adv})] \right], \tag{3}$$

where $h(x; m)$ expresses the model with input x and dropout mask m, and D is a measure of divergence. Specifically, we employ the Kullback-Leibler (KL) divergence in our implementation. Equation 3 becomes L_{fDTA} by using adversarial dropout on the feature extractor, while it becomes L_{cDTA} when adversarial dropout is used on the classifier,

The entropy loss L_E penalizes target samples for being too close to the decision boundary:

$$L_E(\mathcal{T}) = -\mathbb{E}_{\mathbf{x} \sim \mathcal{T}} \left[h(\mathbf{x})^T \log h(\mathbf{x}) \right]. \tag{4}$$

Finally, we employ Stochastic Weight Averaging (SWA) [4] during optimization. SWA has shown to significantly improve the generalization of neural networks. This is also true for consistency-based methods [1], such as DTA. By averaging the weights over multiple epochs of the training, SWA helps our baseline model achieve consistently high single-model performances on both validation and test datasets.

3.3. A Two-Stage Training Procedure

At the first stage, we train a baseline network to minimize the following loss:

$$L(\mathcal{S}, \mathcal{T}) = L_C(\mathcal{S}) + L_{DTA}(\mathcal{S}) + L_{DTA}(\mathcal{T}) + L_E(\mathcal{T}). \tag{5}$$

We train K baseline models independently from each other, for different combinations of image representations (RGB and edge) and backbone architectures (Res-101 and SENet-154). We augment source datasets with the target training samples of which K different baseline models make identical label predictions, using the common label predictions

Table 1. Accuracy using SENet-154 as a backbone architecture (%) : with (w/o) test time augmentation

		Clipart	Painting	Avg.
Single Model	Image	74.14 (74.11)	62.43 (61.38)	67.03 (66.38)
	Pseudo	75.60	64.55	68.89
Ensemble	Image + Edge	(74.56)	(62.11)	(67.00)
	Image + Edge + Pseudo	76.37	64.87	69.39
	Image1 + Image2 + Edge1 + Edge2 + Pseudo	76.32	65.31	69.64

Table 2. Accuracy of different backbone architectures (%)

Sketch	Clipart	Painting	Avg.
ResNet-50	70.12	55.32	61.13
ResNet-101	71.78	56.53	62.52
SEResNeXt-101	68.17	57.67	61.80
SENet-154	74.11	61.38	66.38

Table 3. Accuracy comparison with existing method (%) on ResNet-101

	Clipart	Painting	Avg.
M^{3}SDA-β [6]	58.60	52.30	55.45
Ours	**71.78**	**56.53**	**62.52**

as pseudo labels. Formally, a pseudo labeled domain $\tilde{\mathcal{T}}$ and the augmented source datasets can be expressed as:

$$\tilde{\mathcal{T}} = \{(\mathbf{x}, g(\mathbf{x}))|\mathbf{x} \in \mathcal{T}, g(\mathbf{x}) = g_1(\mathbf{x}) = \cdots g_K(\mathbf{x})\} \quad (6)$$

$$\tilde{\mathcal{S}} = \mathcal{S} \cup \tilde{\mathcal{T}} \quad (7)$$

At the second stage, we train a network to minimize the following loss with the augmented source datasets.

$$L(\tilde{\mathcal{S}}, \mathcal{T}) = L_C(\tilde{\mathcal{S}}) + L_{DTA}(\tilde{\mathcal{S}}) + L_{DTA}(\mathcal{T}) + L_E(\mathcal{T}) \quad (8)$$

4. Experiments

4.1. DomainNet Dataset

DomainNet dataset [6] consists of 0.6 million images with 345 categories across six domains: *Clipart, Infograph, Painting, Quickdraw, Real, Sketch*. In the multi-source domain adaptation challenge, we utilize *Infograph, Quickdraw, Real, Sketch* as source domains and *Clipart, Painting* as target domains each.

4.2. Implementation Details

The proposed model consists of a feature extractor and classification network. For the feature extractor, we use modified version of Fully Convolutional Networks (FCN) on SENet-154 [3] as a backbone architecture. For the classifier, we utilize a nonlinear classifier which consists of three fully connected layers with 1024 channels for each with ReLU activations. More specifically, we apply the channel-wise adversarial dropout (CAdD) and element-wise adversarial dropout (EAdD) proposed in [5], which are inserted in the last convolution layer of SENet-154 models and the second fully connected layer, respectively.

Images are resized to 224×224 for model input. We augment the source training data with random resized crop and random horizontal flip, and the target training data with resize and random horizontal flip. For better prediction, we apply test-time augmentation by ensembling the output logits of four different augmentations of the same image: 1) resize, 2) resize + horizontal flip, 3) random resized crop, and 4) random resized crop + horizontal flip. The test time augmentation allows us to extract more accurate and reliable logits.

We train our model with Stochastic Gradient Descent (SGD), on a minibatch size of 64 images. The momentum is set to 0.9, and the learning rate, which is halved after 3 epochs, is initialized as 4×10^{-3}. As mentioned earlier, we employ SWA in our optimization process. During training, SWA is identical to SGD, but it stores and additional set of the model weights as the running mean over multiple epochs. We select SWA to start 1 epoch after the learning rate decay (3rd epoch), and update the running mean after every epoch until our training ends (10 epochs). At the end of training, the model weights are substituted with the running means, and one last iteration over the training set is performed to stabilize the batch normalization layers. Note that in this last step, we do not update any weights, but only pass the images through the network to update the running statistics in batch normalization. From Eq. 1, we set $\lambda_1, \lambda_2, \lambda_3$ as $2, 4, 0.02$, respectively. For all other hyperparameters specific to the DTA loss - such as δ_c, δ_e, and ramp-up factor - we use the same hyperparameters as Lee *et al*. [5] in the VisDA-2017 classification experiments. Note that, to tune hyperparameters, we used the validation domain (sketch) as the target dataset. All hyperparameters used for both test domains are identical to the best hyperparameters we found on the sketch domain.

For the second stage training, we mostly use the same hyperparameters as our baseline training; however, we change the SWA frequency to two times per epoch (once every half-epoch) and only train for 7 epochs in total. Our final model is an ensemble of 5 independently trained mod-

els: 1,2) two DTA models with two random seeds, 3,4) two DTA models using edge images with two random seeds, and 5) a model trained using our second stage procedure.

4.3. Evaluation on DomainNet Dataset

In Table 1 we present our model's performances on the test domains, as displayed on CodaLab. Note that our pseudo (second stage) model outperforms the all other single model performances by a large margin. Finally, multiple single models can be ensembled, resulting in a much stronger model. Table. 2 shows the accuracy for different backbone architectures using the baseline training method. Again, these individual models were submitted to CodaLab to measure performance. By examining the accuracy of each backbone model, we chose SENet-154 as our main backbone architecture. Finally, we compare our method with existing domain adaptation method in Table. 3, which shows that our method largely outperforms the existing method in both cases when clipart or painting is a target domain.

5. Conclusion

In this abstract, we presented our domain adaptation method based on self-training. By using this method, our team was able to place 3rd in the Visual Domain Adaptation Challenge (VisDA-2019).

References

[1] Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. There are many consistent explanations of unlabeled data: Why you should average. In *ICLR*, 2019.

[2] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *CVPR*, 2019.

[3] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[4] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[5] Seungmin Lee, Kim Dongwan, Namil Kim, and Jeong Seong-Gyun. Drop to Adapt: Learning Discriminative Features for Unsupervised Domain Adaptation. In *ICCV*, 2019.

[6] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. *arXiv preprint arXiv:1812.01754*, 2018.

[7] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *ICLR*, 2015.

[8] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *ICML*, 2017.

[9] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

[10] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *ICML*, 2018.

[11] Zhiding Yu, Weiyang Liu, Yang Zou, Chen Feng, Srikumar Ramalingam, BVK Vijaya Kumar, and Jan Kautz. Simultaneous edge alignment and learning. In *ECCV*, 2018.

[12] Weichen Zhang, Wanli Ouyang, Wen Li, and Dong Xu. Collaborative and adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[13] Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. Confidence regularized self-training. *ICCV*, 2019.

[14] Yang Zou, Zhiding Yu, B. V. K. Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018.