Efficient Pre-trained Features and Recurrent Pseudo-Labeling in Unsupervised Domain Adaptation

Youshan Zhang Brian D. Davison
Computer Science and Engineering, Lehigh University, Bethlehem, PA, USA
{yoz217, bdd3}@lehigh.edu

Abstract

Domain adaptation (DA) mitigates the domain shift problem when transferring knowledge from one annotated domain to another similar but different unlabeled domain. However, existing models often utilize one of the ImageNet models as the backbone without exploring others, and fine-tuning or retraining the backbone ImageNet model is also time-consuming. Moreover, pseudo-labeling has been used to improve the performance in the target domain, while how to generate confident pseudo labels and explicitly align domain distributions has not been well addressed. In this paper, we show how to efficiently opt for the best pre-trained features from seventeen well-known ImageNet models in unsupervised DA problems. In addition, we propose a recurrent pseudo-labeling model using the best pre-trained features (termed PRPL) to improve classification performance. To show the effectiveness of PRPL, we evaluate it on three benchmark datasets, Office+Caltech-10, Office-31, and Office-Home. Extensive experiments show that our model reduces computation time and boosts the mean accuracy to 98.1%, 92.4%, and 81.2%, respectively, substantially outperforming the state of the art.

1. Introduction

With the explosive growth of information in the current era, there are massive amounts of data from multiple sources and corresponding to varied scenarios. However, not all tasks have enough annotated data for training, and collecting sufficient labeled data is a big investment of time and effort. Therefore, in order to build machine learning models it is often necessary to transfer knowledge from one labeled domain to an unlabeled domain. Due to dataset bias or domain shift [17], the generalization ability of the learned model on the unlabeled domain has been severely compromised. Domain adaptation (DA) is proposed to circumvent the domain shift problem.

Unsupervised domain adaptation (UDA) transfers knowledge learned from a label-rich source domain to a fully unlabeled target domain [16]. Most prior methods focus on matching (marginal, conditional, and joint) distributions between two domains to learn domain-invariant representations. Maximum Mean Discrepancy (MMD) is one of the most popular distance metrics when minimizing differences between two distributions. Long et al. [14] proposed a Deep Adaptation Network (DAN) that considered multiple kernels of MMD functions. Recently, Kang et al. [11] extended MMD to the contrastive domain discrepancy loss. However, these distance-based metrics can also mix samples of different classes together. Recently, adversarial learning has shown its power in learning domain invariant representations. The domain discriminator aims to distinguish the source domain from the target domain, while the feature extractor aims to learn domain-invariant representations to fool the domain discriminator [11]. Sometimes, pseudo-labeling is proposed to learn the target discriminative representations [31, 32]. However, the credibility of these pseudo labels is unknown.

To address the above challenges, this paper provides two specific contributions:

1. To reduce computation time, we extract features from seventeen pre-trained ImageNet models and then design a fast and efficient unsupervised metric to select the best pre-trained features for the domain transfer tasks.

2. We develop a recurrent pseudo-labeling paradigm to continuously select high confidence transfer examples from the target domain and minimize the marginal and conditional discrepancies between the two domains.

We conduct extensive experiments on three benchmark datasets (Office + Caltech-10, Office-31, and Office-Home), achieving higher accuracy than state-of-the-art methods.

2. Related work

Pre-training. Pre-training is one of the dominant components of transfer learning. Recent deep networks often apply a pre-trained network (typically trained on the ImageNet dataset) as the initialization for object recognition and segmentation. As in many computer vision tasks, it is often slow and tedious to train a new network from scratch. Hence, using a pre-trained model on one dataset to help another is a major advantage of transfer learning. Traditional DA methods relied on the extracted features from the pre-trained ImageNet models and then aligned the marginal or conditional distributions between different domains [34, 39, 35]. Recent deep networks frequently select ResNet50 as the backbone network for UDA [30, 15]. Notably, other deep networks are not investigated, even though different ImageNet models affect the performance of UDA on traditional methods [36, 40]. The impact of extracted pre-trained features on deep networks are not well explored. In addition, selecting the best pre-trained features based on performance in the target domain requires significant computation to train models in the supervised source domain and infer to the unlabeled target.

Pseudo-labeling. Pseudo-labeling is another technique to address UDA and also achieves substantial performance on multiple tasks. Pseudo-labeling typically generates pseudo labels for the target domain based on the predicted class probability [20, 31, 32, 3, 37]. Under such a regime, some target domain label information can be considered during training. In deep networks, the source classifier is usually treated as an initial pseudo labeler to generate the pseudo labels (and use them as if they were real labels). Different algorithms are proposed to obtain additional pseudo labels and promote distribution alignment between the two domains.

An asymmetric tri-training method for UDA has been proposed to generate pseudo labels for target samples using two networks, and the third can learn from them to obtain target discriminative representations [20]. Xie et al. [31] proposed a Moving Semantic Transfer Network (MSTN) to develop semantic matching and domain adversary losses to obtain pseudo labels. Zhang et al. [32] designed a new criterion to select pseudo-labeled target samples and developed an iterative approach called incremental CAN (iCAN), in which they select samples iteratively and retrain the network using the expanded training set. Progressive Feature Alignment Network (PFAN) [3] aligns the discriminative features across domains progressively and employs an easy-to-hard transfer strategy for iterative learning. Chang et al. [2] proposed to combine the external UDA algorithm and the proposed domain-specific batch normalization to estimate the pseudo labels of samples in the target domain and more effectively learn the domain-specific features. Constrictive Adaptation Network (CAN) also employed batch normalization layers to capture the domain-specific distributions [11].

These methods highly rely on the pseudo labels to compensate for the lack of categorical information in the target domain. However, they did not check the quality of pseudo-labels, as noisy pseudo-labeled samples hurt model performance. In addition, most pseudo-labeling methods employ a two-stage paradigm. The pseudo labels in the first stage are generated and then used to train the model along with the labeled source domain. Differing from previous work [14, 3], we recurrently generate high confidence examples using a novel scheme.

3. Methodology

3.1. Problem

Here we discuss the unsupervised domain adaptation (UDA) problem and introduce some basic notation. Given a labeled source domain $\mathcal{D}_{\mathcal{S}} = \{\mathcal{X}_{\mathcal{S}}^i, \mathcal{Y}_{\mathcal{S}}^i\}_{i=1}^{\mathcal{N}_{\mathcal{S}}}$ with $\mathcal{N}_{\mathcal{S}}$ samples in C categories and an unlabeled target domain $\mathcal{D}_{\mathcal{T}} = \{\mathcal{X}_{\mathcal{T}}^j\}_{j=1}^{\mathcal{N}_{\mathcal{T}}}$ with $\mathcal{N}_{\mathcal{T}}$ samples in the same C categories ($\mathcal{Y}_{\mathcal{T}}$ for evaluation only), our challenge is how to get a well-trained classifier so that domain discrepancy is minimized and generalization error in the target domain is reduced.

In UDA, a typical method would first select one of the ImageNet models as the backbone network for feature extraction. Then, more fine-tuning layers and loss functions are added to minimize the discrepancy between $\mathcal{X}_{\mathcal{S}}$ and $\mathcal{X}_{\mathcal{T}}$. In addition, most pseudo-labeling algorithms do not check the reliability of the generated pseudo labels. These approaches face two critical limitations: (1) they did not explore other pre-trained networks and did not select the best pre-trained features. It is hence necessary to develop an unsupervised metric to quickly determine the best pre-trained features among different ImageNet models. (2) Noisy pseudo labels can deteriorate domain invariant features, and the conditional distributions of two domains are difficult to align at the category level, and the marginal and conditional distributions are not jointly well aligned.

To mitigate these shortcomings, we propose a recurrent pseudo-labeling using the best pre-trained features (PRPL) model. We also jointly align the marginal and conditional distributions of the two domains.

3.2. Pre-training Feature Representation

Feature extraction is an easy and fast way to use the power of deep learning without investing resources into training or fine-tuning a network. It would be especially useful when there are no powerful GPUs to train additional deep networks. However, one disadvantage of feature extraction using pre-trained networks is that it has lower performance than fine-tuning the same network since feature extraction is a single pass over the images. However, previous work [36] suggests that a better ImageNet model will produce better features for UDA. Therefore, we can extract features from a better ImageNet model to compensate for the lower performance of not fine-tuning. Moreover, feature extraction is significantly faster than fine-tuning a neural network. Thus, our design goal in feature representation learning is to find fast and accurate features from well-trained ImageNet models.

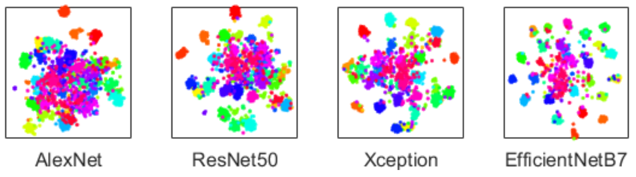


Figure 1: t-SNE view of extracted features from four pre-trained networks (AlexNet [12], ResNet50 [7], Xception [4] and EfficientNetB7 [27]). Different colors represent different classes. EfficientNetB7 has better features than others since the classes are more separate (from Amazon domain in Office31 dataset).

Since there are several well-trained ImageNet models, we employ a feature extractor Φ to extract features from the source and target images using a pre-trained model[1]. Fig. 1 shows extracted features using four different pre-trained ImageNet models. EfficientNetB7 [27] has better performance and better-separated features than others from visual inspection. However, we should have an algorithmic solution that chooses the best pre-trained features. We thus design a fast and accurate unsupervised metric. For a given feature extractor Φ_k, the mean distance between latent represented source $\Phi_k(\mathcal{X_S})$ and target domain $\Phi_k(\mathcal{X_T})$ can be denoted as:

$$Dist_k^{Pre} = ||\frac{1}{\mathcal{N_S}}\sum_{i=1}^{\mathcal{N_S}}\Phi_k(\mathcal{X}_\mathcal{S}^i) - \frac{1}{\mathcal{N_T}}\sum_{j=1}^{\mathcal{N_T}}\Phi_k(\mathcal{X}_\mathcal{T}^j)||_2. \quad (1)$$

where Φ_k is the $k^{th} \in \{1, 2, \cdots K\}$ feature extractor from seventeen pre-trained models ($K = 17$) and $||\cdot||_2$ is the L2 norm. With a different Φ_k, such a distance can be varied. $Dist_k^{Pre}$ is an easy and fast unsupervised metric to quantify the quality of extracted pre-trained features.

Therefore, we can select the best pre-trained features if $Dist_k^{Pre}$ has the shortest distance between two domains. The performance of different pre-trained feature distances is shown in Sec 4.2. The Φ in the following section refers to the best feature extractor, *i.e.*, EfficientNetB7.

3.3. Feature Alignment

3.3.1 Initial Source Classifier

The task in the source domain is to minimize the typical cross-entropy loss in the following equation:

$$\mathcal{L_S} = -\frac{1}{\mathcal{N_S}}\sum_{i=1}^{\mathcal{N_S}}\sum_{c=1}^{C}\mathcal{Y}_{\mathcal{S}_c}^i \log(\mathcal{F}_c(\Phi(\mathcal{X}_\mathcal{S}^i))), \quad (2)$$

where $\mathcal{Y}_{\mathcal{S}_c}^i \in [0,1]^C$ is the binary indicator of each class c in true label for observation $\Phi(\mathcal{X}_\mathcal{S}^i)$, and $\mathcal{F}_c(\Phi(\mathcal{X}_\mathcal{S}^i))$ is the predicted probability of class c (using the softmax function as shown in Fig. 2).

[1] Φ extracts features from the layer prior to last fully connected layer of our examined pre-trained models.

3.3.2 Maximum Mean Discrepancy

Maximum mean discrepancy (MMD) [16] is a non-parametric distance measure to compare the distributions of source and target domains by mapping data into reproducing kernel Hilbert space. After the initial classifier $\mathcal{F}$, it is expressed by

$$\begin{aligned}\mathcal{L_{MMD}} = &\frac{1}{\mathcal{N}_\mathcal{S}^2}\sum_{i,j}^{\mathcal{N_S}}\kappa(L_\mathcal{S}^i, L_\mathcal{S}^j) + \frac{1}{\mathcal{N}_\mathcal{T}^2}\sum_{i,j}^{\mathcal{N_T}}\kappa(L_\mathcal{T}^i, L_\mathcal{T}^j) \\ &- \frac{2}{\mathcal{N_S}\cdot\mathcal{N_T}}\sum_{i,j}^{\mathcal{N_S},\mathcal{N_T}}\kappa(L_\mathcal{S}^i, L_\mathcal{T}^j),\end{aligned} \quad (3)$$

where κ is the mean of a linear combination of multiple RBF kernels and $L_\mathcal{S} = \mathcal{F}(\Phi(\mathcal{X_S}))$, and $L_\mathcal{T} = \mathcal{F}(\Phi(\mathcal{X_T}))$. Therefore, the $\mathcal{L_{MMD}}$ aims to minimize the marginal distance between two domains defined as follows:

$$Dist^{Ma} = ||\frac{1}{\mathcal{N_S}}\sum_{i=1}^{\mathcal{N_S}}L_\mathcal{S}^i - \frac{1}{\mathcal{N_T}}\sum_{j=1}^{\mathcal{N_T}}L_\mathcal{T}^j||_2. \quad (4)$$

3.4. Recurrent Pseudo-Labeling Learning

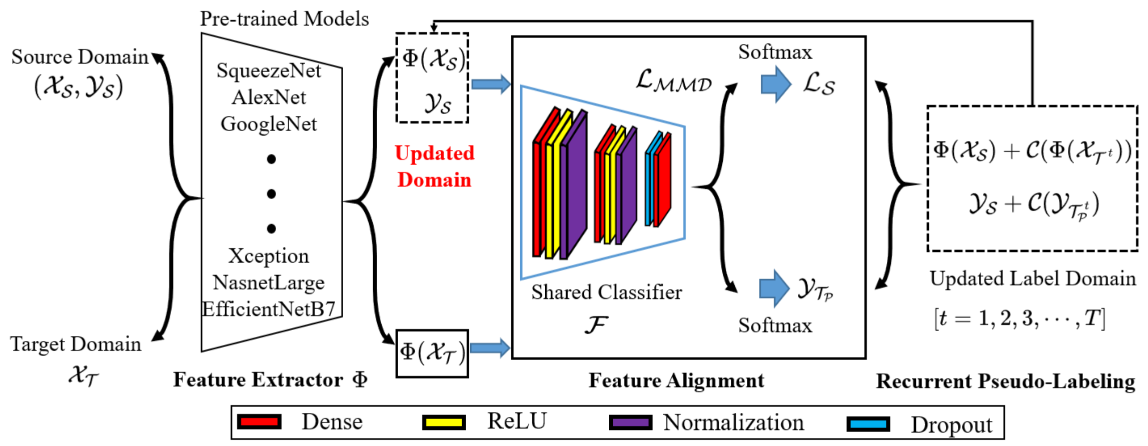
Initial feature alignment only trains the target domain in an unsupervised fashion. To get reliable predicted target domain labels, we train the networks with the instances of the labeled source and pseudo labeled target domains. Pseudo labels of target domain examples will be treated as if they are true labels. The domain invariant features are effective and better adapted in such a paradigm.

3.4.1 Confident Pseudo-Labeling

In this stage, we take advantage of the initial source classifier $\mathcal{F}$ to generate confident pseudo labels and examples for the target domain. In contrast to Hinton et al. [8], we do not use the weighted sum of the soft posteriors and the one-hot hard label to train the model since the prediction of the target domain is not accurate when the classification decision from $\mathcal{F}$ is incorrect. In recurrent pseudo-labeling learning, we continuously bring confident examples and pseudo labels from the target domain to the source domain. A confident pseudo label is defined as

$$\mathcal{C}(\mathcal{Y}_{\mathcal{T_P}}^j) = \arg\max_{c\in C}\{\mathcal{F}_c(\Phi(\mathcal{X}_\mathcal{T}^j))\} \text{ if} \max(\mathcal{F}_c(\Phi(\mathcal{X}_\mathcal{T}^j))) > p,$$

where $\mathcal{F}_c(\Phi(\mathcal{X}_\mathcal{T}^j))$ is the predicted probability of class c of $\Phi(\mathcal{X}_\mathcal{T}^j)$, $\max(\mathcal{F}_c(\Phi(\mathcal{X}_\mathcal{T}^j)))$ is probability of the dominant class, and it should be greater than probability threshold p $(0 \le p \le 1)$ for a confident pseudo label. Also, its corresponding observation $\Phi(\mathcal{X}_\mathcal{T}^j)$ is called a confident example and denoted as $\mathcal{C}(\Phi(\mathcal{X}_\mathcal{T}^j))$, where $\mathcal{C}$ means confident pseudo labels or examples. The advantage of such confident examples and pseudo labels is to support the high quality of predicted target labels and mitigate negative transfer of $\mathcal{F}$.



Pre-trained Models
Source Domain
$(\mathcal{X}_\mathcal{S}, \mathcal{Y}_\mathcal{S})$
SqueezeNet
AlexNet
GoogleNet
Xception
NasnetLarge
EfficientNetB7
Target Domain
$\mathcal{X}_\mathcal{T}$
Feature Extractor Φ
$\Phi(\mathcal{X}_\mathcal{S})$
$\mathcal{Y}_\mathcal{S}$
Updated Domain
$\Phi(\mathcal{X}_\mathcal{T})$
$\mathcal{L}_{\mathcal{MMD}}$
Softmax
$\mathcal{L}_\mathcal{S}$
Shared Classifier
$\mathcal{F}$
$\mathcal{Y}_{\mathcal{T}_\mathcal{P}}$
Softmax
Feature Alignment
$\Phi(\mathcal{X}_\mathcal{S}) + \mathcal{C}(\Phi(\mathcal{X}_{\mathcal{T}^t}))$
$\mathcal{Y}_\mathcal{S} + \mathcal{C}(\mathcal{Y}_{\mathcal{T}^t_\mathcal{P}})$
Updated Label Domain
$[t = 1, 2, 3, \cdots, T]$
Recurrent Pseudo-Labeling
🟥 Dense
🟨 ReLU
🟪 Normalization
🟦 Dropout

Figure 2: Architecture of the PRPL model. We first extract features $\Phi(\mathcal{X}_{\mathcal{S}/\mathcal{T}})$ for both source and target domains via Φ using a pre-trained model, and then train the shared classifier $\mathcal{F}$. In recurrent pseudo-labeling, confident pseudo-labeled examples $(\{\mathcal{C}(\Phi(\mathcal{X}_{\mathcal{T}^t})), \mathcal{C}(\mathcal{Y}_{\mathcal{T}^t_\mathcal{P}})\})$ are generated continuously in each t to form the updated label domain. During training, the updated label domain will keep replacing $\{\Phi(\mathcal{X}_\mathcal{S}), \mathcal{Y}_\mathcal{S}\}$ (the rectangle above updated domain). $\mathcal{L}_\mathcal{S}$ is source classification loss and $\mathcal{L}_{\mathcal{MMD}}$ is maximum mean discrepancy loss. Best viewed in color.

Therefore, we construct an updated label domain $\mathcal{D}_\mathcal{U} = \{\mathcal{X}^n_\mathcal{U}, \mathcal{Y}^n_\mathcal{U}\}^{\mathcal{N}_\mathcal{U}}_{n=1}$, which consists of the labeled source domain and confident target examples with pseudo labels, where $\mathcal{N}_\mathcal{U} \leq \mathcal{N}_\mathcal{S} + \mathcal{N}_\mathcal{T}$, $\mathcal{X}_\mathcal{U} = \Phi(\mathcal{X}_\mathcal{S}) + \mathcal{C}(\Phi(\mathcal{X}_\mathcal{T}))$ and $\mathcal{Y}_\mathcal{U} = \mathcal{Y}_\mathcal{S} + \mathcal{C}(\mathcal{Y}_{\mathcal{T}_\mathcal{P}})$, and $\mathcal{N}_\mathcal{U}$ is controlled by p. $\mathcal{N}_\mathcal{U} = 0$ if $p = 1$, and $\mathcal{N}_\mathcal{U} = \mathcal{N}_\mathcal{S} + \mathcal{N}_\mathcal{T}$ if $p = 0$.

3.4.2 Recurrent learning

Most existing pseudo-labeling methods only generate pseudo labels in a single iteration. However, such a paradigm cannot guarantee reliable predictions of the target domain. Therefore, we propose recurrent pseudo-labeling to continuously generate confident examples for T iterations. In each iteration t, the updated label domain becomes: $\mathcal{D}^t_\mathcal{U} = \{\mathcal{X}^n_{\mathcal{U}^t}, \mathcal{Y}^n_{\mathcal{U}^t}\}^{\mathcal{N}_{\mathcal{U}^t}}_{n=1}$, where $\mathcal{X}_{\mathcal{U}^t} = \Phi(\mathcal{X}_\mathcal{S}) + \mathcal{C}(\Phi(\mathcal{X}_{\mathcal{T}^t}))$ and $\mathcal{Y}_{\mathcal{U}^t} = \mathcal{Y}_\mathcal{S} + \mathcal{C}(\mathcal{Y}_{\mathcal{T}^t_\mathcal{P}})$, and $t \in \{1, 2, 3, \cdots, T\}$. To suppress potentially noisy pseudo labels, in each iteration t, the sample size of updated labels is always not greater than $\mathcal{N}_\mathcal{S} + \mathcal{N}_\mathcal{T}$, *i.e.*, $\mathcal{N}_{\mathcal{U}^t} \leq \mathcal{N}_\mathcal{S} + \mathcal{N}_\mathcal{T}$.

$$\mathcal{C}(\mathcal{Y}_{\mathcal{T}^t_\mathcal{P}}) = \arg\max_{c \in C}\{\mathcal{F}_c(\Phi(\mathcal{X}^j_{\mathcal{T}^t}))\} \ \text{ if} \max(\mathcal{F}_c(\Phi(\mathcal{X}^j_{\mathcal{T}^t}))) > p_t, \tag{5}$$

In addition, $\mathcal{C}(\mathcal{Y}_{\mathcal{T}^t_\mathcal{P}})$ is also updated using Eq. 5 for the probability threshold p_t, and should maintain the condition of $p_{t+1} \geq p_t (0 \leq p_t \leq 1)$ since we only want to generate reliable pseudo labels, which will further avoid negative transfer via pushing the decision boundary toward to the target domain. In all iterations, we have a sequence of p_t, and $p_T = \{p_t\}^T_{t=1}$. Therefore, we can produce confident examples and pseudo labels in each recurrent training.

During training, the updated label domain $\mathcal{D}^t_\mathcal{U}$ will keep replacing the original latent represented source domain $\{\Phi(\mathcal{X}_\mathcal{S}), \mathcal{Y}_\mathcal{S}\}$. The parameters in the feature alignment module will be updated via both labeled source domain and pseudo labeled target domain. Therefore, the loss function in each training iteration is given by:

$$\mathcal{L}^t_\mathcal{U} = \mathcal{L}^t_\mathcal{S} + \mathcal{L}^t_{\mathcal{MMD}}, \tag{6}$$

where $\mathcal{L}^t_\mathcal{S} = -\frac{1}{\mathcal{N}_{\mathcal{U}^t}} \sum^{\mathcal{N}_{\mathcal{U}^t}}_{n=1} \sum^C_{c=1} \mathcal{Y}^n_{\mathcal{U}^t_c} \log(\mathcal{F}_c(\mathcal{X}^n_{\mathcal{U}^t}))$,

$$\mathcal{L}^t_{\mathcal{MMD}} = \frac{1}{\mathcal{N}^2_{\mathcal{U}^t}} \sum^{\mathcal{N}_{\mathcal{U}^t}}_{n,j} \kappa(L^n_{\mathcal{U}^t}, L^j_{\mathcal{U}^t}) + \frac{1}{\mathcal{N}^2_\mathcal{T}} \sum^{\mathcal{N}_\mathcal{T}}_{n,j} \kappa(L^n_\mathcal{T}, L^j_\mathcal{T}) - \frac{2}{\mathcal{N}_{\mathcal{U}^t} \cdot \mathcal{N}_\mathcal{T}} \sum^{\mathcal{N}_{\mathcal{U}^t}, \mathcal{N}_\mathcal{T}}_{n,j} \kappa(L^n_{\mathcal{U}^t}, L^j_\mathcal{T}).$$

In $\mathcal{L}^t_\mathcal{S}$, $\mathcal{Y}^n_{\mathcal{U}^t_c} \in [0, 1]^C$ is the binary indicator of each class c for observation $\mathcal{X}^n_{\mathcal{U}^t}$ in the t^{th} training iteration, and $\mathcal{F}_c(\mathcal{X}^n_{\mathcal{U}^t})$ is the predicted probability of class c. In $\mathcal{L}^t_{\mathcal{MMD}}$, $L_{\mathcal{U}^t} = \mathcal{F}(\mathcal{X}_{\mathcal{U}^t})$, $L_\mathcal{T} = \mathcal{F}(\Phi(\mathcal{X}_\mathcal{T}))$, and it also measures the discrepancy between updated label domain and the target domain in each t. Unlike Eq. 3, $\mathcal{L}^t_{\mathcal{MMD}}$ also includes the confident examples from the target domain. The networks will be jointly optimized using the updated domain, which is equivalent to minimizing the following conditional distance during training.

$$Dist^{Co}_t = Dist \sum^C_{c=1} (\mathcal{Y}_{\mathcal{S}^c} | \mathcal{X}_{\mathcal{S}^c}, \mathcal{C}(\mathcal{Y}_{\mathcal{T}^{tc}_\mathcal{P}}) | \mathcal{C}(\mathcal{X}_{\mathcal{T}^{tc}}))$$
$$= \frac{1}{C} \sum^C_{c=1} || \frac{1}{\mathcal{N}^c_\mathcal{S}} \sum^{\mathcal{N}^c_\mathcal{S}}_{i=1} \Phi(\mathcal{X}^i_{\mathcal{S}^c}) - \frac{1}{\mathcal{C}(\mathcal{N}^c_{\mathcal{T}^t})} \sum^{\mathcal{C}(\mathcal{N}^c_{\mathcal{T}^t})}_{n=1} \mathcal{C}(\Phi(\mathcal{X}^n_{\mathcal{T}^{tc}})) ||_2 \tag{7}$$

where C is the number of categories, $\mathcal{Y}_{\mathcal{S}^c}|\mathcal{X}_{\mathcal{S}^c}$ (or $\mathcal{C}(\mathcal{Y}_{\mathcal{T}_{\mathcal{P}}^{tc}})|\mathcal{C}(\mathcal{X}_{\mathcal{T}^{tc}}))$) represents c^{th} category data in the source domain or confident pseudo labeled target domain. $\mathcal{N}_{\mathcal{S}}^c$ or $\mathcal{C}(\mathcal{N}_{\mathcal{T}^t}^c)$ is the number of samples in the c^{th} category in the source domain or confident pseudo labeled target domain in each t.

In each iteration, we first use the initial classifier to generate pseudo labels for the target domain. Then, the pseudo labels will be progressively refined. We empirically demonstrate that such iterative learning is effective and efficient in improving target domain accuracy.

3.5. PRPL model

The framework of our proposed PRPL model is depicted in Fig. 2. Taken altogether, our model minimizes the following objective function:

$$\mathcal{L}(\mathcal{X}_{\mathcal{S}}, \mathcal{Y}_{\mathcal{S}}, \mathcal{X}_{\mathcal{T}}) = \arg\min(\mathcal{L}_{\mathcal{S}} + \mathcal{L}_{\mathcal{MMD}} + \sum_{t=1}^{T} \mathcal{L}_{\mathcal{U}}^{t}) \qquad (8)$$

where $\mathcal{L}_{\mathcal{S}}$ is the source classification loss and $\mathcal{L}_{\mathcal{MMD}}$ minimizes the distance between initial source and target represented data. $\mathcal{L}_{\mathcal{U}}^{t}$ is the loss function of each t. T represents the number of iterations of training.

3.6. Domain Adaptation Theory

We formalize the theoretical error bound of the target domain for proposed recurrent learning in Lemma 1.

Lemma 1. *Let h be a hypothesis in a hypothesis space H. $\epsilon_{\mathcal{S}}(h)$ and $\epsilon_{\mathcal{T}}(h)$ represent the source and target domain risk, respectively [1]. We have*

$$\epsilon_{\mathcal{T}}(h) \leq \epsilon_{\mathcal{S}}(h) + d_{\mathcal{H}}(P(\Phi(\mathcal{X}_{\mathcal{S}})), P(\Phi(\mathcal{X}_{\mathcal{T}}))) + \gamma,$$

where $d_{\mathcal{H}}(P(\Phi(\mathcal{X}_{\mathcal{S}})), P(\Phi(\mathcal{X}_{\mathcal{T}})))$ is the $\mathcal{H}$ divergence between the probability distribution of the source and target domain. $\gamma = \epsilon_{\mathcal{S}}(h^*, \mathcal{Y}_{\mathcal{S}}) + \epsilon_{\mathcal{T}}(h^*, \mathcal{F}(\Phi(\mathcal{X}_{\mathcal{T}})))$ is the adaptability to quantify the error in ideal hypothesis h^* space of source and target domain, which should be small.

During the recurrent pseudo-labeling, we expect the $\mathcal{H}$ divergence between the distributions of latent feature space can be minimized, and that an ideal hypothesis exists with low risk on both domains, which is corresponding to a small β in Lemma 1. In addition, such a divergence is assessed by $d_{\mathcal{H}}(P(\Phi(\mathcal{X}_{\mathcal{S}})), P(\Phi(\mathcal{X}_{\mathcal{T}}))) \approx Dist^{Ma} + \frac{1}{T}\sum_{t=1}^{T} Dist_t^{Co}$. Therefore, with the implicitly minimized training risk, domain divergence, and the adaptability of true hypothesis h, the generalization bound of $\epsilon_{\mathcal{T}}(h)$ can be achieved.

4. Experiments

4.1. Setup

Datasets. **Office + Caltech-10** [6] consists of Office 10 and Caltech 10 datasets with 2,533 images from ten classes in four domains: Amazon (A), Webcam (W), DSLR (D) and Caltech (C). There are twelve tasks in Office + Caltech-10 dataset. **Office-31** [19] consists of 4,110 images in 31 classes from three domains: Amazon (A), Webcam (W), and DSLR (D). We evaluate methods across all six transfer tasks. **Office-Home** [29] contains 15,588 images from 65 categories. It has four domains: Art (Ar), Clipart (Cl), Product (Pr) and Real-World (Rw). There are also twelve tasks in this dataset. Therefore, we have 30 tasks in our experiment. In experiments, C→A represents learning knowledge from domain C which is applied to domain A.

Table 1: Pre-trained feature mean distance (1.0e+05) between two domains and feature extraction time (minutes) for three datasets (MD: mean distance; OC10: Office + Caltech-10; IR: Inceptionresnetv2; EB7: EfficientNetB7; NM: Nasnetmobile).

Networks	OC10		Office-31		Office-Home	
	MD	Time	MD	Time	MD	Time
SqueezeNet [10]	41.93	0.79	32.61	1.16	28.38	7.23
AlexNet [12]	20.05	0.29	18.74	0.46	19.68	4.20
GoogleNet [25]	15.25	0.28	15.74	0.47	13.89	4.25
ShuffleNet [33]	24.97	0.32	25.81	0.54	21.82	4.47
ResNet18 [7]	19.27	0.29	18.99	0.48	17.27	4.30
Vgg16 [22]	15.92	0.40	16.11	0.64	16.70	4.80
Vgg19 [22]	15.49	0.43	16.17	0.68	16.86	4.94
MobileNetv2 [21]	8.53	0.34	8.13	0.57	8.17	4.62
NM [41]	6.03	0.90	5.44	1.21	6.66	6.21
ResNet50 [7]	18.94	0.39	19.62	0.64	18.13	4.78
ResNet101 [7]	20.25	0.48	20.17	0.75	19.11	5.18
DenseNet201 [9]	22.1	1.32	22.05	2.04	18.80	9.56
Inceptionv3 [26]	5.74	0.43	5.47	0.68	5.94	4.86
Xception [4]	6.02	0.70	5.75	1.13	7.03	6.76
IR [24]	5.40	0.81	5.73	1.19	6.80	6.29
NasnetLarge [41]	4.19	2.45	4.04	3.64	6.15	14.65
EB7 [41]	**1.13**	4.06	**1.27**	8.31	**1.49**	23.64

Implementation details. As in Zhang and Davison [36], we first extract features from the last fully connected layer from 17 different networks for the three datasets. Parameters in recurrent pseudo labeling are $T = 3$ and $p_T = [0.5, 0.8, 0.9]$. Learning rate ($\epsilon = 0.001$), batch size (64), and number of epochs (9) are determined by performance on the source domain. We compare our results with 12 state-of-the-art methods. For fair comparison, we highlight those methods in bold that are re-implemented using our extracted features, and other methods are directly reported from their original papers. Specifically, we modify the architecture of DAN [14], DANN [5] and DCORAL [23], and replace the feature extractor with the best pre-trained features, while maintaining original loss functions[2]. Experiments are tested with a GeForce 1080 Ti.

[2]Source code is available at: https://github.com/YoushanZhang/Transfer-Learning/tree/main/Code/Deep/PRPL.

Table 2: Accuracy (%) on Office + Caltech-10 dataset

Task	C→A	C→W	C→D	A→C	A→W	A→D	W→C	W→A	W→D	D→C	D→A	D→W	**Ave.**
DAN [14]	**96.8**	95.9	96.2	93.1	88.8	92.4	94.5	95.4	**100**	89.1	95.9	95.9	94.5
DANN [5]	96.7	94.9	97.5	95.8	94.9	91.1	94.7	94.5	**100**	94.9	92.4	93.9	95.1
DCORAL [23]	96.5	97.6	96.8	96.3	98.3	96.8	94.9	95.8	**100**	94.6	95.8	99.0	96.9
DDC [28]	91.9	85.4	88.8	85.0	86.1	89.0	78.0	83.8	**100**	79.0	87.1	97.7	86.1
RTN [16]	93.7	96.9	94.2	88.1	95.2	95.5	86.6	92.5	**100**	84.6	93.8	99.2	93.4
MDDA [18]	93.6	95.2	93.4	89.1	95.7	96.6	86.5	94.8	**100**	84.7	94.7	99.4	93.6
PRPL$_{t=0}$	96.7	98.6	98.1	95.5	99.0	**100**	95.3	**96.7**	**100**	95.1	**96.2**	**99.7**	97.6
PRPL$_{t=1}$	96.7	98.6	99.4	96.2	**99.3**	**100**	96.4	96.6	**100**	96.1	96.1	**99.7**	97.9
PRPL$_{t=2}$	**96.8**	98.6	**100**	96.4	**99.3**	**100**	96.4	96.5	**100**	96.1	**96.2**	**99.7**	98.0
PRPL$_{t=3}$	96.7	**99.0**	**100**	**96.6**	**99.3**	**100**	**96.6**	96.6	**100**	**96.2**	**96.2**	**99.7**	**98.1**

Table 3: Accuracy (%) on Office-Home dataset

Task	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	**Ave.**
DAN [14]	55.0	72.5	79.1	66.5	72.9	74.3	69.7	57.4	82.7	76.1	59.1	85.4	70.9
DANN [5]	57.1	75.7	80.3	69.2	76.7	74.3	70.9	58.0	83.0	77.8	59.8	87.0	72.5
DCORAL [23]	59.1	78.4	82.3	71.1	79.0	77.4	71.2	57.8	84.3	78.7	60.4	87.0	73.9
CDAN-RM [15]	49.2	64.8	72.9	53.8	62.4	62.9	49.8	48.8	71.5	65.8	56.4	79.2	61.5
CDAN-M [15]	50.6	65.9	73.4	55.7	62.7	64.2	51.8	49.1	74.5	68.2	56.9	80.7	62.8
ETD [13]	51.3	71.9	85.7	57.6	69.2	73.7	57.8	51.2	79.3	70.2	57.5	82.1	67.3
TADA [30]	53.1	72.3	77.2	59.1	71.2	72.1	59.7	53.1	78.4	72.4	60.0	82.9	67.6
SymNets [38]	47.7	72.9	78.5	64.2	71.3	74.2	64.2	48.8	79.5	74.5	52.6	82.7	67.6
PRPL$_{t=0}$	65.0	83.2	88.3	78.0	83.9	85.2	75.3	65.3	87.6	82.1	66.2	90.6	79.2
PRPL$_{t=1}$	67.0	84.3	**89.4**	79.4	85.2	85.7	78.1	68.1	88.5	83.4	68.6	91.3	80.8
PRPL$_{t=2}$	**68.0**	**84.5**	**89.4**	79.5	**85.7**	86.0	79.1	68.9	88.6	83.3	68.5	91.4	81.1
PRPL$_{t=3}$	67.6	**84.5**	**89.4**	**79.8**	**85.7**	**86.3**	**79.2**	**69.1**	**88.7**	**83.8**	**68.9**	**91.5**	**81.2**

Table 4: Accuracy (%) on Office-31 dataset

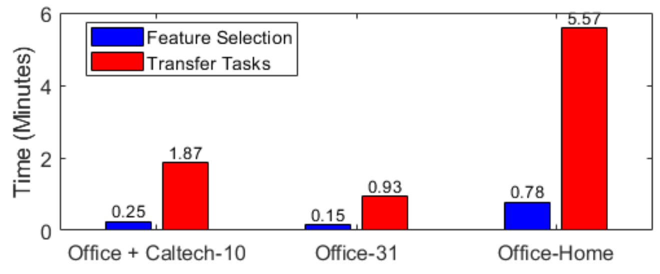
Task	A→W	A→D	W→A	W→D	D→A	D→W	**Ave.**
DAN [14]	85.8	88.0	75.7	98.4	74.7	95.0	86.3
DANN [5]	85.9	90.2	76.6	98.8	78.2	96.2	87.7
DCORAL [23]	90.7	90.6	79.0	98.8	78.7	97.1	89.2
RTN [16]	84.5	77.5	64.8	99.4	66.2	96.8	81.6
ETD [13]	92.1	88.0	67.8	**100**	71.0	**100**	86.2
TADA [30]	94.3	91.6	73.0	99.8	72.9	98.7	88.4
SymNets [38]	90.8	93.9	72.5	**100**	74.6	98.8	88.4
CAN [11]	94.5	95.0	77.0	99.8	78.0	99.1	90.6
PRPL$_{t=0}$	92.1	96.0	80.4	98.6	80.1	96.1	90.6
PRPL$_{t=1}$	95.4	**97.0**	81.8	99.2	82.1	97.0	92.1
PRPL$_{t=2}$	95.6	96.8	82.2	99.2	82.8	97.1	92.3
PRPL$_{t=3}$	**95.9**	**97.0**	**82.4**	99.2	**83.0**	97.1	**92.4**

Table 5: Ablation experiments on Office-31

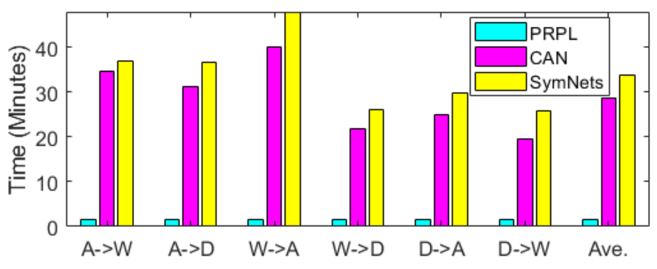
Task	A→W	A→D	W→A	W→D	D→A	D→W	**Ave.**
PRPL$_{t=0}$-$\mathcal{M}$	91.6	96.0	79.9	98.8	80.6	96.0	90.5
PRPL$_{t=1}$-$\mathcal{M}$	94.1	96.6	81.4	**99.2**	82.5	97.0	91.8
PRPL$_{t=2}$-$\mathcal{M}$	94.2	96.3	81.5	**99.2**	82.9	97.0	91.9
PRPL$_{t=3}$-$\mathcal{M}$	94.3	96.4	81.9	**99.2**	82.9	97.0	92.0
PRPL$_{t=3}$	**95.9**	**97.0**	**82.4**	**99.2**	**83.0**	**97.1**	**92.4**

4.2. Results

Pre-trained feature selection. We first conduct experiments to select the best pre-trained features for both the source and target domains. We calculate the distance between two domains using the aforementioned pre-trained feature distance in Eq. 1. The smallest distance between two domains reveals the corresponding pre-trained network that is the best for feature extraction. In Tab. 1, we first report the mean distance between two domains of three datasets (for Office + Caltech-10 and Office-Home, the mean distance is the average of twelve tasks. For Office-31, the mean distance is the average of six tasks). It is obvious that pre-trained features from EfficientNetB7 have the smallest distance between two domains, which suggests the EfficientNetB7 is the best deep network compared to the other 16 networks. In addition, this observation is consistent with [36], that a better ImageNet model will produce better pre-trained features for UDA. Furthermore, we also list the computation time for feature extraction of each dataset. We notice that EfficientNetB7 consumes more time than other networks since it has more complex layers, and it needs more memory to process images. However, the longest time is 23.64 minutes, and these features will not be extracted again during the training. Therefore, we opt for EfficientNetB7 as the feature extractor to extract pre-trained features for the benchmark datasets. We then use the extracted features $\Phi(\mathcal{X}_{\mathcal{S}})$ and $\Phi(\mathcal{X}_{\mathcal{T}})$ to perform the domain transfer tasks. In **supplementary material**, we also validate the effectiveness of the proposed distance function $Dist_k^{Pre}$ in choosing the best pre-trained features by comparing to MMD and mean cosine distance function.



(a) Time of three datasets



(b) Time of each task in Office31

Figure 3: Computation time comparison. (a) is the total computation time, that includes the pre-trained feature selection and all transfer tasks (twelve for Office + Caltech-10, six for Office-31 and another twelve for Office-Home). (b) compares the PRPL model with the other two baselines in each task of Office31 (feature extraction time is also included). On average, our PRPL model is approximately 18 times faster than CAN [11], and 21 times faster than SymNets [38].

Domain transfer accuracy. The performance on Office + Caltech-10, Office-Home and Office-31 are shown in Tables 2-4. Our PRPL model outperforms all state-of-the-art methods in terms of average accuracy (especially in the Office-Home dataset). It is compelling that our PRPL model substantially enhances the classification accuracy on difficult adaptation tasks (e.g., W→A task in the Office-31 dataset and the challenging Office-Home dataset, which has a larger number of categories and different domains are visually dissimilar). Our model also outperforms three re-implemented baselines (DAN, DANN, and DCORAL), which use the same EfficientNetB7 features as our model.

In Office + Caltech-10, although the final accuracy in recurrent learning is 98.1%, it does not improve much from 97.6% to 98.1% (from first recurrent learning to the third recurrent learning). One reason is the classification accuracy is high (more than 97%). It is hence difficult to make a large improvement). However, our model still provides a 1.2% improvement over the best baseline (DCORAL). The mean accuracy on the Office-31 dataset is increased from 90.6% to 92.4%. We notice that accuracy is obviously improved when $t = 1$, and then refined when $t = 2/3$. A similar tendency is observed on Office-Home dataset. Therefore, the best pre-trained features are powerful, and the recurrent learning is effective in improving the classification accuracy. In addition, we also show the computation time of our proposed PRPL model in Fig. 3. We require relatively less computation time of all three datasets in Fig. 3a. In particular, we compare the computation time of our model with two other methods (CAN [11] and SymNets [38]) on each task in Office-31 datasets. Our model is obviously faster than the other two (18 and 21 times faster). Therefore, our PRPL model is fast and accurate in UDA tasks.

4.3. Ablation study

To demonstrate the effects of $\mathcal{L}_{\mathcal{MMD}}$ loss on classification accuracy, we present an ablation study in Tab. 5. We observe that $\mathcal{L}_{\mathcal{MMD}}$ loss is useful in improving performance during each training iteration comparing with Tab. 4.

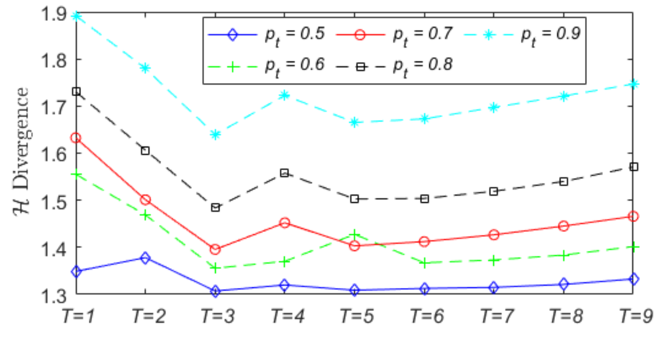
4.4. Parameter Analysis

There are two hyperparameters T and p_t in PRPL that control the number of recurrent learning repetitions and the probability of selecting the confident examples. To get the optimal parameters, we randomly opt the task Rw→Pr and run a set of experiments regarding different values of each parameter. Notice that it is inappropriate to tune parameters using the target domain accuracy since we do not have any labels in the target domain. Therefore, we report the $\mathcal{H}$ divergence between two domains, as stated in Sec. 3.6. Since $\mathcal{H}$ divergence can be assessed by $d_{\mathcal{H}}(P(\Phi(\mathcal{X}_{\mathcal{S}})), P(\Phi(\mathcal{X}_{\mathcal{T}}))) \approx Dist^{Ma} + \frac{1}{T}\sum_{t=1}^{T} Dist_t^{Co}$, we calculate the marginal distance and recurrent conditional distance to tune these two parameters. T is selected from $\{1,2,3,4,5,6,7,8,9\}$, and p_t is selected from $\{0.5, 0.6, 0.7, 0.8, 0.9\}$, we fix one parameter and vary another one at a time.

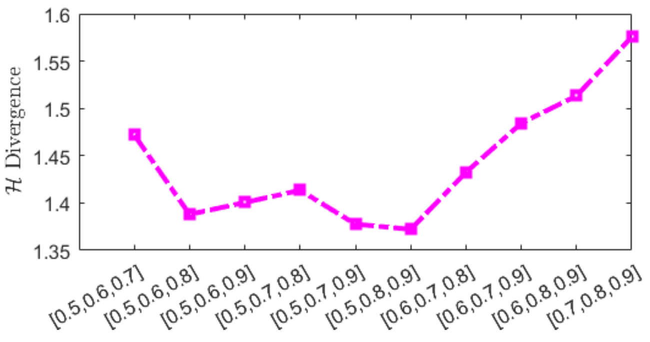
Results presented in Fig. 4 demonstrate that our PRPL model is not very sensitive to a wide range of parameter values since the $\mathcal{H}$ divergence ($d_{\mathcal{H}}$) is not significantly changed. In Fig. 4a, we first tune the parameter T across five different p_t, and it consistently shows that $d_{\mathcal{H}}$ achieves the minimum value when $T = 3$. Therefore, the hyperparameter $T = 3$ is the best since the discrepancy between two domains is minimized. After fixing T, we then present the effect of different p_t on $d_{\mathcal{H}}$ in Fig. 4b. When $p_1 = 0.5, p_2 = 0.8$ and $p_3 = 0.9$, $d_{\mathcal{H}}$ achieves the minimum value. In Fig. 4, a large p_t tends to have a larger $d_{\mathcal{H}}$ (e.g., $p_t = 0.9$ in Fig. 4a and $p_T = [0.7, 0.8, 0.9]$ in Fig. 4b) since a larger p_t will select relatively fewer examples during the training *i.e.,* $\mathcal{N}_{\mathcal{U}}$ is small, and the discrepancy between two domains cannot be well minimized. Therefore, the parameter analysis is useful in finding the best hyperparameters for our PRPL model.

4.5. Feature Visualization

To further investigate the quality of invariant representation learned during the transition from the source domain to the target domain, Fig. 5 visualizes embeddings of the task Rw→Pr in the Office-Home dataset. In this figure, the



(a) Effect of different T on $d_{\mathcal{H}}$



(b) Effect of different p_t on $d_{\mathcal{H}}$

Figure 4: Parameter analysis for T and p_t. In (a), $d_{\mathcal{H}}$ is minimum when $T = 3$. In (b), the x-axis denotes different p_t, and in each array, it contains p_1, p_2 and p_3 since $T = 3$. $d_{\mathcal{H}}$ is minimum when $p_T = [0.5, 0.8, 0.9]$.

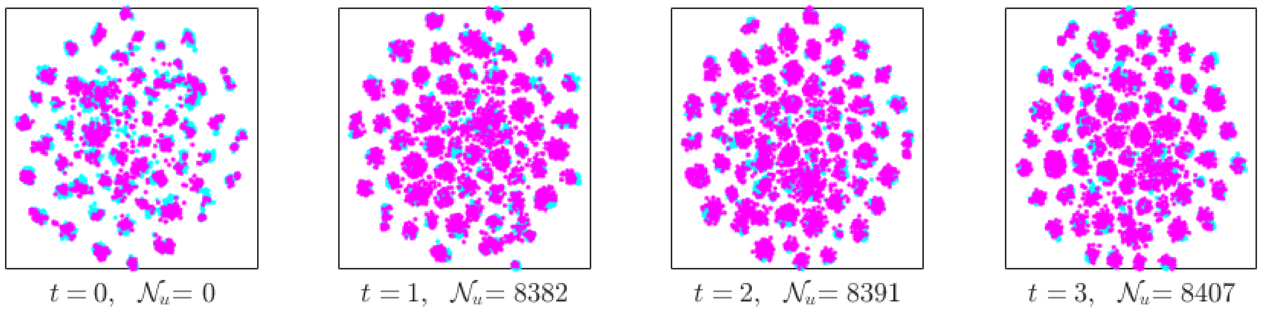


Figure 5: t-SNE of feature visualization of the task Rw→Pr in our recurrent pseudo-labeling learning when $T = 3$. Our PRPL model improves the consistency of representations across domains. Also, the number of updated label domain $\mathcal{N}_{\mathcal{U}}$ is growing with the increasing of time t (magenta color: source domain, cyan color: target domain). Best viewed in color.

magenta dots represent the source domain, and the cyan dots denote the target domain. We can observe that the representation becomes more discriminative when $t = 1$, compared with $t = 0$ (no recurrent learning). Although the representations of $t = 2$ and $t = 3$ are slightly improved, PRPL keeps producing confident examples in $\mathcal{D}^t_{\mathcal{U}}$ since $\mathcal{N}^t_{\mathcal{U}}$ is increasing.

5. Discussion

What can we learn from PRPL? Recurrent learning is effective and accurate to improve target domain accuracy. The architecture of our proposed PRPL is neat and straightforward. However, our model outperforms state-of-the-art methods and achieves the highest accuracy so far. There are two compelling advantages: 1) we extract pre-trained features from 17 well-trained ImageNet networks, and we select the best pre-trained features based on the domain distance. EfficientNetB7 produces high-quality features for the datasets. 2) the proposed recurrent pseudo labeling effectively keeps improving the target domain accuracy in each iteration. Therefore, the generated confident pseudo labels are useful in updating the network parameters, which further reduces the domain discrepancy.

Is a pre-trained ImageNet model helpful? Yes. ImageNet pre-training is important in improving the quality of extracted features. Most existing work focused on fine-tuning the ResNet50 network to perform domain transfer tasks. One underlying reason is that ResNet50 is not a very complex model, and it is easier to re-train the network without the need of multiple GPUs. However, as we can see from the results, the pre-trained features are efficient and easy to use. Therefore, we recommend selecting better pre-trained features for UDA.

6. Conclusion

In this paper, we efficiently and effectively select the best pre-trained features among seventeen well-trained ImageNet models in an unsupervised fashion. The EfficientNetB7 model shows the highest quality in extracting image features. We then propose recurrent pseudo-labeling training to progressively generate confident labels for the target domain. Extensive experiments demonstrate that the proposed PRPL model achieves superior accuracy, noticeably higher than state-of-the-art domain adaptation methods.

References

[1] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 137–144, 2007. 5

[2] W. Chang, T. You, S. Seo, S. Kwak, and B. Han. Domain-specific batch normalization for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7354–7362, 2019. 2

[3] C. Chen, W. Xie, W. Huang, Y. Rong, X. Ding, Y. Huang, T. Xu, and J. Huang. Progressive feature alignment for unsupervised domain adaptation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 627–636, 2019. 2

[4] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017. 3, 5

[5] M. Ghifary, W. B. Kleijn, and M. Zhang. Domain adaptive neural networks for object recognition. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, pages 898–904. Springer, 2014. 5, 6

[6] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2066–2073. IEEE, 2012. 5

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 3, 5

[8] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[9] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 5

[10] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 5

[11] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4893–4902, 2019. 1, 2, 6, 7

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 3, 5

[13] M. Li, Y. Zhai, Y. Luo, P. Ge, and C. Ren. Enhanced transport distance for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13936–13944, 2020. 6

[14] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015. 1, 2, 5, 6

[15] M. Long, Z. Cao, J. Wang, and M. I. Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1647–1657, 2018. 2, 6

[16] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*, pages 136–144, 2016. 1, 3, 6

[17] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. 1

[18] M. M. Rahman, C. Fookes, M. Baktashmotlagh, and S. Sridharan. On minimum discrepancy estimation for deep domain adaptation. In *Domain Adaptation for Visual Understanding*, pages 81–94. Springer, 2020. 6

[19] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *Proceedings of the European Conference on Computer Vision*, pages 213–226. Springer, 2010. 5

[20] K. Saito, Y. Ushiku, and T. Harada. Asymmetric tri-training for unsupervised domain adaptation. *arXiv preprint arXiv:1702.08400*, 2017. 2

[21] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 5

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[23] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Proc. of European Conference on Computer Vision*, pages 443–450. Springer, 2016. 5, 6

[24] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017. 5

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 5

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016. 5

[27] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 3

[28] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 6

[29] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017. 5

[30] X. Wang, L. Li, W. Ye, M. Long, and J. Wang. Transferable attention for domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5345–5352, 2019. 2, 6

[31] S. Xie, Z. Zheng, L. Chen, and C. Chen. Learning semantic representations for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 5423–5432, 2018. 1, 2

[32] W. Zhang, W. Ouyang, W. Li, and D. Xu. Collaborative and adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3801–3809, 2018. 1, 2

[33] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018. 5

[34] Y. Zhang and B. D. Davison. Modified distribution alignment for domain adaptation with pre-trained Inception ResNet. *arXiv preprint arXiv:1904.02322*, 2019. 2

[35] Y. Zhang and B. D. Davison. Domain adaptation for object recognition using subspace sampling demons. *Multimedia Tools and Applications*, pages 1–20, 2020. 2

[36] Y. Zhang and B. D. Davison. Impact of ImageNet model selection on domain adaptation. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision Workshops*, pages 173–182, 2020. 2, 5, 6

[37] Y. Zhang and B. D. Davison. Adversarial continuous learning in unsupervised domain adaptation. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part II*, pages 672–687. Springer International Publishing, 2021. 2

[38] Y. Zhang, H. Tang, K. Jia, and M. Tan. Domain-symmetric networks for adversarial domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5031–5040, 2019. 6, 7

[39] Y. Zhang, S. Xie, and B. D. Davison. Transductive learning via improved geodesic sampling. In *Proceedings of the 30th British Machine Vision Conference*, 2019. 2

[40] Y. Zhang, H. Ye, and B. D. Davison. Adversarial reinforcement learning for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 635–644, 2021. 2

[41] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018. 5