On Unsupervised Domain Adaptation: Pseudo Label Guided Mixup for Adversarial Prompt Tuning

Fanshuang Kong[1], Richong Zhang[1,2*], Ziqiao Wang[3], Yongyi Mao[3]
[1]SKLSDE, School of Computer Science and Engineering, Beihang University, Beijing, China
[2]Zhongguancun Laboratory, Beijing, China
[3]School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada
{kongfs, zhangrc}@act.buaa.edu.cn, {zwang286, ymao}@uottawa.ca

Abstract

To date, a backbone of methods for unsupervised domain adaptation (UDA) involves learning label-discriminative features via a label classifier and domain-invariant features through a domain discriminator in an adversarial scheme. However, these methods lack explicit control for aligning the source data and target data within the same label class, degrading the classifier's performance in the target domain. In this paper, we propose PL-Mix, a pseudo label guided Mixup method based on adversarial prompt tuning. Specifically, our PL-Mix facilitates class-dependent alignment and can alleviate the impact of noisy pseudo-labels. We then theoretically justify that PL-Mix can improve the generalization for UDA. Extensive experiments of the comparison with existing models also demonstrate the effectiveness of PL-Mix.

Introduction

Deep learning's success is partially attributed to extensive datasets with abundant labels (Zhao et al. 2018). However, due to the significant costs of data collection and label annotation, it is important to develop the model capable of transferring knowledge from label-abundant domains to label-scarce domains, a concept known as domain adaptation (Pan and Yang 2010). Particularly, unsupervised domain adaptation (UDA) is a more complex but crucial situation, where target domain labels are unavailable during training. This paper focuses on such a UDA setting, where various methods have been proposed for better performance (Wu and Shi 2022; Du et al. 2020; Zhao et al. 2018; Zhang et al. 2019; Luo et al. 2022; Cai and Wan 2019; Qu et al. 2019). These approaches commonly involve two aspects: learning label-discriminative features using the label classifier, and acquiring domain-invariant features via the domain discriminator (Ganin and Lempitsky 2015). The goal is to align target data with the source data while maintaining source domain label discriminability.

With the development of the pre-trained language model (PLM), prompt tuning, wherein soft prompts are learned for specific downstream tasks, has showcased the resilience and robustness under domain shifts (Brown et al. 2020). AdSPT (Wu and Shi 2022) proposes an adversarial soft prompt

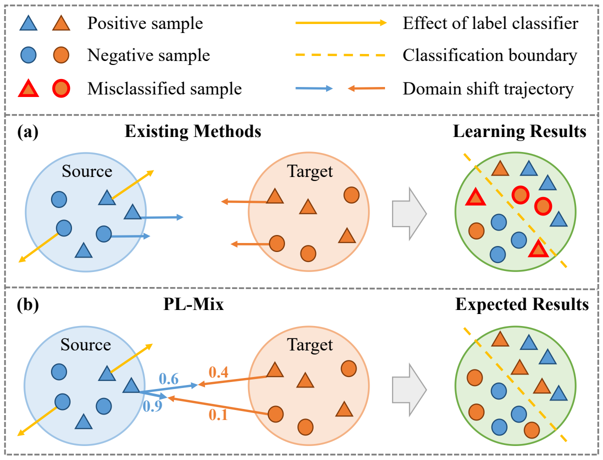

Figure 1: Illustration of existing methods and PL-Mix.

tuning method to solve the UDA problem. This method adopts distinct soft prompts for each individual domain to alleviate the domain discrepancy. Despite these efforts of further aligning target data with source data through domain-specific soft prompt representation, as depicted in Figure 1 (a), there is no explicit mechanism facilitates the positive (or negative) data of the source domain to be attracted towards the corresponding positive (or negative resp.) data of the target domain. That is, source data might align with different class-labeled target data, leading to suboptimal classification performance in the target domain, despite good alignment between source and target data.

In this work, to tackle the problem in Figure 1(a), we enhance the original adversarial prompt tuning (Wu and Shi 2022) with pseudo label guided Mixup (Zhang et al. 2017; Xu et al. 2020), namely PL-Mix. Specifically, we apply Mixup to train both the label classifier and the domain discriminator. While the domain discriminator is supervised by the ground truth domain labels, the label classifier suffers the challenge of lacking labels in the target domain. Accordingly, following previous works (Wu, Inkpen, and El-Roby 2020; Yan et al. 2020; Mao et al. 2019; Sahoo et al. 2023), we use the label classifier to assign the pseudo label to target data for Mixup synthetic label generation. The fusion of pseudo labels and Mixup enables the creation of intermedi-

*Corresponding author

ate synthetic data points between source and target data of the same class, thereby promoting the alignment between the two domains in a class-dependent manner. However, pseudo labels will inevitably bring noise and potentially undermine the performance of the label classifier. Therefore, to mitigate the impact of these noisy pseudo labels on classifier training, we introduce a novel confidence-dependent Mixup ratio (see Eqn. 10). Precisely, we adjust the Mixup ratio based on the confidence level associated with assigning pseudo labels by the classifier. In our setup, a larger Mixup ratio indicates a closer interpolation point to the source data. When a target data receives a high-confidence pseudo label and shares the same label as a source data, we apply a smaller Mixup ratio to encourage the source data to move toward the target data. Conversely, when a source data has a different label from the high-confidence pseudo label assigned to a target data, we adopt a larger Mixup ratio, reducing the extent of the source data's movement toward the target data. For example, as illustrated in Figure 1 (b), a smaller Mixup ratio (0.6) applied to source positive data encourages more movement towards target data with an identical positive label. In contrast, a larger Mixup ratio (0.9) renders the source data move less, given the presence of a different negative label of target data. Regarding low-confidence pseudo labels, deemed as noisy target labels, we either eliminate them directly (by Eqn.11) or reduce source data movement (by Eqn.10). Our intuition behind this adjustment strategy is that the source data should move more toward the target data with a same and highly confident label, while the movement should be limited for a distinct and highly confident label.

Furthermore, we also provide a theoretical justification for the effectiveness of PL-Mix through a generalization upper bound (see Theorem 1) for UDA, which builds upon the recently developed information-theoretic analysis for UDA (Wang and Mao 2023). Our theoretical result reveals that PL-Mix improves the generalization guarantee by diminishing or controlling each component within the generalization bound. Particularly, PL-Mix generates more reliable data, aligns the marginal distributions of the two domains more effectively, and better controls the discrepancies between conditional label distributions. As a result, these contribute to the reduction of the first term, the second term, and the last two terms in the bound (i.e. Eqn. 16), respectively.

In a nutshell, we conclude the contributions as follows:

- We propose a pseudo label guided Mixup method to improve the existing adversarial prompt tuning framework for UDA. Particularly, to mitigate the impact of noisy pseudo labels, we design a novel strategy for adjusting the Mixup ratio via the pseudo label confidence.
- We present a generalization bound for UDA. Notably, we illustrate how our PL-Mix can improve the generalization guarantee by reducing or more effectively controlling each term within the given generalization bound.
- Through extensive experimental comparisons with state-of-the-art (SOTA) models in both single-source and multi-source domain adaptation settings, as well as various other Mixup variants, we empirically validate the effectiveness of PL-Mix.

Related Work

Unsupervised Domain Adaptation In UDA, representative methods optimize domain discrepancy using an adversarial training scheme, which involves learning label-discriminative features through a label classifier and domain-invariant features via a domain discriminator (Ganin and Lempitsky 2015). Various works (Qu et al. 2019; Du et al. 2020; Zhao et al. 2018; Zhang et al. 2019), including AdSPT (Wu and Shi 2022), achieve significant empirical and theoretical advancements using this approach. Additionally, COBE (Luo et al. 2022) explores contrastive learning with an in-batch negative method for cross-domain tasks, yielding improved generalization.

Mixup Mixup (Zhang et al. 2017) is an efficient data augmentation method, creating convex combinations by linearly interpolating input samples and their labels. DM-ADA (Xu et al. 2020) introduces domain discriminator mixup, ensuring a continuous domain-invariant latent space with intermediate statuses between source and target domains. DMRL (Wu, Inkpen, and El-Roby 2020) and IIMT (Yan et al. 2020) not only enriches the intrinsic structures by domain mixup regularization but also guides the better label classifier in enhancing consistent predictions of in-between samples. VMT (Mao et al. 2019) incorporates the locally-Lipschitz constraint to in-between samples and constructs the combination sample via virtual labels.

UDA via Adversarial Prompt Tuning

Problem Formulation Let $\mathcal{S} = \{x_i^s, y_i^s\}_{i=1}^{N^s}$ be labeled source domain dataset. In the context of text learning, $x_i^s = [w_1^s, w_2^s, ..., w_m^s]$ represents the input sentence with m words, y_i^s is the corresponding label of x_i^s, and N^s is the number of source domain data. Additionally, let $\mathcal{T} = \{x_i^t\}_{i=1}^{N^t}$ be the unlabeled target domain dataset, where each $x_i^t = [w_1^t, w_2^t, ..., w_m^t]$ is the unlabeled input sentence, and N^t is the number of target domain data.

The objective of UDA is to predict the label of unseen target domain data using the knowledge from the source domain. Both source domain and target domain share the same label set, namely $\mathcal{Y}$. Furthermore, each label of source domain data can be represented as a one-hot vector $\mathbf{Y}_i^s \in \mathbb{R}^{|\mathcal{Y}|}$.

Prompt Tuning Classifier By adding extra soft prompt representations, prompt tuning has shown its capability to improve the robustness under domain shifts (Lester, Al-Rfou, and Constant 2021; Brown et al. 2020). Moreover, a recent empirical study of Wu and Shi (2022) also demonstrates the effectiveness of prompt tuning for UDA.

Given an input sentence x, we first add n soft prompt tokens and a [mask] token at the beginning of x. Formally, the prompt function T is defined as follows:

$$T(x) = [\mathbf{S}_1, ..., \mathbf{S}_n, \mathbf{E}([\text{mask}]), \mathbf{E}(w_1), ..., \mathbf{E}(w_m)], \quad (1)$$

where $\mathbf{E}$ represents the embedding layer of pre-trained language model (PLM), denoted as M_θ, and $\mathbf{S}$ is the soft prompt token embedding. Let H be the dimension of hidden state of M_θ, and let $|\mathcal{V}|$ be the size of vocabulary set $\mathcal{V}$

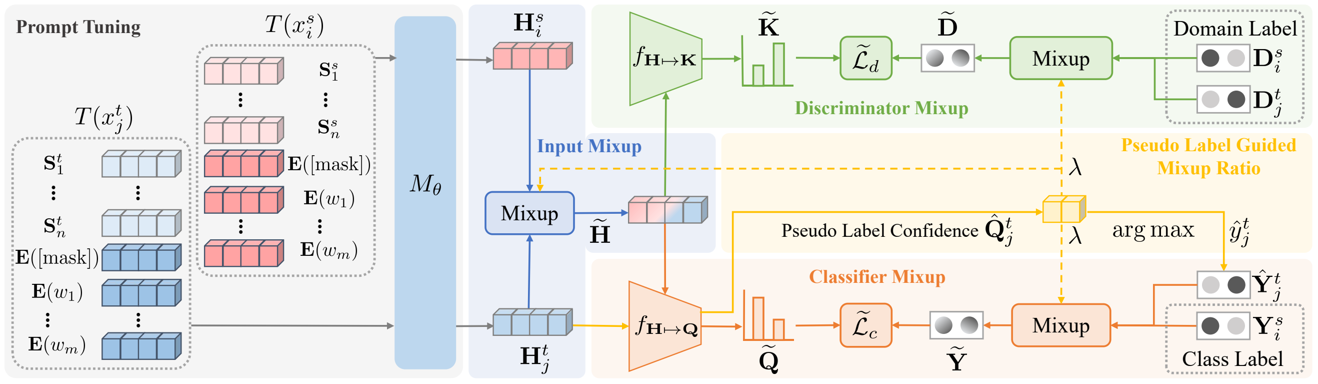

Figure 2: The illustration of PL-Mix. Icons framed by dotted lines are the input of the model. Better viewed in color.

in M_θ. Then, $T(x)$ is fed to the masked language model to obtain the hidden states representation $\mathbf{H} \in \mathbb{R}^H$. In the end, a pre-trained classification head $f_{\mathbf{H}\mapsto\mathbf{P}}$ generates the word occurrence probability $\mathbf{P} \in \mathbb{R}^{|\mathcal{V}|}$ of [mask] token based on $\mathbf{H}$. That is,

$$\mathbf{H} = M_\theta(T(x)), \qquad \mathbf{P} = f_{\mathbf{H}\mapsto\mathbf{P}}(\mathbf{H}). \tag{2}$$

Furthermore, the label words of label y is manually defined as $\mathcal{V}_y \in \mathcal{V}$. Let $\mathbf{Q} \in \mathbb{R}^{|\mathcal{Y}|}$ be the classification probability vector, and for each y, we let

$$\mathbf{Q}^{[y]} = \sum_{v\in\mathcal{V}_y} \mathbf{P}^{[v]}, \tag{3}$$

where $[\cdot]$ is the element selection function. Elements in $\mathbf{Q}$ represent the relative probability of labels, consisting of the label words occurrence probabilities at the [mask] slot. We then obtain the normalized $\mathbf{Q}$ as:

$$\mathbf{Q}^{[y]} := \frac{\exp \mathbf{Q}^{[y]}}{\sum_{y'\in\mathcal{Y}} \exp \mathbf{Q}^{[y']}}. \tag{4}$$

Since the label words engineering is beyond the scope of this study, we simply define the mapping from $\mathbf{H}$ to $\mathbf{Q}$ as $f_{\mathbf{H}\mapsto\mathbf{Q}}$ that has the same parameters with $f_{\mathbf{H}\mapsto\mathbf{P}}$. In UDA, we use source domain data $\mathcal{S}$ to train the classifier $f_{\mathbf{H}\mapsto\mathbf{Q}}$, and the loss of classifier $\mathcal{L}_c$ is defined as:

$$\mathcal{L}_c(\mathcal{S}; M_\theta, f_{\mathbf{H}\mapsto\mathbf{Q}}, \mathbf{S}) = -\sum_{i=1}^{N^s}\sum_{j=1}^{|\mathcal{Y}|} \mathbf{Y}_i^{[j]} \log \mathbf{Q}_i^{[j]}. \tag{5}$$

Adversarial Domain Discriminator In order to obtain domain-invariant features and compel source data close to target data (in the representation space), previous works (Wu and Shi 2022; Du et al. 2020; Ganin and Lempitsky 2015) adopt an adversarial training strategy for a discriminator.

Formally, consider a binary classification task with the domain label set $\mathcal{D} = \{0, 1\}$. Let $\mathcal{S}_d = \{x_i^s, 0\}_{i=1}^{N^s}$ and $\mathcal{T}_d = \{x_i^t, 1\}_{i=1}^{N^t}$ be the source domain data and target domain data, respectively, along with their corresponding domain labels. Let $\mathbf{D} \in \mathbb{R}^{|\mathcal{D}|}$ be the one-hot domain label vector. Then, a discriminator is represented by a multi-layer perception $f_{\mathbf{H}\mapsto\mathbf{K}}$ with ReLU activation layers. The discriminator predicts the domain label of the data, defined as follows:

$$\mathbf{K} = f_{\mathbf{H}\mapsto\mathbf{K}}(\mathbf{H}), \tag{6}$$

where $\mathbf{K}$ is a $|\mathcal{D}|$-dimensional vector corresponding to $\mathbf{D}$. Notice that training the discriminator is a supervised learning problem since both source data and target data have their ground truth domain labels. Hence, the loss of the discriminator is defined as:

$$\mathcal{L}_d(\mathcal{S}_d, \mathcal{T}_d; M_\theta, f_{\mathbf{H}\mapsto\mathbf{K}}, \mathbf{S}) = -\sum_{i=1}^{N^s+N^t}\sum_{j=1}^{|\mathcal{D}|} \mathbf{D}_i^{[j]} \log \mathbf{K}_i^{[j]}. \tag{7}$$

To recap, the adversarial training scheme in UDA involves the minimization of the label classifier loss $\mathcal{L}_c$ and the maximization of the (optimal) domain discriminator loss $\mathcal{L}_d$. The former fosters the acquisition of discriminative features, while the latter facilitates the emergence of domain-invariant features. Thus, the overall training objective $\mathcal{L}$ is:

$$\begin{aligned}\mathcal{L} = &\min_{M_\theta, f_{\mathbf{H}\mapsto\mathbf{Q}}, \mathbf{S}} \mathcal{L}_c(\mathcal{S}; M_\theta, f_{\mathbf{H}\mapsto\mathbf{Q}}, \mathbf{S}) \\ &+ \beta \max_{M_\theta, \mathbf{S}} \min_{f_{\mathbf{H}\mapsto\mathbf{K}}} \mathcal{L}_d(\mathcal{S}_d, \mathcal{T}_d; M_\theta, f_{\mathbf{H}\mapsto\mathbf{K}}, \mathbf{S}),\end{aligned} \tag{8}$$

where β is a trade-off parameter.

Pseudo Label Guided Mixup

As explained in the Introduction and depicted in Figure 1(a), the existing adversarial prompt tuning framework focuses solely on aligning source and target data, disregarding label compatibility. Given that the classifier is exclusively trained on labeled source data, this class-independent alignment can negatively impact performance in the target domain. Hence, we present PL-Mix as a solution to this issue.

In the following, we will first introduce our novel adjustment method of Mixup ratio based on pseudo label confidence. Then, we apply such a method in label classifier training (referred to as classifier Mixup), and we will apply the vanilla Mixup technique in domain discriminator training (referred to as discriminator Mixup).

Confidence-dependent Mixup Ratio In the context of UDA, where access to the true labels of target domain data is unavailable, basically, all previous works (Wu, Inkpen, and El-Roby 2020; Yan et al. 2020; Mao et al. 2019) employ pseudo labels of target data for classifier Mixup. Under the notation system of this paper, the predicted label distribution vector $\hat{\mathbf{Q}}^t_j$ (i.e. the confidence vector) of target data x^t_j and its corresponding pseudo label $\hat{y}^t_j$ are defined as follows:

$$\hat{\mathbf{Q}}^t_j = f_{\mathbf{H}\mapsto\mathbf{Q}}(M_\theta(T(x^t_j))), \qquad \hat{y}^t_j = \arg\max(\hat{\mathbf{Q}}^t_j). \quad (9)$$

Moreover, let $\hat{\mathbf{Y}}^t_j$ be the corresponding one-hot label vector of $\hat{y}^t_j$. The vanilla Mixup interpolates $\hat{\mathbf{Y}}^t_j$ and $\mathbf{Y}^s_j$ by a Mixup ratio $\lambda \sim Beta(\alpha, \alpha)$, where $Beta(\alpha, \alpha)$ is a Beta Distribution. Note that λ represents the proportion of the movement from x^s_i to x^t_j.

Notably, pseudo labels inevitably bring noise and may degrade the performance of the label classifier. In order to alleviate the not-always-favorable influence of pseudo label, we propose a pseudo label guided Mixup method to adjust the Mixup ratio of source and target data. Intuitively, source data should shift 'more' toward target data with the same label, while it should shift 'less' toward target data with different labels. These degrees of 'more' or 'less' are controlled by the confidence of the pseudo labels. More precisely, we will manipulate the Beta Distribution parameter α using the pseudo label confidence $\hat{\mathbf{Q}}^{t[\hat{y}^t_j]}_j$, thereby governing the Mixup ratio λ, as formalized below:

$$\alpha = \begin{cases} \hat{\mathbf{Q}}^{t[\hat{y}^t_j]}_j, & \text{if} \quad \hat{y}^t_j = y^s_i, \\ 1 - \hat{\mathbf{Q}}^{t[\hat{y}^t_j]}_j, & \text{otherwise.} \end{cases} \quad (10)$$

Given that high confidence pseudo labels are assumed to contain less noise, this strategy suggests that the higher the confidence of a pseudo label that shares the same source data label, the more the source data shifts toward the target data. Conversely, when the confidence is higher for differing labels between the source and target data, fewer movements will be applied. By this means, the data in the source domain moves directionally toward the target data with matching labels. Ultimately, this class-dependent interpolation enables the classifier to separate the target data as well.

Meanwhile, to further mitigate the impact of pseudo labels with lower confidence, which contains considerable noise, we introduce a threshold τ to determine whether to apply Mixup, namely

$$\lambda := \begin{cases} 1, & \text{if} \quad \hat{\mathbf{Q}}^{t[\hat{y}^t_j]}_j < \tau, \\ \max(1-\lambda, \lambda), & \text{otherwise.} \end{cases} \quad (11)$$

It is important to note that, source data with ground truth labels should make up the majority of Mixup synthetic data, leading to higher λ than $1 - \lambda$. Therefore, we adjust λ to follow a symmetric distribution around 0.5.

Classifier Mixup Collaborating with the confidence-dependent Mixup ratio, classifier Mixup can be formulated as input mixup and class label mixup, defined as:

$$\begin{aligned} \widetilde{\mathbf{H}} &= \lambda \mathbf{H}^s_i + (1-\lambda)\mathbf{H}^t_j, \\ \widetilde{\mathbf{Y}} &= \lambda \mathbf{Y}^s_i + (1-\lambda)\hat{\mathbf{Y}}^t_j. \end{aligned} \quad (12)$$

Eventually, the Mixup loss for the classifier, $\widetilde{\mathcal{L}}_c$, is calculated by the synthetic label $\widetilde{\mathbf{Y}}$ and its predicted soft label $\widetilde{\mathbf{Q}} = f_{\mathbf{H}\mapsto\mathbf{Q}}(\widetilde{\mathbf{H}})$:

$$\widetilde{\mathcal{L}}_c(\mathcal{S}; M_\theta, f_{\mathbf{H}\mapsto\mathbf{Q}}, \mathbf{S}) = -\sum_{i=1}^{N^s+N^t}\sum_{j=1}^{|\mathcal{Y}|} \widetilde{\mathbf{Y}}^{[j]}_i \log \widetilde{\mathbf{Q}}^{[j]}_i. \quad (13)$$

Discriminator Mixup Since both source data and target data have ground truth domain labels, we apply the vanilla Mixup method and define the discriminator Mixup as:

$$\begin{aligned} \widetilde{\mathbf{H}} &= \lambda \mathbf{H}^s_i + (1-\lambda)\mathbf{H}^t_j, \\ \widetilde{\mathbf{D}} &= \lambda \mathbf{D}^s_i + (1-\lambda)\mathbf{D}^t_j, \end{aligned} \quad (14)$$

where λ remains consistent with the classifier Mixup.

Similarly, the loss of discriminator Mixup is:

$$\widetilde{\mathcal{L}}_d(\mathcal{S}, \mathcal{T}; M_\theta, f_{\mathbf{H}\mapsto\mathbf{K}}, \mathbf{S}) = -\sum_{i=1}^{N^s+N^t}\sum_{j=1}^{|\mathcal{D}|} \widetilde{\mathbf{D}}^{[j]}_i \log \widetilde{\mathbf{K}}^{[j]}_i, \quad (15)$$

where $\widetilde{\mathbf{K}} = f_{\mathbf{H}\mapsto\mathbf{K}}(\widetilde{\mathbf{H}})$.

Accordingly, the overall loss of PL-Mix is the weighted sum of $\widetilde{\mathcal{L}}_c$ and $\widetilde{\mathcal{L}}_d$ as in Eqn. 8.

PL-Mix Improves Generalization

Inspired by the work of Wang and Mao (2023), we now provide a theoretical analysis to illustrate how PL-Mix contributes to enhancing the performance of UDA.

To this end, we introduce some additional notations. Let $P^s_{XY} = P^s_X P^s_{Y|X}$ and $P^t_{XY} = P^t_X P^t_{Y|X}$ be the source domain distribution and the target domain distribution, respectively. Then, the source data $\mathcal{S} = \{(x^s_i, y^s_i)\}_{i=1}^{N^s} \overset{\text{iid}}{\sim} P^s_{XY}$ and the unlabelled target data $\mathcal{T} = \{x^t_i\}_{i=1}^{N^t} \overset{\text{iid}}{\sim} P^t_X$. Let the function F be the composition $f_{\mathbf{H}\mapsto\mathbf{Q}} \circ M_\theta \circ T$. With a little abuse of the notation, we define the empirical risk of the labelled source data as $\hat{L}(F, \mathcal{S}) := \frac{1}{n}\sum_{i=1}^n \mathcal{L}_c(x^s_i, y^s_i; F)$, and define the population risk of the target domain as $L(F, P^t_{XY}) := \mathbb{E}_{P^t_{XY}}[\mathcal{L}_c(x^t, y^t; F)]$. Thus, the generalization error is defined as $\mathcal{E}(F) := L(F, P^t_{XY}) - \hat{L}(F, \mathcal{S})$. Notice that the generalization error is defined in a different way from Wang and Mao (2023). For the sake of simplicity, we let $z = M_\theta(T(x))$ be the representation of x. In this case, for any given F, the marginal distribution P^s_Z (or P^t_Z) is obtained by pushing forward P^s_X (or P^t_X resp.) via $M_\theta \circ T$.

We are now in a position to give the generalization bound, and the proof is deferred to Appendix.

Theorem 1. *Let the function space of F have the finite Natarajan dimension d_N (Daniely et al. 2011). Assume that the loss function $\mathcal{L}_c(\cdot, \cdot; F)$ is R-subgaussian*[1] *under P^s_{XY}. Then, for any F, there exists a constant $C > 0$ such that*

[1]A random variable X is R-subgaussian if for any ρ, $\log \mathbb{E}\exp(\rho(X - \mathbb{E}X)) \le \rho^2 R^2/2$.

with probability $1-\delta$

$$\mathcal{E}(F) \leq C\sqrt{\frac{d_N \log|\mathcal{Y}| + \log\frac{1}{\delta}}{N_s}} + \sqrt{2R^2 \mathrm{D}_{\mathrm{KL}}\left(P_Z^t || P_Z^s\right)} + \sqrt{2R^2 \mathrm{D}_{\mathrm{KL}}\left(P_{Y|Z}^t || P_{\hat{Y}|Z}^t\right)} + \sqrt{2R^2 \mathrm{D}_{\mathrm{KL}}\left(P_{\hat{Y}|Z}^t || P_{Y|Z}^s\right)}, \tag{16}$$

where $\mathrm{D}_{\mathrm{KL}}(\cdot||\cdot)$ *denotes the KL divergence and* $P_{\hat{Y}|Z}^t$ *is the conditional pseudo label distribution of the target data.*

Remark 1. *We remark that the subgaussian assumption is not strong. For example, a bounded loss is guaranteed to be subgaussian. Although the cross-entropy loss in general is not bounded, by using the prevalent training methodologies, one can typically ensure that the loss curve consistently descends until it reaches convergence, effectively bounding the loss values during training by the initial value.*

Subsequently, we will show that PL-Mix can improve the generalization guarantee presented in Theorem 1.

Classifier Mixup Reduces the First Term By using the Mixup technique, we will reduce the first term because Mixup generates additional data, effectively increasing N_s, while preserving the model capacity d_N.

Discriminator Mixup Reduces the Second Term Within the adversarial promt tuning framework, the discriminator is trained with the goal of minimizing the discrepancy between P_Z^t and P_Z^s in the sense of Jensen-Shannon divergence (JSD) (Goodfellow et al. 2014, Theorem 1), and the global minimum is achieved when $P_Z^t = P_Z^s$, in which case $\mathrm{D}_{\mathrm{KL}}\left(P_Z^t || P_Z^s\right) = 0$. Additionally, Mixup plays a significant role in enabling the discriminator to find the optimal decision boundary for this binary classification task. In other words, Mixup aids in finding the optimal discriminator. The ability of Mixup in helping binary classification is theoretically justified in the recent work of Oh and Yun (2023). In addition, it is worth noting that prompt tuning has also been recognized to provide certain advantages in minimizing the discrepancy between the marginal distributions of two domains (Wu and Shi 2022). Nonetheless, theoretically understanding the impact of prompt tuning remains challenging and falls beyond the scope of this paper.

Classifier Mixup Controls the Last Two Terms The pseudo label distribution $P_{\hat{Y}|Z}^t$ is entirely induced by the classifier $f_{\mathbf{H}\mapsto\mathbf{Q}}$. Note that the previous adversarial prompt tuning (Wu and Shi 2022) does not align the conditional label distributions as given in the last two terms of Eqn. 16. In fact, the failure to control the last two terms is the fundamental reason for the issue given in Figure 1(a). To see this, note that without PL-Mix, $f_{\mathbf{H}\mapsto\mathbf{Q}}$ is solely trained on source domain data with hard labels. Then, an erroneous pseudo label can lead to unbounded growth of the third term in Eqn. 16. Concretely, for a given input x^t, it's possible that the probability of a certain label y is non-zero for target domain, denoted as $P_{Y|Z}^t(Y=y) \neq 0$, while the probability of the corresponding pseudo label is zero, i.e. $P_{\hat{Y}|Z}^t(\hat{Y}=y) = 0$, resulting in $\mathrm{D}_{\mathrm{KL}}\left(P_{Y|Z}^t || P_{\hat{Y}|Z}^t\right) \to \infty$. Therefore, overfitting to the source domain label distribution, i.e. $P_{Y|Z}^s$, will hurt the generalization in the target domain. This emphasizes the importance of class-dependent alignment.

When applying the PL-Mix, the classifier $f_{\mathbf{H}\mapsto\mathbf{Q}}$ is trained using soft supervision signals rather than one-hot vectors. Furthermore, by incorporating the Mixup ratio adjustment method proposed in Eqn. 10, the probability of generating synthetic target data with incorrect labels is significantly reduced. Specifically, when the pseudo label of the target data differs from that of the source data, PL-Mix directs the synthesized data to fall in the source domain instead of target domain. This ensures that the classifier is not trained using synthetic target data bearing incorrect labels. Consequently, PL-Mix results in a smoother and closer-to-optimal pseudo label distribution, which mitigates the unbounded issue of the third term in Eqn. 16. Note that the last term in Eqn. 16 will still be small when the classification error in Eqn. 13 is sufficiently minimized.

Experiment

Datasets We adopt Amazon Reviews (Blitzer, Dredze, and Pereira 2007) to evaluate PL-Mix. It contains four domains: Book (B), DVD (D), Electronics (E) and Kitchen (K). For each domain, there are 1000 positive and 1000 negative labeled reviews, and 4000 randomly chosen unlabeled data. Following the previous work (Wu and Shi 2022; Luo et al. 2022), we cross-combined data from the source and target domains to construct 12 tasks for model evaluation.

Implementation Details We exploit the pre-trained model Bert-base-uncased and Roberta-base from Hugging Face Transformers [2] to evaluate PL-Mix and comparable methods. We set the batch size as 8 and run 10 epochs for model training. And we use Adam (Kingma and Ba 2014) to be the optimizer with learning rate 1e-5 for $\widetilde{\mathcal{L}}_c$, 5e-5 for $\widetilde{\mathcal{L}}_d$, and $\beta \in$[0.001, 0.1]. For soft prompt token, we randomly initialize the embedding and choose n from [1,5] as (Wu and Shi 2022) recommended. To speed up the model convergence, we first train the model without Mixup for 5 epochs and then train the model with Mixup for the remaining 5 epochs subsequently. And the final results are tested on the best checkpoint of all 10 epochs, which is determined by a valid set containing 20% samples leaving from labeled data. All experiments are completed on Tesla V100-SXM2-32GB GPU, costing about 1200s for training and testing on the Roberta-base encoder. And we run our experiments five times using different random seeds on all models and report the average accuracy for each task.

Baselines Several models (Du et al. 2020; Li et al. 2018; Zhou et al. 2020) are proposed to solve the UDA problem and achieve certain results. In order to verify the effect of PL-Mix more widely while saving space simultaneously, we mainly choose the classical adversarial model **DANN** (Ganin and Lempitsky 2015), and **COBE** (Luo et al. 2022) and **AdSPT** (Wu and Shi 2022) which achieve SOTA

[2]https://github.com/huggingface/transformers

Bert-base-uncased													
Model	B → D	B → E	B → K	D → B	D → E	D → K	E → B	E → D	E → K	K → B	K → D	K → E	Avg.
DANN	89.70	87.30	89.55	89.55	86.05	87.69	87.15	86.05	91.91	87.65	87.72	86.05	88.56
COBE	90.05	90.45	**92.90**	90.98	90.67	**92.00**	87.90	87.87	93.33	88.38	87.43	92.58	90.39
AdSPT	-	-	-	-	-	-	-	-	-	-	-	-	-
DANN	89.54	88.15	89.76	89.62	88.27	89.87	87.89	88.19	92.25	87.69	87.72	91.14	89.17
COBE*	90.13	90.92	92.28	91.05	89.75	91.67	88.25	**88.88**	93.88	89.18	87.68	**92.87**	90.55
AdSPT	90.10	90.55	92.25	90.55	89.40	90.95	88.35	87.40	93.75	88.45	87.80	92.00	90.13
PL-Mix	**90.91**	**91.04**	91.82	**91.19**	**91.12**	91.84	**88.86**	88.56	**93.93**	**89.25**	**88.27**	92.77	**90.80**

Roberta-base													
Model	B → D	B → E	B → K	D → B	D → E	D → K	E → B	E → D	E → K	K → B	K → D	K → E	Avg.
DANN	-	-	-	-	-	-	-	-	-	-	-	-	-
COBE	-	-	-	-	-	-	-	-	-	-	-	-	-
AdSPT*	92.00	93.75	93.10	92.15	94.00	93.25	92.70	**93.15**	94.75	92.35	**92.55**	93.95	93.14
DANN	91.79	92.60	93.12	92.60	91.58	93.30	90.48	90.27	94.24	91.40	90.15	93.85	92.11
COBE	92.19	92.79	95.02	93.27	93.24	94.47	92.01	90.00	95.31	91.70	90.14	94.63	92.90
AdSPT	92.86	93.08	94.45	93.97	93.16	94.97	91.75	89.72	95.43	91.33	90.76	**94.70**	93.02
PL-Mix	**93.60**	**94.22**	**95.36**	**94.19**	**94.11**	**95.29**	**92.77**	92.02	**95.67**	**92.50**	91.71	94.65	**93.84**

Table 1: Overall performance on Bert-base-uncased and Roberta-base. B → D represents source domain Book (B) to target domain DVD (D), and Avg. represents the average accuracy of the 12 subtasks. The first three lines of results are taken from the public paper, and the next three lines in italics are the results we reproduce. The model represented by superscript $\star$ is the current SOTA method. Bolded results are the best performance of each task.

Roberta-base													
Model	B → D	B → E	B → K	D → B	D → E	D → K	E → B	E → D	E → K	K → B	K → D	K → E	Avg.
AdSPT	92.86	93.08	94.45	93.97	93.16	94.97	91.75	89.72	95.43	91.33	90.76	94.70	93.02
DM-ADA	92.59	93.77	95.11	94.14	93.64	95.13	92.09	90.96	95.41	91.62	90.22	**94.74**	93.29
IIMT	93.43	93.55	95.22	93.89	**94.16**	94.78	92.45	90.85	95.59	91.79	90.61	94.49	93.40
DMRL	93.58	93.37	94.38	94.13	94.05	94.83	91.97	91.48	95.55	91.98	91.10	94.70	93.43
PL-Mix-D	93.20	93.63	95.14	94.13	93.71	95.04	91.73	90.96	95.38	92.43	90.94	94.65	93.41
PL-Mix-C	93.43	93.89	95.03	94.00	93.98	95.31	92.93	91.62	95.64	92.08	91.28	94.48	93.64
PL-Mix	**93.60**	**94.22**	**95.36**	**94.19**	94.11	**95.29**	**92.77**	**92.02**	**95.67**	**92.50**	**91.71**	94.65	**93.84**

Table 2: Mixup variants comparison in Roberta. Bolded results represent the best performance among DM-ADA, IIMT, DMRL and PL-Mix. Underlined results represent the best performance among PL-Mix-D, PL-Mix-C and PL-Mix.

results on Bert and Roberta respectively. For a fair comparison, we report the results of these models in their paper and reproduce them on both Bert and Roberta. And our code will be released in our Github [3].

Overall Performance The overall performance on Bert-base-uncased and Roberta-base is shown in Table 1. From the Table, we find that our proposed model, PL-Mix, outperforms other models on average (Avg.) both in Bert and Roberta. Especially in Roberta, PL-Mix archives a significant improvement compared with other models, including the SOTA model AdSPT. It is trivial that COBE achieves closer improvement to PL-Mix in Bert, as COBE possesses a gap with PL-Mix in Roberta. Overall, PL-Mix obtains convincing results and proves the effectiveness of the pseudo label guided Mixup on adversarial prompt tuning for UDA.

[3]https://github.com/fskong/PL-Mix

Mixup Variants Comparison Since PL-Mix is also a Mixup strategy, we compare it with other Mixup variants and report the results in Table 2. Although DM-ADA (Xu et al. 2020), IIMT (Yan et al. 2020) and DMRL (Wu, Inkpen, and El-Roby 2020) are models proposed for UDA in terms of image classification rather than text, we capture the core idea and re-implement these models based on the same adversarial prompt tuning framework. The main difference between these models is shown in Table 3. Combining with results reported in Table 2, we find that all Mixup variants perform better than AdSPT, indicating the effectiveness of Mixup. Among these variants, particularly in comparison with IIMT, PL-Mix achieves the most superior performance which demonstrates the efficacity of our PL strategy.

Ablation Study We also conduct ablation experiments of PL-Mix to examine the contribution of each module in PL-Mix. For space consideration, the results are also shown in

Model	DM-ADA	IIMT	DMRL	PL-Mix
Mix-D	✓	✓	✓	✓
Mix-C	✗	✓	✓	✓
Mix-Cross	✓	✓	✗	✓
PL	✗	✗	✗	✓

Table 3: Mixup variants comparison. Mix-D represents discriminator Mixup, Mix-C represents classifier Mixup, Mix-Cross represents mixing cross-domain data, and PL represents the pseudo label guided Mixup strategy we proposed.

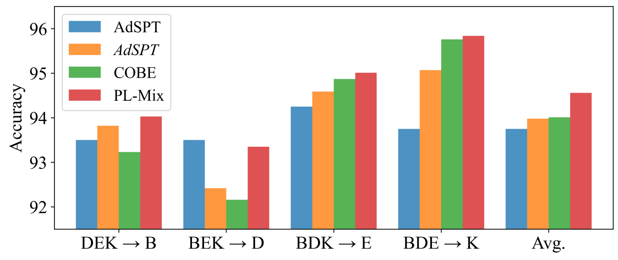

Figure 3: Performance of AdSPT, COBE and PL-Mix on multi-source domain adaption setting.

Table 2 in which PL-Mix-D represents PL-Mix only apply Mixup on discriminator and PL-Mix-C represents only Mixup classifier. From these results, we discover that each module contributes positively to the final results. If comparing PL-Mix-D with DM-ADA, which also only applies discriminator Mixup, we find PL-Mix-D wins slightly. It further indicates the efficiency of PL strategy.

Multi-source Domain Adaptation Except for comparing PL-Mix with other UDA models in single-source tasks, we evaluate PL-Mix with AdSPT and COBE in the multi-source domain adaptation setting. We follow the setup as in (Wu and Shi 2022) that leave a domain as the target and utilize the others as the multiple source domain. For instance, BDE → K represents that the source domain contains Book, DVD and Electronics, and the target domain is Kitchen.

The results of multi-source domain adaptation are shown in Figure 3. As illustrated in this Figure, the performance of accuracy in the target domain using multi-source data is overall better than single-source (reported in Table 1), which follows the expectation and is consistent with that in AdSPT paper. And it's worth noting that PL-Mix still outperforms AdSPT (including the results we re-implement which is better than AdSPT published) and COBE, which further proves the effectiveness and adaptability of PL-Mix.

Soft Prompts Analysis Prompt tuning (Lester, Al-Rfou, and Constant 2021) has demonstrated the resilience of domain shift by soft prompts. Here, we analyze the effect of soft prompt under PL-Mix as in AdSPT. More specially, we plot the variation of AdSPT and DANN, the best performance of AdSPT and PL-Mix, with the parameter n.

Similar to AdSPT paper, both AdSPT and PL-Mix first rise and then fall when $n \in [1, 5]$, and achieve the sweet point at $n = 2$. More specially, AdSPT is better than DANN

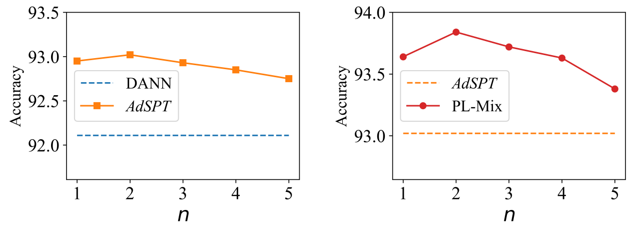

Figure 4: Results of AdSPT and PL-Mix in single-source setting via the different n using in Roberta.

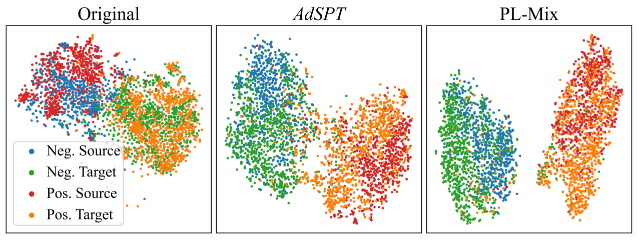

Figure 5: Visualization of sentence representations reducing dimensionality by t-SNE. Original represents the distribution before training, AdSPT and PL-Mix represent the distribution learned by AdSPT and PL-Mix separately.

and PL-Mix performs better than AdSPT on all n, which indicates the effectiveness of prompt tuning and pseudo label guided Mixup respectively.

Visualization of Data In order to analyze the effect of models on the alignment of data distribution, we utilize t-SNE (Van der Maaten and Hinton 2008) to visualize the sentence representations of positive and negative samples from the source and target domain. The visualization of sentence representations on K → B is shown in Figure 5.

As shown in Figure 5, both AdSPT and PL-Mix basically modulate source data aligning to target data according to their labels, which indicates adversarial prompt tuning framework indeed transfers knowledge from the source to target. Comparing AdSPT with PL-Mix, we find there are more samples across the classification boundary of AdSPT than PL-Mix. We believe that this is because of the pseudo label guided Mixup ratio mechanism, which explicitly and directionally aligns source data to target data and improves the performance naturally.

Conclusion

In this paper, we propose a pseudo label guided Mixup method on the adversarial prompt tuning framework, called PL-Mix, to explicitly and directionally align source data with target data and alleviate the noise delivered from the pseudo label. We first theoretically justify that our PL-Mix can improve the generalization guarantee for UDA. Then we empirically demonstrate the efficacy of PL-Mix, by comparing PL-Mix with SOTA methods on both single-source and multi-source settings, as well as against other popular Mixup variants.

Appendices

Proof of Theorem 1

To prove Theorem 1, similar to Wang and Mao (2023), we will invoke the the Donsker-Varadhan variational representation of KL divergence (Donsker and Varadhan 1983):

Lemma 1 (Donsker and Varadhan's variational formula)**.** *Let Q, P be probability measures on Θ, for any bounded measurable function $f : \Theta \to \mathbb{R}$, we have* $\mathrm{D_{KL}}(Q||P) = \sup_f \mathbb{E}_{\theta\sim Q}[f(\theta)] - \log \mathbb{E}_{\theta\sim P}[\exp f(\theta)]$.

We are now ready to prove Theorem 1.

Proof of Theorem 1.

$$\begin{aligned}\mathcal{E}(F) =& L(F, P^t_{XY}) - \hat{L}(F, \mathcal{S})\\ =& \underbrace{L(F, P^t_{XY}) - L(F, P^s_{XY})}_{\mathcal{E}_1} + \underbrace{L(F, P^s_{XY}) - \hat{L}(F, \mathcal{S})}_{\mathcal{E}_2},\end{aligned}$$

where $L(F, P^s_{XY}) \triangleq \mathbb{E}_{P^s_{XY}}[\mathcal{L}_c(x^s, y^s; F)]$ is the population risk of source domain.

Notice that the second gap term $\mathcal{E}_2$ is just the standard generalization gap in the supervised learning under i.i.d. assumption case. Hence, we invoke the Natarajan dimension based multi-class classification generalization bound (Daniely et al. 2011, Theorem 5). For any F, the following bound hold for some constant $C > 0$ with probability $1-\delta$,

$$L(F, P^s_{XY}) - \hat{L}(F, \mathcal{S}) \leq C\sqrt{\frac{d_N \log|\mathcal{Y}| + \log\frac{1}{\delta}}{N_s}}. \quad (17)$$

The remaining task is to bound the first term $\mathcal{E}_1$, which is the generalization gap between the population risk of target domain and the population risk of the source domain. We will apply Lemma 1.

Plugging $\theta = (x, y)$, $Q = P^t_{XY}$, $P = P^s_{XY}$ and $f(\cdot) = t\mathcal{L}_c(\cdot; F)$ for $t \in \mathbb{R}$ into Lemma 1, we have

$$\begin{aligned}\mathrm{D_{KL}}(P^t_{XY}||P^s_{XY}) \geq& \sup_t t \cdot \mathbb{E}_{P^t_{XY}}\left[\mathcal{L}_c(x^t, y^t; F)\right]\\ &- \log \mathbb{E}_{P^s_{XY}}\left[e^{t\cdot\mathcal{L}_c(x^s,y^s;F)}\right]. \quad (18)\end{aligned}$$

By the subgaussian assumption of $\mathcal{L}_c(\cdot; F)$, we have

$$\log \mathbb{E}_{P^s_{XY}}\left[e^{t\cdot\mathcal{L}_c(x^s,y^s;F)}\right] \leq t\cdot\mathbb{E}_{P^s_{XY}}[\mathcal{L}_c(x^s, y^s; F)] + \frac{t^2R^2}{2}.$$

Plugging the above into Eq. (18),

$$\begin{aligned}&\mathrm{D_{KL}}(P^t_{XY}||P^s_{XY})\\ \geq& \sup_t t\cdot\left[\mathbb{E}_{P^t_{XY}}\left[\mathcal{L}_c(x^t, y^t; F)\right] - \mathbb{E}_{P^s_{XY}}\left[\mathcal{L}_c(x^s, y^s; F)\right]\right]\\ &+ \frac{t^2R^2}{2}\\ =& \sup_t t\cdot\mathcal{E}_1 + \frac{t^2R^2}{2}.\end{aligned}$$

Then, we optimize the LHS above over t (or simply by AM-GM inequality), we have

$$\mathcal{E}_1 \leq \sqrt{2R^2 \mathrm{D_{KL}}(P^t_{XY}||P^s_{XY})}. \quad (19)$$

Furthermore,

$$\begin{aligned}\mathrm{D_{KL}}(P^t_{XY}||P^s_{XY}) =& \mathbb{E}_{P^t_{XY}} \log\frac{P^t_{XY}}{P^s_{XY}}\\ =& \mathbb{E}_{P^t_X}\mathbb{E}_{P^t_{Y|X}} \log\frac{P^t_{Y|X}}{P^s_{Y|X}} + \mathbb{E}_{P^t_X}\log\frac{P^t_X}{P^s_X}\\ =& \mathrm{D_{KL}}\Big(P^t_{Y|X}||P^s_{Y|X}\big|P^t_X\Big) + \mathrm{D_{KL}}\big(P^t_X||P^s_X\big).\end{aligned} \quad (20)$$

Combining Eq. (17), Eq. (19) and Eq. (20), we have

$$\begin{aligned}\mathcal{E}(F) =& \mathcal{E}_1 + \mathcal{E}_2\\ \leq& C\sqrt{\frac{d_N \log|\mathcal{Y}| + \log\frac{1}{\delta}}{N_s}}\\ &+ \sqrt{2R^2\left(\mathrm{D_{KL}}\Big(P^t_{Y|X}||P^s_{Y|X}\big|P^t_X\Big) + \mathrm{D_{KL}}(P^t_X||P^s_X)\right)}\\ \leq& C\sqrt{\frac{d_N \log|\mathcal{Y}| + \log\frac{1}{\delta}}{N_s}}\\ &+ \sqrt{2R^2\mathrm{D_{KL}}\Big(P^t_{Y|X}||P^s_{Y|X}\big|P^t_X\Big)} + \sqrt{2R^2\mathrm{D_{KL}}(P^t_X||P^s_X)},\end{aligned}$$

where the last inequality is by $\sqrt{\sum_{i=1}^n a_i} \leq \sum_{i=1}^n \sqrt{a_i}$.

Finally, since the bound above holds for any given F, then for a given $M_\theta \circ T$, the distribution $P^s_{M_\theta(T(X))}$ (or $P^t_{M_\theta(T(X))}$) is simply obtained by pushing forward P^s_X (or P^t_X resp.) via $M_\theta \circ T$. In this case, we have

$$\begin{aligned}\mathcal{E}(F) \leq& C\sqrt{\frac{d_N \log|\mathcal{Y}| + \log\frac{1}{\delta}}{N_s}} + \sqrt{2R^2\mathrm{D_{KL}}\left(P^t_Z||P^s_Z\right)}\\ &+ \sqrt{2R^2\mathrm{D_{KL}}\left(P^t_{Y|Z}||P^s_{Y|Z}\right)}\\ \leq& C\sqrt{\frac{d_N \log|\mathcal{Y}| + \log\frac{1}{\delta}}{N_s}} + \sqrt{2R^2\mathrm{D_{KL}}\left(P^t_Z||P^s_Z\right)}\\ &+ \sqrt{2R^2\mathrm{D_{KL}}\Big(P^t_{Y|Z}||P^t_{\hat{Y}|Z}\Big)} + \sqrt{2R^2\mathrm{D_{KL}}\Big(P^t_{\hat{Y}|Z}||P^s_{Y|Z}\Big)},\end{aligned}$$

where the last inequality is by

$$\mathrm{D_{KL}}\Big(P^t_{Y|Z}||P^s_{Y|Z}\Big) = \mathrm{D_{KL}}\Big(P^t_{Y|Z}||P^t_{\hat{Y}|Z}\Big) + \mathrm{D_{KL}}\Big(P^t_{\hat{Y}|Z}||P^s_{Y|Z}\Big)$$

and $\sqrt{\sum_{i=1}^n a_i} \leq \sum_{i=1}^n \sqrt{a_i}$.

This completes the proof. □

Acknowledgements

This work was supported by the STI 2030-Major Projects under Grant 2022ZD0120200, in part by the National Natural Science Foundation of China (No. U23B2056), in part by the Fundamental Research Funds for the Central Universities, and in part by the State Key Laboratory of Software Development Environment.

References

Blitzer, J.; Dredze, M.; and Pereira, F. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. *Meeting of the Association for Computational Linguistics,Meeting of the Association for Computational Linguistics*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Cai, Y.; and Wan, X. 2019. Multi-Domain Sentiment Classification Based on Domain-Aware Embedding and Attention. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*.

Daniely, A.; Sabato, S.; Ben-David, S.; and Shalev-Shwartz, S. 2011. Multiclass learnability and the erm principle. In *Proceedings of the 24th Annual Conference on Learning Theory*, 207–232.

Donsker, M. D.; and Varadhan, S. S. 1983. Asymptotic evaluation of certain Markov process expectations for large time. IV. *Communications on pure and applied mathematics*, 36(2): 183–212.

Du, C.; Sun, H.; Wang, J.; Qi, Q.; and Liao, J. 2020. Adversarial and Domain-Aware BERT for Cross-Domain Sentiment Analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 4019–4028.

Ganin, Y.; and Lempitsky, V. 2015. Unsupervised Domain Adaptation by Backpropagation. *International Conference on Machine Learning*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Li, Z.; Wei, Y.; Zhang, Y.; and Yang, Q. 2018. Hierarchical Attention Transfer Network for Cross-domain Sentiment Classification. *Proceedings of the AAAI Conference on Artificial Intelligence,Proceedings of the AAAI Conference on Artificial Intelligence*.

Luo, Y.; Guo, F.; Liu, Z.; and Zhang, Y. 2022. Mere Contrastive Learning for Cross-Domain Sentiment Analysis. In *Proceedings of the 29th International Conference on Computational Linguistics*.

Mao, X.; Ma, Y.; Yang, Z.; Chen, Y.; and Li, Q. 2019. Virtual Mixup Training for Unsupervised Domain Adaptation. *arXiv: Computer Vision and Pattern Recognition,arXiv: Computer Vision and Pattern Recognition*.

Oh, J.; and Yun, C. 2023. Provable Benefit of Mixup for Finding Optimal Decision Boundaries. In *International Conference on Machine Learning*, 26403–26450. PMLR.

Pan, S. J.; and Yang, Q. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 1345–1359.

Qu, X.; Zou, Z.; Cheng, Y.; Yang, Y.; and Zhou, P. 2019. Adversarial Category Alignment Network for Cross-domain Sentiment Classification. In *Proceedings of the 2019 Conference of the North*.

Sahoo, A.; Panda, R.; Feris, R.; Saenko, K.; and Das, A. 2023. Select, Label, and Mix: Learning Discriminative Invariant Feature Representations for Partial Domain Adaptation. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Wang, Z.; and Mao, Y. 2023. Information-Theoretic Analysis of Unsupervised Domain Adaptation. In *The Eleventh International Conference on Learning Representations*.

Wu, H.; and Shi, X. 2022. Adversarial Soft Prompt Tuning for Cross-Domain Sentiment Analysis. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2438–2447. Dublin, Ireland: Association for Computational Linguistics.

Wu, Y.; Inkpen, D.; and El-Roby, A. 2020. Dual mixup regularized learning for adversarial domain adaptation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, 540–555. Springer.

Xu, M.; Zhang, J.; Ni, B.; Li, T.; Wang, C.; Tian, Q.; and Zhang, W. 2020. Adversarial Domain Adaptation with Domain Mixup. *Proceedings of the AAAI Conference on Artificial Intelligence*, 6502–6509.

Yan, S.; Song, H.; Li, N.; Zou, L.; and Ren, L. 2020. Improve Unsupervised Domain Adaptation with Mixup Training. *arXiv: Machine Learning,arXiv: Machine Learning*.

Zhang, H.; Cisse, M.; Dauphin, Y.; and Lopez-Paz, D. 2017. mixup: Beyond Empirical Risk Minimization. *Cornell University - arXiv,Learning*.

Zhang, Y.; Liu, T.; Long, M.; and Jordan, M. 2019. Bridging theory and algorithm for domain adaptation. In *International conference on machine learning*, 7404–7413. PMLR.

Zhao, H.; Zhang, S.; Wu, G.; Moura, J. M. F.; Costeira, J. P.; and Gordon, G. J. 2018. Adversarial Multiple Source Domain Adaptation. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Zhou, J.; Tian, J.; Wang, R.; Wu, Y.; Xiao, W.; and He, L. 2020. SentiX: A Sentiment-Aware Pre-Trained Model for Cross-Domain Sentiment Analysis. In *Proceedings of thes International Conference on Computational Linguistics*.