Reply to Reviewer#1

1. How does the domain discriminator G_d work? G_d should be explained when it first appears.

Response:

Thanks for your comments and suggestions. The proposed *Thresholded Domain Adversarial Network (**ThDAN**)* is built upon *Domain Adversarial Neural Network (**DANN**)* [1]. DANN is a model proposed for close set domain adaptation to align distributions of the source and target domain by performing domain adversarial training. The domain adversarial training is a two-player minmax game: a domain discriminator G_d as the first player aims to separate the feature representation of the source domain from the target domain, at the same time, a feature generator G_f as the second player is trained to deceive the domain discriminator. Formally, the domain adversarial training can be written as:

$$\begin{aligned}\min_{G_f}\max_{G_d}\mathcal{L}(G_f,G_d) = &\mathbb{E}_{x\sim p_t(x)}\left[\log\left(G_d\left(G_f\left(x\right)\right)\right)\right] \\ &+\mathbb{E}_{x\sim p_s(x)}\left[\log\left(1-G_d\left(G_f\left(x\right)\right)\right)\right].\end{aligned} \tag{1}$$

where G_f is a feature extractor for samples, and G_d is a binary domain classifier with all the source samples labelled as 0 and all the target samples labelled as 1.

To clarify how the domain discriminator G_d and DANN works, we revise Section 3 by adding the introduction of DANN as the preliminary of our work. Please refer to Section 3.1 in the revised manuscript for details.

2. In Eq.(2), p_s and p_t are probability distributions, so what do $p_s(z_t)$ and $p_t(z_t)$ mean? And why is the optimal G_d this?

Response:

Thanks for your precious comments. Eq.(2) in the original paper is as follows,

$$G_d^*(z_t) = \frac{p_s(z_t)}{p_s(z_t)+p_t(z_t)}. \tag{2}$$

Here p_s is probability distribution of source domain, and p_t is that of target domain. z is deep represent of samples in feature space, *i.e.,* $z = G_f(x)$. Therefore $p_s(z)$ is probability of source samples in feature space and $p_t(z)$ is that of target samples. The subscript t in the Eq.(2) is used to indicate the features extracted from samples of the target domain. Writing like this can be confusing, so we rewrite Eq.2 in the revised manuscript as:

$$G_d^*(z) = \frac{p_s(z)}{p_s(z)+p_t(z)}. \tag{3}$$

Eq.(3) works for both the source and target samples. And it gives the optimal G_d for Eq.(1). We give the proof as follows,

Proof. For any G_f, we train G_d to maximize Eq.(1):

$$\begin{aligned}\max_{G_d}\mathcal{L}(G_f,G_d) &= \int_x p_s(x)\log\left(G_d\left(G_f\left(x\right)\right)\right)+p_t(x)\log\left(1-G_d\left(G_f\left(x\right)\right)\right)\,dx. \\ &= \int_x p_s(x)\log\left(G_d\left(z\right)\right)+p_t(x)\log\left(1-G_d\left(z\right)\right)\,dx.\end{aligned} \tag{4}$$

For any $(a, b) \in \mathbb{R}^2 \backslash \{0, 0\}$, the function $y \to a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. Therefore the optimal G_d is Eq.(1). □

The proof has been added in Section 3.1 of the revised manuscript.

3. In Eq.(4) and Eq.(5), authors use the entropy of probabilities of known classes to measure the probability that the sample comes from a known class. Why do not directly use the output of G_c? The $K + 1$ dimension of G_c (i.e. G_c^{K+1})) seems like the same probability.

Response:

Thanks for your precious comments. We use superscript k to denote $k'th$ dimensional output of G_c. Then $G_d = G_c^{k+1}$ gives probabilities of being the target samples. And $G_{c,known}$ gives probability distributions among the known classes:

$$G_{c,\,known} = softmax([G_c^1, G_c^2, ..., G_c^K]). \tag{5}$$

Eq.(4) and Eq.(5) of the original paper are respectively as follows,

$$w_d(x) = 1 - G_d^*(z). \tag{6}$$
$$w_c(x) = 1 - H(G_{c,\,known}(z)). \tag{7}$$

Both of these equations are transferability criterion that we proposed to select target samples from the known class for domain adversarial training. The statement "use the entropy of probabilities of known classes to measure the probability that the sample comes from a known class" in the original paper could be confusing. So we eger to re-explained Eq.(6) and Eq.(7) for clearer understanding.

Eq.(6) is motivated by the fact that **target samples from the known classes are able to confuse G_d.** Considering network G_d has converged to the optimal value for the current feature extractor, the output value of it delivers the likelihood of the sample from the target domain. For a target sample, if $G_d^*(z)$ approaches to 1, then the sample has a high probability of coming from the unknown classes. That is because the unknown class only included in the target domain and can be almost perfectly discriminated from the source samples. On the other hand, if $G_d^*(z)$ approaches to 0, then the sample is more likely from the known classes that shared by domains. Therefore, transferable target samples are able to confuse G_d to label them as the source samples. Then we can define the transferability $w_d(x)$ as inversely related to $G_d^*(z)$ as Eq.(6)

Eq.(7) is motivated by the fact that **target samples from the known classes can be categorized by $G_{c,known}$.** As *entropy* measures the degree to which the probability of the model is spread out over different possible states, it can be used to measure the uncertainty of predictions. Due to the overlapping in the marginal distributions, the target samples from the known classes should be categorized by the classifier G_c that trained on the source samples, leading to low entropy. And for samples from the unknown classes, because they cannot be aligned with a specific class from the source domain, the prediction tends to be uncertain over K known categories, leading to high entropy. Therefore we can define the transferability $w_c(x)$ as inversely related to the classification entropy as Eq.(7).

Since $G_{c,known}$ and G_d gives different probabilities and used to measure transferability from different aspects, they are not replaceable to each other. The above explanation has been added in the revised manuscript accordingly. Please refer to Section 3.3 in the revised manuscript for details.

4. In Eq.(6), why is the probability of being a certain known class and the probability of being the unknown classes independent?

Response:

Thanks for your precious comments. This question is related to the previous question. Eq.(6) of the original paper is as follows,

$$w(x) = 1 - G_d(z) \cdot H(G_{c,\,known}(z)). \tag{8}$$

Here $H(G_{c,\,known}(z))$ is used to estimate the probability of being a certain known class, and $G_d(z)$ gives the probability of being the target samples.

In domain adaptation tasks, the classifier that leverages domain-invariant features gives the probability of being a certain known class. The domain discriminator that leverages domain-relevant features gives the probability of from a certain domain. Since the classifier and domain discriminator work independently, and domain-invariant and domain-relevant can be well separated from each other [2], these probabilities can be considered as independent in domain adaptation tasks.

In this work, we do not directly give the probability of being the unknown class. Instead, since samples from the unknown class are usually untransferable and are prone to be labeled as the target sample, we use the probability of being the target domain (*i.e.,* the output of $G_c^{(}K + 1)$) to approximate the probability of being the unknown class. Thus, the probability of being a certain known class and the probability of being the unknown classes can be considered independent in our work.

5. In Fig.3, $G_f s$ for the source domain and target domain share weights. However, samples from target domain should be transferred, so are there some differences between these two G_fs? Or are there some extra structures after G_f for target domain?

Response:

Thanks for your precious comments. There is no extra structures after G_f for target domain The proposed ThDAN adopts domain adversarial training (Eq.(1)) to transfer knowledge between domains. Specifically, we train a feature generator G_f to extracted embedding features, and a classifier G_c to leverage feature extracted by G_f to correctly label source samples. Then we train G_f with adversarial leaning to generate domain-invariant features, so the classifier trained on the source domain can perform well for the target domain. Since G_f can generate domain-invariant features for both the source and targe domain, there is no need use separate $G_f s$ for two domains or to add extra structures for the target domain.

6. The writing could be improved, especially for some confusing description. Some typos and grammar errors should be corrected, such as "a certain known classes".

Response:

Thanks for pointing out the mistake. As suggested, the paper has been proofread and the language errors have been corrected in the revised manuscript.

References

[1] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, V. Lempitsky, Domain-adversarial training of neural networks, The Journal of Machine Learning Research 17 (1) (2016) 2096–2030.

[2] X. Peng, Z. Huang, X. Sun, K. Saenko, Domain agnostic learning with disentangled representations, arXiv preprint arXiv:1904.12347 (2019).