Learning to Match Distributions for Domain Adaptation

Chaohui Yu[1], Jindong Wang[2]*, Chang Liu[2], Tao Qin[2],
Renjun Xu[3], Wenjie Feng[1], Yiqiang Chen[1], Tie-Yan Liu[2]
[1]ICT, CAS [2]Microsoft Research [3]Zhejiang University

Abstract

When the training and test data are from different distributions, domain adaptation is needed to reduce dataset bias to improve the model's generalization ability. Since it is difficult to directly match the cross-domain joint distributions, existing methods tend to reduce the marginal or conditional distribution divergence using predefined distances such as MMD and adversarial-based discrepancies. However, it remains challenging to determine which method is suitable for a given application since they are built with certain priors or bias. Thus they may fail to uncover the underlying relationship between transferable features and joint distributions. This paper proposes Learning to Match (L2M) to automatically learn the cross-domain distribution matching without relying on hand-crafted priors on the matching loss. Instead, L2M reduces the inductive bias by using a meta-network to learn the distribution matching loss in a data-driven way. L2M is a general framework that unifies task-independent and human-designed matching features. We design a novel optimization algorithm for this challenging objective with self-supervised label propagation. Experiments on public datasets substantiate the superiority of L2M over SOTA methods. Moreover, we apply L2M to transfer from pneumonia to COVID-19 chest X-ray images with remarkable performance. L2M can also be extended in other distribution matching applications where we show in a trial experiment that L2M generates more realistic and sharper MNIST samples.

1 Introduction

Traditional machine learning generally assumes that training and test data are from the same distribution. In reality, this i.i.d. assumption barely holds. When an algorithm is trained on one domain and then tested on another domain, the performance is likely to drop due to the different data distribution [1]. Since the collection of massive labeled data is expensive and time-consuming, a more promising approach is to perform domain adaptation (DA) to enable the consistent performance of a predictive function on different domains.

The core challenge of DA is to match the cross-domain joint distributions [2]. However, the labels on the target domain are often unavailable in unsupervised DA. Therefore, a trend is to approximately match the joint distributions by matching the marginal and conditional distributions as theoretically verified in [2, 3]. Existing approaches achieve this goal via learning a domain-invariant representation by minimizing predefined distribution distances such as MMD [4–7], or an implicit discrepancy by an adversarial min-max game [8–11]. Recent works suggest that in addition to jointly matching these two distributions with equal weights [12], an adaptive weighting scheme is necessary to achieve better distribution matching performance [7, 13–15].

[*] The first two authors contributed equally. Correspondence to: jindong.wang@microsoft.com

Preprint. Under review.

Unfortunately, it remains challenging to apply DA to new applications. Existing methods are built with their own priors and inductive bias in approximating the joint distribution matching, which may fail to uncover the underlying relationship between transferable features and joint distributions [16]. For instance, MMD [17] may not be discriminative enough for high-dimensional data, and Jensen-Shannon divergence is not sensitive to mode collapse [18–21]. A recent work Learning to Transfer (L2T) [22] aims to reduce such bias by learning the "transfer experience" from thousands of pre-computed tasks before applying to new problems. However, L2T needs to build historical tasks from large auxiliary datasets, which is expensive and burdensome. Since deep learning makes it possible to learn features directly from the original datasets, can we design an automatic distribution matching strategy in a data-driven way?

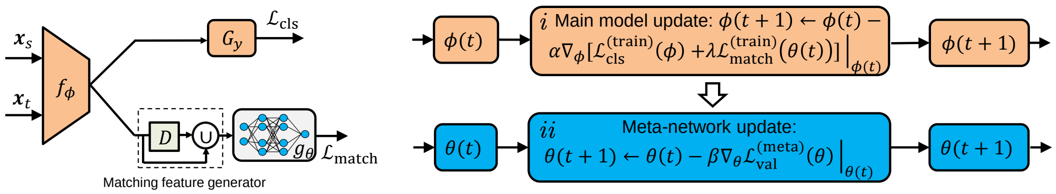
In this work, we propose a Learning to Match (L2M) framework to automatically match the cross-domain distributions while reducing the inductive bias on matching functions. Stepping back from the hand-crafted and predefined distances, we construct a meta-network to learn the distribution matching functions directly from the source and target domains. The meta-network is an MLP network which is theoretically a universal approximator for almost any continuous function [23]. We design a novel matching feature generator to L2M, where both task-independent and human-designed matching features can be taken as inputs to the meta-network for better distribution matching. Therefore, L2M can be seen as a general framework that unifies the deep features and human-crafted features (pre-defined distances) from the view of traditional vs. deep learning. Since it is challenging to optimize L2M with the unavailability of target domain labels, we propose to construct and update meta-data in a self-supervised manner [42] for updating the distribution matching loss. On the basis of matching features and meta-data, we propose an online optimization algorithm for L2M which can achieve accurate and steady performance.

Experiments show that L2M outperforms several state-of-the-art methods on public DA datasets. L2M is a general and flexible framework that can be used in other cross-domain tasks. We apply L2M to COVID-19 X-ray image classification by transferring knowledge from normal pneumonia to COVID-19, where L2M outperforms other methods in this data-hungry and imbalanced task. As an extension, L2M can be used for generating more realistic and sharper hand-written digits. The code of L2M will be released soon at `https://github.com/jindongwang/transferlearning/tree/master/code/deep/Learning-to-Match`.

2 Related Work

Transfer learning and domain adaptation. Domain adaptation (DA) is a specific area of transfer learning [24]. Existing works tend to explicitly or implicitly reduce the distribution divergence. The explicit distances are predefined divergence, such as Maximum Mean Discrepancy (MMD) [17], KL or JS divergence, cosine similarity, mutual information, and higher-order moments [25], which are well investigated in recent DA works [4–7, 26]. Optimal transport (OT) is another popular measure for distribution matching [11, 27–30]. There are other geometrical distances or transforms such GFK [31] and subspace learning [32, 33]. The implicit distance indirectly bridges the distribution gap through adversarial nets [34], or learnable metrics [35]. GAN-based DA methods learn domain-invariant features by confusing the feature extractor and discriminator [8–10], while metric learning [35] focuses on the sample-wise distance. Recent research implies performance improvement by adding more prior to the matching strategy such as adaptive weights between marginal and conditional distributions [7, 14, 15] with weights generated by the $\mathcal{A}$-distance [2]. Learning to transfer (L2T) [22] is similar to our idea in spirit. However, L2T has to manually construct thousands of transfer tasks to learn a linear transformation matrix using MMD, while L2M does not rely on historical tasks and learns non-linear feature maps, which is more efficient and general. There are several works aiming at bridging two domains by normalization such as BN [36], AutoDIAL [37], AdaBN [38], and TransNorm [39], which did not focus on direct learning the cross-domain joint distributions.

Distribution matching. Generative adversarial nets (GANs) [34] matches distributions between training and generated samples by iteratively training a domain discriminator and generator to confuse the discriminator. Our L2M is model-agnostic that can be applied in an adversarial manner by adopting GAN-based schemes such as DANN [8] or can also work without GAN. The pixel-level DA [40] learns the distribution matching in pixel-space.


x_s
x_t
f_ϕ
G_y
$\mathcal{L}_{\text{cls}}$
D
$\cup$
g_θ
$\mathcal{L}_{\text{match}}$
Matching feature generator
$\phi(t)$
i Main model update: $\phi(t+1) \leftarrow \phi(t) - \alpha\nabla_\phi[\mathcal{L}_{\text{cls}}^{(\text{train})}(\phi) + \lambda\mathcal{L}_{\text{match}}^{(\text{train})}(\theta(t))]\big|_{\phi(t)}$
$\phi(t+1)$
$\theta(t)$
ii Meta-network update: $\theta(t+1) \leftarrow \theta(t) - \beta\nabla_\theta\mathcal{L}_{\text{val}}^{(\text{meta})}(\theta)\big|_{\theta(t)}$
$\theta(t+1)$

(a) The architecture

(b) The learning procedure

Figure 1: The framework and computing flow of the proposed L2M approach.

3 Methodology

3.1 Learning to Match

We can decompose h_ϕ into a feature extractor f_ϕ and a classification layer G_y, where f_ϕ is explicitly parameterized by ϕ since it is more important for domain-invariant representation learning. Under the principle of structural risk minimization (SRM) [41], the optimal model parameter can be learned as:

$$\phi^\star = \arg\min_\phi \mathcal{L}_{\text{cls}}(G_y \circ f_\phi; \mathcal{D}_s) + \lambda\mathcal{L}_{\text{match}}(f_\phi(\mathcal{D}_s), f_\phi(\mathcal{D}_t)), \tag{1}$$

where $\mathcal{L}_{\text{cls}}$ is the classification loss on the source domain, $\mathcal{L}_{\text{match}}$ is the distribution matching loss, and λ is a trade-off parameter.

It is challenging to directly match the cross-domain joint distributions since the labels for the target domain are not available. Therefore, existing methods tend to approximate $\mathcal{L}_{\text{match}}$ using different priors. For instance, if we let $\mathcal{L}_{\text{match}} = d(\mathcal{D}_s, \mathcal{D}_t)$ where d is a predefined distance such as MMD [4], then we can get the explicit distribution matching. If $\mathcal{L}_{\text{match}} = \mathbb{D}(\mathcal{D}_s, \mathcal{D}_t)$ where $\mathbb{D}$ is the adversarial discriminator [8], then we get the implicit distribution matching. In a nutshell, the main difference among existing works is the design of explicit or implicit $\mathcal{L}_{\text{match}}$.

In this paper, we postulate L2M to automatically match the distributions across domains. The core of L2M is a meta-network that learns the distribution matching in a data-driven manner. To be specific, the meta-network is a Multi-Layer Perceptron (MLP) that has the ability to approximate any continuous functions [23]. Therefore, L2M can learn the distribution matching loss directly from the source and target domains:

$$\mathcal{L}_{\text{match}}(\mathcal{D}_s, \mathcal{D}_t) = g_\theta(f_\phi(\mathcal{D}_s), f_\phi(\mathcal{D}_t)), \tag{2}$$

where $g(\cdot)$ is the distribution matching function (network) parameterized by θ. It is clear that this formulation is a general form that can theoretically include existing pre-defined distances.

With the meta-network g_θ, we build a general framework as shown in Fig. 1(a). The architecture consists of four parts: feature extractor f_ϕ, label classifier G_y, meta-network g_θ, and matching feature generator. Specifically, f_ϕ is a CNN network to extract the features of the input domains, G_y is trained to minimize the prediction loss on the labeled (source) domain, and g_θ is an MLP network, which is used to match the cross-domain distributions (learn $\mathcal{L}_{\text{match}}$). The most important part is the matching feature generator, which generates useful inputs to the meta-network g_θ. For a general framework that allows both deep and human-designed features, we concatenate the deep features (direct link) with the human-designed distances (the green module D) via the concatenation module $\cup$. Then, the matching features can be taken as inputs by the meta-network g_θ to learn the distribution matching functions.

Our learning objective is well in accordance with the DA theory proposed in [2] that directly learns to reduce the distribution divergence between domains, such that the risk on the target domain can be bounded. The learning objective of L2M can be obtained as:

$$\min_{\theta,\phi} = \arg\min \mathcal{L}_{\text{cls}}(\phi) + \lambda\mathcal{L}_{\text{match}}(\phi; \theta). \tag{3}$$

However, it remains challenging to optimize the above equation due to three reasons. Firstly, what kind of matching features should we take as inputs to the meta-network g_θ for better distribution matching? Secondly, we only have the labeled source domain and the unlabeled target domain, how to compute the distribution matching loss $\mathcal{L}_{\text{match}}$ without the target domain labels? Thirdly, even if we have the matching features and optimization data, we cannot use a simple EM algorithm for optimization since updating $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{match}}$ on the same training data will definitely lead to overfitting and local optimum. Therefore, it is still non-trivial to optimize L2M.

In the next sections, we introduce how to tackle the above three challenges.

3.2 Matching feature generator

The matching feature generator generates useful representations as inputs to the meta-network g. We use $\mathbf{F}$ to denote the matching features. Technically, $\mathbf{F}$ can be any useful representations. In this paper, we propose two different kinds of matching features as shown in Table 1: (1) **Task-independent features**, which are general and can be automatically computed by the main network f_ϕ as shown in the direct link of Fig. 1(a); (2) **Human-designed distances (features)** as indicated in the green module D in Fig. 1(a), which are the pre-defined distances such as MMD or adversarial game. These features can either be used alone, or be concatenated by the concatenation module $\cup$. In later experiments, it is surprising to find that the combination of these two features can be seen as the combination of deep and human-designed features, which generally leads to better performance. More details of the matching features are in the supplementary file.

Table 1: Description of the matching features. $q(\cdot)$ is the function of last layer before softmax. d_{m} and d_{c} are marginal and conditional explicit distances using MMD. $\mathbb{D}_{\text{m}}$ and $\mathbb{D}_{\text{c}}$ are marginal and conditional implicit distances using adversarial min-max game. $[a, b]$ is the concatenation of a and b.

Feature Type	Notation	Description	Calculation
Task-independent	$\mathbf{F}_{\text{emb}}$	Feature embedding	$\mathbf{F}_{\text{emb}} = [f_\phi(\boldsymbol{x}_i), f_\phi(\boldsymbol{x}_j)]$
	$\mathbf{F}_{\text{logit}}$	logit	$\mathbf{F}_{\text{logit}} = [q(f_\phi(\boldsymbol{x}_i)), q(f_\phi(\boldsymbol{x}_j))]$
Human-designed (pre-defined dist.)	$\mathbf{F}_{\text{mmd}}$	Explicit distribution distance	$\mathbf{F}_{\text{mmd}} = [d_{\text{m}}(f_\phi(\boldsymbol{x}_i), f_\phi(\boldsymbol{x}_j)), d_{\text{c}}(f_\phi(\boldsymbol{x}_i), f_\phi(\boldsymbol{x}_j))]$
	$\mathbf{F}_{\text{adv}}$	Implicit distribution distance	$\mathbf{F}_{\text{adv}} = [\mathbb{D}_{\text{m}}(f_\phi(\boldsymbol{x}_i), f_\phi(\boldsymbol{x}_j)), \mathbb{D}_{\text{c}}(f_\phi(\boldsymbol{x}_i), f_\phi(\boldsymbol{x}_j))]$

3.3 The construction of meta-data

We introduce the idea of "meta-data". Since direct computation of the distribution matching loss $\mathcal{L}_{\text{match}}$ is hard due to the unavailability of target labels, we turn to using the meta-data $\mathcal{D}_{\text{meta}}$ instead. To be more specific, $\mathcal{D}_{\text{meta}} = \{\boldsymbol{x}_j^t\}_{j=1}^{m \times C} \sim P_t(\boldsymbol{x}, \hat{y})$ where $\hat{y}$ is the predicted (pseudo) label on the target domain. In each iteration, we randomly sample m instances for each class with high prediction scores calculated by the main network as the ground truth of the meta-data. This selection is iterated in the whole learning process for better performance. The pseudo labels of the meta-data can get more confident since the meta-data are chosen from the target domain data with the highest prediction probabilities. This assumption is validated in early works [7, 10] and can also be seen as a self-supervised technique [42]. Therefore, the matching loss is calculated on the training data ($\mathcal{L}_{\text{match}} = \mathcal{L}_{\text{match}}^{(\text{train})}$) when updating the main network f_ϕ, and the meta-network g_θ is updated on the meta-data.

3.4 Learning algorithm

In this paper, we propose an online updating algorithm for L2M. Fig. 1(b) illustrates the key learning steps. It should be noted that the data for updating ϕ and θ are different: when updating ϕ, we use the normal training data from the source domain to calculate the cross-entropy loss; when updating θ, we use the source domain and the pseudo-labeled target domain meta-data. The learning procedure of L2M consists of two main steps: main network update and meta-network update. In the following, we use t to denote learning steps.

Main network update. This step is mainly for updating ϕ for the main network. To enforce the update of θ in the next step, we construct an assist model which is a copy of the main model by inheriting the same architecture and parameters from the main model (f_ϕ, G_y, g_θ) and use it for calculating the loss. We employ SGD for optimizing the classification loss $\mathcal{L}_{\text{cls}}$ and distribution matching loss $\mathcal{L}_{\text{match}}$. $\mathcal{L}_{\text{cls}}$ can be formulated as:

$$\mathcal{L}_{\text{cls}}^{(\text{train})} = \mathbb{E}_{(\boldsymbol{x},y)\sim B_s} \ell^{(\text{CE})}(G_y(f_\phi(\boldsymbol{x})), y), \tag{4}$$

where $\ell^{(\mathrm{CE})}$ is the cross-entropy loss and B_s denotes a mini-batch data sampled from $\mathcal{D}_s$. The distribution matching loss $\mathcal{L}_{\mathrm{match}}$ is calculated by the meta-network g_θ:

$$\mathcal{L}_{\mathrm{match}}^{(\mathrm{train})} = \mathbb{E}_{\boldsymbol{x}_i \sim B_s, \boldsymbol{x}_j \sim B_t} g_\theta(f_\phi(\boldsymbol{x}_i), f_\phi(\boldsymbol{x}_j); \phi), \tag{5}$$

where B_t is a mini-batch data sampled from $\mathcal{D}_t$. Note that this step does not need the meta-data from the target domain since we only sample a batch of source and target domain data ($\boldsymbol{x}$) and do not need the target domain label y. Therefore, we do not update the matching loss.

After getting the training loss, the updating equation of the copied main model can be obtained by moving the current $\phi(t)$ towards the descent direction of objective in Eq. (3):

$$\phi(t+1) = \phi(t) - \alpha \nabla_\phi [\mathcal{L}_{\mathrm{cls}}^{(\mathrm{train})}(\phi) + \lambda \mathcal{L}_{\mathrm{match}}^{(\mathrm{train})}(\phi; \theta(t))]|_{\phi(t)}, \tag{6}$$

where α is the learning rate of the assist model. $\mathcal{L}_{\mathrm{cls}}^{(\mathrm{train})}(\phi)$ and $\mathcal{L}_{\mathrm{match}}^{(\mathrm{train})}(\phi; \theta(t))$ are computed by Eq. (4) and (5).

Meta-network update. This step is for updating θ for the meta-network g_θ on the meta-data $\mathcal{D}_{\mathrm{meta}}$. Similar to updating ϕ, it is natural that updating θ requires "ground-truth" available for the distribution matching loss $\mathcal{L}_{\mathrm{match}}$. However, this is not available in UDA problems. To solve this challenge, we employ a self-supervised strategy with the assumption that after one epoch of updating $\phi(t)$ to $\phi(t+1)$, the distribution matching loss can get smaller with the increasing confidence of the target pseudo labels. This pseudo-label assumption is widely adopted in previous DA works [6, 7, 10]. Therefore, this validation loss can be updated by computing the discrepancy between the distribution matching loss on $\phi(t)$ and $\phi(t+1)$:

$$\mathcal{L}_{\mathrm{val}}^{(\mathrm{meta})} = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_{\mathrm{meta}}} \tanh(g_\theta(f(\boldsymbol{x}; \phi(t))) - g_\theta(f(\boldsymbol{x}; \phi(t+1)))), \tag{7}$$

where $\tanh(\cdot)$ is an activation function. Note that we fix ϕ in this step and minimize Eq. (7) w.r.t. θ can gradually update the meta-network g_θ. The pseudo labels can be easily obtained by a single forward-pass and then selected according to the confidence (softmax probability). To ensure their confidence, we choose the samples with probabilities ≥ 0.8 in our experiments.

Denote β the learning rate of meta-network g_θ, then θ can be updated as:

$$\theta(t+1) = \theta(t) - \beta \nabla_\theta \mathcal{L}_{\mathrm{val}}^{(\mathrm{meta})}(\theta; \phi(t), \phi(t+1))|_{\theta(t)}. \tag{8}$$

The above two steps are used iteratively as the pseudo labels of the meta-data can be more confident and all the losses can be iteratively minimized. In our experiments, we observe that the network will converge in dozens of epochs. The complete algorithm and convergence analysis are presented in the supplementary file.

As for inference, L2M is the same as existing DA methods [6, 7, 10, 43]. We simply fix ϕ and θ and use the main model to perform a single forward-pass to get the results for the test data.

4 Experiments on Public Datasets

4.1 Experimental setup

Datasets. We adopt four public datasets: ImageCLEF-DA [44], Office-Home [45], VisDA-2017 [46] and Office-31 [47]. They are widely used by most UDA approaches [7, 9, 10, 43]. The detailed dataset descriptions are presented in the supplementary file.

Baselines and implementations. We comapre L2M with several recent DA methods: **ResNet** [48], **DDC** [6], **DAN** [5], **DANN** [8], **JAN** [49], **MADA** [50], **CAN** [51], **MEDA** [7], **DAAN** [15], **CDAN** [43], **DeepJDOT** [29], **MDD** [10], and **TransNorm** [39]. The main network of all methods including L2M are based on ImageNet-pretrained ResNet50. The hyperparameter setting for L2M are presented in the supplementary file. We follow the standard protocols for UDA and take classification accuracy on the target domain as the evaluation metric and target labels are only used for evaluation. The best parameters are tuned according to [52]. The results are the average accuracy of 10 experiments by following the same protocol [6, 7, 10, 43].

4.2 Analysis of matching features

Before using L2M, a natural question is which matching feature should be used for better performance. Moreover, how is the performance of MMD and adversarial discrepancy in L2M compared to existing MMD or adversarial-based DA methods? To answer these questions, we randomly choose two pairs of DA tasks from Office-Home dataset (R → A, R → P, and vice versa) to compare the performance of existing distance-based methods (DAN [5] and MEDA [7] use MMD while DANN [8], CDAN [43], and DAAN [15] use adversarial-based discrepancy) with L2M. Technically, all matching features can be combined, which will result in $2^4 = 16$ different matching features. For computational issue, we construct eight matching features: $\{\mathbf{F}_{\mathrm{emb}}, \mathbf{F}_{\mathrm{logit}}, \mathbf{F}_{\mathrm{mmd}}, \mathbf{F}_{\mathrm{adv}}, [\mathbf{F}_{\mathrm{emb}}, \mathbf{F}_{\mathrm{mmd}}], [\mathbf{F}_{\mathrm{emb}}, \mathbf{F}_{\mathrm{adv}}], [\mathbf{F}_{\mathrm{logit}}, \mathbf{F}_{\mathrm{mmd}}], [\mathbf{F}_{\mathrm{logit}}, \mathbf{F}_{\mathrm{adv}}]\}$. It should be noted that both $\mathbf{F}_{\mathrm{emb}}$ and $\mathbf{F}_{\mathrm{logit}}$ can be applied to both explicit (deep) and implicit (adversarial) matching networks, leading to ten features in total. In addition, we do not combine three or four features since their performance can naturally be better but with more computations.

Table 2: Matching features of L2M.

	Method	R→A	A→R	P→R	R→P
Explicit	DAN (marginal) [5]	63.1	67.9	67.7	74.3
	MEDA (joint) [7]	61.2	68.8	72.9	76.0
	L2M (emb)	71.1	76.1	79.1	83.7
	L2M (logit)	70.3	76.6	79.4	83.6
	L2M (mmd)	69.3	73.4	75.2	83.2
	L2M (emb+mmd)	71.1	76.9	78.6	83.1
	L2M (logit+mmd)	71.5	76.7	78.5	82.8
Implicit	DANN (marginal) [8]	63.2	70.1	76.8	68.5
	DAAN (joint) [15]	66.3	73.7	74.0	78.8
	CDAN (conditional) [43]	70.9	76.0	77.3	81.6
	L2M (emb)	70.8	77.8	79.3	83.2
	L2M (logit)	71.6	71.7	79.4	83.6
	L2M (adv)	72.7	78.5	80.3	83.1
	L2M (emb+adv)	71.8	79.3	80.6	83.5
	L2M (logit+adv)	71.4	76.6	78.6	82.8

The feature dimensions of each matching feature are presented in the supplementary file. The comparison results are in Table 2. For better clarification, we compare the performance of best MMD- and adversarial-based methods in Fig. 2(a), along with the *average* performance of L2M using these features. More experiments can be found at the supplementary. Firstly, we see that in both explicit and implicit distribution matching, L2M can generally achieve competitive performance with different matching features. This verifies that L2M is effective for distribution matching. Secondly, in some cases, the performance of L2M with MMD distances are better than previous adversarial-based methods. Since adversarial-based methods require much more training time, this makes L2M+MMD suitable solutions for resource-constrained applications. Thirdly, the performance of L2M with both task-independent features and pre-defined distances are generally better than using each feature solely, indicating the common practice is useful that deep learning performance can be boosted by combining deep features ($\mathbf{F}_{\mathrm{emb}}$ or $\mathbf{F}_{\mathrm{logit}}$) with human-designed features ($\mathbf{F}_{\mathrm{mmd}}$ or $\mathbf{F}_{\mathrm{adv}}$). We also observe that L2M with embeddings are generally better than logits, which is probably because those embeddings contain richer information than logits. Therefore, in the next experiments, we adopt $[\mathbf{F}_{\mathrm{emb}}, \mathbf{F}_{\mathrm{adv}}]$ for a balance between computation and better performance. In real applications, more domain-dependent matching features can be added according to domain knowledge.

4.3 Performance against SOTA methods

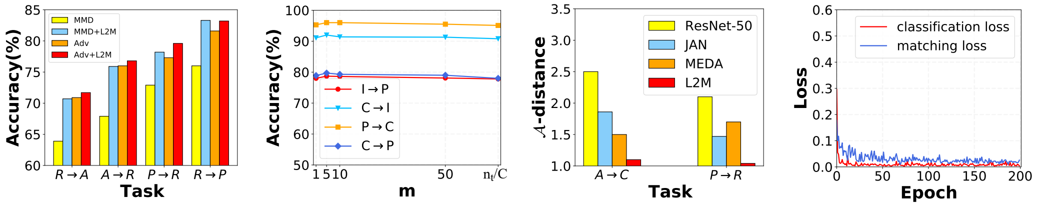
The results on Office-Home dataset are shown in Table 3, while the results on ImageCLEF-DA and VisDA-17 datasets are in Table 4. The results on Office-31 are provided in supplementary file due to space limits. From the results, we see that the L2M outperforms all comparison methods. Specifically, on ImageCLEF-DA dataset, although the baseline for this dataset is very high, L2M still achieves an average accuracy of 89.1% with a 0.6% improvement over the second-best baseline. On Office-Home dataset, L2M achieves an average accuracy of 69.6% with a 1.5% improvement compared to the second-best. Office-Home dataset is rather complicated and involves more samples and categories, which indicates the effectiveness of L2M. On Office-31 dataset, L2M achieves an average accuracy of 89.5%, which is also highly competitive. Last, on the VisDA-17 dataset, which

Table 3: Accuracy (%) on Office-Home for UDA (ResNet-50).

Method	A→C	A→P	A→R	C→A	C→P	C→R	P→A	P→C	P→R	R→A	R→C	R→P	AVG
ResNet [48]	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DAN [5]	43.6	57.0	67.9	45.8	56.5	60.4	44.0	43.6	67.7	63.1	51.5	74.3	56.3
DANN [8]	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
JAN [49]	45.9	61.2	68.9	50.4	59.7	61.0	45.8	43.4	70.3	63.9	52.4	76.8	58.3
MEDA [7]	46.6	68.9	68.8	49.0	66.4	66.1	51.8	45.0	72.9	61.2	50.3	76.0	60.2
DAAN [15]	50.5	65.0	73.7	53.7	62.7	64.6	53.5	45.2	74.0	66.3	54.0	78.8	61.8
CDAN [43]	50.7	70.6	76.0	57.6	70.0	70.0	57.4	50.9	77.3	70.9	56.7	81.6	65.8
ALDA [64]	53.7	70.1	76.4	60.2	72.6	71.5	56.8	51.9	77.1	70.2	56.3	82.1	66.6
CDAN+TransNorm [39]	50.2	71.4	77.4	59.3	72.7	73.1	61.0	53.1	79.5	71.9	59.0	82.9	67.6
MDD [10]	54.9	73.7	77.8	60.0	71.4	71.8	61.2	53.6	78.1	**72.5**	**60.2**	82.3	68.1
L2M	**57.5**	**74.0**	**78.5**	**63.0**	**73.1**	**72.5**	**63.5**	**56.6**	**80.5**	72.0	**60.2**	**83.6**	**69.6**

Table 4: Accuracy (%) on ImageCLEF-DA and VisDA-2017 for UDA (ResNet-50).

	ImageCLEF-DA							VisDA-2017	
Method	I→P	P→I	I→C	C→I	C→P	P→C	AVG	Method	Syn→Real
ResNet [48]	74.8	83.9	91.5	78.0	65.5	91.2	80.7	DAN [5]	49.8
DAN [5]	75.0	86.2	93.3	84.1	69.8	91.3	83.3	JAN [49]	61.6
DANN [8]	75.0	86.0	96.2	87.0	74.3	91.5	85.0	DANN+TransNorm [39]	66.3
MEDA [7]	78.1	90.4	93.1	86.4	73.2	91.7	85.5	DeepJDOT [29]	66.9
CAN [51]	78.2	87.5	94.2	89.5	75.8	89.2	85.7	MCD [53]	69.2
JAN [49]	76.8	88.0	94.7	89.5	74.2	91.7	85.8	GTA [54]	69.5
MADA [50]	75.0	87.9	96.0	88.8	75.2	92.2	85.8	CDAN [43]	70.0
CDAN [43]	77.7	90.7	**97.7**	91.3	74.2	94.3	87.7	CDAN+TransNorm [39]	71.4
CDAN+TransNorm [39]	78.3	90.8	96.7	**92.3**	78.0	94.8	88.5	MDD [10]	74.6
L2M	**78.7**	**91.0**	97.0	92.0	**79.7**	**96.0**	**89.1**	L2M	**77.5**



(a) Matching features (b) Meta-data (c) Distribution distance (d) Convergence

Figure 2: (a) Comparison between the best existing methods with predefined distance and the average of L2M. (b) Analysis of the number of meta-data m. (c) Distribution discrepancy between two domains. (d) Convergence of L2M.

is rather larger compared to the other datasets (280,000+ images), L2M achieves an accuracy of 77.5% with a significant improvement of **2.9**%. All these results demonstrate that L2M can achieve competitive performance on DA tasks.

4.4 Detailed analysis

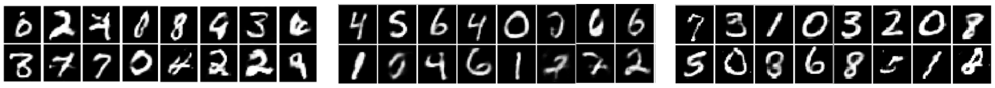
Analysis of meta-data. We empirically analyze the batch size m of the meta-data $\mathcal{D}_{\text{meta}}$. It is obvious that a larger m will bring more uncertainty, and a smaller m is likely to make the meta-network unstable. We record the performance of L2M using different values of m on several randomly selected tasks in Fig. 2(b). The results indicate that L2M is robust to m and a small m can lead to competitive performance. Therefore, we set $m = 5$ in our experiments for computational efficiency.

Distribution discrepancy. The $\mathcal{A}$-distance [2] measures the distribution discrepancy that is defined as $d_{\mathcal{A}} = 2(1 - 2\epsilon)$, where ϵ is the classifier loss to discriminate the source and target domains. Smaller $\mathcal{A}$-distance indicates better domain-invariant features. Fig. 2(c) shows that L2M can achieve a lower $d_{\mathcal{A}}$, implying a lower generalization error of L2M.

Convergence analysis. L2M introduces a meta-network, which may make the training process harder. In this section, we empirically evaluate the convergence of L2M. As shown in Fig. 2(d), the results on a randomly-chosen task show that L2M can reach a quick and steady convergence in a limited number of iterations. Therefore, L2M can be easily trained.

Table 5: Results on COVID-19 X-ray adaptation (normal pneumonia → COVID-19, ResNet-18). Here we use the 95% confidence interval, where the corresponding value of z is 1.96. The computed confidence interval r is around 1.3%.

Method	Precision (%)	Recall (%)	F1 (%)
Train on source	63.5	66.7	65.0
Train on target *(ideal state)*	**91.7**	55.0	68.8
Fine-tuning	56.3	75.0	64.3
DLAD [55]	62.0	73.3	67.2
DANN [8]	61.4	71.7	66.2
MCD [53]	63.2	60.0	61.5
CDAN+TransNorm [39]	85.0	39.2	63.7
L2M	70.1	**78.3**	**74.0**



(a) GAN (b) GMMN (c) L2M

Figure 3: Generated samples from GAN, GMMN, and L2M.

5 Application to COVID-19 Chest X-ray Image Classification

Other than public datasets, we apply L2M to a COVID-19 chest X-ray image classification dataset [56], where the source domain is normal or pneumonia, and the target domain is normal or COVID-19 pneumonia. Note that this is a class-imbalanced task which is more challenging and realistic. We use F1, Precision, and Recall as the evaluation metrics for this highly-imbalanced binary classification task. As shown in Table 7, L2M achieves better results compared to finetune and other DA methods. Here we use the 95% confidence interval, where the corresponding value of z is 1.96. The computed confidence interval r is around 1.3%. More details about the dataset, comparison methods, and results are in the supplementary file.

6 Discussions

Extending L2M for image generation. We show the potential of L2M in generating MNIST hand-written digits. We use GAN [34] (adversarial distance) and GMMN [57] (MMD distance) as the baselines. We replace the MMD module in GMMN with L2M. Hyperparameter settings and training details are in supplementary. The generated samples are shown in Fig. 3. L2M can generate more realistic samples compared to GAN, and sharper samples compared to GMMN. This indicates the potential of L2M in image generation. It should be noted that this is only a *trial* experiment and more efforts are needed for achieving SOTA performance on image generation.

Limitations and solutions. L2M can be roughly regarded as that requires updating two networks iteratively. Therefore, compared with regular DA methods (*e.g.*, DANN, CDAN, MDD), L2M needs more than more training time. It is suggested to use a smaller batch size of meta-data compared to training data to reduce GPU memory increment and speed up training. However, the inference time is the same as other methods for using the same backbone. L2M can be more efficient by adopting knowledge distillation as suggested in meta-pseudo-labels (MPL) [58], which is left for future research. Additionally, a pre-trained L2M model can be deployed to the edge devices which can achieve accurate and fast inference.

7 Conclusions

In this paper, for the first time, we step back from focusing on designing distribution matching features according to human knowledge, and instead, propose L2M to automatically match the cross-domain joint distributions for domain adaptation. Our work shows that by taking diverse matching features including task-independent and human-designed distances, L2M can directly learn the distribution matching in a data-driven way. L2M can be seen as a general framework that unifies deep feature learning and human-designed feature learning for better distribution matching. Experiments on public datasets substantiate the superiority of L2M over state-of-the-art approaches on DA and image generation tasks. We apply L2M to COVID-19 X-ray image adaptation experiment, where it significantly outperforms existing methods in such a highly imbalanced task. We believe that L2M can be helpful in other problems such as domain generalization, open-set DA, and partial transfer learning, which will be the focus of future research.

References

[1] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328, 2014.

[2] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *NIPS*, pages 137–144, 2007.

[3] Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J Gordon. On learning invariant representation for domain adaptation. In *ICML*, 2019.

[4] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE TNN*, 22(2):199–210, 2011.

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

[6] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint:1412.3474*, 2014.

[7] Jindong Wang, Wenjie Feng, Yiqiang Chen, Han Yu, Meiyu Huang, and Philip S Yu. Visual domain adaptation with manifold embedded distribution alignment. In *MM*, pages 402–410, 2018.

[8] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

[9] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, pages 7167–7176, 2017.

[10] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael I Jordan. Bridging theory and algorithm for domain adaptation. In *ICML*, 2019.

[11] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE TPAMI*, 39(9):1853–1865, 2016.

[12] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer feature learning with joint distribution adaptation. In *ICCV*, 2013.

[13] Jindong Wang, Yiqiang Chen, Shuji Hao, Wenjie Feng, and Zhiqi Shen. Balanced distribution adaptation for transfer learning. In *ICDM*, pages 1129–1134, 2017.

[14] Zhen Fang, Jie Lu, Feng Liu, Junyu Xuan, and Guangquan Zhang. Open set domain adaptation: Theoretical bound and algorithm. *IEEE TNNLS*, 2019.

[15] Chaohui Yu, Jindong Wang, Yiqiang Chen, and Meiyu Huang. Transfer learning with dynamic adversarial adaptation network. In *ICDM*, 2019.

[16] Fredrik D Johansson, David Sontag, and Rajesh Ranganath. Support and invertibility in domain-invariant representations. *arXiv preprint arXiv:1903.03448*, 2019.

[17] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *JMLR*, 13(Mar):723–773, 2012.

[18] Aaditya Ramdas, Sashank Jakkam Reddi, Barnabás Póczos, Aarti Singh, and Larry Wasserman. On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions. In *AAAI*, 2015.

[19] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P Xing. On unifying deep generative models. *arXiv preprint arXiv:1706.00550*, 2017.

[20] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

[21] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[22] Ying Wei, Yu Zhang, Junzhou Huang, and Qiang Yang. Transfer learning via learning to transfer. In *ICML*, pages 5085–5094, 2018.

[23] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[24] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE TKDE*, 22(10):1345–1359, 2010.

[25] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. In *ICLR*, 2017.

[26] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, pages 443–450, 2016.

[27] Nicolas Courty, Rémi Flamary, and Devis Tuia. Domain adaptation with regularized optimal transport. In *ECML PKDD*, pages 274–289. Springer, 2014.

[28] Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *NIPS*, pages 3730–3739, 2017.

[29] Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *ECCV*, pages 447–463, 2018.

[30] Zhen Zhang, Mianzhi Wang, and Arye Nehorai. Optimal transport in reproducing kernel hilbert spaces: Theory and applications. *IEEE TPAMI*, 2019.

[31] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, pages 2066–2073. IEEE, 2012.

[32] Baochen Sun and Kate Saenko. Subspace distribution alignment for unsupervised domain adaptation. In *BMVC*, pages 24–1, 2015.

[33] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, volume 6, page 8, 2016.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[35] Yong Luo, Yonggang Wen, Ling-Yu Duan, and Dacheng Tao. Transfer metric learning: Algorithms, applications and outlooks. *arXiv preprint arXiv:1810.03944*, 2018.

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

[37] Fabio Maria Cariucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. Autodial: Automatic domain alignment layers. In *ICCV*, pages 5077–5085. IEEE, 2017.

[38] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018.

[39] Ximei Wang, Ying Jin, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Transferable normalization: Towards improving transferability of deep neural networks. In *NeurIPS*, pages 1951–1961, 2019.

[40] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, pages 3722–3731, 2017.

[41] Masashi Sugiyama. *Introduction to statistical machine learning*. Morgan Kaufmann, 2015.

[42] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[43] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, pages 1640–1650, 2018.

[44] The imageclef-da challenge 2014. https://www.imageclef.org/2014.

[45] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, pages 5018–5027, 2017.

[46] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge, 2017.

[47] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV*, pages 213–226. Springer, 2010.

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[49] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, 2017.

[50] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. In *AAAI*, 2018.

[51] Weichen Zhang, Wanli Ouyang, Wen Li, and Dong Xu. Collaborative and adversarial network for unsupervised domain adaptation. In *CVPR*, pages 3801–3809, 2018.

[52] Kaichao You, Ximei Wang, Mingsheng Long, and Michael Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In *ICML*, pages 7124–7133, 2019.

[53] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, pages 3723–3732, 2018.

[54] Swami Sankaranarayanan, Yogesh Balaji, Carlos D Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *CVPR*, pages 8503–8512, 2018.

[55] Jianpeng Zhang, Yutong Xie, Yi Li, Chunhua Shen, and Yong Xia. Covid-19 screening on chest x-ray images using deep learning based anomaly detection. *arXiv preprint arXiv:2003.12338*, 2020.

[56] Yifan Zhang, Shuaicheng Niu, Zhen Qiu, Ying Wei, Peilin Zhao, Jianhua Yao, Junzhou Huang, Qingyao Wu, and Mingkui Tan. Covid-da: Deep domain adaptation from typical pneumonia to covid-19. *arXiv preprint arXiv:2005.01577*, 2020.

[57] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *ICML*, 2015.

[58] Hieu Pham, Qizhe Xie, Zihang Dai, and Quoc V Le. Meta pseudo labels. *arXiv preprint arXiv:2003.10580*, 2020.

[59] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018.

[60] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 2019.

[61] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030, 2016.

[62] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *NIPS*, pages 136–144, 2016.

[63] Ruichu Cai, Zijian Li, Pengfei Wei, Jie Qiao, Kun Zhang, and Zhifeng Hao. Learning disentangled semantic representation for domain adaptation. In *IJCAI*, 2019.

[64] Minghao Chen, Shuai Zhao, Haifeng Liu, and Deng Cai. Adversarial-learned loss for domain adaptation. In *AAAI*, 2020.

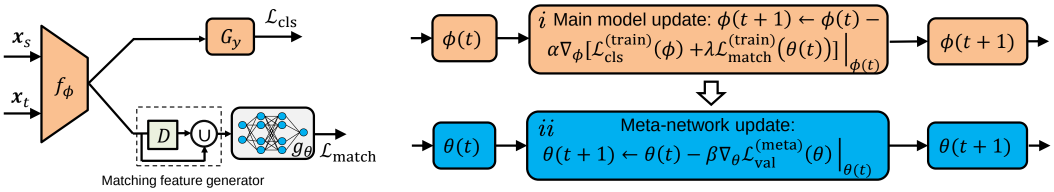
[65] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[66] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, pages 2951–2959, 2012.

Supplementary: Learning to Match Distributions for Domain Adaptation

A Learning algorithm for L2M

We also put the framework and key learning steps of L2M here for better illustration. The complete learning procedure of L2M is listed in Algorithm 1.


x_s
x_t
f_ϕ
G_y
$\mathcal{L}_{\text{cls}}$
D
$\cup$
g_θ
$\mathcal{L}_{\text{match}}$
Matching feature generator
$\phi(t)$
i Main model update: $\phi(t+1) \leftarrow \phi(t) - \alpha\nabla_\phi[\mathcal{L}_{\text{cls}}^{(\text{train})}(\phi) + \lambda\mathcal{L}_{\text{match}}^{(\text{train})}(\theta(t))]\big|_{\phi(t)}$
$\phi(t+1)$
$\theta(t)$
ii Meta-network update:
$\theta(t+1) \leftarrow \theta(t) - \beta\nabla_\theta\mathcal{L}_{\text{val}}^{(\text{meta})}(\theta)\big|_{\theta(t)}$
$\theta(t+1)$

(a) The architecture

(b) The learning procedure

Figure 1: The framework and computing flow of the proposed L2M approach.

Algorithm 1 Learning algorithm of L2M

```
Input: Source domain $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$, target domain $\mathcal{D}_t = \{\mathbf{x}_j^t\}_{j=1}^{n_t}$, learning rate $\alpha, \beta$, max epochs $T$.
Output: $\{\phi^\star, \theta^\star\}$.
Initialize $\phi(0)$ and $\theta(0)$.
while epoch $t < T$ do
    Build an assist model with its parameter inherited from the main model $\phi(t)$.
    Sample a mini-batch data $B_s, B_t$ from both the source and target domain.
    Update $\phi$ by step $i$ in Fig. 1(b). The loss consists of $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{match}}$, we only update the assist model $\phi$ and the meta-network $\theta$ only be updated in step $ii$.
    Select the data with the highest prediction confidence from $\mathcal{D}_t$ to construct meta-data $\mathcal{D}_{\text{meta}}$.
    Update the meta-network $\theta$ by step $ii$ in Fig. 1(b).
end while
return $\{\phi^\star, \theta^\star\}$
```

It is worth noting that this optimization is general and can be naturally used in image generation tasks. Hence, we also use the same optimization step in the MNIST digits generation experiments by injecting this process directly to the GMMN [57] models. Therefore, L2M is a general and flexible framework that can work for most cross-domain distribution matching tasks.

A.1 Matching features

Task-independent matching features. It is natural to use the extracted feature embedding by f_ϕ as one kind of task-independent features, which is denoted as $\mathbf{F}_{\text{emb}} \in \mathbb{R}^d$, where d is the number of neurons in this layer. For classification tasks, another kind of features is the network logit: $\mathbf{F}_{\text{logit}} \in \mathbb{R}^C$, which is the activation of the last FC layer before softmax. Note that in fact, $\mathbf{F}_{\text{logit}}$ should be computed by G_y. For symbolic brievity we also draw it in the same way as $\mathbf{F}_{\text{emb}}$ in Fig. 1(a). Denote q the function of last FC layer, then they can be computed as:

$$\mathbf{F}_{\text{emb}} = [f_\phi(\mathbf{x}_i).f_\phi(\mathbf{x}_j)]. \tag{1}$$

$$\mathbf{F}_{\text{logit}} = [q(f_\phi(\mathbf{x}_i)).q(f_\phi(\mathbf{x}_j))]. \tag{2}$$

Human-designed matching features. We adopt two popular distances as human-designed matching features: explicit distribution matching distance using MMD ($\mathbf{F}_{\text{mmd}} \in \mathbb{R}$), and implicit distribution matching distance using adversarial nets ($\mathbf{F}_{\text{adv}} \in \mathbb{R}$). Their basic idea is to approximate the joint distributions using marginal or conditional distributions. A recent work MEDA [7] showed that matching both conditional and marginal distributions can be useful. Therefore, we denote $d_{\text{m}}, d_{\text{c}}$ the marginal and conditional distances (losses) respectively. Then, these features can be computed as:

$$\mathbf{F}_{\text{mmd}} = [d_{\text{m}}(f_\phi(\mathbf{x}_i), f_\phi(\mathbf{x}_j)), d_{\text{c}}(f_\phi(\mathbf{x}_i), f_\phi(\mathbf{x}_j))]. \tag{3}$$

$$\mathbf{F}_{\mathrm{adv}} = [\mathbb{D}_{\mathrm{m}}(f_\phi(\mathbf{x}_i), f_\phi(\mathbf{x}_j)), \mathbb{D}_{\mathrm{c}}(f_\phi(\mathbf{x}_i), f_\phi(\mathbf{x}_j))]. \tag{4}$$

For explicit distribution matching using MMD [17], the marginal and conditional distances can be computed as:

$$\begin{aligned} d_{\mathrm{m}} &= \left\| \mathbb{E}_{\mathbf{x}\sim B_s}\phi(\mathbf{x}) - \mathbb{E}_{\mathbf{x}\sim B_t}\phi(\mathbf{x}) \right\|^2_{\mathcal{H}_k}, \\ d_{\mathrm{c}} &= \mathbb{E}_{c\sim\mathcal{C}} \left\| \mathbb{E}_{\mathbf{x}\sim B_s^{(c)}}\phi(\mathbf{x}) - \mathbb{E}_{\mathbf{x}\sim B_t^{(c)}}\phi(\mathbf{x}) \right\|^2_{\mathcal{H}_k}, \end{aligned} \tag{5}$$

where $\mathcal{H}_k$ is the Reproducing Kernel Hilbert Space (RKHS) induced by kernel k, $B^{(c)}$ denotes samples belonging to class c, and $\phi(\cdot)$ some feature mapping function.

For implicit distribution matching using GAN [34], the main idea is to design a domain discriminator G_d to identify which domain the samples belong to. We train f_ϕ and G_y to confuse G_d, and eventually G_d gets confused and fails to discriminate the domains. In this situation, the marginal and conditional adversarial distances can be respectively computed as:

$$\begin{aligned} \mathbb{D}_{\mathrm{m}} &= \mathbb{E}_{\mathbf{x}\sim B_s\cup B_t}\ell_d(G_d(f_\phi(\mathbf{x})), d), \\ \mathbb{D}_{\mathrm{c}} &= \mathbb{E}_{c\sim\mathcal{C}}\mathbb{E}_{\mathbf{x}\sim B_s\cup B_t}\ell_d^{(c)}(G_d^{(c)}(\hat{y}^{(c)}f_\phi(\mathbf{x})), d), \end{aligned} \tag{6}$$

where ℓ_d is the cross-entropy loss for domain classification, and d is the domain label (0 or 1) of the input sample $\mathbf{x}$. $G_d^{(c)}$ and $\ell_d^{(c)}$ are the conditional domain discriminator and its cross-entropy loss associated with class c, respectively. $\hat{y}^{(c)}$ is the predicted label over the class c of the input sample $\mathbf{x}$.

Note that the target domain $\mathcal{D}_t$ has no labels, making it difficult to compute the conditional distance d_c. We apply prediction to $\mathcal{D}_t$ using the classifier G_y trained on $\mathcal{D}_s$ to obtain soft labels, which will be iteratively refined. Clearly, MMD and adversarial distance are only two options for predefined distance and others can be used. In specific problems, more task-dependent features can be used. This makes L2M a general and flexible framework.

A.2 Convergence analysis and theoretical insights

In addition to empirically analyzing the convergence of L2M, we provide some theoretical analysis. The convergence of L2M depends on two items: the classification loss $\mathcal{L}_{\mathrm{cls}}$ on the training data, and the distribution matching loss $\mathcal{L}_{\mathrm{match}}$ on the meta-data. The convergence of $\mathcal{L}_{\mathrm{cls}}$ is well ensured since it is a standard cross-entropy loss in deep neural networks. The convergence of $\mathcal{L}_{\mathrm{match}}$ depends on two factors: the construction of meta-data and the loss itself. We adopt an iterative way to construct the meta-data by using the pseudo labels provided by the trained network. According to several recent works [7, 10, 43], the convergence of such an iterative pseudo-label can be ensured, i.e., the pseudo labels will be more accurate, providing a strong support to the construction of the meta-data. On the other hand, the convergence of $\mathcal{L}_{\mathrm{match}}$ can also be ensured as long as the meta-network g_θ is differential (in our work, it is differential) by following [59, 60]. Therefore, the convergence of $\mathcal{L}_{\mathrm{match}}$ can be ensured.

In the view of domain adaptation theory, L2M is designed by following the DA theory according to [2] that the risk on the target domain is bounded by the following theorem:

Theorem 1 *Let $h \in \mathcal{H}$ be a hypothesis, $\epsilon_s(h)$ and $\epsilon_t(h)$ be the expected risks on the source and target domain, respectively, then*

$$\epsilon_t(h) \leqslant \epsilon_s(h) + d_{\mathcal{H}}(p, q) + C_0, \tag{7}$$

where C_0 is a constant for the complexity of hypothesis and plus the risk of an ideal hypothesis for both domains. $d_{\mathcal{H}}(p, q)$ refers to the distribution divergence between domains. As can be seen, L2M is directly minimizing the distribution distance (distribution matching loss) $\mathcal{L}_{\mathrm{match}}$, which is in consistence with the above theorem.

A.3 Remarks

L2M vs. AutoML. L2M shares the same goal with AutoML: both are trying to reduce the human intervention in a machine learning process. However, AutoML focuses more on “auto” while L2M

Table 1: Public datasets description.

Dataset	#Sample	#Class	#Domain
Office-31	4,110	31	A, W, D
ImageCLEF-DA	1,800	12	C, I, P
Office-Home	15,588	65	A, C, P, R
VisDA-2017	280,000	12	Synthetic, Real

can be seen as a combination of deep features and human-designed features. Moreover, AutoML focuses on architecture design, hyperparameter search, and channel pruning, which are different from L2M. The main goal of L2M is to learn a good and automatic distribution matching between domains. From this point of view, L2M can also be seen as an "automated" DA method. Future works may lay emphasis on domain adaptation architecture design, which is more like automl.

L2M is not yet another "SOTA" and is not intending replacing other methods. The results in this paper demonstrates that the performance of L2M outperforms several SOTA methods. However, our goal is not to develop yet another SOTA to the community, but to introduce another kind of DA algorithm that can be easily applied to real applications without specific concentration on the loss function design and distribution matching module. Therefore, for a new application, both L2M and other existing SOTA methods are applicable. The advantage of using L2M is that it requires less human intervention of algorithm selection, while a simple embedding matching feature can achieve a competitive performance. If you need better results, you still need to have a deep domain knowledge and integrate it in L2M with the embeddings or logits features. Therefore, L2M can be used to enhance other methods.

B Experimental Details

B.1 Datasets

ImageCLEF-DA. ImageCLEF-DA [44] is a benchmark dataset for ImageCLEF 2014 domain adaptation challenge, and it is collected by selecting the 12 common categories shared by the following public datasets and each of them is considered as a domain: $Caltech-256$ (**C**), $ImageNet\ ILSVRC$ 2012 (**I**), $Pascal\ VOC$ 2012 (**P**). There are 50 images in each category and 600 images in each domain. We use all domain combinations and build 6 transfer tasks.

Office-Home. Office-Home [45] consists of images from 4 different domains: $Artistic\ images$ (**A**), $Clip\ Art$ (**C**), $Product\ images$ (**P**) and $Real-World\ images$ (**R**). For each domain, the dataset contains images of 65 object categories collected in office and home settings. Similarly, we use all domain combinations and construct 12 transfer tasks.

VisDA-2017. VisDA-2017 [46] is a simulation-to-real dataset with two extremely distinct domains: Synthetic renderings of 3D models and Real collected from photo-realistic or real-image datasets. With 280K images in 12 classes, the scale of VisDA-2017 brings challenges to domain adaptation.

Office-31. Office-31 dataset [47] is a standard and maybe the most popular benchmark for unsupervised domain adaptation. It consists of 4,110 images within 31 categories collected from everyday objects in an office environment. It consists of three domains: $Amazon$ (**A**), which contains images downloaded from `amazon.com`, $Webcam$ (**W**) and $DSLR$ (**D**), which contain images respectively taken by web camera and digital SLR camera under different settings. We evaluate all our methods across six transfer tasks on all three domains.

The statistics of these datasets are shown in Table 1.

B.2 Implementation Details

For different variants of L2M using different matching features, we report the dimension information of eight matching features of each dataset in Table 2.

All methods use the ImageNet-pretrained ResNet-50 as the backbone network. Results of the comparison methods are obtained from original papers. For L2M, we set max iterations to be

Table 2: Dimension of matching features of the datasets.

Dataset	$\mathbf{F}_{\text{emb}}$	$\mathbf{F}_{\text{logit}}$	$\mathbf{F}_{\text{mmd}}$	$\mathbf{F}_{\text{adv}}$	$[\mathbf{F}_{\text{emb}}, \mathbf{F}_{\text{mmd}}]$	$[\mathbf{F}_{\text{emb}}, \mathbf{F}_{\text{adv}}]$	$[\mathbf{F}_{\text{logit}}, \mathbf{F}_{\text{mmd}}]$	$[\mathbf{F}_{\text{logit}}, \mathbf{F}_{\text{adv}}]$
Office-31	2,048	31	2	2	2,050	2,050	33	33
ImageCLEF-DA	2,048	12	2	2	2,050	2,050	14	14
Office-Home	2,048	65	2	2	2,050	2,050	67	67
VisDA-2017	2,048	12	2	2	2,050	2,050	14	14

Table 3: Accuracy (%) on Office-31 for UDA (ResNet-50).

Method	A→W	A→D	D→W	D→A	W→D	W→A	AVG
ResNet [48]	68.4	68.9	96.7	62.5	99.3	60.7	76.1
DDC [6]	75.6	76.5	96.0	62.2	98.2	61.5	78.3
DAN [5]	80.5	78.6	97.1	63.6	99.6	62.8	80.4
D-CORAL [26]	77.0	81.5	97.1	65.9	99.6	64.3	80.9
RTN [62]	84.5	77.5	96.8	66.2	99.4	64.8	81.6
DANN [8]	82.0	79.7	96.9	68.2	99.1	67.4	82.2
ADDA [9]	86.2	77.8	96.2	69.5	98.4	68.9	82.9
JAN [49]	85.4	84.7	97.4	68.6	99.8	70.0	84.3
MADA [50]	90.0	87.8	97.4	70.3	99.6	66.4	85.2
MEDA [7]	86.0	86.3	97.1	72.1	99.2	73.2	85.7
CAN [51]	92.5	90.1	98.8	72.1	100.0	69.9	87.2
DSR [63]	93.1	92.4	98.7	73.5	99.8	73.9	88.6
CDAN [43]	94.1	92.9	98.6	71.0	100.0	69.3	87.7
ALDA [64]	**95.6**	94.0	97.7	72.2	100.0	72.5	88.7
MDD [10]	94.5	93.5	98.7	74.6	100.0	72.2	88.9
L2M	93.2	**94.1**	**98.8**	**75.9**	**100.0**	**74.7**	**89.5**

200000. The mini-batch SGD with nesterov momentum of 0.9 and batchsize 32 is used as the optimization strategy. The learning rate α of the meta-model and the overall model changes by following [8]: $\alpha_k = \frac{\alpha}{(1+\gamma k)^{-\upsilon}}$, where k is the training iteration linearly changing from 1 to max iterations, $\gamma = 0.001$, $\alpha = 0.004$, and decay rate $\upsilon = 0.75$. The initial learning rate β of the meta-network is 0.01 and will gradually decrease to 0.0001 during training. Meta-network g_θ uses a $d - 1024 - 1024 - 1$ structure where d is the dimension of input matching features, and more information of different matching features can be seen in Table 2. We follow the standard protocols for unsupervised domain adaptation [61], we use classification accuracy on the target domain as the evaluation metric and target labels are only used for evaluation. The results are the average accuracy of 10 experiments by following the same protocol [6, 7, 10, 43]. We use Pytorch to implement L2M and it is trained on a Linux machine with a 16GB P100 GPU.

B.3 Results on Office-31 dataset

Table 3 reports the results on Office-31, which indicates that L2M outperforms all the recent DA methods in classification accuracy.

B.4 More ablation experiments of L2M

We show more ablation experiments of L2M on Office-Home and ImageCLEF-DA in Table 4 and Table 5, respectively. We did not run ablation experiments on VisDA-17 since this dataset is rather larger and needs more computations. The ablation results on other datasets are enough for observing the patterns of L2M variants. Combining these results with that from the main paper, more insightful conclusions can be made. **(1)** L2M achieves the best performance on multiple datasets, which indicates the efficiency of L2M. **(2)** All the 4 variants of L2M can achieve competitive performance, implying the effectiveness of the meta-network on matching functions and L2M is able to fit a wide range of matching features. **(3)** L2M (emb+adv) outperforms the other 3 variants of L2M, which indicates L2M can learn more representative and transferable features by taking as input deep features and human-designed features.

Despite the performance on these public datasets, we want to emphasis that in real applications, L2M (emb+adv) is perhaps not always the best matching features. Therefore, in order to achieve the best performance, users can try several combinations of matching features along with their own domain experience before finding the suitable features. Since the performance of most matching features are with a low variance, any matching feature can achieve competitive performance compared to existing methods.

Table 4: Accuracy (%) on Office-Home for UDA (ResNet-50).

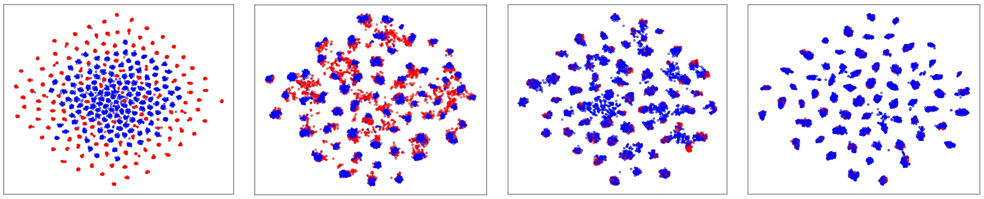
Method	A→C	A→P	A→R	C→A	C→P	C→R	P→A	P→C	P→R	R→A	R→C	R→P	AVG
L2M (emb)	55.8	73.9	77.8	61.4	72.9	73.1	62.5	54.5	79.3	71.1	60.2	83.2	69.1
L2M (logit)	56.2	70.2	71.7	57.7	68.1	71.1	60.7	57.1	79.4	71.6	61.5	83.6	67.4
L2M (logit+adv)	55.4	70.6	76.6	58.1	69.2	70.1	61.1	55.5	78.6	71.4	60.7	82.8	67.5
L2M (emb+adv)	56.1	75.0	79.3	62.8	73.3	73.7	63.8	54.1	80.6	71.8	60.2	83.5	69.5

Table 5: Accuracy (%) on ImageCLEF-DA for UDA (ResNet-50).

Method	I→P	P→I	I→C	C→I	C→P	P→C	AVG
L2M (emb)	78.0	90.5	96.2	91.4	78.3	94.1	88.1
L2M (logit)	76.2	88.0	96.8	89.5	76.8	94.5	87.0
L2M (logit+adv)	77.0	89.2	95.7	89.8	77.3	93.3	87.1
L2M (emb+adv)	78.7	91.0	97.0	92.0	79.7	96.0	89.1

B.5 Feature visualization.

We visualize the network activation (before FC layer) on task P→R using t-SNE in Fig. 2. ResNet-50 does not align the distributions. JAN aligns both marginal and conditional distributions with equal weights, while MEDA adaptively aligns these two distributions whose results are better. However, the source and target domains are not fully matched by MEDA. For L2M, both the cross-domain distributions and categories are aligned well, implying that L2M learns more discriminative features.



(a) ResNet50 (b) JAN (c) MEDA (d) L2M

Figure 2: (Best viewed in color) The t-SNE visualization of network activation on task P→R. Red circles are the source samples and blue circles are the target samples.

C Application to COVID-19 Chest X-ray Image Adaptation

Other than benchmarking L2M on popular public datasets including Office-31, Office-Home, ImageCLEF-DA, and VisDA-17, we compare the performance of several DA methods including L2M in a real application. Different from public datasets, this application will prove the effectiveness of L2M and other DA methods in a real-world task, which is more appealing and inspiring.

We present more details for applying L2M to COVID-19 chest X-ray image adaptation tasks. COVID-19 is a specific type of pneumonia compared to the normal kind of pneumonia, and there is not too much COVID-19 data available, it becomes necessary and feasible to use the sufficient labeled pneumonia data to help classify the COVID-19 symptom. Therefore, this task is a binary classification task, where the source domain is the well-labeled pneumonia data to classify whether this patient is having pneumonia or not, and the target domain is the unlabeled COVID-19 data. Out task is to classify whether each of the the target domain samples is having a COVID-19 symptom or not.

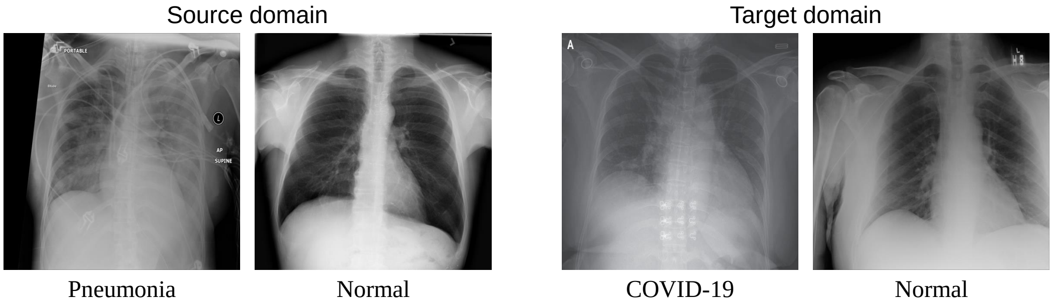
This is a binary classification task, i.e., the normal category vs. pneumonia on the source domain, and the normal category vs. COVID-19 on the target domain. We also notice that this dataset is highly-imbalanced (as shown in the next section). Therefore, for better illustrate the results, we adopt F1 score, Recall, and Precision as the evaluation metrics rather than classification accuracy. These metrics are better for imbalanced classification tasks. It also demonstrates our contribution that L2M can achieve robust preformance in imbalanced tasks compared to other DA methods.

C.1 Dataset

Table 6 shows the description of the dataset[2]. Note that in this task, we use some COVID-19 data as the validation set to better tune the hyperparameters. In the source domain, there are two classes: normal and pneumonia, while there are normal and COVID-19 classes in the target and validation dataset. Fig. 3 shows some examples from the source and target domain. It is clear that data from two domains are very similar especially for pneumonia and COVID-19 classes. Therefore, it is feasible to perform domain adaptation or transfer learning between these two domains.

Table 6: Dataset description of pneumonia and COVID-19 dataset

Domain	Symptom	#Normal	#Pneumonia	#COVID-19	#Total
Source	Pneumonia	5,613	2,306	0	7,919
Target	COVID-19	885	0	60	945
Validation	COVID-19	254	0	25	279


Source domain

Target domain

Pneumonia

Normal

COVID-19

Normal

Figure 3: Samples from the source and target domain.

C.2 Baselines and experimental details

We mainly compare the performance of L2M with three categories of methods: (1) Deep learning baselines, (2) Deep diagnostic methods, and (3) unsupervised domain adaptation methods. The deep learning baselines include three baselines:

- Train on source: train a network on the source domain, and then apply the pretrained model on the target domain.
- Train on target: this is an ideal state since there are no labels for the target domain in our task. Therefore, we directly use several extra labeled COVID-19 data from the dataset (they are 30% of the target domain data) and train a network on these data. Then, we can apply prediction on the target data.
- Fine-tuning: This is a combination of the above two baselines. Firstly, we train a network on the source domain. Then, we fine-tune the pretrained model on the extra labeled target domain data. Finally, we apply prediction on the target data.

The deep diagnostic method is DLAD [55].

The unsupervised DA methods are DANN [8], MCD [53], and CDAN+TransNorm [39]. All methods are using ResNet-18 as the backbone network following [56]. The results of these methods are obtained from COVID-DA [56] to ensure a fair comparison. Note that we did not compare COVID-DA since this method is a semi-supervised method that explicitly uses the labeled data on the target domain. Therefore, we only use the report of unsupervised methods and train L2M with the same experimental settings.

[2] The dataset is available at https://github.com/qiuzhen8484/COVID-DA

C.3 Analysis of the results

The results are shown in Table 7. Here we use the 95% confidence interval, where the corresponding value of z is 1.96. The computed confidence interval r is around 1.3%.

This table is the same as the main paper but with more analysis of the results. From the results, we see that L2M outperforms all comparison methods in terms of F1 score and Recall. In Precision, the performance of Training on labeled target data achieves the best results, which is reasonable since this approach trains on the labeled target domain data and is expected to achieve the best precision. The UDA methods, namely DANN, MCD, and CDAN+TransNorm can sometimes achieve worse results than baselines, indicating that the different distribution distance of pneumonia and COVID-19 data are not that easy to compute by adversarial distance (DANN and MCD are using adversarial distances) or statistical alignment (TransNorm uses a source-target normalization technique) since these methods are built with their own priors and biases. In this situation, it is necessary to perform domain adaptation in a data-driven way stepping back from these predefined distances. Therefore, L2M can be useful in real-world applications. It also demonstrates our contribution that L2M can achieve robust preformance in imbalanced tasks compared to other DA methods.

Table 7: Results on COVID-19 X-ray adaptation (normal pneumonia → COVID-19, ResNet-18). Here we use the 95% confidence interval, where the corresponding value of z is 1.96. The computed confidence interval r is around 1.3%.

Method	Precision (%)	Recall (%)	F1 (%)
Train on source	63.5	66.7	65.0
Train on target *(ideal state)*	**91.7**	55.0	68.8
Fine-tuning	56.3	75.0	64.3
DLAD [55]	62.0	73.3	67.2
DANN [8]	61.4	71.7	66.2
MCD [53]	63.2	60.0	61.5
CDAN+TransNorm [39]	85.0	39.2	63.7
L2M	70.1	**78.3**	**74.0**

On the other hand, we also notice that the performance of fine-tuning is worse than training on source and target, which is probably due to the distribution gap between the source and target domains. In a nutshell, among baselines and DA methods other than L2M, training on target achieves the best performance, indicating the importance of labeled data. We can also see that in COVID-DA [56], authors used semi-supervised settings to improve the F1, Precision, and Recall score to over 90%, which clearly shows optimistic performance. Therefore, L2M and other methods can also be applied to semi-supervised DA tasks by adopting several labeled target samples. This is left for future work since this work mainly focuses on unsupervised DA.

C.4 Ablation study

We show the ablation study of L2M on this COVID-19 data in Table 8. It is shown that by combining different matching features, L2M can generally achieve better performance than the comparison methods.

Table 8: Ablation experiments of L2M on COVID-19 experiment. Here we use the 95% confidence interval, where the corresponding value of z is 1.96. The computed confidence interval r is around 1.3%.

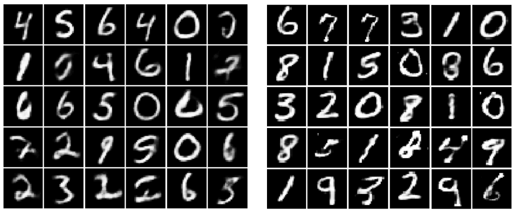
L2M variant	F1	Recall	Precision
L2M (emb)	73.2	75.0	71.4
L2M (logit)	68.3	70.0	66.7
L2M (emb+mmd)	69.1	78.3	61.8
L2M (logit+mmd)	74.0	78.3	72.3
L2M (emb+adv)	68.9	65.4	78.5
L2M (logit+adv)	69.4	71.7	67.2
L2M (mmd)	71.2	70.0	72.4
L2M (adv)	65.5	65.0	66.1

D Details for Image Generation

We train GMMNs on the benchmark datasets MNIST [65]. We use the standard test set of 10,000 images, and randomly select 5000 from the standard 60,000 training images for validation. The remaining 55,000 are used for training. We train the GMMN network in both the input data space and code space of an auto-encoder. For all the networks, a uniform distribution in [-1, 1]^H is used as the prior for the H-dimensional stochastic hidden layer at the top of the GMMN, which is followed by 4 ReLU layers. The output layer is a logistic sigmoid function, which guarantees that the code space dimensions lay in [0, 1]. The auto-encoder has 4 layers, 2 for the encoder and 2 for the decoder. For more details about the architecture of GMMN and auto-encoder, please refer to the original paper [57].

We train the GMMNs with mini-batch of size 1000, for each mini-batch, a set of 1000 samples will be generated from the network. The loss and gradient are computed from these 2000 samples. We replace the original square root loss function $\mathcal{L}_{\text{MMD}}$ with $\mathcal{L}_{\text{match}}$ of L2M to get the result GMMN+L2M. We set max epochs to be 500 and use Adam as the optimization strategy. The learning rate and momentum for both GMMN and auto-encoder, dropout rate for the auto-encoder are tuned using Bayesian optimization [66].

Fig. 4 shows more MNIST samples generated by GMMN with MMD and L2M. It is clear that L2M generates sharper samples than MMD. We believe that L2M has more potential in image generation and this is only a test experiment. We are well aware that there are lots of existing works for image generation these years and adhere to hope that L2M could be significantly extended for this task in the future.



(a) GMMN

(b) L2M

Figure 4: More MNIST samples generated by GMMN and L2M.