Distribution-Informed Neural Networks for Domain Adaptation Regression

Jun Wu, Jingrui He, Sheng Wang, Kaiyu Guan, Elizabeth Ainsworth
University of Illinois Urbana-Champaign
{junwu3,jingrui,sheng12,kaiyug,ainswort}@illinois.edu

Abstract

In this paper, we study the problem of domain adaptation regression, which learns a regressor for a target domain by leveraging the knowledge from a relevant source domain. We start by proposing a distribution-informed neural network, which aims to build distribution-aware relationship of inputs and outputs from different domains. This allows us to develop a simple domain adaptation regression framework, which subsumes popular domain adaptation approaches based on domain invariant representation learning, reweighting, and adaptive Gaussian process. The resulting findings not only explain the connections of existing domain adaptation approaches, but also motivate the efficient training of domain adaptation approaches with overparameterized neural networks. We also analyze the convergence and generalization error bound of our framework based on the distribution-informed neural network. Specifically, our generalization bound focuses explicitly on the maximum mean discrepancy in the RKHS induced by the neural tangent kernel of distribution-informed neural network. This is in sharp contrast to the existing work which relies on domain discrepancy in the latent feature space heuristically formed by one or several hidden neural layers. The efficacy of our framework is also empirically verified on a variety of domain adaptation regression benchmarks.

1 Introduction

Domain adaptation tackles the knowledge transfer from a source domain with adequate label information to a relevant target domain with little or no label information. It is shown [5, 39, 11, 13] that the generalization performance on the target domain can be improved by leveraging the source domain knowledge in both classification [59] and regression [12] settings.

In this paper, we focus on studying the domain adaptation regression problem, as classification can be naturally formulated as regression [34]. In the past decades, most domain adaptation approaches were developed using the following paradigms: learning domain invariant representation [42, 43], reweighting the source examples [30, 11, 13], or deriving adaptive transfer kernel for Gaussian process [7, 44]. More recently, modern practice for domain adaptation repeatedly demonstrates the benefit of considering overparameterized neural networks as the backbones for feature extraction [37, 20, 47, 1]. In this case, a L-layer neural network would be manually divided into two parts: the feature extraction function (e.g., the first l layers) and the prediction function (e.g., the left $L - l$ layers). Then, the core idea of domain adaptation with overparameterized neural networks is to enhance the adaptation of source and target domains in the hidden feature space learned by the feature extraction layers. Nevertheless, it is unclear how those overparameterized neural networks affect the convergence and generalization of domain adaptation approaches during model training. Moreover, the heuristic selection of feature extraction layers might lead to suboptimal performance of domain adaptation approaches in practice, as the lower and higher layers of an overparameterized neural network might encode different information [57].

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Compared with standard supervised learning, the key challenge of domain adaptation lies in the distribution shift between source and target domains. This indicates that the relationship between inputs and outputs might be different across domains. As illustrated in Figure 1, due to the distribution shift across domains, an input data point (e.g., $x_2 = x_3$) might have significantly different outputs at source and target domains (i.e., $y_2 \neq y_3$). This is also observed in recent work [51]. To solve this problem, we propose a distribution-informed neural network to build the unified relationship of inputs and outputs from different domains. This neural network can be explained as the integration of standard feature representation learning and input-oriented distribution representation learning. Then, a simple domain adaptation regression framework named `DINO` is proposed based on the distribution-informed neural network, followed by the theoretical analysis on its convergence and generalization performance.

In particular, we show that the distribution discrepancy of source and target domains can be defined by the evolution of distribution-informed neural network during training. That is, we measure the distribution shift using the maximum mean discrepancy (MMD) [24] in the reproducing kernel Hilbert space (RKHS) induced by the neural tangent kernel of distribution-informed neural network. Compared to previous works [37, 20, 1], our domain discrepancy measure has the following benefits. First, it can be estimated from the entire neural network, whereas previous works usually estimate the domain discrepancy in the pre-defined feature extraction layers. For example, [37] extracts features of input source and target examples and then estimates the discrepancy using MMD with standard kernels (e.g., RBF kernel). Intuitively, it is a two-fold composition kernel of the neural kernel induced by the feature extractor and the standard kernel. Second, our domain discrepancy measure focuses on the training dynamics of the neural network (e.g., $f_{\theta+\Delta\theta}(x) - f_\theta(x)$). This is in sharp contrast to the previous works which measure the distribution shift using the static state of the neural network (e.g., $f_\theta(x)$). As shown in Figure 1, a source data point x_3 might have similar output with both x_1 and x_4 from target domain, when simply considering the trained neural network $f_\theta(x)$. But we see that the neural network has similar evolution patterns at x_3 and x_4 but different patterns at x_3 and x_1. That is, x_3 is more likely be aligned with x_4 after distribution minimization across domains, That explained why the training dynamics of the neural network can better identify the distribution shift across domains.

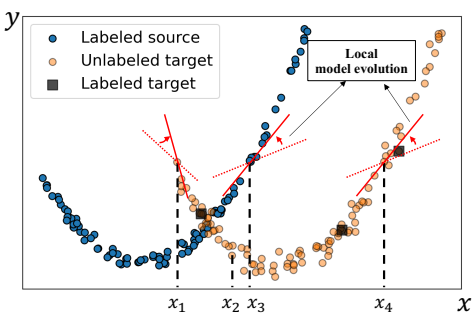


Figure 1: Illustration of the domain adaptation regression, where x_1, x_3 denotes two source data points and x_2, x_4 denotes two target data points. Here solid and dotted lines denote two consecutive states of the neural network around data points during training. (1) Similar data points (e.g., x_2 and x_3 from different domains) might have different outputs. (2) Source data point x_3 is more likely to be aligned with target data point x_4, as they have similar local model evolution.

In addition, we show that in special cases, the training dynamic of the distribution-informed neural network can also explain the rationale of the existing domain adaptation techniques. To be more specific, we have the following observations. (1) At random initialization, our distribution-informed neural network is equivalent to the adaptive Gaussian process [7, 44]. (2) Under gradient descent training, our domain adaptation framework based on the distribution-informed neural network can recover the prediction function of the reweighting domain adaptation approaches [30, 11, 13]. (3) When the distribution representation is shared by all the examples, our framework degenerates into standard domain-invariant representation learning [37, 20, 1]. Compared to previous works, our contributions can be summarized as follows.

- *Unified Framework:* We propose a simple domain adaptation regression framework `DINO` based on the distribution-informed neural network, and then identify its connections to previous techniques, including domain invariant representation learning, reweighting and adaptive Gaussian process.
- *Theoretical Results:* It is shown that for the distribution-informed neural network with sufficient width, the convergence and generalization of our framework can be theoretically guaranteed. Different from previous deep domain adaptation theories, we propose to measure the distribution shift across domains using the training dynamics of the neural network.

- *Empirical Performance:* We experimentally investigate the performance of our DINO framework on a variety of domain adaptation regression benchmarks, and show its effectiveness over state-of-the-art baselines.

The rest of this paper is organized as follows. We introduce the related work in Section 2, and summarize the preliminaries of previous domain adaptation techniques in Section 3. Section 4 shows the proposed domain adaptation regression framework, followed by its theoretical analysis. The experimental results are presented in Section 5. Finally, we conclude the paper in Section 6.

2 Related Work

2.1 Domain Adaptation

Domain adaptation [5, 51, 52, 61, 62] improves the generalization performance of a learning algorithm on the target domain, by leveraging the knowledge from a relevant source domain with adequate labeled data. There are three solutions to bridge the distribution gap between source and target domains. (1) *Domain invariant representation:* It maps the source and target examples into a new feature space such that the distribution discrepancy between source and target domains can be explicitly minimized [42, 37, 43, 20, 60, 59, 10]. (2) *Reweighting:* The core idea is to correct the difference between distributions by multiplying the prediction loss of each source example by a non-negative weight [30, 14, 11, 13, 47]. (3) *Gaussian process:* It generalized the standard Gaussian process regression to the domain adaptation setting by learning an adaptive transfer kernel [7, 44, 38, 50]. By using the neural networks as the backbones, modern domain adaptation techniques have achieved state-of-the-art performance in a variety of real-world tasks [28, 8]. Nevertheless, little effort has been devoted to theoretically analyzing the domain adaptation techniques with overparameterized neural networks.

More recently, it has been revealed [10] that there is a gap between domain adaptation regression and domain adaptation classification problems. That is, one common loss function of domain adaptation regression approaches is mean square error (MSE), whereas domain adaptation classification approaches [37, 20] often uses cross-entropy loss with softmax. Specifically, softmax changes the feature scales (i.e., Frobenius norm of feature matrix $||H||_F$ where H is the hidden feature learned by the neural network), and the change of feature scales might lead to the performance degradation of domain adaptation [10]. Therefore, in this paper, we focus on the neural network with MSE loss for domain adaptation regression. Different from the existing domain adaptation regression approaches, e.g., domain invariant representation learning [10], reweighting [16], and adaptive Gaussian process [7], we develop the domain adaptation regression algorithms based on a novel distribution-informed neural network and derive the theoretical analysis regarding the convergence and generalization bound for our algorithms.

2.2 Overparameterized Neural Networks

It has been revealed from the perspective bias-variance decomposition [55] that the generalization error of neural networks decreases with respect to the model complexity (e.g., number of neurons). The benefit of increasing the number of neurons has also been empirically confirmed in modern neural networks [32, 58]. Moreover, the convergence and generalization of overparameterized neural networks have been studied in the neural tangent kernel (NTK) regime [31, 3, 4, 35, 17, 18, 2, 48, 9, 33]. However, those works focus on the standard supervised learning with the assumption that training and testing examples follow the same distribution. In real scenarios, this assumption often fails due to the distribution shift [38]. More recently, it is found that previous theoretical results can be generalized to both federated learning [29] and multi-task learning [45]. Our work fundamentally differs from them in that little label information is available in the target domain for our studied domain adaptation problem. This motivates us to explicitly analyze the domain discrepancy when using neural networks for domain adaptation.

3 Preliminaries

In this section, we introduce the notations and background of different domain adaptation techniques.

3.1 Notation and Neural Network Architecture

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input space and output space. Following [6], we assume that the joint probability distribution $\mathbb{P}$ of any domain is drawn from a probability distribution space $\mathscr{P}$ over $\mathcal{X} \times \mathcal{Y}$. For domain adaptation regression, we have a source domain with labeled examples $\{x_i^{\text{src}}, y_i^{\text{src}}\}_{i=1}^{n_{\text{src}}}$ drawn from joint probability distribution $\mathbb{P}^{\text{src}}$, and a target domain with both label examples $\{x_j^{\text{tgt}}, y_j^{\text{tgt}}\}_{j=1}^{n_{\text{tgt}}^l}$ from $\mathbb{P}^{\text{tgt}}$ and unlabeled examples $\{x_j^{\text{tgt}}\}_{j=1}^{n_{\text{tgt}}^u}$ ($n_{\text{tgt}}^l \ll n_{\text{tgt}}^u$) drawn from marginal probability distribution $\mathbb{P}_X^{\text{tgt}}$. The goal is to predict the outputs of unlabeled target examples by leveraging the label information from the source domain. In this paper, we consider an L-layer fully-connected neural network $f_\theta(\cdot)$, which maps the input $x \in \mathcal{X} \subset \mathbb{R}^d$ into the output $y \in \mathcal{Y} \subset \mathbb{R}$. Here θ denotes all the parameters of $f(\cdot)$ (e.g., θ_0 denotes the initialized parameters at time stamp $t = 0$) and d denotes the dimensionality of input data. In next section, we generalize the standard fully-connected neural network $f_\theta(\cdot)$ to the distribution-informed neural network $\tilde{f}(\cdot)$ for domain adaptation. For notation simplicity, we let $X^{\text{src}} = \{x_i^{\text{src}}\}_{i=1}^{n_{\text{src}}}$ and $Y^{\text{src}} = \{y_i^{\text{src}}\}_{i=1}^{n_{\text{src}}}$ for source examples. We use similar notations X_l^{tgt} and Y_l^{tgt} for target examples. Then we use $X = X^{\text{src}} \cup X_l^{\text{tgt}}$ denote all the labeled training inputs, and $Y = Y^{\text{src}} \cup Y_l^{\text{tgt}}$ be the corresponding outputs.

3.2 Domain Adaptation Techniques

We first review the existing domain adaptation techniques. In Subsection 4.4, we show that these techniques can be explained in a unified framework.

3.2.1 Domain Invariant Representation

The key idea of domain invariant representation learning is to map the source and target examples into a new feature space where the distribution discrepancy across domains is explicitly minimized [20, 59]. The objective function of this framework can be summarized as follows.

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathbb{P}^{\text{src}}} \left[L(f_\theta(x), y)\right] + d\left(\mathbb{P}^{\text{src}}, \mathbb{P}^{\text{tgt}}\right) \tag{1}$$

where $d\left(\mathbb{P}^{\text{src}}, \mathbb{P}^{\text{tgt}}\right)$ denotes the domain discrepancy measure in the hidden feature space learned by $f_\theta(\cdot)$, and $L(\cdot, \cdot)$ is the loss function.

3.2.2 Reweighting

Reweighting aims to correct the difference between source and target distributions by reweighting the source examples, i.e., multiplying the prediction loss of every source example by a non-negative weight [14, 11, 13].

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathbb{P}^{\text{src}}} \left[w^{\text{src}}(x, y) \cdot L(f_\theta(x), y)\right] \tag{2}$$

where $w^{\text{src}}(x, y) = \frac{\mathbb{P}^{\text{tgt}}(x,y)}{\mathbb{P}^{\text{src}}(x,y)} \geq 0$ denotes the importance of every source example. The importance ratio $w^{\text{src}}(x, y)$ can be estimated using kernel mean matching (KMM) [30, 56] or adversarial learning [47, 16].

3.2.3 Adaptive Gaussian Process

Following [7], the domain adaptation regression problem can also be formulated as the Gaussian process with an adaptive transfer kernel. Specifically, the adaptive transfer kernel can be defined as

$$K' = \begin{bmatrix} K(X^{\text{src}}, X^{\text{src}}) & \tau \cdot K(X^{\text{src}}, X_l^{\text{tgt}}) \\ \tau \cdot K(X_l^{\text{tgt}}, X^{\text{src}}) & K(X_l^{\text{tgt}}, X_l^{\text{tgt}}) \end{bmatrix} \tag{3}$$

where $\tau \in [0, 1]$ is a trainable parameter explicitly indicating the relatedness of source and target domains, and $K(\cdot, \cdot)$ is a base kernel, e.g., RBF kernel. Then following the standard Gaussian process regression [49], the predictions of this adaptive Gaussian process over testing target data can be analytically derived.

4 A Unified Framework

In this section, we propose the distribution-informed neural network, and then present a simple domain adaptation regression framework.

4.1 Distribution-Informed Neural Network

An L-layer fully-connected neural network $f(\cdot)$ can be written as $f_\theta(x) = \phi_{\theta^{<L}}(x)^T w$ where $\theta^{<L}$ is the vector of parameters in the first $L-1$ layers and w is the parameter of the output layer (we assume the output layer has no bias). For a single task, the neural network $f(\cdot)$ can model the relationship between input data and output label. But it might suffer in the domain adaptation scenarios due to the distribution shift. Thus, given a domain distribution $\mathbb{P} \in \mathscr{P}$ and an input example $x \sim \mathbb{P}$, we propose to explicitly learn the distribution-related output as $g_{w_g}(\mathbb{P}|x) = \Phi_x(\mathbb{P})^T w_g$ where $\Phi_x(\mathbb{P})$ is the input-oriented distribution feature representation, and w_g is the parameter. Given finite basis examples $\tilde{x}_1, \cdots, \tilde{x}_n \sim \mathbb{P}$, $\Phi_x(\mathbb{P})$ can be formally defined as follows.

$$\Phi_x(\mathbb{P}) = \sum_{i=1}^{n} \beta_{x,\tilde{x}_i} \langle \cdot, \tilde{x}_i \rangle_{\mathcal{K}_\mathcal{X}} \tag{4}$$

where $\beta_{x,\tilde{x}_i} \in \mathbb{R}$ is the coefficient of $\Phi_x(\mathbb{P})$ in the space spanned by $\langle \cdot, \tilde{x}_i \rangle_{\mathcal{K}_\mathcal{X}}$ and indicates the similarity of input example x and basis example $\tilde{x}_i$. Here $\langle \cdot, \tilde{x}_i \rangle_{\mathcal{K}_\mathcal{X}}$ denotes the feature map of $\tilde{x}_i$ in the NNGP kernel space induced by infinitely-wide $f(\cdot)$. Therefore, $\Phi_x(\mathbb{P})$ represents the distribution representation of $\mathbb{P}$ when an example x is observed in the domain associated with sampling distribution $\mathbb{P}$.

One special case is when $\beta_{x,\tilde{x}_i} = 1/n$, it degenerates into the mean mapping of $\mathbb{P}$ in the kernel space, which is commonly used for measuring the distribution discrepancy/similarity [24]. Compared with this plain distribution representation $\frac{1}{n}\sum_{i=1}^{n} \langle \cdot, \tilde{x}_i \rangle_{\mathcal{K}_\mathcal{X}}$, $\Phi_x(\mathbb{P})$ pays more attention to the region around x with large $\beta_{x,\tilde{x}_i}$ for $\tilde{x}_i \in \mathcal{X}$. For example, for data points x_1 and x_4 in Figure 1, they would share the same plain distribution representation, as they are sampled from the same target domain. But it is feasible to differentiate them using the parametric distribution representation $\Phi_x(\mathbb{P})$.

Generally, by taking both example x and its associated probability $\mathbb{P}$ as random variables, Bayes' theorem tells us that $P(x, \mathbb{P}) = P(x) \cdot P(\mathbb{P}|x)$ where $P(\cdot)$ denotes the probability of an observed event. The event involving x can be described by observing its representation $f(x)$; given x, the event involving $\mathbb{P}$ is the input-oriented distribution representation $\Phi_x(\mathbb{P})$. This motivates us to develop a distribution-informed neural network as follows.

$$\tilde{f}(x, \mathbb{P}) := f_\theta(x) \cdot g_{w_g}(\mathbb{P}|x) = \left(\phi_{\theta^{<L}}(x)^T w\right) \cdot \left(\Phi_x(\mathbb{P})^T w_g\right) = w^T \left(\phi_{\theta^{<L}}(x) \Phi_x(\mathbb{P})^T\right) w_g \tag{5}$$

The intuition behind Eq. (5) can be explained as follows. For domain adaptation, the feature representation of an input example x is domain-dependent, as two similar examples ($x^{\text{src}} \approx x^{\text{tgt}}$) from different domains might have distinctive outputs ($y^{\text{src}} \neq y^{\text{tgt}}$). The input-oriented distribution representation $\Phi_x(\mathbb{P})$ allows us to identify the source and target examples with similar inputs but distinctive outputs. By analyzing the distribution-informed neural network at initialization and under gradient descent training, we propose two domain adaptation algorithms: `DINO-INIT` (see Subsection 4.2) and `DINO-TRAIN` (see Subsection 4.3), respectively.

4.2 Initialization

We start by studying the distribution-informed neural network at initialization, e.g., all the parameters of distribution-informed neural network $\tilde{f}(\cdot)$ are initialized as standard normal variables. Inspired by the connections between standard neural networks and Gaussian processes [15, 34, 22, 53], we show that the distribution-informed neural network $\tilde{f}(x, \mathbb{P})$ in Eq (5) is equivalent to a domain adaptive Gaussian process, which explains the existing domain adaption regression algorithm [7].

Lemma 4.1. *Assume that all the parameters of distribution-informed neural network $\tilde{f}(\cdot)$ are initialized as standard normal variables, when the network width goes to infinity, the output function of $\tilde{f}(x)$ in Eq (5) at initialization is iid centered Gaussian process, i.e., $\tilde{f}(\cdot) \sim \mathcal{N}\left(0, \mathcal{K}^{DA}\right)$ with*

$$\mathcal{K}^{DA}\left((x, \mathbb{P}), (x', \mathbb{P}')\right) = \mathcal{K}_\mathcal{X}(x, x') \cdot \mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x')$$

where $\mathcal{K}_\mathcal{X}(\cdot, \cdot)$ is the NNGP kernel induced by the neural network $f(\cdot)$ over input space $\mathcal{X}$, and $\mathcal{K}_\mathscr{P}(\cdot, \cdot)$ is a distribution kernel identifying the similarity of two input-oriented distributions $\mathbb{P}$ and $\mathbb{P}'$ over distribution space $\mathscr{P}$:

$$\mathcal{K}_{\mathscr{P}|\mathcal{X}}\left(\mathbb{P}, \mathbb{P}'|x, x'\right) = \sum_{i=1}^{n} \sum_{j=1}^{n'} \beta_{x,\tilde{x}_i} \beta_{x',\tilde{x}'_j} \mathcal{K}_\mathcal{X}(\tilde{x}_i, \tilde{x}'_j) \tag{6}$$

where n (n') is the number of basis examples sampled from the distribution $\mathbb{P}$ ($\mathbb{P}'$).

This lemma motivates us to propose an adaptive Gaussian process regression algorithm named DINO-INIT (see Algorithm 1 in Appendix A.10) with adaptive transfer kernel defined as $\mathcal{K}^{DA}$. For notation simplicity, we represent the distribution-informed kernel $\mathcal{K}^{DA}((x,\mathbb{P}),(x',\mathbb{P}'))$ as $\mathcal{K}^{DA}(x,x')$ in the following. In this case, we consider a noisy model $y_i^r = \tilde{f}(x_i^r,\mathbb{P}^r) + \epsilon_i^r$ ($r \in \{\text{src},\text{tgt}\}$), where the noise ϵ_i^r follows a zero-mean Gaussian $\mathcal{N}(0,\sigma_r^2)$. Given labeled training examples (X,Y), we assume a prior Gaussian process $p(Y) = \mathcal{N}(0,\mathcal{K}^{DA})$ where $\mathcal{K}^{DA}(X,X)$ is a block matrix. That is, $\mathcal{K}^{DA}(X,X) = \begin{bmatrix}\mathcal{K}_{11}^{DA} & \mathcal{K}_{12}^{DA}\\ \mathcal{K}_{21}^{DA} & \mathcal{K}_{22}^{DA}\end{bmatrix}$ where $\mathcal{K}_{11}^{DA} := \mathcal{K}^{DA}(X^{\text{src}},X^{\text{src}})$ ($\mathcal{K}_{22}^{DA} := \mathcal{K}^{DA}(X_l^{\text{tgt}},X_l^{\text{tgt}})$) denotes the kernel matrix of source (target) data, and $\mathcal{K}_{12}^{DA} = (\mathcal{K}_{21}^{DA})^T := \mathcal{K}^{DA}(X^{\text{src}},X_l^{\text{tgt}})$ denotes the kernel matrix across domains. Then for testing target examples X_*^{tgt}, their output can be inferred using the predictive distribution $p(Y|X_*^{\text{tgt}},X^{\text{src}},X_l^{\text{tgt}}) = \mathcal{N}(\bar{\mu},\bar{\Sigma})$. Similar to standard Gaussian process regression [49], the mean and variance of the predictive distribution can be exactly calculated as follows.

$$\bar{\mu} = \mathcal{K}^{DA}(X_*^{\text{tgt}},X)C^{-1}Y \qquad \bar{\Sigma} = \mathcal{K}^{DA}(X_*^{\text{tgt}},X_*^{\text{tgt}}) - \mathcal{K}^{DA}(X_*^{\text{tgt}},X)C^{-1}\mathcal{K}^{DA}(X_*^{\text{tgt}},X)^T$$

where $C = \mathcal{K}^{DA}(X,X) + \begin{bmatrix}\sigma_{\text{src}}^2\mathbb{I}_{n_{\text{src}}} & 0\\ 0 & \sigma_{\text{tgt}}^2\mathbb{I}_{n_{\text{tgt}}^l}\end{bmatrix}$. Here $\mathbb{I}$ denotes the identity matrix.

This adaptive Gaussian process involves the following parameters: coefficient β_{x,x_i} in the kernel function $\mathcal{K}^{DA}$ and noise variance σ_{src}, σ_{tgt}. Following [7], we optimize these parameters by maximizing the conditional likelihood $p(Y_l^{\text{tgt}}|X_l^{\text{tgt}},X^{\text{src}},Y^{\text{src}})$. It can be seen that $p(Y_l^{\text{tgt}}|X_l^{\text{tgt}},X^{\text{src}},Y^{\text{src}})$ is also a Gaussian process, i.e., $p(Y_l^{\text{tgt}}|X_l^{\text{tgt}},X^{\text{src}},Y^{\text{src}}) = \mathcal{N}\Big(\mathcal{K}_{21}^{DA}\left(\mathcal{K}_{11}^{DA}+\sigma_{\text{src}}^2\mathbb{I}_{n_{\text{src}}}\right)^{-1}Y^{\text{src}}, \Big(\mathcal{K}_{22}^{DA}+\sigma_{\text{tgt}}^2\mathbb{I}_{n_{\text{tgt}}^l}\Big) - \mathcal{K}_{21}^{DA}\left(\mathcal{K}_{11}^{DA}+\sigma_{\text{src}}^2\mathbb{I}_{n_{\text{src}}}\right)^{-1}\mathcal{K}_{12}^{DA}\Big)$.

Instantition of the coefficient $\beta_{x,\tilde{x}_i}$: It is notable that it can only optimize the coefficient $\beta_{x,\tilde{x}_i}$ of Eq. (4) for training (source or target) examples. The coefficient of testing target examples X_*^{tgt} is unknown. To solve this problem, in this paper, we formulate the estimate of $\beta_{x^r,\tilde{x}_i^r}$ ($r \in \{\text{src},\text{tgt}\}$) as $\beta_{x,\tilde{x}_i^r} = [x^r \circ \tilde{x}_i^r]^T w^r$, where w^r is a domain-specific parameter vector and $\circ$ denotes vector concatenation. Note that domain adaptation assumes that the domain labels of all input examples are known. As shown in Eq. (4), the input-oriented distribution representation can be learned from the domain-specific vector w^r and the basis examples $\tilde{x}_1^r,\cdots,\tilde{x}_{n_r}^r$ from domain r. Here r is determined by the domain label of the input example x. As illustrated in Algorithm 1, we use all the training source (target) examples as the basis source (target) examples $\tilde{x}_1^r,\cdots,\tilde{x}_{n_r}^r$. As a result, the input-oriented distribution representation learning of testing target examples X_*^{tgt} can be inferred by w^{tgt} and the pre-defined basis target examples.

4.3 Gradient Descent Training

Here we assume that the model parameters of the distribution-informed neural network can be updated by gradient descent during training. Based on the distribution-informed neural network, we propose a novel DINO-TRAIN algorithm. Its objective function using mean square error is formulated as follows.

$$\mathcal{L}(\theta) = \frac{\alpha}{2n_{\text{src}}}\sum_{i=1}^{n_{\text{src}}}\left(\tilde{f}(x_i^{\text{src}},\mathbb{P}^{\text{src}}) - y_i^{\text{src}}\right)^2 + \frac{1-\alpha}{2n_{\text{tgt}}^l}\sum_{j=1}^{n_{\text{tgt}}^l}\left(\tilde{f}(x_j^{\text{tgt}},\mathbb{P}^{\text{tgt}}) - y_j^{\text{tgt}}\right)^2 + \frac{\mu}{2}\hat{\text{MMD}}_{\Theta_{DA}}^2\left(\mathbb{P}^{\text{src}},\mathbb{P}^{\text{tgt}}\right) \tag{7}$$

where $\alpha \in (0,1)$ and $\mu \geq 0$ are hyperparameters to balance different terms. The first two terms are standard supervised learning losses over labeled source and target examples, and the third one is the empirical maximum mean discrepancy (MMD) [24] in the RKHS $\mathcal{H}_{DA}$ induced by the neural tangent kernel of our distribution-informed neural network, i.e.,

$$\hat{\text{MMD}}_{\Theta_{DA}}^2\left(\mathbb{P}^{\text{src}},\mathbb{P}^{\text{tgt}}\right) = \left\|\frac{1}{n_{\text{src}}}\sum_{i=1}^{n_{\text{src}}}\nabla_\theta\tilde{f}(x_i^{\text{src}},\mathbb{P}^{\text{src}}) - \frac{1}{n_{\text{tgt}}}\sum_{j=1}^{n_{\text{tgt}}}\nabla_\theta\tilde{f}(x_j^{\text{tgt}},\mathbb{P}^{\text{tgt}})\right\|_{\mathcal{H}_{DA}}^2 \tag{8}$$

where $n_{\text{tgt}} = n_{\text{tgt}}^l + n_{\text{tgt}}^u$ is the total number of target training examples. As shown in Theorem 4.5 below, our framework empirically minimizes the upper error bound of the expected prediction error in the target domain (see Algorithm 2 in Appendix A.10).

Remark. In our framework of Eq. (7), we measure the distribution shift using the training dynamics of the distribution-informed neural network over the source and target examples. This is fundamentally different from previous works [20, 59, 1] associated with a two-stage domain discrepancy measurement. That is, those works would first learn the feature representation in hidden neural layers and then measure the domain discrepancy in the learned feature space. In the second stage, they usually require additional modules, e.g., auxiliary neural networks [10, 21, 19] or pre-defined distribution distances [37]. Therefore, compared to previous works, our discrepancy measure Eq. (8) can not only unify the domain adaptation regression framework with neural networks, but also enable the theoretical convergence and generalization analysis in the following.

Before analyzing the dynamics of the distribution-informed neural network under Eq. (7), we first introduce the following assumption.

Assumption 4.2. Given labeled data $X = X^{\text{src}} \cup X_l^{\text{tgt}}$, we assume $\lambda_{\min}\left(\Theta(X,X)\right) > 0$ and $\lambda_{\min}\left(\mathcal{K}_{\mathscr{P}|\mathcal{X}}\left(X,X\right)\right) > 0$, where $\Theta(X,X) = \nabla_\theta f(X)\nabla_\theta f(X)^T$ is standard neural tangent kernel [31] induced by $f(\cdot)$ with infinite width, and $\mathcal{K}_{\mathscr{P}|\mathcal{X}}\left(X,X\right)$ is the input-oriented distribution kernel[1] (see Eq. (6)). Here, $\lambda_{\min}(\cdot)$ denotes the smallest eigenvalue.

The assumption $\lambda_{\min}\left(\Theta(X,X)\right) > 0$ has been studied in previous works [31, 18, 3]. It holds as long as no two inputs x_i and x_j are parallel in real scenarios. The implication of this assumption is that the input data (e.g., images) would not be linearly changed by a single factor, i.e., there is no two inputs such that $x_i = \varsigma \cdot x_j$ for some $\varsigma \in \mathbb{R}$.

Lemma 4.3. *Given the labeled training data $X = X^{src} \cup X_l^{tgt}$ and the basis examples $\{\tilde{x}_i^r\}_{i=1}^{\tilde{n}_r}$ ($r \in \{src, tgt\}$), we let $\Upsilon \in \mathbb{R}^{(n_{src}+n_{tgt}^l)\times(\tilde{n}_{src}+\tilde{n}_{tgt})}$ denote the coefficient matrix of $\mathcal{K}_{\mathscr{P}|\mathcal{X}}\left(X,X\right)$, where for $x_k \in X$ ($k = 1, \cdots, n_{src}+n_{tgt}^l$), its k^{th} row $[\Upsilon]_{k,:} = [\beta_{x_k,\tilde{x}_1^{src}}, \cdots, \beta_{x_k,\tilde{x}_{\tilde{n}_{src}}^{src}}, 0, \cdots, 0]$ when $x_k \in X^{src}$, $[\Upsilon]_{k,:} = [0, \cdots, 0, \beta_{x_k,\tilde{x}_1^{tgt}}, \cdots, \beta_{x_k,\tilde{x}_{\tilde{n}_{tgt}}^{tgt}}]$ otherwise. Then, the assumption $\lambda_{min}(\mathcal{K}_{\mathscr{P}|\mathcal{X}}(X,X)) > 0$ holds when $\Upsilon^T\Upsilon$ is positive definite.*

Lemma 4.3 shows that in real scenarios, the assumption $\lambda_{\min}(\mathcal{K}_{\mathscr{P}|\mathcal{X}}(X,X)) > 0$ can be guaranteed by the positive definiteness of the coefficient matrix Υ. We also empirically evaluate the smallest eigenvalue $\lambda_{\min}\left(\mathcal{K}_{\mathscr{P}|\mathcal{X}}\left(X,X\right)\right)$ in Subsection 5.2.

Theorem 4.4. *For any coefficient $\beta_{x,\tilde{x}_i}$ of the input-oriented distribution representation in Eq. (4), there exists $\eta^* \in \mathbb{R}_+$ such that for the infinitely-wide distribution-informed neural network $\tilde{f}(\cdot)$ trained under gradient flow with learning rate $\eta < \eta^*$, the test prediction $\tilde{f}_{\theta_t}(X_*^{tgt})$ of the domain adaptation regression in Eq. (7) over test target data X_*^{tgt} is*

$$\tilde{f}_{\theta_t}(X_*^{tgt}) = \tilde{f}_{\theta_0}(X_*^{tgt}) - \Theta_{DA}(X_*^{tgt},X)\Theta_{DA}(X,X)^{-1}\left(\mathbb{I} - e^{-\eta\Theta_{DA}\tilde{C}t}\right)\left(\tilde{f}_{\theta_0}(X) - Y\right)$$

where $\Theta_{DA}(\cdot,\cdot)$ is the distribution-informed NTK, i.e., $\Theta_{DA}(x,x') = \Theta(x,x')\cdot\mathcal{K}_{\mathscr{P}|\mathcal{X}}\left(\mathbb{P},\mathbb{P}'|x,x'\right)$ and $\tilde{C} = diag\{\underbrace{\alpha/n_{src},\cdots,\alpha/n_{src}}_{n_{src}},\underbrace{(1-\alpha)/n_{tgt}^l,\cdots,(1-\alpha)/n_{tgt}^l}_{n_{tgt}^l}\}$ is a diagonal matrix. Moreover, under the assumption 4.2, when the network width goes to infinity, $\lim_{t\to\infty}\tilde{f}_{\theta_t}(X_*^{tgt})$ converges to a Gaussian process with mean $\mu(X_*^{tgt})$ and variance $\Sigma(X_*^{tgt},X_*^{tgt})$ as follows.

$$\begin{aligned}\mu(X_*^{tgt}) &= \Theta_{DA}\left(X_*^{tgt},X\right)\Theta_{DA}(X,X)^{-1}Y\\ \Sigma(X_*^{tgt},X_*^{tgt}) &= \mathcal{K}^{DA}\left(X_*^{tgt},X_*^{tgt}\right) + \Theta_{DA}\left(X_*^{tgt},X\right)\Theta_{DA}(X,X)^{-1}\mathcal{K}^{DA}\Theta_{DA}(X,X)^{-1}\Theta_{DA}\left(X,X_*^{tgt}\right)\\ &\quad - \left(\Theta_{DA}\left(X_*^{tgt},X\right)\Theta_{DA}(X,X)^{-1}\mathcal{K}^{DA}\left(X,X_*^{tgt}\right) + h.c.\right)\end{aligned}$$

where "$+h.c.$" means "plus the Hermitian conjugate".

In addition, our result on generalization bound is given in the following theorem.

Theorem 4.5. *Assume for any training example $(x,y) \in \mathcal{X}\times\mathcal{Y}$, we have $(\tilde{f}(x,\mathbb{P}) - y)^2 \le M_0$ for some constant $M_0 \ge 0$. Let $\tilde{\mathcal{F}}$ be the hypothesis space induced by infinitely-wide neural networks $\tilde{f}$. Then, for any $\tilde{f} \in \tilde{\mathcal{F}}$ and $\delta > 0$, with probability at least $1-\delta$, the expected error in the target*

[1] Here we let $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(x,x') = \mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P},\mathbb{P}'|x,x')$ for brevity.

domain can be bounded as follows.

$$\mathcal{E}_{\mathbb{P}^{tgt}}(\tilde{f}) \leq \frac{\alpha}{n_{src}} \sum_{i=1}^{n_{src}} \left(\tilde{f}(x_i^{src}, \mathbb{P}^{src}) - y_i^{src}\right)^2 + \frac{1-\alpha}{n_{tgt}^l} \sum_{j=1}^{n_{tgt}^l} \left(\tilde{f}(x_j^{tgt}, \mathbb{P}^{tgt}) - y_j^{tgt}\right)^2 + 8\alpha M_0 \cdot \mathit{MMD}_{\Theta_{DA}}\left(\mathbb{P}^{src}, \mathbb{P}^{tgt}\right) + \Omega$$

where $\Omega = 2\alpha\Re_{n_{src}}(\mathcal{H}_{src}) + 2(1-\alpha)\Re_{n_{tgt}^l}(\mathcal{H}_{tgt}) + M\sqrt{\frac{(n_{src}+n_{tgt}^l)\log(1/\delta)}{2}}$, $\mathcal{H}_r = \{(x,y) \to (\tilde{f}(x^r, \mathbb{P}^r) - y^r)^2 : \tilde{f} \in \mathcal{F}\}$ *is a set of functions* ($r \in \{src, tgt\}$), $\Re_{n_r}(\mathcal{H}_r)$ *is the Rademacher complexity of* $\mathcal{H}_r$ *given* n_r *examples, and* $M = \max\{\alpha M_0/n_{src}, (1-\alpha)M_0/n_{tgt}^l\}$.

Though domain adaptation theories [5, 39] have been generalized to deep learning scenarios in recent years, neural networks are simply considered as an expressive feature extractor in modern generalization errors [59]. Nevertheless, recent works [60, 36] reveal that the feature space learned by the neural networks might worsen the adaptation between source and target domains. As a comparison, Theorem 4.5 provides a unified generalization error for neural network based domain adaptation, as it directly measures the domain discrepancy over input source and target examples in the RKHS induced by the distribution-informed NTK Θ_{DA}.

4.4 Discussion

4.4.1 Gaussian Process at Initialization

One special case of distribution-informed neural network at initialization is that when the coefficient $\beta_{x,x_i} = 1/n||\bar{\Phi}_x(\mathbb{P})||_{\mathcal{K}_\mathcal{X}}$ and $\bar{\Phi}_x(\mathbb{P}) = \frac{1}{n}\sum_{i=1}^n \langle \cdot, x_i\rangle_{\mathcal{K}_\mathcal{X}}$, it can be seen that $\Phi_x(\mathbb{P}) = \frac{1}{n\bar{\Phi}_x(\mathbb{P})}\sum_{i=1}^n \langle \cdot, x_i\rangle_{\mathcal{K}_\mathcal{X}}$ is the normalized mean mapping [24] of data distribution $\mathbb{P}$ in the NNGP kernel space. Then, the distribution kernel $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x')$ in Eq. (6) can be explained as the inner product of the mean mappings of $\mathbb{P}$ and $\mathbb{P}'$. Moreover, using the definition of maximum mean discrepancy (MMD) [24], it can be shown that $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x') = \frac{c - \hat{\text{MMD}}^2_{\mathcal{K}_\mathcal{X}}(\mathbb{P}, \mathbb{P}')}{2\bar{\Phi}_x(\mathbb{P})\bar{\Phi}_x(\mathbb{P}')}$, where $\hat{\text{MMD}}_{\mathcal{K}_\mathcal{X}}(\cdot)$ denotes the empirical MMD in the NNGP kernel space, and $c = \frac{1}{n^2}\sum_{i,j=1}^n \mathcal{K}_\mathcal{X}(x_i, x_j) + \frac{1}{n'^2}\sum_{i,j=1}^{n'} \mathcal{K}_\mathcal{X}(x_i', x_j')$. It implies that $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x')$ is negatively correlated with the popular MMD estimator. MMD measures the distribution distance using the difference of mean mappings of distributions, whereas $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x')$ measures the distribution similarity via the inner product of the mean mappings of the distributions $\mathbb{P}$ and $\mathbb{P}'$ in the NNGP kernel space.

The following corollary shows that the existing domain adaptive Gaussian process [7] can be explained as implicitly learning the distribution-informed representation in the kernel space.

Corollary 4.6. *With the assumptions in Lemma 4.1, when* $\beta_{x,\tilde{x}_i} = 1/||\sum_{i=1}^n \langle \cdot, \tilde{x}_i\rangle_{\mathcal{K}_\mathcal{X}}||$, *the Gaussian process induced by* $\tilde{f}(x)$ *at the initialization would be equivalent to the adaptive Gaussian process in [7] over NNKP kernel.*

4.4.2 Fully Trained Model

When there exist some labled target examples, the objective function of the reweighting domain adaptation techniques (see Subsection 3.2.2) can be rewritten as follows.

$$\min_\theta \frac{\alpha}{2n_{\text{src}}} \sum_{i=1}^{n_{\text{src}}} w_i^{\text{src}} \left(f(x_i^{\text{src}}) - y_i^{\text{src}}\right)_2^2 + \frac{1-\alpha}{2n_{\text{tgt}}} \sum_{j=1}^{n_{\text{tgt}}^l} \left(f(x_j^{\text{tgt}}) - y_j^{\text{tgt}}\right)_2^2 \tag{9}$$

The following corollary holds that our framework Eq. (7) can recover the reweighting approach Eq. (9) in the function space.

Corollary 4.7. *With the assumption in Theorem 4.4, our framework Eq. (7) with distribution-informed neural network* $\tilde{f}(\cdot)$ *can recover the popular reweighting domain adaptation approach Eq. (9) in the function space.*

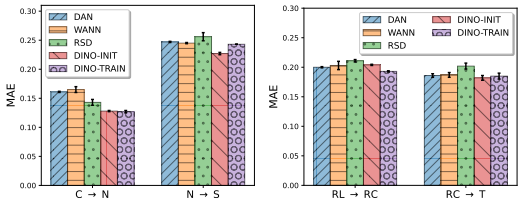
Besides, we show that when the distribution representation is shared by all the examples, our framework can be naturally degenerated into the domain-invariant representation learning [37].

Methods	C → N	C → S	N → C	N → S	S → C	S → N	Avg.
NNGP [34]	$2.041_{\pm 0.001}$	$1.823_{\pm 0.001}$	$0.445_{\pm 0.002}$	$0.624_{\pm 0.001}$	$0.197_{\pm 0.002}$	$0.459_{\pm 0.002}$	0.932
NTKGP [25]	$1.345_{\pm 0.002}$	$1.227_{\pm 0.000}$	$0.323_{\pm 0.002}$	$0.529_{\pm 0.004}$	$0.248_{\pm 0.001}$	$0.425_{\pm 0.002}$	0.683
AT-GP [7]	$0.194_{\pm 0.005}$	$0.259_{\pm 0.002}$	$\mathbf{0.104_{\pm 0.001}}$	$0.252_{\pm 0.005}$	$0.118_{\pm 0.003}$	$0.189_{\pm 0.006}$	0.186
TL-NTK [38]	$0.164_{\pm 0.001}$	$\mathbf{0.231_{\pm 0.000}}$	$0.124_{\pm 0.005}$	$0.242_{\pm 0.002}$	$0.125_{\pm 0.001}$	$0.197_{\pm 0.004}$	0.181
DINO-INIT (ours)	$0.128_{\pm 0.001}$	$0.233_{\pm 0.003}$	$0.114_{\pm 0.002}$	$\mathbf{0.227_{\pm 0.002}}$	$\mathbf{0.112_{\pm 0.001}}$	$\mathbf{0.181_{\pm 0.005}}$	**0.166**
DINO-TRAIN (ours)	$\mathbf{0.127_{\pm 0.002}}$	$0.240_{\pm 0.003}$	$0.127_{\pm 0.000}$	$0.243_{\pm 0.000}$	$0.128_{\pm 0.001}$	$0.194_{\pm 0.001}$	0.177

Table 1: Results of domain adaptation regression on dSprites

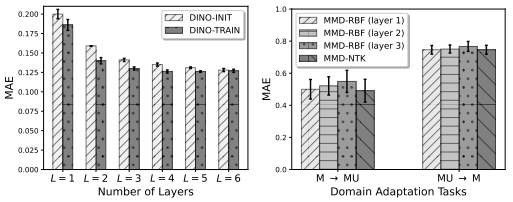
Methods	RL → RC	RL → T	RC → RL	RC → T	T → RL	T → RC	Avg.
NNGP [34]	$0.313_{\pm 0.001}$	$0.438_{\pm 0.004}$	$0.356_{\pm 0.005}$	$0.515_{\pm 0.008}$	$0.367_{\pm 0.001}$	$0.324_{\pm 0.004}$	0.386
NTKGP [25]	$0.396_{\pm 0.001}$	$0.365_{\pm 0.001}$	$0.200_{\pm 0.007}$	$0.390_{\pm 0.003}$	$0.390_{\pm 0.000}$	$0.354_{\pm 0.003}$	0.349
AT-GP [7]	$0.214_{\pm 0.011}$	$0.209_{\pm 0.002}$	$0.227_{\pm 0.010}$	$0.198_{\pm 0.002}$	$0.236_{\pm 0.000}$	$0.249_{\pm 0.000}$	0.222
TL-NTK [38]	$0.206_{\pm 0.004}$	$0.200_{\pm 0.002}$	$0.213_{\pm 0.000}$	$0.197_{\pm 0.000}$	$0.226_{\pm 0.001}$	$0.218_{\pm 0.000}$	0.210
DINO-INIT (ours)	$0.204_{\pm 0.001}$	$\mathbf{0.185_{\pm 0.006}}$	$\mathbf{0.207_{\pm 0.003}}$	$\mathbf{0.182_{\pm 0.004}}$	$\mathbf{0.218_{\pm 0.001}}$	$\mathbf{0.212_{\pm 0.001}}$	**0.201**
DINO-TRAIN (ours)	$\mathbf{0.193_{\pm 0.001}}$	$0.194_{\pm 0.003}$	$\mathbf{0.207_{\pm 0.003}}$	$0.188_{\pm 0.002}$	$0.226_{\pm 0.001}$	$0.218_{\pm 0.001}$	0.204

Table 2: Results of domain adaptation regression on MPI3D



(a) dSprites (b) MPI3D

Figure 2: Model comparison



(a) Number of layers L (b) Discrepancy

Figure 3: Model Analysis

Corollary 4.8. *Under mild conditions, our framework Eq. (7) with distribution-informed neural network $\tilde{f}(\cdot)$ can recover the standard domain invariant representation learning in Eq. (1), where the domain discrepancy measure $d(\cdot, \cdot)$ is instantiated with MMD in RKHS induced by standard NTK Θ.*

5 Experiments

Data Sets: We use three domain adaptation regression benchmarks. Following [10], we use two image data sets: dSprites [40] and MPI3D [23]. Specifically, dSprites has 737,280 images from three domains: Color (C), Noisy (N) and Scream (S). For each image, it has three regression tasks, i.e., predicting three factors of variations (scale, position X and Y). MPI3D contains over 3M images from three domains: Toy (T), Realistic (RC) and Real (RL). It is shown [10] that for each image, it involves two regression tasks, i.e., predicting three factors of variations (position X and Y). In addition, we also use a plant phenotyping data set. It predicts diverse traits (e.g., Nitrogen) of plants related to the plants' growth using leaf hyperspectral reflectance. Here we consider the following two domains [46]: Maize (M) and Maize_UNL (MU). In our case, the task is to predict the Nitrogen content of maize using the leaf hyperspectral reflectance (see Appendix A.11 for more details).

Baselines: In the experiments, we consider the following baseline methods. (1) Plain Gaussian process: NNGP [34] and NTKGP [25]. (2) Domain adaptive Gaussian processes: AT-GP [7] and TL-NTK [38]. (3) Deep domain adaptation methods: DAN [37], WANN [16] and RSD [10].

Implementations: In the experiments, our algorithms are implemented using a L-layer ($L = 6$) fully-connected neural network with ReLU (see Appendix A.11 for more details). The induced NNGP and neural tangent kernels induced can be estimated using the Neural Tangents package [41]. In addition, we set $\alpha = 0.5$ and $\mu = 0.1$ for DINO-TRAIN.

5.1 Results

Table 1, Table 2 and Table 3 provide the domain adaptation regression results on dSprites, MPI3D and Plant Phenotyping data sets. Following [10], we report the Mean Absolute Error (MAE) between the predicted outputs and the ground-truth outputs in the target test set (the best results are indicated in bold). It is observed that our method DINO achieves competitive performance over

the Gaussian process baselines. The weakly-trained DINO-INIT slightly outperforms DINO-TRAIN in some cases. This observation is consistent with previous work [33]. The relatively unstable performance of DINO-TRAIN might be caused by the large covariance of Gaussian process with NTK [25]. In addition, we also compare DINO with domain adaptation methods instantiated by overparameterized neural networks. WANN [16] reweights the source examples, and DAN [37] and RSD [10] learn the domain invariant representation. The performance comparison in Figure 2 confirms the effectiveness of our DINO approach over those state-of-the-art baselines.

Methods	M $\rightarrow$ MU	MU $\rightarrow$ M
NNGP [34]	$0.562_{\pm 0.001}$	$0.672_{\pm 0.010}$
NTKGP [25]	$0.562_{\pm 0.004}$	$0.702_{\pm 0.010}$
AT-GP [7]	$\mathbf{0.308_{\pm 0.006}}$	$0.593_{\pm 0.025}$
TL-NTK [38]	$0.316_{\pm 0.008}$	$0.488_{\pm 0.027}$
DINO-INIT (ours)	$0.316_{\pm 0.007}$	$0.645_{\pm 0.017}$
DINO-TRAIN (ours)	$0.314_{\pm 0.009}$	$\mathbf{0.443_{\pm 0.030}}$

Table 3: Results on Plant Phenotyping

5.2 Analysis

Ablation Study: We investigate the impact of input-oriented distribution representation in our DINO framework. We consider the following variants. DINO-INIT (DINO-TRAIN) w/o distribution feature: it would not utilize the input-oriented distribution representation, i.e., $g_{w_g}(\mathbb{P}|x) = 1$ for all $x \in \mathcal{X}$. DINO-INIT (DINO-TRAIN) w uniform distribution feature: the coefficient $\beta_{x,\tilde{x}_i} = 1/n$ is a constant. The results are given in Table 4. We observe that the distribution representation improves the domain adaptation performance. Compared to globally shared distribution representation with $\beta_{x,\tilde{x}_i} = 1/n$, our parametric $\Phi_x(\mathbb{P})$ in Eq. (4) encourages to learn the local distribution representation around the input data point x. The empirical results in Table 4 confirmed the efficacy of the input-oriented distribution representation.

Methods	C $\rightarrow$ N
DINO-INIT w/o distribution feature	$0.189_{\pm 0.002}$
DINO-INIT w uniform distribution feature	$0.171_{\pm 0.001}$
DINO-INIT	$0.128_{\pm 0.001}$
DINO-TRIAN w/o distribution feature	$0.161_{\pm 0.002}$
DINO-TRIAN w uniform distribution feature	$0.147_{\pm 0.001}$
DINO-TRIAN	$0.127_{\pm 0.002}$

Table 4: Ablation study on dSprites

Impact of Model Architecture and Discrepancy Measure: Figure 3a shows the results of DINO with different number of neural layers L on dSprites (C $\rightarrow$ N). The results indicate that the performance of DINO is positively related to the expressiveness of NTK and NNGP kernel induced by the neural network. In addition, we compare the proposed $\hat{\text{MMD}}_{\Theta_{DA}}(\cdot,\cdot)$ with MMD-RBF (i.e., MMD with standard RBF kernel) in domain adaptation regression, where MMD-RBF is estimated in the feature space learned by different neural layers (see Appendix A.11 for more details). As shown in Figure 3b, the results on Plant Phenotyping data set reveal that the selection of feature space affects the model performance when using MMD-RBF, and our proposed $\hat{\text{MMD}}_{\Theta_{DA}}(\cdot,\cdot)$ outperforms the MMD-RBF.

Positive Definiteness of $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(X,X)$: We report the smallest eigenvalue of the distribution kernel $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(X,X)$ over the training examples on Plant Phenotyping data set. The smallest eigenvalues of $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(X,X)$ in DINO-TRAIN after training are $3e-6$ and $8e-7$ on M $\rightarrow$ MU and MU $\rightarrow$ M, respectively. This empirically confirms the positive definiteness of $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(X,X)$ in DINO-TRAIN.

6 Conclusion

In this paper, we study the domain adaptation regression by proposing a distribution-informed neural network. Our domain adaptation framework with distribution-informed neural network subsumes the existing adaptation approaches based on domain invariant representation, reweighting, and adaptive Gaussian process. We also analyze the convergence and generalization bound of our framework. The experiments demonstrate the effectiveness of our DINO framework over state-of-the-art baselines.

Acknowledgments and Disclosure of Funding

This work is supported by National Science Foundation under Award No. IIS-1947203, IIS-2117902, IIS-2137468, and Agriculture and Food Research Initiative (AFRI) grant no. 2020-67021-32799/project accession no.1024178 from the USDA National Institute of Food and Agriculture. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

References

[1] David Acuna, Guojun Zhang, Marc T. Law, and Sanja Fidler. f-domain adversarial learning: Theory and algorithms. In *International Conference on Machine Learning*, pages 66–75. PMLR, 2021.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

[3] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[4] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, 32, 2019.

[5] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1):151–175, 2010.

[6] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in Neural Information Processing Systems*, 24:2178–2186, 2011.

[7] Bin Cao, Sinno Jialin Pan, Yu Zhang, Dit-Yan Yeung, and Qiang Yang. Adaptive transfer learning. In *proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

[8] Yu Cao, Meng Fang, Baosheng Yu, and Joey Tianyi Zhou. Unsupervised domain adaptation on reading comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7480–7487, 2020.

[9] Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. *Advances in Neural Information Processing Systems*, 32:10836–10846, 2019.

[10] Xinyang Chen, Sinan Wang, Jianmin Wang, and Mingsheng Long. Representation subspace distance for domain adaptation regression. In *International Conference on Machine Learning*, pages 1749–1759. PMLR, 2021.

[11] Corinna Cortes, Yishay Mansour, and Mehryar Mohri. Learning bounds for importance weighting. *Advances in Neural Information Processing Systems*, 10:442–450, 2010.

[12] Corinna Cortes and Mehryar Mohri. Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519:103–126, 2014.

[13] Corinna Cortes, Mehryar Mohri, and Andrés Munoz Medina. Adaptation based on generalized discrepancy. *The Journal of Machine Learning Research*, 20(1):1–30, 2019.

[14] Corinna Cortes, Mehryar Mohri, Michael Riley, and Afshin Rostamizadeh. Sample selection bias correction theory. In *International Conference on Algorithmic Learning Theory*, pages 38–53. Springer, 2008.

[15] Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

[16] Antoine de Mathelin, Guillaume Richard, François Deheeger, Mathilde Mougeot, and Nicolas Vayatis. Adversarial weighting for domain adaptation in regression. In *IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 49–56. IEEE, 2021.

[17] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR, 2019.

[18] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[19] Zhekai Du, Jingjing Li, Hongzu Su, Lei Zhu, and Ke Lu. Cross-domain gradient discrepancy minimization for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3937–3946, 2021.

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

[21] Zhiqiang Gao, Shufei Zhang, Kaizhu Huang, Qiufeng Wang, and Chaoliang Zhong. Gradient distribution alignment certificates better adversarial domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8937–8946, 2021.

[22] Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2019.

[23] Muhammad Waleed Gondal, Manuel Wuthrich, Djordje Miladinovic, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. *Advances in Neural Information Processing Systems*, 32:15740–15751, 2019.

[24] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[25] Bobby He, Balaji Lakshminarayanan, and Yee Whye Teh. Bayesian deep ensembles via the neural tangent kernel. In *Advances in Neural Information Processing Systems*, volume 33, pages 1010–1022, 2020.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Fumio Hiai and Minghua Lin. On an eigenvalue inequality involving the hadamard product. *Linear Algebra and its Applications*, 515:313–320, 2017.

[28] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International Conference on Machine Learning*, pages 1989–1998. PMLR, 2018.

[29] Baihe Huang, Xiaoxiao Li, Zhao Song, and Xin Yang. Fl-ntk: A neural tangent kernel-based framework for federated learning analysis. In *International Conference on Machine Learning*, pages 4423–4434. PMLR, 2021.

[30] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. *Advances in Neural Information Processing Systems*, pages 601–608, 2006.

[31] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8580–8589, 2018.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[33] Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33:15156–15172, 2020.

[34] Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

[35] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in Neural Information Processing Systems*, 32:8572–8583, 2019.

[36] Hong Liu, Mingsheng Long, Jianmin Wang, and Michael Jordan. Transferable adversarial training: A general approach to adapting deep classifiers. In *International Conference on Machine Learning*, pages 4013–4022. PMLR, 2019.

[37] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[38] Wesley Maddox, Shuai Tang, Pablo Moreno, Andrew Gordon Wilson, and Andreas Damianou. Fast adaptation with linearized neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 2737–2745. PMLR, 2021.

[39] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *22nd Conference on Learning Theory*, 2009.

[40] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

[41] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020.

[42] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.

[43] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[44] Neeti Wagle and Eric W Frew. Forward adaptive transfer of gaussian process regression. *Journal of Aerospace Information Systems*, 14(4):214–231, 2017.

[45] Haoxiang Wang, Han Zhao, and Bo Li. Bridging multi-task learning and meta-learning: Towards efficient training and effective adaptation. In *International Conference on Machine Learning*, pages 10991–11002. PMLR, 2021.

[46] Sheng Wang, Kaiyu Guan, Zhihui Wang, Elizabeth A Ainsworth, Ting Zheng, Philip A Townsend, Kaiyuan Li, Christopher Moller, Genghong Wu, and Chongya Jiang. Unique contributions of chlorophyll and nitrogen to predict crop photosynthetic capacity from leaf spectroscopy. *Journal of experimental botany*, 72(2):341–354, 2021.

[47] Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11293–11302, 2019.

[48] Colin Wei, Jason D Lee, Qiang Liu, and Tengyu Ma. Regularization matters: Generalization and optimization of neural nets vs their induced kernel. *Advances in Neural Information Processing Systems*, 32, 2019.

[49] Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[50] Jun Wu, Elizabeth A Ainsworth, Sheng Wang, Kaiyu Guan, and Jingrui He. Adaptive transfer learning for plant phenotyping. *arXiv preprint arXiv:2201.05261*, 2022.

[51] Jun Wu and Jingrui He. Indirect invisible poisoning attacks on domain adaptation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1852–1862, 2021.

[52] Jun Wu and Jingrui He. Domain adaptation with dynamic open-set targets. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2039–2049, 2022.

[53] Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[54] Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*, 2020.

[55] Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. Rethinking bias-variance trade-off for generalization of neural networks. In *International Conference on Machine Learning*, pages 10767–10777. PMLR, 2020.

[56] Yao-Liang Yu and Csaba Szepesvári. Analysis of kernel mean matching under covariate shift. In *International Conference on Machine Learning*, pages 1147–1154, 2012.

[57] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[58] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[59] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *International Conference on Machine Learning*, pages 7404–7413. PMLR, 2019.

[60] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pages 7523–7532. PMLR, 2019.

[61] Yao Zhou, Lei Ying, and Jingrui He. MultiC^2: an optimization framework for learning from task and worker dual heterogeneity. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 579–587. SIAM, 2017.

[62] Yao Zhou, Lei Ying, and Jingrui He. Multi-task crowdsourcing via an optimization framework. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(3):1–26, 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...

(a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Abstract and Section 1.

(b) Did you describe the limitations of your work? [Yes] See Appendix A.10.

(c) Did you discuss any potential negative societal impacts of your work? [No] Our work does not involve any potential negative societal impacts.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes]

(b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.2 - A.9.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the supplemental material.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 and Appendix A.11.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.11.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [N/A]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

In the appendix, we have the following results.

- In Appendix A.1, we summarize the main notations used in this paper.
- In Appendix A.2 - A.9, we show all the proofs of our theoretical results.
- In Appendix A.10, we present the overall training procedures (e.g., pseudo code) of our proposed `DINO-INIT` and `DINO-TRAIN` algorithms, as well as the limitations of our work.
- In Appendix A.11, we present additional data description and implementation details.

A.1 Notation

Notation	Definition
$\mathcal{X}, \mathcal{Y}, \mathscr{P}$	Input, output and probability distribution space
$\mathbb{P}^{\text{src}}, \mathbb{P}^{\text{tgt}} \in \mathscr{P}$	Source and target joint distributions
$\mathbb{P}^{\text{src}}_X, \mathbb{P}^{\text{tgt}}_X$	Source and target marginal distributions
$\{x_i^{\text{src}}, y_i^{\text{src}}\}_{i=1}^{n_{\text{src}}}$	Labeled source training examples
$\{x_j^{\text{tgt}}, y_j^{\text{tgt}}\}_{j=1}^{n_{\text{tgt}}_l}$	Labeled target training examples
$\{x_j^{\text{tgt}}\}_{j=1}^{n_{\text{tgt}}_u}$	Unlabeled target training examples
X_*^{tgt}	Target testing examples
$X^{\text{src}} = \{x_i^{\text{src}}\}_{i=1}^{n_{\text{src}}}, Y^{\text{src}} = \{y_i^{\text{src}}\}_{i=1}^{n_{\text{src}}}$	Labeled source input and output sets
$X_l^{\text{tgt}} = \{x_j^{\text{tgt}}\}_{j=1}^{n_{\text{tgt}}_l}, Y_l^{\text{tgt}} = \{y_j^{\text{tgt}}\}_{j=1}^{n_{\text{tgt}}_l}$	Labeled target input and output sets
$X = X^{\text{src}} \cup X_l^{\text{tgt}}, Y = Y^{\text{src}} \cup Y_l^{\text{tgt}}$	All the labeled training inputs, and the corresponding outputs
$\tilde{x}_i$	Basis example of input-oriented distribution representation learning
$f(\cdot)$	L-layer fully-connected neural network
$\tilde{f}(\cdot)$	Distribution-informed neural network
$\mathcal{K}_{\mathcal{X}}(\cdot,\cdot)$	NNGP kernel over $\mathcal{X}$
$\mathcal{K}_{\mathscr{P}\|\mathcal{X}}(\cdot,\cdot)$	Distribution kernel over $\mathscr{P}$
$\mathcal{K}^{DA}(\cdot,\cdot)$	Distribution-informed NNGP kernel
$\Theta(\cdot,\cdot)$	Neural tangent kernel (NTK)
$\Theta_{DA}(\cdot,\cdot)$	Distribution-informed NTK

Table 5: Notation

A.2 Proof of Lemma 4.1

Lemma 4.1. Assume that all the parameters of $\tilde{f}(\cdot)$ follows standard normal distribution, in the limits as the layer width $d \to \infty$, the output function of the distribution-informed neural network $\tilde{f}(x)$ in Eq (5) at initialization is iid centered Gaussian process, i.e., $\tilde{f}(\cdot) \sim \mathcal{N}\left(0, \mathcal{K}^{DA}\right)$ where

$$\mathcal{K}^{DA}\left((x,\mathbb{P}),(x',\mathbb{P}')\right) = \mathcal{K}_{\mathcal{X}}(x,x') \cdot \mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P},\mathbb{P}'|x,x')$$

where $\mathcal{K}_{\mathcal{X}}(\cdot,\cdot)$ is the NNGP kernel induced by the neural network $f(\cdot)$ over input space $\mathcal{X}$, and $\mathcal{K}_{\mathscr{P}}(\cdot,\cdot)$ is a distribution kernel identifying the similarity of the distributions $\mathbb{P}$ and $\mathbb{P}'$ over distribution space $\mathscr{P}$:

$$\mathcal{K}_{\mathscr{P}|\mathcal{X}}\left(\mathbb{P},\mathbb{P}'|x,x'\right) = \sum_{i=1}^{n}\sum_{j=1}^{n'} \beta_{x,\tilde{x}_i}\beta_{x',\tilde{x}_j}\mathcal{K}_{\mathcal{X}}(\tilde{x}_i,\tilde{x}'_j) \tag{10}$$

where n (n') is the number of examples in the domain associated with distribution $\mathbb{P}$ ($\mathbb{P}'$).

Proof. Following [34], it can be shown that the output function of a fully-connected neural networks $f(\cdot)$ is an iid centered Gaussian process with zero mean and varaince $\mathcal{K}_{\mathcal{X}}(x,x')$ (i.e., NNGP kernel). Using definition of distribution-informed neural network $\tilde{f}(x)$ in Eq (5), we have $\mathbb{E}[\tilde{f}(x,\mathbb{P})] = 0$ (due to $\mathbb{E}[w] = 0$), and $\mathcal{K}^{DA}((x,\mathbb{P}),(x',\mathbb{P}')) = \mathbb{E}[\phi(x)^T\phi(x')] \cdot \Phi_x(\mathbb{P})^T\Phi_{x'}(\mathbb{P}')$ (due to $\mathbb{E}[w^Tw] = 1$). Using the definition of $\Phi_x(\mathbb{P})$ in Eq. (4), we have $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P},\mathbb{P}'|x,x') = \sum_{i=1}^{n}\sum_{j=1}^{n'}\beta_{x,\tilde{x}_i}\beta_{x',\tilde{x}'_j}\mathcal{K}_{\mathcal{X}}(\tilde{x}_i,\tilde{x}'_j)$. □

A.3 Proof of Lemma 4.3

Lemma 4.3. Given the labeled training data $X = X^{\text{src}} \cup X_l^{\text{tgt}}$ and the basis examples $\{\tilde{x}_i^r\}_{i=1}^{\tilde{n}_r}$ ($r \in \{\text{src}, \text{tgt}\}$), we let $\Upsilon \in \mathbb{R}^{(n_{\text{src}}+n_{\text{tgt}}^l)\times(\tilde{n}_{\text{src}}+\tilde{n}_{\text{tgt}})}$ denote the coefficient matrix of $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(X, X)$, where for $x_k \in X$ ($k = 1, \cdots, n_{\text{src}} + n_{\text{tgt}}^l$), its k^{th} row $[\Upsilon]_{k,:} = [\beta_{x_k,\tilde{x}_1^{\text{src}}}, \cdots, \beta_{x_k,\tilde{x}_{\tilde{n}_{\text{src}}}^{\text{src}}}, 0, \cdots, 0]$ when $x_k \in X^{\text{src}}$, $[\Upsilon]_{k,:} = [0, \cdots, 0, \beta_{x_k,\tilde{x}_1^{\text{tgt}}}, \cdots, \beta_{x_k,\tilde{x}_{\tilde{n}_{\text{tgt}}}^{\text{tgt}}}]$ otherwise. Then, the assumption $\lambda_{\min}(\mathcal{K}_{\mathscr{P}|\mathcal{X}}(X, X)) > 0$ holds when $\Upsilon^T\Upsilon$ is positive definite.

Proof. Using the definition of the distribution kernel in Eq. (6), we have $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(X, X) = \Upsilon\mathcal{K}_{\mathcal{X}}(\tilde{X}, \tilde{X})\Upsilon^T$ where $\tilde{X}$ denotes all the basis examples. Then, we have $\lambda_{\min}(\mathcal{K}_{\mathscr{P}|\mathcal{X}}(X, X)) = \lambda_{\min}(\Upsilon\mathcal{K}_{\mathcal{X}}(\tilde{X}, \tilde{X})\Upsilon^T) \geq \lambda_{\min}(\mathcal{K}_{\mathcal{X}}(\tilde{X}, \tilde{X})) \cdot \lambda_{\min}(\Upsilon^T\Upsilon)$. Here $\mathcal{K}_{\mathcal{X}}(\tilde{X}, \tilde{X})$ is the NNGP kernel matrix of the basis examples induced by the neural network $f(\cdot)$ with infinite width. It is shown [4] that the key difference between NNGP kernel and NTK is that NTK is generated by a fully-trained neural network, whereas NNGP kernel is produced by a weakly-trained neural network. That is, NNGP kernel is a special case of NTK when training only the output layer. Following [18, 3], when there is no two parallel inputs in $\tilde{X}$, we have $\lambda_{\min}(\mathcal{K}_{\mathcal{X}}(\tilde{X}, \tilde{X})) > 0$. Therefore, when $\Upsilon^T\Upsilon$ is positive definite, the assumption $\lambda_{\min}(\mathcal{K}_{\mathscr{P}|\mathcal{X}}(X, X)) > 0$ can hold. □

A.4 Proof of Theorem 4.4

Theorem 4.4. For any coefficient β_{x,x_i} of the input-oriented distribution representation in Eq. (4), there exists $\eta^* \in \mathbb{R}_+$ such that for the infinitely-wide distribution-informed neural network $\tilde{f}(\cdot)$ trained under gradient flow with learning rate $\eta < \eta^*$, the test prediction $\tilde{f}_{\theta_t}(X_*^{\text{tgt}})$ of the domain adaptation regression in Eq. (7) over test target data X_*^{tgt} is

$$\tilde{f}_{\theta_t}(X_*^{\text{tgt}}) = \tilde{f}_{\theta_0}(X_*^{\text{tgt}}) - \Theta_{DA}(X_*^{\text{tgt}}, X)\Theta_{DA}(X, X)^{-1}\left(\mathbb{I} - e^{-\eta\Theta_{DA}\tilde{C}t}\right)\left(\tilde{f}_{\theta_0}(X) - Y\right)$$

where $\Theta_{DA}(\cdot, \cdot)$ is the distribution-informed NTK, i.e., $\Theta_{DA}(x, x') = \Theta(x, x') \cdot \mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x')$ and $\tilde{C} = \text{diag}\{\underbrace{\alpha/n_{\text{src}}, \cdots, \alpha/n_{\text{src}}}_{n_{\text{src}}}, \underbrace{(1-\alpha)/n_{\text{tgt}}^l, \cdots, (1-\alpha)/n_{\text{tgt}}^l}_{n_{\text{tgt}}^l}\}$ is a diagonal matrix. Moreover, under the assumption 4.2, when the network width goes to infinity, $\lim_{t\to\infty} \tilde{f}_{\theta_t}(X_*^{\text{tgt}})$ converges to a Gaussian process with mean $\mu(X_*^{\text{tgt}})$ and variance $\Sigma(X_*^{\text{tgt}}, X_*^{\text{tgt}})$ as follows.

$$\begin{aligned}
\mu(X_*^{\text{tgt}}) &= \Theta_{DA}\left(X_*^{\text{tgt}}, X\right)\Theta_{DA}(X, X)^{-1}Y \\
\Sigma(X_*^{\text{tgt}}, X_*^{\text{tgt}}) &= \mathcal{K}^{DA}\left(X_*^{\text{tgt}}, X_*^{\text{tgt}}\right) + \Theta_{DA}\left(X_*^{\text{tgt}}, X\right)\Theta_{DA}(X, X)^{-1}\mathcal{K}^{DA}\Theta_{DA}(X, X)^{-1}\Theta_{DA}\left(X, X_*^{\text{tgt}}\right) \\
&\quad - \left(\Theta_{DA}\left(X_*^{\text{tgt}}, X\right)\Theta_{DA}(X, X)^{-1}\mathcal{K}^{DA}\left(X, X_*^{\text{tgt}}\right) + h.c.\right)
\end{aligned}$$

where "$+h.c.$" means "plus the Hermitian conjugate".

Proof. The objective function of Eq. (7) can be rewritten as follows.

$$\begin{aligned}
\mathcal{L}(\theta) &= \frac{\alpha}{2n_{\text{src}}}\sum_{i=1}^{n_{\text{src}}}\left(\tilde{f}(x_i^{\text{src}}, \mathbb{P}^{\text{src}}) - y_i^{\text{src}}\right)^2 + \frac{1-\alpha}{2n_{\text{tgt}}^l}\sum_{j=1}^{n_{\text{tgt}}^l}\left(\tilde{f}(x_j^{\text{tgt}}, \mathbb{P}^{\text{tgt}}) - y_j^{\text{tgt}}\right)^2 + \frac{\mu}{2}\hat{\text{MMD}}^2_{\Theta_{DA}}\left(\mathbb{P}^{\text{src}}, \mathbb{P}^{\text{tgt}}\right) \\
&= \frac{1}{2}\left|\left|C\tilde{f}(X) - CY\right|\right|_2^2 + \frac{\mu}{2}\hat{\text{MMD}}^2_{\Theta_{DA}}\left(\mathbb{P}^{\text{src}}, \mathbb{P}^{\text{tgt}}\right)
\end{aligned}$$

where $\tilde{f}(X) = \text{vec}(\{\tilde{f}(x_i)\}_{i=1}^{n_{\text{src}}+n_{\text{tgt}}^l})$, and $C = \text{diag}\{\underbrace{\sqrt{\alpha/n_{\text{src}}}, \cdots, \sqrt{\alpha/n_{\text{src}}}}_{n_{\text{src}}}, \underbrace{\sqrt{(1-\alpha)/n_{\text{tgt}}^l}, \cdots, \sqrt{(1-\alpha)/n_{\text{tgt}}^l}}_{n_{\text{tgt}}^l}\}$ is a constant diagonal matrix. Then the tangent kernel can be defined as

$$\Theta_{DA}(X, X) = \lim_{d\to\infty} \nabla_{\theta_0}\tilde{f}(X)\nabla_{\theta_0}\tilde{f}(X)^T$$

Moreover, we obtain

$$\nabla_{\theta_0}\tilde{f}(x,\mathbb{P}) = \nabla_{\theta_0} f(x)\cdot g_{w_g}(\mathbb{P}|x)$$

$$\lim_{d\to\infty}\left\langle \nabla_{\theta_0}\tilde{f}(x,\mathbb{P}), \nabla_{\theta_0}\tilde{f}(x',\mathbb{P}')\right\rangle = \Theta(x,x')\cdot\mathcal{K}_{\mathscr{P}|\mathcal{X}}\left(\mathbb{P},\mathbb{P}'|x,x'\right)$$

$$\Theta_{DA}(X,X) = \Theta(X,X)\odot\mathcal{K}_{\mathscr{P}|\mathcal{X}}\left(X,X\right)$$

where $\Theta(x,x') = \lim_{d\to\infty}\langle\nabla_{\theta_0} f(x), \nabla_{\theta_0} f(x')\rangle$ is standard neural tangent kernel [31] induced by $f(\cdot)$ with infinite width.

Following [35], we have the linearized neural network given by its first-order Taylor expansion.

$$f^{\text{lin}}_{\theta_t}(x) = f_{\theta_0}(x) + \nabla_{\theta_0} f_{\theta_0}(x)\left(\theta_t - \theta_0\right)$$

and $f_{\theta_t}(x) - f^{\text{lin}}_{\theta_t}(x) = \mathcal{O}(1/\sqrt{d^*}) \to 0$ $(d^*\to 0)$. Then we have $\tilde{f}_{\theta_t}(X) = \tilde{f}_{\theta_0}(X) + \nabla_{\theta_0}\tilde{f}_{\theta_0}(X)\left(\theta_t-\theta_0\right) + \mathcal{O}(1/\sqrt{d^*})$ where d^* denotes the network width. Using the lineraized function $\tilde{f}^{\text{lin}}_{\theta_t}(X) = \tilde{f}_{\theta_0}(X) + \nabla_{\theta_0}\tilde{f}_{\theta_0}(X)\left(\theta_t-\theta_0\right)$, we have the dynamics of lineraized neural network as follows.

$$\begin{aligned}\dot{\theta}_t = -\eta\nabla_\theta\mathcal{L}(\theta) &= -\eta\nabla_\theta\tilde{f}^{\text{lin}}_{\theta_t}(X)^T C^T C\left(\tilde{f}^{\text{lin}}_{\theta_t}(X) - Y\right)\\ &= -\eta\nabla_\theta\tilde{f}_{\theta_0}(X)^T C^T C\left(\tilde{f}_{\theta_0}(X) + \nabla_{\theta_0}\tilde{f}_{\theta_0}(X)\left(\theta_t-\theta_0\right) - Y\right)\end{aligned}$$

Then the ODE has closed form solution as

$$\theta_t = \theta_0 - \nabla_\theta\tilde{f}_{\theta_0}(X)^T\Theta^{-1}_{DA}\left(\mathbb{I} - e^{-\eta\Theta_{DA}C^TCt}\right)\left(\tilde{f}_{\theta_0}(X) - Y\right)$$

Then given random initialization θ_0, the predictions of this neural network over training X and testing example X^{tgt}_* are

$$\tilde{f}_{\theta_t}(X) \approx \tilde{f}^{\text{lin}}_{\theta_t}(X) = \tilde{f}_{\theta_0}(X) - \left(\mathbb{I} - e^{-\eta\Theta_{DA}C^TCt}\right)\left(\tilde{f}_{\theta_0}(X) - Y\right)$$

$$\tilde{f}_{\theta_t}(X^{\text{tgt}}_*) \approx \tilde{f}^{\text{lin}}_{\theta_t}(X^{\text{tgt}}_*) = \tilde{f}_{\theta_0}(X^{\text{tgt}}_*) - \Theta_{DA}(X^{\text{tgt}}_*, X)\Theta_{DA}(X,X)^{-1}\left(\mathbb{I} - e^{-\eta\Theta_{DA}C^TCt}\right)\left(\tilde{f}_{\theta_0}(X) - Y\right)$$

with up to an error of $\mathcal{O}(1/\sqrt{d^*})$.

Here, the minimum eigenvalue $\lambda_{\min}(\Theta_{DA}C^TC) \geq \lambda_{\min}(\Theta_{DA})\lambda_{\min}(C^TC) = \lambda_{\min}(\Theta_{DA})\cdot\min\{\alpha/n_{\text{src}}, (1-\alpha)/n^l_{\text{tgt}}\}$. Following [27], using $\lambda_{\min}\left(\Theta(X,X)\right) > 0$ and $\lambda_{\min}\left(\mathcal{K}_{\mathscr{P}|\mathcal{X}}\left(X,X\right)\right) > 0$, we have $\lambda_{\min}(\Theta_{DA}) \geq 0$. When $\alpha\in(0,1)$, $\lim_{t\to\infty}\lim_{d^*\to\infty}\tilde{f}_{\theta_t}(X^{\text{tgt}}_*) = \tilde{f}_{\theta_0}(X^{\text{tgt}}_*) - \Theta_{DA}(X^{\text{tgt}}_*, X)\Theta_{DA}(X,X)^{-1}\left(\tilde{f}_{\theta_0}(X) - Y\right)$.

Over random initialization of θ_0, $\tilde{f}(X^{\text{tgt}}_*)$ converges to a Gaussian distribution, as it is a linear transformation of $\tilde{f}_{\theta_0}(X^{\text{tgt}}_*)$ associated with Gaussian distribution. Using $\mathbb{E}[\tilde{f}_{\theta_0}(X^{\text{tgt}}_*)] = 0$ and $\mathbb{E}[\tilde{f}_{\theta_0}(X^{\text{tgt}}_*)\tilde{f}_{\theta_0}(X^{\text{tgt}}_*)] = \mathcal{K}^{DA}\left(X^{\text{tgt}}_*, X^{\text{tgt}}_*\right)$, when $\alpha\in(0,1)$, we have the following results:

$$\begin{aligned}\mu(X^{\text{tgt}}_*) &= \Theta_{DA}\left(X^{\text{tgt}}_*, X\right)\Theta_{DA}(X,X)^{-1}Y\\ \Sigma(X^{\text{tgt}}_*, X^{\text{tgt}}_*) &= \mathcal{K}^{DA}\left(X^{\text{tgt}}_*, X^{\text{tgt}}_*\right) + \Theta_{DA}\left(X^{\text{tgt}}_*, X\right)\Theta_{DA}(X,X)^{-1}\mathcal{K}^{DA}\Theta_{DA}(X,X)^{-1}\Theta_{DA}\left(X, X^{\text{tgt}}_*\right)\\ &\quad - \left(\Theta_{DA}\left(X^{\text{tgt}}_*, X\right)\Theta_{DA}(X,X)^{-1}\mathcal{K}^{DA}\left(X, X^{\text{tgt}}_*\right) + h.c.\right)\end{aligned}$$

which completes the proof. □

A.5 Proof of Theorem 4.5

Theorem 4.5. Assume for any training example $(x,y)\in\mathcal{X}\times\mathcal{Y}$, we have $(\tilde{f}(x,\mathbb{P}) - y)^2 \leq M_0$ for some constant $M_0\geq 0$. Let $\tilde{\mathcal{F}}$ be the hypothesis space induced by infinitely-wide neural networks $\tilde{f}$. Then, for any $\tilde{f}\in\tilde{\mathcal{F}}$ and $\delta > 0$, with probability at least $1-\delta$, the expected error in the target domain can be bounded as follows.

$$\mathcal{E}_{\mathbb{P}^{\text{tgt}}}(\tilde{f}) \leq \frac{\alpha}{n_{\text{src}}}\sum_{i=1}^{n_{\text{src}}}\left(\tilde{f}(x^{\text{src}}_i, \mathbb{P}^{\text{src}}) - y^{\text{src}}_i\right)^2 + \frac{1-\alpha}{n^l_{\text{tgt}}}\sum_{j=1}^{n^l_{\text{tgt}}}\left(\tilde{f}(x^{\text{tgt}}_j, \mathbb{P}^{\text{tgt}}) - y^{\text{tgt}}_j\right)^2 + 8\alpha M_0\cdot\text{MMD}_{\Theta_{DA}}\left(\mathbb{P}^{\text{src}}, \mathbb{P}^{\text{tgt}}\right)$$

$$+ 2\alpha\Re_{n_{\text{src}}}(\mathcal{H}_{\text{src}}) + 2(1-\alpha)\Re_{n_{\text{tgt}}^l}(\mathcal{H}_{\text{tgt}}) + M\sqrt{\frac{(n_{\text{src}} + n_{\text{tgt}}^l)\log(1/\delta)}{2}}$$

where $\mathcal{H}_{\text{src}} = \{(x, y) \to (\tilde{f}(x^{\text{src}}, \mathbb{P}^{\text{src}}) - y^{\text{src}})^2 : \tilde{f} \in \tilde{\mathcal{F}}\}$ is a set of functions, $\Re_{n_{\text{src}}}(\mathcal{H}_{\text{src}})$ is the Rademacher complexity of $\mathcal{H}_{\text{src}}$ given n_{src} examples, and $M = \max\{\alpha M_0/n_{\text{src}}, (1-\alpha)M_0/n_{\text{tgt}}^l\}$.

Proof. Let $\Psi(X) = \sup_{\tilde{f}\in\tilde{\mathcal{F}}} \mathcal{L}_{\mathbb{P}^{\text{tgt}}}(\tilde{f}) - \alpha \cdot \mathcal{L}_{\hat{\mathbb{P}}^{\text{src}}}(\tilde{f}) - (1-\alpha) \cdot \mathcal{L}_{\hat{\mathbb{P}}^{\text{tgt}}}(\tilde{f})$ where $\mathcal{L}_{\hat{\mathbb{P}}^{\text{src}}}(\tilde{f}) = \frac{1}{n_{\text{src}}} \sum_{i=1}^{n_{\text{src}}} \left(\tilde{f}(x_i^{\text{src}}, \mathbb{P}^{\text{src}}) - y_i^{\text{src}}\right)^2$ and $\mathcal{L}_{\hat{\mathbb{P}}^{\text{tgt}}}(\tilde{f}) = \frac{1}{n_{\text{tgt}}^l} \sum_{j=1}^{n_{\text{tgt}}^l} \left(\tilde{f}(x_j^{\text{tgt}}, \mathbb{P}^{\text{tgt}}) - y_j^{\text{tgt}}\right)^2$.

Then when one point of X is changed, $\Psi(X)$ will change at most $M = \max\{\alpha M_0/n_{\text{src}}, (1-\alpha)M_0/n_{\text{tgt}}^l\}$. Using McDiarmid's inequality, it holds that

$$\Pr[\Psi(X) - \mathbb{E}[\Psi(X)] > \epsilon] \leq \exp\left(-\frac{2\epsilon^2}{(n_{\text{src}} + n_{\text{tgt}}^l)M^2}\right)$$

Thus, for any $\delta > 0$, with probability at least $1-\delta$, we have

$$\mathcal{L}_{\mathbb{P}^{\text{tgt}}}(\tilde{f}) \leq \alpha \cdot \mathcal{L}_{\hat{\mathbb{P}}^{\text{src}}}(\tilde{f}) + (1-\alpha) \cdot \mathcal{L}_{\hat{\mathbb{P}}^{\text{tgt}}}(\tilde{f}) + \mathbb{E}[\Psi(X)] + M\sqrt{\frac{(n_{\text{src}} + n_{\text{tgt}}^l)\log(1/\delta)}{2}}$$

Moreover,

$$\begin{aligned}
\mathbb{E}[\Psi(X)] &= \mathbb{E}\left[\sup_{\tilde{f}\in\tilde{\mathcal{F}}} \alpha \cdot \mathcal{L}_{\hat{\mathbb{P}}^{\text{src}}}(\tilde{f}) + (1-\alpha) \cdot \mathcal{L}_{\hat{\mathbb{P}}^{\text{tgt}}}(\tilde{f}) - \mathcal{L}_{\mathbb{P}^{\text{tgt}}}(\tilde{f})\right] \\
&\leq \alpha \cdot \mathbb{E}\left[\sup_{\tilde{f}\in\tilde{\mathcal{F}}} \mathcal{L}_{\hat{\mathbb{P}}^{\text{src}}}(\tilde{f}) - \mathcal{L}_{\mathbb{P}^{\text{tgt}}}(\tilde{f})\right] + (1-\alpha) \cdot \mathbb{E}\left[\sup_{\tilde{f}\in\tilde{\mathcal{F}}} \mathcal{L}_{\hat{\mathbb{P}}^{\text{tgt}}}(\tilde{f}) - \mathcal{L}_{\mathbb{P}^{\text{tgt}}}(\tilde{f})\right] \\
&\leq 2\alpha\Re_{n_{\text{src}}}(\mathcal{H}_{\text{src}}) + 2(1-\alpha)\Re_{n_{\text{tgt}}^l}(\mathcal{H}_{\text{tgt}}) + \alpha \cdot \sup_{\tilde{f}\in\tilde{\mathcal{F}}} \mathcal{L}_{\mathbb{P}^{\text{src}}}(\tilde{f}) - \mathcal{L}_{\mathbb{P}^{\text{tgt}}}(\tilde{f})
\end{aligned}$$

When $(\tilde{f}(x, \mathbb{P}) - y)_2 \leq M_0$ for any training example $(x, y) \in \mathcal{X} \times \mathcal{Y}$, Lemma 23 of [13] shows that $\left|(\tilde{f}(x, \mathbb{P}) - y)^2 - (\tilde{f}'(x, \mathbb{P}) - y)^2\right| \leq 2M_0 \left|\tilde{f}(x, \mathbb{P}) - \tilde{f}'(x, \mathbb{P})\right|$. Therefore, we have $(\tilde{f}(x, \mathbb{P}) - y)^2 \leq 2M_0 \left|\tilde{f}(x, \mathbb{P}) - y\right|$. It is shown in [31] that a infinitely-wide neural network would behave as its linearization around the initialization and achieves zero training loss under MSE loss. Thus, there exists a small perturbation $\Delta\theta$ over parameters for each example (x, y) such that $\tilde{f}_{\theta+\Delta\theta}(x, \mathbb{P}) = y$.

$$\begin{aligned}
&\sup_{\tilde{f}\in\tilde{\mathcal{F}}} \mathcal{L}_{\mathbb{P}^{\text{src}}}(\tilde{f}) - \mathcal{L}_{\mathbb{P}^{\text{tgt}}}(\tilde{f}) \leq \sup_{\tilde{f}\in\tilde{\mathcal{F}}} \left|\mathbb{E}_{\mathbb{P}^{\text{src}}}\left(\tilde{f}(x^{\text{src}}, \mathbb{P}^{\text{src}}) - y^{\text{src}}\right)^2 - \mathbb{E}_{\mathbb{P}^{\text{tgt}}}\left(\tilde{f}(x^{\text{tgt}}, \mathbb{P}^{\text{tgt}}) - y^{\text{tgt}}\right)^2\right| \\
&\leq 4M_0 \cdot \sup_{\tilde{f}\in\tilde{\mathcal{F}}} \left|\mathbb{E}_{\mathbb{P}^{\text{src}}}\left|\tilde{f}(x^{\text{src}}, \mathbb{P}^{\text{src}}) - y^{\text{src}}\right| - \mathbb{E}_{\mathbb{P}^{\text{tgt}}}\left|\tilde{f}(x^{\text{tgt}}, \mathbb{P}^{\text{tgt}}) - y^{\text{tgt}}\right|\right| \\
&\leq 8M_0 \cdot \sup_{\tilde{f}\in\tilde{\mathcal{F}}} \left|\mathbb{E}_{\mathbb{P}^{\text{src}}}\left[\tilde{f}(x^{\text{src}}, \mathbb{P}^{\text{src}}) - y^{\text{src}}\right] - \mathbb{E}_{\mathbb{P}^{\text{tgt}}}\left[\tilde{f}(x^{\text{tgt}}, \mathbb{P}^{\text{tgt}}) - y^{\text{tgt}}\right]\right| \\
&\leq 8M_0 \cdot \sup_{\Delta\theta\in\mathcal{H}_{DA}, \tilde{f}\in\tilde{\mathcal{F}}} \left|\mathbb{E}_{\mathbb{P}^{\text{src}}}\left[\tilde{f}_\theta(x^{\text{src}}, \mathbb{P}^{\text{src}}) - \tilde{f}_{\theta+\Delta\theta}(x^{\text{src}}, \mathbb{P}^{\text{src}})\right] - \mathbb{E}_{\mathbb{P}^{\text{tgt}}}\left[\tilde{f}(x^{\text{tgt}}, \mathbb{P}^{\text{tgt}}) - \tilde{f}_{\theta+\Delta\theta}(x^{\text{tgt}}, \mathbb{P}^{\text{tgt}})\right]\right| \\
&= 8M_0 \cdot \sup_{\Delta\theta\in\mathcal{H}_{DA}, \tilde{f}\in\tilde{\mathcal{F}}} \left|\mathbb{E}_{\mathbb{P}^{\text{src}}}\left[\nabla_\theta \tilde{f}_\theta(x^{\text{src}}, \mathbb{P}^{\text{src}})\Delta\theta\right] - \mathbb{E}_{\mathbb{P}^{\text{tgt}}}\left[\nabla_\theta \tilde{f}(x^{\text{tgt}}, \mathbb{P}^{\text{tgt}})\Delta\theta\right]\right| \\
&= 8M_0 \cdot \text{MMD}_{\Theta_{DA}}\left(\mathbb{P}^{\text{src}}, \mathbb{P}^{\text{tgt}}\right)
\end{aligned}$$

which $\mathcal{H}_{DA}$ is the RKHS induced by kernel Θ_{DA}. □

A.6 Proof of Corollary 4.6

Corollary 4.6. With the assumptions in Lemma 4.1, when $\beta_{x,\tilde{x}_i} = 1/||\sum_{i=1}^n \langle\cdot, \tilde{x}_i\rangle_{\mathcal{K}_\mathcal{X}}||$, the Gaussian process induced by $\tilde{f}(x)$ at the initialization would be equivalent to the adaptive Gaussian process in [7] over NNKP kernel.

Proof. One special case of distribution-informed neural network at initialization is that when the coefficient $\beta_{x,\tilde{x}_i} = 1/||\sum_{i=1}^{n}\langle\cdot,\tilde{x}_i\rangle_{\mathcal{K}_\mathcal{X}}||$, it can be seen that $\Phi_x(\mathbb{P}) = \frac{1}{||\sum_{i=1}^{n}\langle\cdot,\tilde{x}_i\rangle_{\mathcal{K}_\mathcal{X}}||}\sum_{i=1}^{n}\langle\cdot,\tilde{x}_i\rangle_{\mathcal{K}_\mathcal{X}}$ is the normalized mean mapping [24] of data distribution $\mathbb{P}$ in the NNGP kernel space. In this case, the distribution kernel $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P},\mathbb{P}'|x,x')$ in Eq. (6) can be explained as the inner product of the mean mappings of $\mathbb{P}$ and $\mathbb{P}'$. Moreover, using the definition of maximum mean discrepancy (MMD) [24], it can be shown that $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P},\mathbb{P}'|x,x') = \frac{c-\hat{\text{MMD}}^2_{\mathcal{K}_\mathcal{X}}(\mathbb{P},\mathbb{P}')}{\bar{\Phi}_x(\mathbb{P})\bar{\Phi}_x(\mathbb{P}')}$, where $\hat{\text{MMD}}_{\mathcal{K}_\mathcal{X}}(\cdot)$ denotes the empirical MMD in the NNGP kernel space, and $c = \frac{1}{n^2}\sum_{i,j=1}^{n}\mathcal{K}_\mathcal{X}(\tilde{x}_i,\tilde{x}_j) + \frac{1}{n'^2}\sum_{i,j=1}^{n'}\mathcal{K}_\mathcal{X}(\tilde{x}'_i,\tilde{x}'_j)$. It implies that $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P},\mathbb{P}'|x,x')$ is negatively correlated with the popular MMD estimator. Moreover, when $\mathbb{P} = \mathbb{P}'$, $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P},\mathbb{P}'|x,x') = 1$, otherwise $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P},\mathbb{P}'|x,x') = \frac{1}{||\sum_{i=1}^{n}\langle\cdot,\tilde{x}_i\rangle_{\mathcal{K}_\mathcal{X}}||\cdot||\sum_{j=1}^{n'}\langle\cdot,\tilde{x}'_j\rangle_{\mathcal{K}_\mathcal{X}}||}\sum_{i=1}^{n}\sum_{j=1}^{n'}\mathcal{K}_\mathcal{X}(\tilde{x}_i,\tilde{x}'_j)$. In this case, the adaptive transfer kernel $\mathcal{K}^{DA}(X,X)$ induced by $\tilde{f}(x)$ at the initialization would be equivalent to [7], by setting the transfer parameter τ of Eq. (3) as $\mathcal{K}_{\mathscr{P}|\mathcal{X}}(\mathbb{P}^{\text{src}},\mathbb{P}^{\text{tgt}}|x^{\text{src}},x^{\text{tgt}})$ for any inputs $x^{\text{src}},x^{\text{tgt}}$. □

A.7 Proof of Theorem A.1

Theorem A.1. When $w_i^{\text{src}} > 0$ for all $i = 1,\cdots,n_{\text{src}}$, the reweighting domain adaptation approach Eq. (9) and standard supervised learning have identical predictions on test target data, i.e., $\lim_{t\to\infty}\lim_{d^*\to\infty} f_{\theta_t}(X_*^{\text{tgt}}) = f_{\theta_0}(X_*^{\text{tgt}}) - \Theta(X_*^{\text{tgt}},X)\Theta^{-1}\left(f_{\theta_0}(X) - Y\right)$.

Proof. The objective function of Eq. (7) can be rewritten as follows.

$$\mathcal{L}_{RW}(\theta) = \frac{\alpha}{2n_{\text{src}}}\sum_{i=1}^{n_{\text{src}}} w_i^{\text{src}}\,||f(x_i^{\text{src}}) - y_i^{\text{src}}||_2^2 + \frac{1-\alpha}{2n_{\text{tgt}}}\sum_{j=1}^{n_{\text{tgt}}^l}\left|\left|f(x_j^{\text{tgt}}) - y_j^{\text{tgt}}\right|\right|_2^2 = \frac{1}{2}\,||CBf(X) - CBY||_2^2$$

where $f(X) = \text{vec}(\{f(x_i)\}_{i=1}^{n_{\text{src}}+n_{\text{tgt}}^l})$, and $C = \text{diag}\{\underbrace{\sqrt{\alpha/n_{\text{src}}},\cdots,\sqrt{\alpha/n_{\text{src}}}}_{n_{\text{src}}},\underbrace{\sqrt{(1-\alpha)/n_{\text{tgt}}^l},\cdots,\sqrt{(1-\alpha)/n_{\text{tgt}}^l}}_{n_{\text{tgt}}^l}\}$ is a diagonal matrix and $B = \text{diag}\{\underbrace{\sqrt{w_1^{\text{src}}},\cdots,\sqrt{w_{n_{\text{src}}}^{\text{src}}}}_{n_{\text{src}}},\underbrace{1,\cdots,1}_{n_{\text{tgt}}^l}\}$. We have the dynamics of lineraized neural network as follows (let $A = CB$ for brevity).

$$\begin{aligned}\dot{\theta}_t = -\eta\nabla_\theta\mathcal{L}_2(\theta) &= -\eta\nabla_\theta f_{\theta_t}^{\text{lin}}(X)^T A^T A\left(f_{\theta_t}^{\text{lin}}(X) - Y\right)\\ &= -\eta\nabla_\theta f_{\theta_0}(X)^T A^T A\left(f_{\theta_0}(X) + \nabla_{\theta_0} f_{\theta_0}(X)\left(\theta_t - \theta_0\right) - Y\right)\end{aligned}$$

Then the ODE has closed form solution as

$$\theta_t = \theta_0 - \nabla_\theta f_{\theta_0}(X)^T\Theta^{-1}\left(\mathbb{I} - e^{-\eta\Theta A^T A t}\right)\left(f_{\theta_0}(X) - Y\right)$$

Then given random initialization θ_0, the predictions of this neural network over training X and testing examples X_*^{tgt} are

$$f_{\theta_t}(X_*^{\text{tgt}}) = f_{\theta_0}(X_*^{\text{tgt}}) - \Theta(X_*^{\text{tgt}},X)\Theta^{-1}\left(\mathbb{I} - e^{-\eta\Theta A^T A t}\right)\left(f_{\theta_0}(X) - Y\right)$$

Therefore, when $w_i^{\text{src}} > 0$ for all $i = 1,\cdots,n_{\text{src}}$, it can be shown that $\lambda_{\min}(\Theta A^T A) \geq \lambda_{\min}(\Theta)\lambda_{\min}(A^T A) > 0$. Then we have $\lim_{t\to\infty}\lim_{d^*\to\infty} f_{\theta_t}(X_*^{\text{tgt}}) = f_{\theta_0}(X_*^{\text{tgt}}) - \Theta(X_*^{\text{tgt}},X)\Theta^{-1}\left(f_{\theta_0}(X) - Y\right)$.

As indicated in previous work [35], the neural network $f(\cdot)$ with standard MSE loss also has the following objective function

$$\mathcal{L}_{sup}(\theta) = \frac{1}{2}\sum_{i=1}^{n_{\text{src}}}||f(x_i^{\text{src}}) - y_i^{\text{src}}||_2^2 + \frac{1}{2}\sum_{j=1}^{n_{\text{tgt}}^l}\left|\left|f(x_j^{\text{tgt}}) - y_j^{\text{tgt}}\right|\right|_2^2 = \frac{1}{2}\,||f(X) - Y||_2^2$$

For any test examples X_*^{tgt}, it has

$$f_{\theta_t}^{sup}(X_*^{\text{tgt}}) = f_{\theta_0}(X_*^{\text{tgt}}) - \Theta(X_*^{\text{tgt}}, X)\Theta^{-1}\left(\mathbb{I} - e^{-\eta\Theta t}\right)\left(f_{\theta_0}(X) - Y\right)$$

and $\lim_{t\to\infty}\lim_{d^*\to\infty} f_{\theta_t}^{sup}(X_*^{\text{tgt}}) = f_{\theta_0}(X_*^{\text{tgt}}) - \Theta(X_*^{\text{tgt}}, X)\Theta^{-1}\left(f_{\theta_0}(X) - Y\right)$. This indicates that when $w_i^{\text{src}} > 0$ for all $i = 1, \cdots, n_{\text{src}}$, the reweighting domain adaptation approach and standard supervised learning have identical predictions on test target data. □

A.8 Proof of Corollary 4.7

Corollary 4.7. With the assumption in Theorem 4.4, our framework Eq. (7) with distribution-informed neural network $\tilde{f}(\cdot)$ can recover the popular reweighting domain adaptation approach Eq. (9) in the function space.

Proof. We can consider the following special case of the distribution representation in Eq. (4). For source example x, we can set $\beta_{x,x_i} = \frac{1}{n_{\text{src}}||x_i||_{\mathcal{K}_\mathcal{X}}} \cdot \delta[w_i^{\text{src}} > 0]$, where $\delta[\cdot]$ is a Kronecker delta function. For target example x, we can set $\beta_{x,x_j} = \frac{1}{n_{\text{tgt}}||x_j||_{\mathcal{K}_\mathcal{X}}}$ where $n_{\text{tgt}} = n_{\text{tgt}}^l + n_{\text{tgt}}^u$, as we use all the target training examples as the basis examples for learning the input-oriented representation of the target domain. In this case, for distribution-informed NTK $\Theta_{DA}(X, X)$, we have $\Theta_{DA}(x_i, x_j) = 0$ if x_i (or x_j) are source example with $w_i^{\text{src}} > 0$ (or $w_j^{\text{src}} > 0$), $\Theta_{DA}(x_i, x_j) = \Theta(x_i, x_j)$ otherwise. That is, when $w_i^{\text{src}} > 0$ for all $i = 1, \cdots, n_{\text{src}}$, we have $\Theta_{DA}(X, X) = \Theta(X, X)$. Theorem 4.4 indicates that it can produce the same prediction function over random initialization as the reweighting domain adaptation approach Eq. (9). When there are some source examples with $w_i^{\text{src}} = 0$, the objective function of our domain adaptation will also simply filter out those source examples, and the final prediction function depends only on the source examples with $w_i^{\text{src}} > 0$. □

A.9 Proof of Corollary 4.8

Corollary 4.8. Under mild conditions, our framework Eq. (7) with distribution-informed neural network $\tilde{f}(\cdot)$ can recover the standard domain invariant representation learning in Eq. (1), where the domain discrepancy measure $d(\cdot, \cdot)$ is instantiated with MMD in RKHS induced by standard NTK Θ.

Proof. It can be shown when the distribution representation of Eq. (4) is shared by all the input examples. In this case, for any w_g, $\tilde{f}(x, \mathbb{P}) = a \cdot f_\theta(x)$ where $a = \Phi_x(\mathbb{P})^T w_g \in \mathbb{R}$ is shared by all examples. Then, we have $\tilde{f}(x, \mathbb{P}) = a \cdot f_\theta(x) = a \cdot \phi_{\theta^{<L}}(x)^T w = \phi_{\theta^{<L}}(x)^T(aw)$. Therefore, in this case, $\tilde{f}(\cdot)$ is equivalent to a simple distribution-free neural network. The overall framework Eq. (7) would be degenerated into the standard domain invariant representation learning in Eq. (1). Notice that when using the distribution-free neural network, the framework Eq. (1) requires an additional discrepancy minimization regularization to guarantee the success of domain adaptation. In our case, the discrepancy minimization regularization would be given by the MMD in RKHS induced by the NTK Θ_{DA}. Here, when the distribution representation of Eq. (4) is shared by all the input examples, it holds that for any $x, x' \in \mathcal{X}$, $\Theta_{DA}(x, x') = b \cdot \Theta(x, x')$ where $b = ||\Phi_x(\mathbb{P})||_{\mathcal{K}_\mathcal{X}}^2 \in \mathbb{R}$. Thus, $\hat{\text{MMD}}_{\Theta_{DA}}^2(\mathbb{P}^{\text{src}}, \mathbb{P}^{\text{tgt}}) = b \cdot \hat{\text{MMD}}_{\Theta}^2(\mathbb{P}^{\text{src}}, \mathbb{P}^{\text{tgt}})$. □

A.10 Algorithms

DINO-INIT vs. DINO-TRAIN: `DINO-INIT` is weakly-trained (i.e., only the last layer is trained), while `DINO-TRAIN` is fully-trained (i.e., all network layers are trained). We observe from the experimental results that the weakly-trained `DINO-INIT` might outperform `DINO-TRAIN` in some cases. This observation is consistent with previous work [33] on standard neural networks.

The overall training procedures of `DINO-INIT` are illustrated in Algorithm 1. Here we use both labeled target examples and unlabeled target examples $X_l^{\text{tgt}} \cup X_u^{\text{tgt}}$ as the basis target examples of Eq. (4). This allows the proposed `DINO-INIT` to be applied for domain adaptation scenarios [5, 13, 47] where both limited labeled and adequate unlabeled target data are available during model training. When no unlabeled target data is available [7], i.e., only limited labeled target data is given, we can simply use those labeled target examples as the basis target examples of Eq. (4).

Algorithm 1 DINO-INIT

Input: Labeled source examples $(X^{\text{src}}, Y^{\text{src}})$, labeled target examples $(X_l^{\text{tgt}}, Y_l^{\text{tgt}})$ and unlabeled target examples X_u^{tgt}, neural network architecture of $f(\cdot)$.
Output: Predictions on testing target examples X_*^{tgt}.
1: Set X^{src} to be basis source examples of Eq. (4);
2: Set $X_l^{\text{tgt}} \cup X_u^{\text{tgt}}$ to be basis target examples of Eq. (4);
3: Calculate the basis NNGP kernels of Eq. (6);
4: Estimate the coefficient w^r and noise variance σ_{src}, σ_{tgt} by maximizing $p(Y_l^{\text{tgt}}|X_l^{\text{tgt}}, X^{\text{src}}, Y^{\text{src}})$;
5: Calculate the posterior distribution with $\bar{\mu}$ and $\bar{\Sigma}$;
6: Output $\hat{Y}_*^{\text{tgt}}|X_*^{\text{tgt}} \sim \mathcal{N}(\bar{\mu}, \bar{\Sigma})$.

In addition, the overall training procedures of DINO-TRAIN are illustrated in Algorithm 2. Note that in this paper, we focus on analyzing the training dynamics of our model in the adaptation scenarios where limited labeled data in the target domain is available. But our theoretical results can be extended to unsupervised domain adaptation setting where only unlabeled data is available in the target domain. For example, the proposed DINO-TRAIN framework of Eq. (7) can be trained with $\alpha = 1$[2] for unsupervised domain adaptation. Then we can show similar convergence and generalization results of this framework.

Analysis of the reweighting approach Eq. (9): Notice that if we do not consider the distribution shift between source and target domain, the prediction function of reweighting approach Eq. (9) can be simply learned by minimizing the standard supervised learning loss: $\mathcal{L}_{sup}(\theta) = \frac{1}{2}\sum_{i=1}^{n_{\text{src}}} (f(x_i^{\text{src}}) - y_i^{\text{src}})_2^2 + \frac{1}{2}\sum_{j=1}^{n_{\text{tgt}}^l} \left(f(x_j^{\text{tgt}}) - y_j^{\text{tgt}}\right)_2^2$ over all the labeled training examples. The following theorem shows that for an infinitely-wide neural network $f(\cdot)$, the reweighting domain adaptation approach Eq. (9) and the standard supervised learning have identical prediction function when they converges.

Theorem A.1. *When $w_i^{src} > 0$ for all $i = 1, \cdots, n_{src}$, in the limit of infinite network width, the reweighting domain adaptation approach Eq. (9) and the standard supervised learning would have identical prediction function on testing target data, i.e.,* $\lim_{t\to\infty} f_{\theta_t}(X_*^{tgt}) = f_{\theta_0}(X_*^{tgt}) - \Theta(X_*^{tgt}, X)\Theta^{-1}\left(f_{\theta_0}(X) - Y\right)$.

This result can be generalized to the scenarios where $w_i^{\text{src}} \geq 0$. In this case, the reweighting domain adaptation approach Eq. (9) considers only the source examples with $w_i^{\text{src}} > 0$. It is equivalent to the standard supervised learning over those source examples. Therefore, we have the following observations. (1) The weight w_i^{src} can only filter out some unrelated source examples (i.e., $w_i^{\text{src}} = 0$) for the reweighting approach. (2) For neural networks with infinite width, the learned predictive functions of the reweighting approach and the standard supervised learning are equivalent in the function space. Compared to the reweighting approach Eq. (9), our framework of Eq. (7) learns the distribution-informed representation for both source and target data. The resulting prediction function is explicitly determined by the domain discrepancy (indicated by the distribution-informed NTK Θ_{DA}).

Extension: We would like to point out that the proposed DINO framework can be easily generalized to other network architectures. This is because previous works [4, 22, 53, 54] have shown the existence of NNGP and NTK in different network architectures, including (residual) convolutional neural networks, recurrent neural networks, transformer, etc. Therefore, we can adapt our algorithms and theoretical analysis to those network architectures as well. In this paper, we focus on the most used fully connected network, and the exploration of network architecture comparison in our framework is beyond the scope of this paper.

Limitations: In this paper, we assume that there is only one source domain. But in real scenarios, it is possible to gather source information from multiple domains. In the context of multi-source domain adaptation regression, the generalization and convergence analysis of domain adaptation regression with neural network might provide the insight on selecting high-quality source data for better knowledge transfer. We would like to leave it as our future work.

[2] In Subsection 4.3, we consider $\alpha \in (0, 1)$, as the framework with $\alpha = 1$ or $\alpha = 0$ will produce a slightly different convergence result.

Algorithm 2 DINO-TRAIN

Input: Labeled source examples $(X^{\text{src}}, Y^{\text{src}})$, labeled target examples $(X_l^{\text{tgt}}, Y_l^{\text{tgt}})$ and unlabeled target examples X_u^{tgt}, neural network architecture of $f(\cdot)$.
Output: Predictions on testing target examples X_*^{tgt}.

1: Set X^{src} to be basis source examples of Eq. (4);
2: Set $X_l^{\text{tgt}} \cup X_u^{\text{tgt}}$ to be basis target examples of Eq. (4);
3: Calculate the basis NNGP kernels of Eq. (6);
4: **repeat**
5: Minimize the objective function of Eq. (7);
6: **until** It is converged
7: Output $\hat{Y}_*^{\text{tgt}} = \tilde{f}(X_*^{\text{tgt}})$.

A.11 Experimental Details

A.11.1 Data Sets

dSprites [40]: It is composed of 737,280 images from three domains: Color (C), Noisy (N) and Scream (S). Following [10], we evaluate all the baselines on six adaptation benchmarks: C $\rightarrow$ N, C $\rightarrow$ S, N $\rightarrow$ C, N $\rightarrow$ S, S $\rightarrow$ C, and S $\rightarrow$ N. For each image, it has three regression tasks, i.e., predicting three factors of variations (scale, position X and Y).

MPI3D [23]: It contains over 3M images from three domains: Toy (T), Realistic (RC) and Real (RL). We evaluate all the baselines on six benchmarks: T $\rightarrow$ RC, T $\rightarrow$ RL, RC $\rightarrow$ T, RC $\rightarrow$ RL, RL $\rightarrow$ T, and RL $\rightarrow$ RC. It is shown [10] that for each image, it involves two regression tasks, i.e., predicting three factors of variations (position X and Y).

Plant Phenotyping: It aims to predict diverse traits (e.g., Nitrogen) of plants related to the plants' growth using leaf hyperspectral reflectance (i.e., spectral wavelengths 500-2400 nm). Then the input example with spectral wavelengths 500-2400 nm is formulated as a 1901-dimensional feature vector. Here we consider the following two domains [46]: Maize (M) and Maize_UNL (MU). In our case, the task is to predict the Nitrogen content of maize using the leaf hyperspectral reflectance.

For dSprites and MPI3D, following [10], we use the default train/test split for the target domain. In this case, we randomly choose 100 training target images as the labeled examples, and others as the unlabeled ones for all our experiments. For Plant Phenotyping, for the target domain, we randomly 5% of data as the label examples, and others as the unlabeled ones.

A.11.2 Implementations

All the experiments are performed on a Windows machine with four 3.80GHz Intel Cores, 64GB RAM and two NVIDIA Quadro RTX 5000 GPUs. In the experiments, our algorithms are implemented based on a L-layer ($L = 6$) fully-connected neural network with ReLU activation function. In dSprites and MPI3D data sets, the size of input images is $64 \times 64 \times 3$, we simply vectorize the input image to a 12288-dimensional vector. That is, each input image $x \in \mathbb{R}^{12288}$ can be learned by the fully-connected neural network. The NNGP and neural tangent kernels induced by this neural network can be estimated using the Neural Tangents package [41]. In addition, we set $\alpha = 0.5$ and $\mu = 0.1$ for our DINO-TRAIN method. Note that different from previous works [37, 10], all the baselines will be trained from scratch and use the same neural network architecture for domain adaptation. It is shown [59, 10] that initializing the neural network using existing pre-trained models (e.g. ResNet-50 [26]) can improve the domain adaptation performance, but it is out of the scope of this paper.

In our experiments, we use three deep domain adaptation baselines: DAN [37], WANN [16] and RSD [10]. DAN [37] and RSD [10] focus on domain invariant representation learning (see Subsection 3.2.1) by empirically minimizing the prediction error over the labeled training examples and the domain discrepancy (i.e., maximum mean discrepancy or representation subspace distance). WANN uses the reweighting technique in Subsection 3.2.2. Notice that DAN and RSD are proposed for unsupervised domain adaptation with no label information from the target domain. In the experiments, for a fair comparison, we extend them to domain adaptation scenarios with little label information from the target domain, by adding the prediction error over the labeled training target examples.

Besides, Figure 3b shows the comparison of the proposed gradient-based MMD $\hat{\text{MMD}}_{\Theta_{DA}}(\cdot,\cdot)$ and conventional MMD with RBF kernel for domain adaptation in Subsection 5.2. To be more specific, we consider a simple 3-layer fully-connected neural network with ReLU. Following [37], we implement the domain adaptation approach by minimizing the prediction error over the labeled training examples and the MMD with RBF kernel over the source and target features learned by the first $l = 1, 2, 3$ layers. We denote those approaches as 'MMD-RBF (layer 1)', 'MMD-RBF (layer 2)' and 'MMD-RBF (layer 3)' in Figure 3b, respectively. For a fair comparison, we use the same neural network architecture to implement the domain adaptation approach with our proposed $\hat{\text{MMD}}_{\Theta_{DA}}(\cdot,\cdot)$, which is denoted as 'MMD-NTK' in Figure 3b.