Meta Two-Sample Testing: Learning Kernels for Testing with Limited Data

Anonymous Author(s)
Affiliation
Address
email

Abstract

Modern kernel-based two-sample tests have shown great success in distinguishing complex, high-dimensional distributions with appropriate learned kernels. Previous work has demonstrated that this kernel learning procedure succeeds, assuming a considerable number of observed samples from each distribution. In realistic scenarios with very limited numbers of data samples, however, it can be challenging to identify a kernel powerful enough to distinguish complex distributions. We address this issue by introducing the problem of *meta two-sample testing (M2ST)*, which aims to exploit (abundant) auxiliary data on related tasks to find an algorithm that can quickly identify a powerful test on new target tasks. We propose two specific algorithms for this task: a generic scheme which improves over baselines and a more tailored approach which performs even better. We provide both theoretical justification and empirical evidence that our proposed meta-testing schemes outperform learning kernel-based tests directly from scarce observations, and identify when such schemes will be successful.

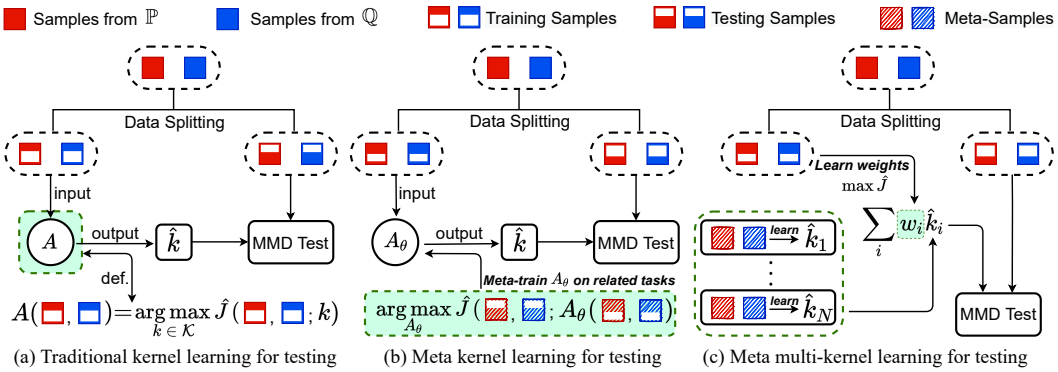
1 Introduction

Two-sample tests ask, "given samples from each, are these two populations the same?" For instance, one might wish to know whether a treatment and control group differ. With very low-dimensional data and/or strong parametric assumptions, methods such as t-tests or Kolmogorov-Smirnov tests are widespread. Recent work in statistics and machine learning has sought tests that cover situations not well-handled by these classic methods [1–16], providing tools useful in machine learning for domain adaptation, causal discovery, generative modeling, fairness, adversarial learning, and more [8, 17–21]. Perhaps the most powerful known widely-applicable scheme is based on a kernel method known as the Maximum Mean Discrepancy, MMD [1] – or, equivalently [22], the energy distance [3] – when one *learns* an appropriate kernel for the task at hand [10, 16]. Here, one divides the observed data into "training" and "testing" splits, identifies a kernel on the training data by maximizing a power criterion $\hat{J}$, then runs an MMD test on the testing data (as illustrated in Figure 1a). This method generally works very well, when enough data is available for both training and testing.

In real-world scenarios, however, two-sample testing tasks can be challenging if we do not have very many data observations. For example, in medical imaging, we might face two small datasets of lung computed tomography (CT) scans of patients with coronavirus diseases, and wish to know if these patients are affected in different ways. If they are from different distributions, the virus causing the disease may have mutated. Here, previous tests are likely to be relatively ineffective; we cannot learn a powerful kernel to distinguish such complex distributions with only a few observations.

In this paper, we address this issue by considering a problem setting where related testing tasks are available. We use those related tasks to identify a kernel selection *algorithm*. Specifically, instead of using a fixed algorithm A to learn a kernel ("maximize $\hat{J}$ among this class of deep kernels"), we

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) Traditional kernel learning for testing (b) Meta kernel learning for testing (c) Meta multi-kernel learning for testing

Figure 1: Comparison among (a) traditional kernel learning, (b) meta kernel learning and (c) meta multi-kernel learning for kernel two-sample testing, where $\hat{k}/\hat{k}_i$ is the learned kernel.

want to learn an algorithm A_θ from auxiliary data (Figure 1b):

$$\arg\max_{A_\theta} \mathbb{E}_{(\mathbb{P},\mathbb{Q})\sim\tau}\left[\hat{J}(S^{te}_{\mathbb{P}}, S^{te}_{\mathbb{Q}}; A_\theta(S^{tr}_{\mathbb{P}}, S^{tr}_{\mathbb{Q}}))\right], \tag{1}$$

where $\mathbb{P}, \mathbb{Q}$ are distributions sampled from a meta-distribution of related tasks τ, with corresponding observed sample sets $S_{\mathbb{P}}, S_{\mathbb{Q}}$. In analogy with meta-learning [23–27], we call this learning procedure *meta kernel learning* (Meta-KL). We can then apply A_θ to select a kernel on our actual testing task, then finally run an MMD test as before (Figure 1b).

The adaptation performed by A_θ, however, might still be very difficult to achieve well with few training observations; even the best A_θ found by a generic adaptation scheme might over-fit to $S^{tr}_{\mathbb{P}}, S^{tr}_{\mathbb{Q}}$. For more stable procedures and, in our experiments, more powerful final tests, we propose *meta multi-kernel learning* (Meta-MKL). This algorithm independently finds the most powerful kernel for each related task; at adaptation time, we select a convex combination of those kernels for testing (Figure 1c), as in standard multiple kernel learning [28] and similarly to ensemble methods in few-shot classification [29, 30]. Because we are only learning a small number of weights rather than all of the parameters of a deep network, this adaptation can quickly find a high-quality kernel.

We provide both theoretical and empirical evidence that Meta-MKL is better than generic Meta-KL, and that both outperform approaches that do not use related task data, in low-data regimes. We find that learned algorithms can output kernels with high test power using only a few samples, where "plain" kernel learning techniques entirely fail.

2 Preliminaries

We will now review the setting of learning a kernel for a two-sample test, following [16]. Let $\mathcal{X} \subset \mathbb{R}^d$ and $\mathbb{P}, \mathbb{Q}$ be (unknown) Borel probability measures on $\mathcal{X}$, with $S_{\mathbb{P}} = \{\boldsymbol{x}_i\}_{i=1}^m \sim \mathbb{P}^m$ and $S_{\mathbb{Q}} = \{\boldsymbol{y}_j\}_{j=1}^m \sim \mathbb{Q}^m$ observed i.i.d. samples from these distributions. We operate in a classical hypothesis testing setup, with the null hypothesis that $\mathbb{P} = \mathbb{Q}$.

Maximum mean discrepancy (MMD). The basic tool we use is a kernel-based distance metric between distributions called the MMD, defined as follows. (The energy distance [3] is a special case of the MMD for a particular choice of k [22].)

Definition 1 (MMD [1])**.** *Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be the bounded kernel of an RKHS $\mathcal{H}_k$ (i.e., $\sup_{x,y\in\mathcal{X}} |k(x,y)| < \infty$). Letting $X, X' \sim \mathbb{P}$ and $Y, Y' \sim \mathbb{Q}$ be independent random variables,*

$$\mathrm{MMD}(\mathbb{P}, \mathbb{Q}; k) = \sup_{f\in\mathcal{H}_k, \|f\|_{\mathcal{H}_k}\leq 1} \mathbb{E}[f(X)] - \mathbb{E}[f(Y)] = \sqrt{\mathbb{E}\left[k(X,X') + k(Y,Y') - 2k(X,Y)\right]}.$$

If k is characteristic*, we have that* $\mathrm{MMD}(\mathbb{P}, \mathbb{Q}; k) = 0$ *if and only if* $\mathbb{P} = \mathbb{Q}$.

We can estimate MMD using the following U-statistic estimator, which is unbiased for MMD^2 (denoted by $\widehat{\mathrm{MMD}}_u^2$) and has nearly minimal variance among unbiased estimators [1]:

$$\widehat{\mathrm{MMD}}_u^2(S_{\mathbb{P}}, S_{\mathbb{Q}}; k) := \frac{1}{n(n-1)} \sum_{i \neq j} H_{ij}^{(k)}, \tag{2}$$

$$H_{ij}^{(k)} := k(\boldsymbol{x}_i, \boldsymbol{x}_j) + k(\boldsymbol{y}_i, \boldsymbol{y}_j) - k(\boldsymbol{x}_i, \boldsymbol{y}_j) - k(\boldsymbol{y}_i, \boldsymbol{x}_j). \tag{3}$$

Testing. Under the null hypothesis $\mathfrak{H}_0$, $m\widehat{\mathrm{MMD}}_u^2$ converges in distribution as $m \to \infty$ to some distribution depending on $\mathbb{P}$ and k [1, Theorem 12]. We can thus build a test with p-value equal to the quantile of our test statistic $m\widehat{\mathrm{MMD}}_u^2$ under this distribution. Although there are several methods to estimate this null distribution, it is usually considered best [10] to use a permutation test [31, 32]: under $\mathfrak{H}_0$, samples from $\mathbb{P}$ and $\mathbb{Q}$ are interchangeable, and repeatedly re-computing the statistic with samples randomly shuffled between $S_{\mathbb{P}}$ and $S_{\mathbb{Q}}$ estimates its null distribution.

Test power. We generally want to find tests likely to reject $\mathfrak{H}_0$ when indeed it holds that $\mathbb{P} \neq \mathbb{Q}$; the probability of doing so (for a particular $\mathbb{P}$, $\mathbb{Q}$, k and m) is called *power*. For reasonably large m, [10, 16] argue that the power is an almost-monotonic function of

$$J(\mathbb{P}, \mathbb{Q}; k) := \frac{\mathrm{MMD}^2(\mathbb{P}, \mathbb{Q}; k)}{\sigma_{\mathfrak{H}_1}(\mathbb{P}, \mathbb{Q}; k)}, \quad \sigma_{\mathfrak{H}_1}^2(\mathbb{P}, \mathbb{Q}; k) := 4\left(\mathbb{E}\left[H_{ij}^{(k)} H_{i\ell}^{(k)}\right] - \mathbb{E}\left[H_{ij}^{(k)}\right]^2\right). \tag{4}$$

Here, $\sigma_{\mathfrak{H}_1}^2$ is the asymptotic variance of $\sqrt{m}\,\widehat{\mathrm{MMD}}_u^2$ under $\mathfrak{H}_1$; it is defined in terms of an expectation of (3) with respect to the data samples $S_{\mathbb{P}}$, $S_{\mathbb{Q}}$, for i, j, ℓ distinct. The criterion (4) depends on the unknown distributions; we can estimate it from samples with the regularized estimator [16]

$$\hat{J}_\lambda(S_{\mathbb{P}}, S_{\mathbb{Q}}; k) := \widehat{\mathrm{MMD}}_u^2(S_{\mathbb{P}}, S_{\mathbb{Q}}; k) \Big/ \sqrt{\hat{\sigma}_{\mathfrak{H}_1}^2(S_{\mathbb{P}}, S_{\mathbb{Q}}; k) + \lambda}, \tag{5}$$

$$\hat{\sigma}_{\mathfrak{H}_1}^2(S_{\mathbb{P}}, S_{\mathbb{Q}}; k) := \frac{4}{m^3} \sum_{i=1}^{m} \left(\sum_{j=1}^{m} H_{ij}^{(k)}\right)^2 - \frac{4}{m^4} \left(\sum_{i=1}^{m} \sum_{j=1}^{m} H_{ij}^{(k)}\right)^2. \tag{6}$$

Kernel choice. Given two samples $S_{\mathbb{P}}$ and $S_{\mathbb{Q}}$, the best kernel is (essentially) the one that maximizes J in (4). If we pick a kernel to maximize our estimate $\hat{J}$ using the same data that we use for testing, though, we will "overfit," and reject $\mathfrak{H}_0$ far too often. Instead, we use data splitting [2, 5, 10]: we partition the samples into two disjoint sets, $S_{\mathbb{P}} = S_{\mathbb{P}}^{tr} \cup S_{\mathbb{P}}^{te}$. We obtain $k^{tr} = A(S_{\mathbb{P}}^{tr}, S_{\mathbb{Q}}^{tr}) \approx \arg\max_k \hat{J}_\lambda(S_{\mathbb{P}}^{tr}, S_{\mathbb{Q}}^{tr}; k)$, then conduct a permutation test based on $\mathrm{MMD}(S_{\mathbb{P}}^{te}, S_{\mathbb{Q}}^{te}; k^{tr})$. This process is summarized in Algorithm 2 and illustrated in Figure 1a.

This procedure has been successfully used not only to, e.g., pick the best bandwidth for a simple Gaussian kernel, but even to learn all the parameters of a kernel like (8) which incorporates a deep network architecture [10, 16]. As argued by [16], classifier two-sample tests [8, 33] (which test based on the accuracy of a classifier distinguishing $\mathbb{P}$ from $\mathbb{Q}$) are also essentially a special case of this framework – and more-general deep kernel MMD tests tend to work better. Although presented here specifically for $\widehat{\mathrm{MMD}}_u$, an analogous procedure has been used for many other problems, including other estimates of the MMD and closely-related quantities [2, 5, 34].

When data splitting, the training split must be big enough to identify a good kernel; with too few training samples m^{tr}, $\hat{J}_\lambda$ will be a poor estimator, and the kernel will overfit. The testing split, however, must also be big enough: for a given $\mathbb{P}$, $\mathbb{Q}$ and k, it becomes much easier to be confident that $\mathrm{MMD}(\mathbb{P}, \mathbb{Q}; k) > 0$ as m^{te} grows and the variance in $\widehat{\mathrm{MMD}}_u(\mathbb{P}, \mathbb{Q}; k)$ accordingly decreases. When the number of available samples is small, both steps suffer. This work seeks methods where, by using related testing tasks, we can identify a good kernel with an extremely small m^{tr}; thus we can reserve most of the available samples for testing, and overall achieve a more powerful test.

Another class of techniques for kernel selection avoids the need for data splitting via selective inference [15]. At least as studied by [15], however, it is currently available only for restricted classes of kernels and with the much-less-accurate "streaming" estimates of the MMD, which for fixed kernels can yield *far* less powerful tests than $\widehat{\mathrm{MMD}}_u$ [35]. In Section 5.4, we will demonstrate that in our settings, the data-splitting approach is empirically far more powerful.

Algorithm 1 Meta Kernel Learning (Meta-KL)

Input: Meta-samples $\boldsymbol{S} = \{(S_{\mathbb{P}^i}, S_{\mathbb{Q}^i})\}_{i=1}^N$; kernel architecture (8) and parameters ω_0; regularization λ
1. Initialize algorithm parameters: $\theta := [\omega_{start}] \leftarrow [\omega_0]$
2. Define a parameterized learning algorithm $A_\theta(S_{\mathbb{P}}, S_{\mathbb{Q}})$ as:
  $\omega \leftarrow \omega_{start}$; **for** $t = 1, \ldots, n_{steps}$ **do** $\omega \leftarrow \omega + \eta \nabla_\omega \hat{J}_\lambda(S_{\mathbb{P}}, S_{\mathbb{Q}}; k_\omega)$; **end for**; **return** k_ω
for $T = 1, 2, \ldots, T_{max}$ **do**
  3: Sample $\mathcal{I}$ as a set of indices in $\{1, 2, \ldots, N\}$ of size n_{batch}
  for $i \in \mathcal{I}$ **do**
    4: Split data as $S_{\mathbb{P}^i} = S^{tr}_{\mathbb{P}^i} \cup S^{te}_{\mathbb{P}^i}$ and $S_{\mathbb{Q}^i} = S^{tr}_{\mathbb{Q}^i} \cup S^{te}_{\mathbb{Q}^i}$;
    5: Apply the learning algorithm: $k_i \leftarrow A_\theta(S^{tr}_{\mathbb{P}^i}, S^{tr}_{\mathbb{Q}^i})$
  end for
  6: Update $\theta \leftarrow \theta + \beta \nabla_\theta \sum_{i \in \mathcal{I}} \hat{J}_\lambda(S^{te}_{\mathbb{P}^i}, S^{te}_{\mathbb{Q}^i}; k_i)$; *# update θ, i.e. w_{start}, to maximize $\mathcal{J}(A_\theta, \tau)$*
end for
7: return A_θ

3 Meta Two-Sample Testing

To handle cases with low numbers of available data samples, we consider a problem setting where *related* testing tasks are available. We use those tasks in a framework inspired by meta-learning [e.g. 23], where we use those related tasks to identify a kernel selection *algorithm*, e.g. (1).

Specifically, we define a *task* as a pair $\mathcal{T} = (\mathbb{P}, \mathbb{Q})$ of distributions over $\mathcal{X}$ we would like to distinguish, and assume a meta-distribution τ over the space of tasks $\mathcal{T}$.

Definition 2 (M2ST)**.** *Assume we are assigned (unobserved) training tasks $\boldsymbol{T} = \{\mathcal{T}^i = (\mathbb{P}^i, \mathbb{Q}^i)\}_{i=1}^N$ drawn from a task distribution τ, and observe meta-samples $\boldsymbol{S} = \{(S^i_{\mathbb{P}}, S^i_{\mathbb{Q}})\}_{i=1}^N$ with $S^i_{\mathbb{P}} \sim (\mathbb{P}^i)^{n_i}$ and $S^i_{\mathbb{Q}} \sim (\mathbb{Q}^i)^{n_i}$. Our goal is to use these meta-samples to find a kernel learning algorithm A_θ, such that for a target task $(\mathbb{P}', \mathbb{Q}') \sim \tau$ and samples $S_{\mathbb{P}'} \sim (\mathbb{P}')^{n'}$, $S_{\mathbb{Q}'} \sim (\mathbb{Q}')^{n'}$, $A_\theta(S_{\mathbb{P}'}, S_{\mathbb{Q}'})$ returns a kernel which will achieve high test power on $(\mathbb{P}', \mathbb{Q}')$.*

We measure the performance of A_θ based on the expected test power criterion for a target task:

$$\mathcal{J}(A_\theta, \tau) = \mathbb{E}_{(\mathbb{P},\mathbb{Q})\sim\tau}\left[\mathbb{E}_{S^{tr}_{\mathbb{P}}\sim\mathbb{P}^m, S^{tr}_{\mathbb{Q}}\sim\mathbb{Q}^m}\left[J(\mathbb{P}, \mathbb{Q}; A_\theta(S^{tr}_{\mathbb{P}}, S^{tr}_{\mathbb{Q}}))\right]\right]. \tag{7}$$

If τ were in some sense "uniform over all conceivable tasks," then a no-free-lunch property would cause M2ST to be hopeless. Instead, our assumption is that tasks from τ are "related" enough that we can make progress at improving (7). We will propose two approaches to finding an A_θ for M2ST.

Neither is specific to any particular kernel parameterization, but for the sake of concreteness, we will follow Liu et al. [16] in choosing the form

$$k_\omega(x, y) = [(1 - \epsilon)\kappa(\phi(x), \phi(y)) + \epsilon]q(x, y), \tag{8}$$

where ϕ is a deep neural network which extracts features from the samples, and κ is a simple kernel (e.g., a Gaussian) on those features, while q is a simple characteristic kernel (e.g. Gaussian) on the input space; $\epsilon \in (0, 1]$ ensures that every kernel of the form k_ω will be characteristic. Here, ω represents all parameters in (8): most will be in ϕ, but κ and q may have a few parameters (e.g. length scales), and we can also learn ϵ.

Meta-KL. We first propose Algorithm 1 as a standard approach to optimizing (7), à la MAML [23]: A_θ takes a small, fixed number of gradient ascent steps in $\hat{J}_\lambda$ for the parameters of k_ω, starting from a learned initialization point $\omega_{start} \in \theta$ (lines 1-2). We differentiate through A_θ, and perform stochastic gradient ascent to find a good value of θ based on the meta-training sets (lines 3-6, also illustrated in Figure 2). Once we have learned a kernel selection procedure, we can again apply it to a testing task with Algorithm 2.

As we will see in the experiments, this approach does indeed use the meta-tasks to improve performance on target tasks. Differently from usual meta-learning settings, as in e.g. classification [23], however, here it is conceivable that there is a single good kernel that works for all tasks from τ; improving on this single baseline kernel, rather than simply overfitting to the very few target points, may be quite difficult. Thus, in practice, the amount of adaptation that A_θ actually performs in its gradient ascent can be somewhat limited.

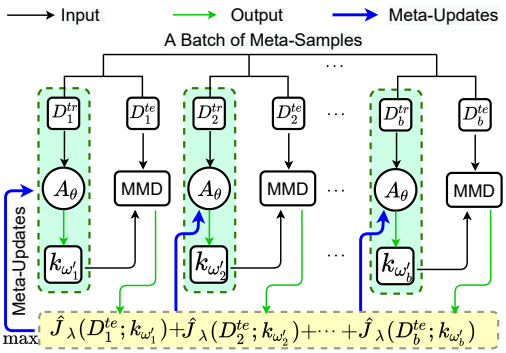


Figure 2: Illustration of Algorithm 1.

Algorithm 2 Testing with a Kernel Learner

1: Input: Two samples: $S_{\mathbb{P}}, S_{\mathbb{Q}}$; algorithm A_θ
2: Split data as $S_{\mathbb{P}} = S_{\mathbb{P}}^{tr} \cup S_{\mathbb{P}}^{te}$ and $S_{\mathbb{Q}} = S_{\mathbb{Q}}^{tr} \cup S_{\mathbb{Q}}^{te}$;
3: Learn a kernel $k_\omega = A_\theta(S_{\mathbb{P}}^{tr}, S_{\mathbb{Q}}^{tr})$;
4: Compute $est \leftarrow \widehat{\mathrm{MMD}}_u^2(S_{\mathbb{P}}^{te}, S_{\mathbb{Q}}^{te}; k_\omega)$;
for $i = 1, 2, \ldots, n_{perm}$ **do**
  5: Shuffle $S_{\mathbb{P}}^{te} \cup S_{\mathbb{Q}}^{te}$ into X and Y
  6: Compute $perm_i \leftarrow \widehat{\mathrm{MMD}}_u^2(X, Y; k_\omega)$;
end for
7: Output: p-value $\frac{1}{n_{perm}} \sum_{i=1}^{n_{perm}} \mathbb{1}(perm_i \geq est)$;

Algorithm 3 Meta Multi-Kernel Learning (Meta-MKL)

Input: Meta-samples $\boldsymbol{S} = \{(S_{\mathbb{P}^i}, S_{\mathbb{Q}^i})\}_{i=1}^N$; kernel architecture as in (8)
for $i = 1, 2, \ldots, N$ **do**
  1: Optimize $\omega_i \leftarrow \widehat{\arg\max}_\omega \hat{J}_\lambda(S_{\mathbb{P}^i}, S_{\mathbb{Q}^i}; k_\omega)$ with some optimization algorithm;
end for
2: Define $\mathcal{K} := \{\sum_{i=1}^N \frac{\exp(w_i)}{\sum_{j=1}^N \exp(w_j)} k_{\omega_i} : w \in \mathbb{R}^N\}$; *# convex combinations of kernels, softmax weights*
3: return the algorithm $A(S_{\mathbb{P}}, S_{\mathbb{Q}}) = \widehat{\arg\max}_{k \in \mathcal{K}} \hat{J}_\lambda(S_{\mathbb{P}}, S_{\mathbb{Q}}; k)$;

Meta-MKL. As an alternative approach, we also consider a different strategy for A_θ which may be able to adapt with many fewer data samples, albeit in a possibly weaker class of possible kernels. Here, to select an A_θ, we simply find the best kernel independently for each of the meta-training tasks. Then A_θ chooses the best *convex combination* of these kernels, as in classical multiple kernel learning [28] and similarly to ensemble methods in few-shot classification [30]. At adaptation time, we only attempt to learn N weights, rather than adapting all of the parameters of a deep network; but, if the meta-training tasks contained some similar tasks to the target task, then we should be able to find a powerful test. This procedure is detailed in Algorithm 3 and illustrated in Figure 1c.

4 Theoretical analysis

We now analyze and compare the theoretical performance of direct optimizing the regularized test power from small sample size with our proposed meta-training procedures. To study our learning objective of approximate test power, we first state the following relevant technical assumptions, made by Liu et al. [16]; they discuss these assumptions in detail, particularly Assumption **(C)**.

(A) The kernels k_ω are uniformly bounded as follows. For the kernels we use in practice, $\nu = 1$.

$$\sup_{\omega \in \Omega} \sup_{x \in \mathcal{X}} k_\omega(x, x) \leq \nu.$$

(B) The possible kernel parameters ω lie in a Banach space of dimension D. Furthermore, the set of possible kernel parameters Ω is bounded by R_ω, $\Omega \subseteq \{\omega \mid \|\omega\| \leq R_\Omega\}$.

(C) The kernel parameterization is Lipschitz: for all $x, y \in \mathcal{X}$ and $\omega, \omega' \in \Omega$,

$$|k_\omega(x, y) - k_{\omega'}(x, y)| \leq L_k \|\omega - \omega'\|.$$

Proposition 3 (Direct training with approximate test power, Theorem 6 of [16])**.** *Under Assumptions **(A)** to **(C)**, let $\bar{\Omega}_s \subseteq \Omega$ be the set of kernel parameters for which $\sigma_\omega^2 \geq s^2$, and assume $\nu \geq 1$ is constant. Take regularized estimate $\hat{\sigma}_{\omega,\lambda}^2 = \hat{\sigma}_\omega^2 + \lambda$ and $\lambda = m^{-1/3}$. Then, with probability at least $1 - \delta$, $\sup_{\omega \in \bar{\Omega}_s} \left| \hat{J}_\lambda(S_{\mathbb{P}}, S_{\mathbb{Q}}; k_\omega) - J(\mathbb{P}, \mathbb{Q}; k_\omega) \right| \leq \beta$, where the generalization gap β is $\mathcal{O}\left(\frac{1}{s^2 m^{1/3}} \left[\frac{1}{s} + \sqrt{D \log(R_\Omega m) + \log \frac{1}{\delta}} + L_k \right] \right)$.*

Since m is small in our settings, and s may also be small for deep kernel classes as noted by Liu et al. [16], this bound may not give satisfying results.

The key mechanism that drives meta-testing to work, intuitively, is training kernels on *related* tasks. How do we quantify the relatedness between different two-sample problems?

Definition 4 (γ-relatedness)**.** *Let $(\mathbb{P}, \mathbb{Q})$ and $(\mathbb{P}', \mathbb{Q}')$ be the underlying distributions for two different two-sample testing tasks. We say the two tasks are γ-related w.r.t. learning objective J if*

$$\max\left\{\left|J(\mathbb{P}\|\mathbb{Q};\tilde{k}) - J(\mathbb{P}'\|\mathbb{Q}';\tilde{k})\right|, \left|J(\mathbb{P}\|\mathbb{Q};\tilde{k}') - J(\mathbb{P}'\|\mathbb{Q}';\tilde{k}')\right|\right\} = \gamma \tag{9}$$

where $\tilde{k} = \arg\max_k J(\mathbb{P}, \mathbb{Q}; k)$, $\tilde{k}' = \arg\max_k J(\mathbb{P}', \mathbb{Q}'; k)$.

The relatedness measure explicitly quantifies how good the "best" kernel learned from one two-sample testing problem applying to another two-sample testing problem, through the approximate test power objective. It also implies the two problems are of similar difficulty, since for small γ, the ability of our MMD test statistics to distinguish the distributions (with optimal kernels) are similar.

Definition 5 (Adaptation by choosing one best kernel)**.** *With the set of base kernels $\{k_\omega^1, \ldots, k_\omega^N\}$, $\hat{k}_\omega = \arg\max_i \hat{J}_{\lambda^{ne}}(S_{\mathbb{P}}^{tr}, S_{\mathbb{Q}}^{tr}; k^{(i)})$ is said to be the best kernel adaptation.*

Definition 6 (Adaptation with Meta-MKL)**.** *Given a set of kernels $\{k_\omega^1, \ldots, k_\omega^N\}$, the Meta-MKL adaptation is the kernel $\hat{k} = \sum_{i\in[N]} \hat{w}_i k_\omega^i$ where $\hat{w} = \arg\max_w \hat{J}_{\lambda^{ne}}(S_{\mathbb{P}}^{tr}, S_{\mathbb{Q}}^{tr}; \sum_i w_i k_\omega^i)$.*

These adaptation steps use the same learning objective, $\hat{J}_\lambda$, as directly training a deep kernel in Proposition 3. Choosing one kernel from well-trained base kernels is expected to be easier than training the deep kernel parameters, based on the following theorem (proved in the appendix).

Theorem 7 (Adaptation by choosing one best base kernel)**.** *Suppose the kernel parameterization k_ω satisfies Assumptions **(A)** to **(C)**, and let $\bar{\Omega}_s \subseteq \Omega$ be the set of kernel parameters for which $\sigma_\omega^2 \geq s^2$. Assume $\nu \geq 1$ is constant. Denote the relatedness between meta-task $(\mathbb{P}^i, \mathbb{Q}^i)$ and target task $(\mathbb{P}, \mathbb{Q})$ as γ_i, and set $\gamma = \min\{\gamma_1, \ldots, \gamma_N\}$. For meta-training with sample size of order n, take $\lambda = n^{-1/3}$; for adaptation phase, take $\lambda^{ne} = m^{-1/3}$. Denote the chosen kernel as $\hat{k}_\omega$. Let $s'^2 = \min_{i\in[N]} \sigma_{\mathfrak{H}_1}^2(\mathbb{P}, \mathbb{Q}; k_\omega^i)$. Then, we have $s' > s$, and with probability at least $1-\delta$,*

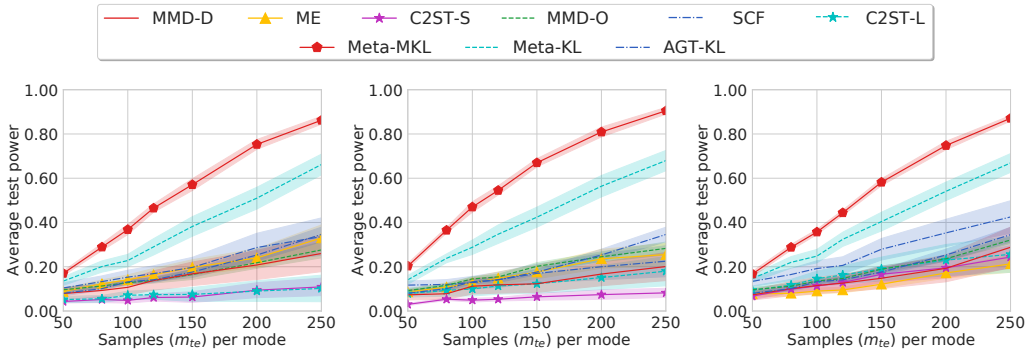
$$\sup_{\omega\in\bar{\Omega}_s} \left|\hat{J}_{\lambda^{ne}}(S_{\mathbb{P}}, S_{\mathbb{Q}}; \hat{k}_\omega) - J(\mathbb{P}, \mathbb{Q}; \hat{k}_\omega)\right| \leq \beta_n + \gamma + \tilde{\beta},$$

with $\beta_n = \mathcal{O}\left(\frac{1}{s^2 n^{1/3}}\left[\frac{1}{s} + \sqrt{D\log(R_\Omega n) + \log\frac{1}{\delta}} + L_k\right]\right)$ the generalization gap for meta-training, and $\tilde{\beta} = \mathcal{O}\left(\frac{1}{s'^2 m^{1/3}}\left[\frac{1}{s'} + \frac{1}{\sqrt{m}} + \sqrt{\log\frac{N}{\delta}}\right]\right)$ the generalization gap for adaptation.

From the theorem, we can see that the β_n term depends on the meta-training sample size $n \gg m$. With enough (relevant) meta-training tasks, γ goes to 0 quickly with the number of meta-training tasks N. So, the overall uniform convergence bound is likely to be dominated by the term $\tilde{\beta}$. Notice that both $\tilde{\beta}$ and the β of Proposition 3 have the same $m^{-1/3}$ rate. This is roughly to be expected; similar optimization objectives are applied for both learning and adaptation, which are limited by sample size m. However, each of the other terms in $\tilde{\beta}$ is much smaller than the corresponding term in β_n: we expect that the variance term $s' \gg s$, that $\log N$ is far smaller than $D\log(R_\omega n)$ since D will be roughly the number of parameters in a neural network, and also that L_k in β_n will be quite large: the bound of Proposition 23 of [16] is exponential in the depth of the network. Combining all of these factors, we expect adaptation by picking the one best related kernel to be far more efficient than direct training. In the appendix, we show a similar result for adaptation with Meta-MKL.

5 Experiments

Following Liu et al. [16], we compare the following baseline tests with our methods on synthetic and real-world datasets: 1) 1) **MMD-D**: MMD with a deep kernel whose parameters are optimized; 2) **MMD-O**: MMD with a Gaussian kernel whose lengthscale is optimized; 3) Mean embedding (**ME**) test [5, 36]; 4) Smooth characteristic functions (**SCF**) test [5, 36], and 5) Classifier two-sample tests, including **C2ST-S** [8] and **C2ST-L** as described in Liu et al. [16]. None of these methods use related tasks at all, so we additionally consider an aggregated kernel learning (**AGT-KL**) method, which



(a) Test power when $m_{tr} = 50$. (b) Test power when $m_{tr} = 100$. (c) Test power when $m_{tr} = 150$.

Figure 3: Test power on synthetic datasets for $\alpha = 0.05$. Average test power when increasing the number of testing samples m_{te} while only using the 50 (a), 100 (b) and 150 (c) training samples per mode (i.e., $m_{tr} = 50, 100, 150$). Shaded regions show standard errors for the mean.

optimizes a deep kernel of the form (8) by maximizing the value of $\hat{J}_\lambda$ averaged over all the related tasks in the meta-training set $\mathbf{S}$.

Again following the the setup of Liu et al. [16], we evaluate tests as follows. For synthetic datasets, we take a single sample set for $S^{tr}_{\mathbb{P}}$ and $S^{tr}_{\mathbb{Q}}$, and learn a kernel/test locations/etc once for each method on that training set. We then evaluate its rejection rate (power or Type-I error, depending on if $\mathbb{P} = \mathbb{Q}$) using 100 new sample sets $S^{te}_{\mathbb{P}}$, $S^{te}_{\mathbb{Q}}$ from the same distribution. For real datasets, we select a subset of the available data for $S^{tr}_{\mathbb{P}}$ and $S^{tr}_{\mathbb{Q}}$ and train on that; we then evaluate on 100 random subsets, disjoint from the training set, of the remaining data. We repeat this full process 10 times, and report the mean rejection rate of each test and the standard error of the mean rejection rate. Implementation details are in Appendix B.

5.1 Results on Synthetic Data

We use a bimodal Gaussian mixtures dataset proposed by [16], known as *high-dimensional Gaussian mixtures* (HDGM): $\mathbb{P}$ and $\mathbb{Q}$ differ in the covariance of a single dimension pair but are otherwise the same. Specifically, $\mathbb{P} = \sum_{i=1}^{2} \frac{1}{2}\mathcal{N}(\mu_i^h, I_d)$, and

$$\mathbb{Q}(\Delta^h) = \sum_{i=1}^{2} \frac{1}{2}\mathcal{N}\left(\mu_i^h, \begin{bmatrix} 1 & (-1)^i\Delta^h & \mathbf{0}_{d-2} \\ (-1)^i\Delta^h & 1 & \mathbf{0}_{d-2} \\ \mathbf{0}_{d-2}^T & \mathbf{0}_{d-2}^T & \mathbf{I}_{d-2} \end{bmatrix}\right), \tag{10}$$

where $\mu_1^h = \mathbf{0}_d$, $\mu_2^h = 0.5 \times \mathbf{1}_d$. In this paper, our target task is $\mathcal{T} = (\mathbb{P}, \mathbb{Q}(0.7))$ and meta-samples are drawn from $\boldsymbol{T} = \{\mathcal{T}^i := (\mathbb{P}, \mathbb{Q}(0.3 + 0.1 \times i/n))\}_{i=1}^N$, where $N = 100$; note that the target task is well outside the scope of training tasks. In experiments, we set d to 2; given the small number of data samples, this is already a difficulty task for existing kernel two-sample tests. To evaluate all tests given limited data, we set the number of training samples ($S^{tr}_{\mathbb{P}}$, $S^{tr}_{\mathbb{Q}}$) to $50, 100, 150$ per mode, and the number of testing samples ($S^{te}_{\mathbb{P}}$, $S^{te}_{\mathbb{Q}}$) from 50 to 250.

Figure 3 illustrates test powers of all tests. Meta-MKL and Meta-KL are the clear winners, with both tests much better when m_{te} is over 100 per mode. It is clear that previous kernel-learning based tests perform poorly due to limited training samples. Comparing Meta-MKL with Meta-KL, apparently, we can obtain much higher power when we consider using multiple trained kernels. Although AGT-KL performs better than baselines, it cannot adapt to the target task very well: it only cares about "in-task" samples, rather than learning to adapt to new distributions.

5.2 Distinguishing CIFAR-10 or -100 from CIFAR-10.1

In this subsection, we distinguish the standard datasets of small images CIFAR-10 and CIFAR-100 [37] from the attempted replication CIFAR-10.1 [38], similar to Liu et al. [16]. Because only a

Table 1: Test power of tests on CIFAR10 vs CIFAR10.1 given very limited training data ($\alpha = 0.05$, $m_{tr} = 100, 200$). The m_{te} represents number of samples when testing. Bold represents the highest mean per column.

Methods	$m_{tr} = 100$			$m_{tr} = 200$		
	$m_{te} = 200$	$m_{te} = 500$	$m_{te} = 900$	$m_{te} = 200$	$m_{te} = 500$	$m_{te} = 900$
ME	0.084±0.009	0.096±0.016	0.160±0.035	0.104±0.013	0.202±0.020	0.326±0.039
SCF	0.047±0.013	0.037±0.011	0.047±0.015	0.026±0.009	0.018±0.006	0.026±0.012
C2ST-S	0.059±0.009	0.062±0.007	0.059±0.007	0.052±0.011	0.054±0.011	0.057±0.008
C2ST-L	0.064±0.009	0.064±0.006	0.063±0.007	0.075±0.014	0.066±0.011	0.067±0.008
MMD-O	0.091±0.011	0.141±0.009	0.279±0.018	0.084±0.007	0.160±0.011	0.319±0.020
MMD-D	0.104±0.007	0.222±0.020	0.418±0.046	0.117±0.013	0.226±0.021	0.444±0.037
AGT-KL	0.170±0.032	0.457±0.052	0.765±0.045	0.152±0.023	0.463±0.060	0.778±0.050
Meta-KL	0.245±0.010	0.671±0.026	0.959±0.013	0.226±0.015	0.668±0.032	0.972±0.006
Meta-MKL	**0.277±0.016**	**0.728±0.020**	**0.973±0.008**	**0.255±0.020**	**0.724±0.026**	**0.993±0.003**

Table 2: Test power of tests on CIFAR100 vs CIFAR10.1 given very limited training data ($\alpha = 0.05$, $m_{tr} = 100, 200$). The m_{te} represents number of samples when testing. Bold represents the highest mean per column.

Methods	$m_{tr} = 100$			$m_{tr} = 200$		
	$m_{te} = 200$	$m_{te} = 500$	$m_{te} = 900$	$m_{te} = 200$	$m_{te} = 500$	$m_{te} = 900$
ME	0.211±0.020	0.459±0.045	0.751±0.054	0.236±0.033	0.512±0.076	0.744±0.090
SCF	0.076±0.027	0.132±0.050	0.240±0.095	0.136±0.036	0.245±0.066	0.416±0.114
C2ST-S	0.064±0.007	0.063±0.010	0.067±0.008	0.324±0.034	0.237±0.030	0.215±0.023
C2ST-L	0.089±0.010	0.077±0.010	0.075±0.010	0.378±0.042	0.273±0.032	0.262±0.023
MMD-O	0.214±0.012	0.624±0.013	0.970±0.005	0.199±0.016	0.614±0.017	0.965±0.006
MMD-D	0.244±0.011	0.644±0.030	0.970±0.010	0.223±0.016	0.627±0.031	0.975±0.006
AGT-KL	0.596±0.044	0.979±0.010	**1.000±0.000**	0.635±0.038	**0.994±0.002**	**1.000±0.000**
Meta-KL	0.771±0.018	0.999±0.001	**1.000±0.000**	0.806±0.017	**1.000±0.000**	**1.000±0.000**
Meta-MKL	**0.820±0.015**	**1.000±0.000**	**1.000±0.000**	**0.838±0.017**	**1.000±0.000**	**1.000±0.000**

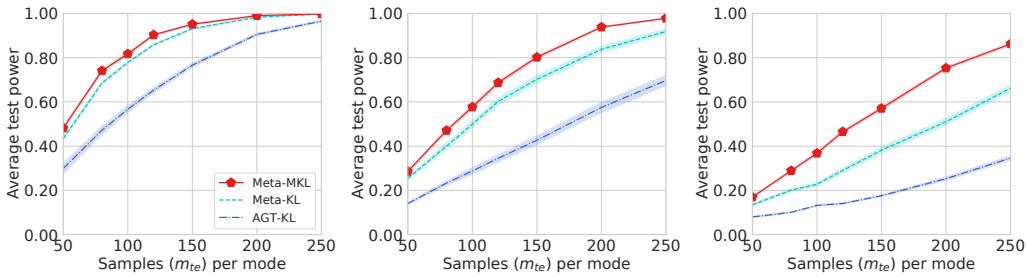
relatively small number of CIFAR-10.1 samples are available, it is of interest to see whether by meta-training only on CIFAR-10's training set (as described in Appendix B), we can find a good test to distinguish CIFAR-10.1, with $m^{tr} \in \{100, 200\}$. Testing samples (i.e., $S^{te}_{\mathbb{P}}$ and $S^{te}_{\mathbb{Q}}$) are from test sets of each dataset. We report test powers of all tests with $200, 500, 900$ testing samples in Table 1 (CIFAR10 compared to CIFAR10.1) and Table 2 (CIFAR100 compared to CIFAR10.1). Since Liu et al. [16] have shown that CIFAR10 and CIFAR10.1 come from different distributions, higher test power is better in both tables. The results demonstrate that our methods have much higher test power than baselines, which is strong evidence that leveraging samples from related tasks can boost test power significantly. Interestingly, C2ST tests almost entirely fail in this setting; it is hard to learn useful information with only a few data points. In Appendix B, we also report results when meta-samples are generated by the training set of CIFAR100 dataset.

5.3 Analysis of closeness between meta training and testing

This subsection studies how closeness between related tasks and the target task affects test powers of our tests. Given the target task $\mathcal{T}$ in synthetic datasets, we define tasks $\boldsymbol{T}$ with closeness C as

$$\boldsymbol{T}(C) = \{\mathcal{T}^i := (\mathbb{P}, \mathbb{Q}((0.6 - C) + 0.1 \times i/n))\}_{i=1}^{n}. \tag{11}$$

It is clear that $\boldsymbol{T}(0)$ will contain our target task $\mathcal{T}$ (i.e., the closeness is zero). In Figure 4, we illustrate the test power of our tests when setting closeness C to 0.1, 0.2 and 0.3, respectively. It can be seen that Meta-MKL and Meta-KL outperforms AGT-KL all the figures, meaning that Meta-MKL and Meta-KL actually learn algorithms that can quickly adapt to new tasks. Another phenomenon is that the gap between test powers of meta based KL and AGT-KL will get smaller if the closeness is smaller, which is expected since AGT-KL has seen closer related tasks.



(a) Closeness is 0.1. (b) Closeness is 0.2. (c) Closeness is 0.3.

Figure 4: Test power when changing closeness between related tasks and the target task ($\alpha = 0.05$). Average test power when closeness is 0.1 (a), 0.2 (b) and 0.3 (c), where only using the 50 training samples per mode (i.e., $m_{tr} = 50$). Shaded regions show standard errors for the mean.

5.4 Ablation Study

In previous sections, we mainly compare with previous kernel-learning tests and have shown that the test power can be improved significantly by our proposed tests. We now show that each component in our tests is effective to improve the test power, and (as previously mentioned) data splitting in Meta-MKL is much better than using the recent test of Kübler et al. [15] which avoids data splitting.

The comparison has been made in synthetic datasets studied in Section 5.1, and results are shown in Table 3. Meta-MKL-A is a test that takes all $w_i = 0$ in Algorithm 3, so that kernels are weighted equally. AGT-MKL uses multiple kernels in AGT-KL (learning weights like Meta-MKL), and AGT-MKL-A does not learn the weights but assign "average" weights $1/K$ directly to all base-kernels. Meta-MKL$_{\text{SI}}$ is a kernel two-sample test using the selective inference technique of Kübler et al. [15] rather than data splitting in its A_θ.

Table 3: The test power of various tests on the synthetic dataset, where for data-splitting methods we use only 10 samples are to select kernels and 200 to test. Meta-MKL-A, AGT-MKL-A and Meta-MKL$_{\text{SI}}$ use 210 test samples.

Meta-MKL	Meta-MKL-A	Meta-KL	AGT-MKL	AGT-MKL-A	AGT-KL	Meta-MKL$_{\text{SI}}$
0.792$\pm$**0.014**	0.780$\pm$0.012	0.509$\pm$0.046	0.364$\pm$0.016	0.358$\pm$0.021	0.253$\pm$0.025	0.058$\pm$0.008

From Table 3, we can see that introducing *multiple kernel learning* (MKL) scheme extensively improves test power, as it combines useful features learned from base kernels covering different aspects of the problems. Moreover, learning with approximate test power with data-splitting in the meta-setting also outperforms the non-splitting testing procedure MetaMKL$_{\text{SI}}$, since MetaMKL$_{\text{SI}}$ requires a poor linear estimator of MMD. The result also indicates that leveraging related tasks is also important to improve the test power even though we need a small set of training samples (e.g., 10).

6 Conclusions

This paper proposes kernel-based non-parametric testing procedures to tackle practical two-sample problems where the sample size is small. By meta-training on related tasks, our work opens a new paradigm of applying learning-to-learn schemes for testing problems, and the potential of very accurate tests in some small-data regimes using our proposed algorithms.

It is worth noting, however, that statistical tests are perhaps particularly ripe for mis-application, e.g. by over-interpreting small marginal differences between sample populations of people to claim "inherent" differences between large groups. Future work will focusing on reliable notions of interpretability in these types of tests is critical. Meta-testing procedures, although they yield much better tests in our domains, may also introduce issues of their own: any rejection of the null hypothesis will be statistically valid, but they favor identifying differences similar to those seen before, and so may worsen gaps in performance between "well-represented" differences and rarer ones.

References

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

[2] Arthur Gretton, Bharath Sriperumbudur, Dino Sejdinovic, Heiko Strathmann, and Massimiliano Pontil. "Optimal kernel choice for large-scale two-sample tests." *NeurIPS*. 2012.

[3] Gábor J. Székely and Maria L. Rizzo. "Energy statistics: A class of statistics based on distances." *Journal of Statistical Planning and Inference* 143.8 (2013), pp. 1249–1272.

[4] Ruth Heller and Yair Heller. "Multivariate tests of association based on univariate tests." *NeurIPS*. 2016.

[5] Wittawat Jitkrittum, Zoltan Szabo, Kacper Chwialkowski, and Arthur Gretton. "Interpretable distribution features with maximum testing power." *NeurIPS*. 2016.

[6] Hao Chen and Jerome H. Friedman. "A new graph-based two-sample test for multivariate and object data." *Journal of the American Statistical Association* 112.517 (2017), pp. 397–409.

[7] Debarghya Ghoshdastidar, Maurilio Gutzeit, Alexandra Carpentier, and Ulrike von Luxburg. "Two-sample tests for large random graphs using network statistics." *COLT*. 2017.

[8] David Lopez-Paz and Maxime Oquab. "Revisiting Classifier Two-Sample Tests." *ICLR*. 2017.

[9] Aaditya Ramdas, Nicolás García Trillos, and Marco Cuturi. "On Wasserstein Two-Sample Testing and Related Families of Nonparametric Tests." *Entropy* 19.2 (Jan. 2017), p. 47.

[10] Danica J. Sutherland, Hsiao-Yu Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alex Smola, and Arthur Gretton. "Generative Models and Model Criticism via Optimized Maximum Mean Discrepancy." *ICLR*. 2017.

[11] Rui Gao, Liyan Xie, Yao Xie, and Huan Xu. "Robust Hypothesis Testing Using Wasserstein Uncertainty Sets." *NeurIPS*. 2018.

[12] Debarghya Ghoshdastidar and Ulrike von Luxburg. "Practical Methods for Graph Two-Sample Testing." *NeurIPS*. 2018.

[13] Shang Li and Xiaodong Wang. "Fully Distributed Sequential Hypothesis Testing: Algorithms and Asymptotic Analyses." *IEEE Trans. Information Theory* 64.4 (2018), pp. 2742–2758.

[14] Matthias Kirchler, Shahryar Khorasani, Marius Kloft, and Christoph Lippert. "Two-sample Testing Using Deep Learning." *AISTATS*. 2020.

[15] Jonas Kübler, Wittawat Jitkrittum, Bernhard Schölkopf, and Krikamol Muandet. "Learning Kernel Tests Without Data Splitting." *NeurIPS*. Vol. 33. 2020, pp. 6245–6255.

[16] Feng Liu, Wenkai Xu, Jie Lu, Guangquan Zhang, Arthur Gretton, and Danica J. Sutherland. "Learning Deep Kernels for Non-Parametric Two-Sample Tests." *ICML*. 2020.

[17] Mingming Gong, Kun Zhang, Tongliang Liu, Dacheng Tao, Clark Glymour, and Behrnhard Schölkopf. "Domain adaptation with conditional transferable components." *ICML*. 2016.

[18] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. "Demystifying MMD GANs." *ICLR*. 2018.

[19] Petar Stojanov, Mingming Gong, Jaime G. Carbonell, and Kun Zhang. "Data-Driven Approach to Multiple-Source Domain Adaptation." *AISTATS*. 2019.

[20] Luca Oneto, Michele Donini, Giulia Luise, Carlo Ciliberto, Andreas Maurer, and Massimiliano Pontil. "Exploiting MMD and Sinkhorn Divergences for Fair and Transferable Representation Learning." *Advances in Neural Information Processing Systems*. 2020.

[21] Ruize Gao, Feng Liu, Jingfeng Zhang, Bo Han, Tongliang Liu, Niu Gang, and Masashi Sugiyama. "Maximum Mean Discrepancy is Aware of Adversarial Attacks." *ICML*. 2021.

[22] Dino Sejdinovic, Bharath Sriperumbudur, Arthur Gretton, Kenji Fukumizu, et al. "Equivalence of distance-based and RKHS-based statistics in hypothesis testing." *The Annals of Statistics* 41.5 (2013), pp. 2263–2291.

[23] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks." *ICML*. 2017.

[24] Jake Snell, Kevin Swersky, and Richard S. Zemel. "Prototypical Networks for Few-shot Learning." *NeurIPS*. 2017.

[25] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. "Learning to propagate labels: Transductive propagation network for few-shot learning." *ICLR*. 2019.

[26] Lu Liu, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. "Attribute Propagation Network for Graph Zero-shot Learning." *AAAI*. 2020.

[27] Lu Liu, Tianyi Zhou, Guodong Long, Jing Jiang, Xuanyi Dong, and Chengqi Zhang. "Isometric Propagation Network for Generalized Zero-shot Learning." *ICLR*. 2021.

[28] Mehmet Gönen and Ethem Alpaydın. "Multiple kernel learning algorithms." *The Journal of Machine Learning Research* 12 (2011), pp. 2211–2268.

[29] Minseop Park, Saehoon Kim, Jungtaek Kim, Yanbin Liu, and Seungjin Choi. "TAEML: Task-Adaptive Ensemble of Meta-Learners." *NeurIPS 2018 Workshop on Meta-learning*. 2018.

[30] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. "Selecting Relevant Features from a Multi-domain Representation for Few-Shot Classification." *ECCV*. Vol. 12355. 2020, pp. 769–786.

[31] Meyer Dwass. "Modified Randomization Tests for Nonparametric Hypotheses." *The Annals of Mathematical Statistics* 28.1 (Mar. 1957), pp. 181–187.

[32] V. Alba Fernández, M.D. Jiménez Gamero, and J. Muñoz García. "A test for the two-sample problem based on empirical characteristic functions." *Computational Statistics & Data Analysis* 52.7 (2008), pp. 3730–3748.

[33] Ilmun Kim, Aaditya Ramdas, Aarti Singh, and Larry Wasserman. "Classification Accuracy as a Proxy for Two Sample Testing." *Annals of Statistics* 49.1 (2021), pp. 411–434.

[34] Wittawat Jitkrittum, Wenkai Xu, Zoltan Szabo, Kenji Fukumizu, and Arthur Gretton. "A linear-time kernel goodness-of-fit test." *NeurIPS*. 2017.

[35] Aaditya Ramdas, Sashank J. Reddi, Barnabas Poczos, Aarti Singh, and Larry Wasserman. "Adaptivity and Computation-Statistics Tradeoffs for Kernel and Distance based High Dimensional Two Sample Testing." 2015. arXiv: 1508.00655.

[36] Kacper Chwialkowski, Aaditya Ramdas, Dino Sejdinovic, and Arthur Gretton. "Fast two-sample testing with analytic representations of probability measures." *NeurIPS*. 2015.

[37] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images." 2009. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[38] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. "Do ImageNet Classifiers Generalize to ImageNet?" *ICML*. 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Conclusion
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In Conclusion
   (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] In Appendix.
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] In the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In the supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] see Figure 3, Figure 4 and all tables.
   (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] In the supplemental material.
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? [Yes]
   (b) Did you mention the license of the assets? [Yes] CIFAR-10 is MIT-licensed
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] In the supplement.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We use only standard datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We use only standard datasets.
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]