MOKD: Cross-domain Finetuning for Few-shot Classification via Maximizing Optimized Kernel Dependence

Hongduan Tian [1 2] Feng Liu [3] Tongliang Liu [4] Bo Du [5] Yiu-ming Cheung [2] Bo Han [1 2]

Abstract

In cross-domain few-shot classification, *nearest centroid classifier* (NCC) aims to learn representations to construct a metric space where few-shot classification can be performed by measuring the similarities between samples and the prototype of each class. An intuition behind NCC is that each sample is pulled closer to the class centroid it belongs to while pushed away from those of other classes. However, in this paper, we find that there exist high similarities between NCC-learned representations of two samples from different classes. In order to address this problem, we propose a bi-level optimization framework, *maximizing optimized kernel dependence* (MOKD) to learn a set of class-specific representations that match the cluster structures indicated by labeled data of the given task. Specifically, MOKD first optimizes the kernel adopted in *Hilbert-Schmidt independence criterion* (HSIC) to obtain the optimized kernel HSIC (opt-HSIC) that can capture the dependence more precisely. Then, an optimization problem regarding the opt-HSIC is addressed to simultaneously maximize the dependence between representations and labels and minimize the dependence among all samples. Extensive experiments on Meta-Dataset demonstrate that MOKD can not only achieve better generalization performance on unseen domains in most cases but also learn better data representation clusters. The project repository of MOKD is available at: https://github.com/tmlr-group/MOKD.

[1]TMLR Group, Hong Kong Baptist University [2]Department of Computer Science, Hong Kong Baptist University [3]TMLR Group, University of Melbourne [4]Sydney AI Centre, The University of Sydney [5]National Engineering Research Center for Multimedia Software, Institute of Artificial Intelligence, School of Computer Science, Wuhan Univeristy. Correspondence to: Bo Han <bhanml@comp.hkbu.edu.hk>.

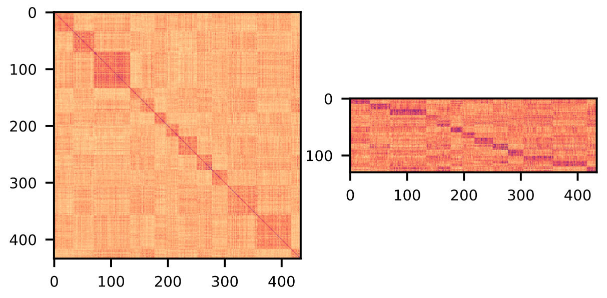
Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Cross-domain few-shot classification (Dvornik et al., 2020; Li et al., 2021a; Liu et al., 2021a; Triantafillou et al., 2020), also known as CFC, is a learning paradigm which aims at learning to perform classification on tasks sampled from previously unseen data or domains with only a few labeled data available. Compared with conventional few-shot classification (Finn et al., 2017; Ravi & Larochelle, 2017; Snell et al., 2017; Vinyals et al., 2016) which learns to adapt to new tasks sampled from unseen data with the same distribution as seen data, cross-domain few-shot classification is a much more challenging learning task since there exist discrepancies between the distributions of source and target domains (Chi et al., 2021; Kuzborskij & Orabona, 2013).

Due to its simplicity and scalability, *nearest centroid classifier* (NCC) (Snell et al., 2017) has been widely applied in recent works (Doersch et al., 2020; Li et al., 2021a; Liu et al., 2021a; Triantafillou et al., 2020) regarding cross-domain few-shot classification. The goal of NCC is to learn representations to construct a metric space where few-shot classification can be performed by measuring the similarities between samples and the prototype of each class. Intuitively, the learning process via NCC is pulling each sample closer to the class centroid it belongs to while pushing it away from other class centroids. Thus, the learned representations are expected to be specific enough to be distinguished from other classes while identified by the class they belong to.

However, in this paper, we find that there exist high similarities between NCC-learned representations of two samples coming from different classes. For example, as shown in Fig. 1(a), the heatmap of the similarity matrix, which depicts the similarities among support data representations, reveals that the NCC-learned representations of each sample not only resemble the samples that belong to the same class but also have high similarities with samples from other classes. Such undesirable high similarities among samples may induce uncertainty and further result in misclassification of samples. Thus, learning a set of *class-specific* representations, where similarities among samples within the same class are maximized while similarities between samples from different classes are minimized, is crucial for CFC.


(a) NCC-based Loss (ImageNet with 13 classes)

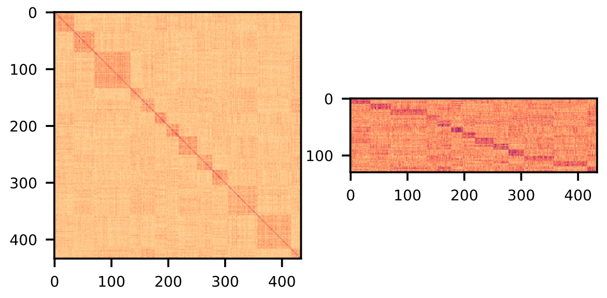

(b) MOKD (ImageNet with 13 classes)

Figure 1. **Heatmaps of similarity matrices of representations respectively learned with NCC-based loss and MOKD.** The left of each figure describes the similarities among all support data representations and the right side describes the similarities between query data and support data representations. As shown in (a), NCC-learned representations of samples are not only similar to samples belonging to their own class but also similar to samples from other classes. (b) shows that the undesirable high similarities existing between samples from different classes are significantly alleviated and the cluster structures of the given set of data are well explored by applying MOKD.

To this end, we propose an efficient and effective approach, *maximizing optimized kernel dependence* (MOKD), to learn a set of class-specific representations, where the similarities among samples belonging to the same class are maximized while the similarities between samples from different classes are minimized, with optimized kernel HSIC measures, where test power is maximized. In general, MOKD is formulated as a bi-level optimization problem for dependence optimization based on optimized kernel measures. Specifically, MOKD first optimizes the kernel used in Hilbert-Schmidt Independence Criterion (HSIC) by maximizing its test power to obtain a powerful kernel dependence measure, the optimized kernel HSIC (opt-HSIC). The goal of this step is to increase the sensitivity of the kernel HSIC to dependence. Then, an optimization objective with respect to the opt-HSIC is optimized to simultaneously maximize the dependence between the kernelized representations and labels and minimize the dependence among all sample representations. In this way, a set of class-specific representations is learned. As shown in Fig 1(b), MOKD does help alleviate the undesirable high similarities between samples from different classes and learn better data clusters.

Extensive experiments on Meta-Dataset (Triantafillou et al., 2020) benchmark under several task settings and further analysis results demonstrate that MOKD is an efficient and effective algorithm. On the one side, numerical results indicate that MOKD can achieve better generalization performance than previous baselines on unseen domains. On the other side, analysis results further reveal that test power maximization is essential for achieving good performance and our proposed MOKD method can mitigate the undesirable high similarities between data representations from different classes and in turn learn better sample clusters.

Our Contribution. In this paper, we find that there exist high similarities between NCC-learned representations of data from different classes, which may further induce uncertainty and in turn result in misclassification. To address this problem, we first provide a new perspective of the NCC-based loss from dependence measure and reveal that NCC-based loss can be expressed in the format of Hilbert-Schmidt Independence Measure (HSIC). Based on this, we further propose a new method, *maximizing optimized kernel dependence* (MOKD), to learn class-specific representations, where the similarities among samples within the same class are maximized while the similarities between samples from different classes are minimized. Extensive experimental results demonstrate that our proposed MOKD not only achieves better generalization performance without increasing running time but also learns better data clusters.

2. Preliminary

Dataset Structure. Let $\mathcal{S}$ denote a meta dataset that contains n sub-datasets with different distributions, i.e. $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{|\mathcal{S}|}\}$. For each sub-dataset $\mathcal{S}_i$, three *disjoint* subsets, which are training set $\mathcal{D}^{\rm tr}_{\mathcal{S}_i}$, validation set $\mathcal{D}^{\rm val}_{\mathcal{S}_i}$ and test set $\mathcal{D}^{\rm test}_{\mathcal{S}_i}$, are included, i.e. $\mathcal{S}_i = \{\mathcal{D}^{\rm tr}_{\mathcal{S}_i}, \mathcal{D}^{\rm val}_{\mathcal{S}_i}, \mathcal{D}^{\rm test}_{\mathcal{S}_i}\}$. In the context of cross-domain few-shot classification, a feature encoder is trained on the training sets of a portion of sub-datasets of $\mathcal{S}$. Thus, the sub-datasets whose training sets are observed by the feature encoder during the pre-training phase are called *seen domains* and denoted as $\mathcal{S}^{\rm seen}$ while the remaining datasets are called *unseen domains* and denoted as $\mathcal{S}^{\rm unseen}$. Note that $\mathcal{S}^{\rm seen}$ and $\mathcal{S}^{\rm unseen}$ are disjoint.

Task Generation. A task $\mathcal{T} = \{\mathcal{D}_\mathcal{T}, \mathcal{Q}_\mathcal{T}\}$, where $\mathcal{D}_\mathcal{T} = \{\boldsymbol{X}^{\rm s}, Y^{\rm s}\} = \{(\boldsymbol{x}_i^{\rm s}, y_i^{\rm s})\}_{i=1}^{|\mathcal{D}_\mathcal{T}|}$ denotes the support data pairs and $\mathcal{Q}_\mathcal{T} = \{\boldsymbol{X}^{\rm q}, Y^{\rm q}\} = \{(\boldsymbol{x}_j^{\rm q}, y_j^{\rm q})\}_{j=1}^{|\mathcal{Q}_\mathcal{T}|}$ denotes query data pairs, is randomly sampled from the specific dataset at the beginning of each episode. The sampling of *vary-way vary-shot* cross-domain few-shot classification tasks follows the rules proposed in Triantafillou et al. (2020). The sampling process can be roughly divided into two steps.

Firstly, the number of classes N is randomly sampled from the interval $[5, N_{\max}]$, where $N_{\max}$ denotes the maximum of the number of classes and is either 50 or as many classes as available. Then, the number of shots of each class, K_n, for $n = \{1, 2, ..., N\}$, in support set is determined with specific rules (refer to Triantafillou et al. (2020) or Appendix D.4 for details). Thus, in the support set of a task, each datapoint belonging to class n can be treated as being independently sampled from the given dataset with the probability $\frac{1}{NK_n}$.

Pre-training Few-shot Classification Pipeline. Consider a pre-trained backbone f_{ϕ^*} parameterized with the optimal pre-trained parameters ϕ^* and a linear transformation head h_θ parameterized with θ. Given a support set $\mathcal{D}_\mathcal{T}$, the corresponding representations can be obtained by applying the pre-trained backbone f_{ϕ^*} and the linear transformation head h_θ: $\mathcal{Z} = \{\boldsymbol{z}_i^{\mathrm{s}}\}_{i=1}^{|\mathcal{D}_\mathcal{T}|} = \{h_\theta \circ f_{\phi^*}(\boldsymbol{x}_i^{\mathrm{s}})\}_{i=1}^{|\mathcal{D}_\mathcal{T}|}$. Then, according to URL (Li et al., 2021a), the learning problem can be solved by minimizing the following NCC-based loss (Snell et al., 2017) on the given support set $\mathcal{D}_\mathcal{T}$:

$$\mathcal{L}_{\mathrm{NCC}}(\theta, \mathcal{D}_\mathcal{T}) = -\frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \log\left(p(y = y_i^{\mathrm{s}}|\boldsymbol{z}_i^{\mathrm{s}}, \theta)\right), \quad (1)$$

where $p(y = c|\boldsymbol{z}, \theta) = \frac{e^{k(\boldsymbol{z}, \boldsymbol{c}_c)}}{\sum_{i=1}^{N_C} e^{k(\boldsymbol{z}, \boldsymbol{c}_i)}}$ denotes the likelihood of a given sample $\boldsymbol{z}$ belonging to class c, $k(\cdot, \cdot)$ denotes a kernel function which is formulated as a cosine similarity function in URL, $\boldsymbol{c}_c$ denotes the prototype of class c and is calculated as $\boldsymbol{c}_c = \frac{1}{|\mathcal{C}_c|}\sum_{\boldsymbol{z}\in\mathcal{C}_c} \boldsymbol{z}, \mathcal{C}_c = \{\boldsymbol{z}_j | y_j = c\}$.

Hilbert-Schmidt Independence Criterion. Given separable *reproducing kernel Hilbert spaces* (RKHSs) $\mathcal{F}$, $\mathcal{G}$ and two feature spaces $\mathcal{X}$, $\mathcal{Y}$, HSIC (Gretton et al., 2005a) measures the dependence between two random variables $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ by evaluating the norm of the cross-variance between the features that are respectively transformed by non-linear kernels: $\varphi : \mathcal{X} \to \mathcal{F}$ and $\psi : \mathcal{Y} \to \mathcal{G}$:

$$\mathrm{HSIC}(X, Y) = ||\mathbb{E}[\varphi(X)\psi(Y)^\top] - \mathbb{E}[\varphi(X)]\mathbb{E}[\psi(Y)]^\top||_{HS}^2, \quad (2)$$

where $||\cdot||_{HS}$ is the Hilbert-Schmidt norm which is the Frobenius norm in finite dimensions. Further, let $(X^{'}, Y^{'})$ and $(X^{''}, Y^{''})$ be the independent copies of (X, Y), the HSIC can be formulated as:

$$\begin{aligned}\mathrm{HSIC}(X, Y) =& \mathbb{E}_{XX^{'}YY^{'}}[k(X, X^{'})l(Y, Y^{'})] \\ &+ \mathbb{E}_{XX^{'}}[k(X, X^{'})]\mathbb{E}_{YY^{'}}[l(Y, Y^{'})] \\ &- 2\mathbb{E}_{XY}[\mathbb{E}_{X^{'}}[k(X, X^{'})]\mathbb{E}_{Y^{'}}[l(Y, Y^{''})]],\end{aligned} \quad (3)$$

where $k(x, x^{'}) = \langle\varphi(x), \varphi(x^{'})\rangle_\mathcal{F}$ and $l(y, y^{'}) = \langle\psi(y), \psi(y^{'})\rangle_\mathcal{G}$ are kernel functions which are defined as inner product operations in reproducing kernel Hilbert space. Note that HSIC takes zero if and only if X and Y are mutually independent (Gretton et al., 2005a; Song et al., 2012).

Test power of HSIC. In this paper, test power is used to measure the probability that, for particular two dependent distributions and the number of samples m, the null hypothesis that the two distributions are independent is correctly rejected. Consider a $\widehat{\mathrm{HSIC}}_{\mathrm{u}}$ as an unbiased HSIC estimator (e.g., U-statistic estimator), under the hypothesis that the two distributions are dependent, the central limit theorem (Serfling, 2009) holds:

$$\sqrt{m}(\widehat{\mathrm{HSIC}}_{\mathrm{u}} - \mathrm{HSIC}) \xrightarrow{d} \mathcal{N}(0, v^2),$$

where v^2 denotes the variance, $\xrightarrow{d}$ denotes convergence in distribution. The CLT implies that test power can be formulated as:

$$\Pr\left(m\widehat{\mathrm{HSIC}}_{\mathrm{u}} > r\right) \to \Phi\left(\frac{\sqrt{m}\mathrm{HSIC}}{v} - \frac{r}{\sqrt{m}v}\right),$$

where r denotes a rejection threshold and Φ denotes the standard normal CDF. Since the rejection threshold r will converge to a constant, and HSIC, v are constants, for reasonably large m, the test power is dominated by the first term. Thus, a feasible way to maximize the test power is to find a kernel function to maximize HSIC/v. The intuition of test power maximization is increasing the sensitivity of the estimated kernel to the dependence among data samples.

3. Motivation: Theoretically Understand NCC via the Kernel HSIC Measure

In this section, we provide an understanding of NCC-based loss from the perspective of HSIC. Specifically, we first reveal two insights behind NCC-based loss. Then, inspired by Li et al. (2021b), we bridge a connection between NCC-based loss and kernel HSIC and find that the upper bound of NCC-based loss can be treated as the surrogate loss under mild assumptions. All proofs are available in Appendix B.

3.1. A Lower Bound of NCC-based Loss

Assumption 3.1 (Li et al. (2021b)). *Given a kernel function $k(\cdot, \cdot)$, for arbitrary two data representations $\boldsymbol{z}$ and $\boldsymbol{z}^{'}$ from support data set $\mathcal{Z} = \{\boldsymbol{z}_i\}_{i=1}^{|\mathcal{D}_\mathcal{T}|}$, we assume that $k(\boldsymbol{z}, \boldsymbol{z}^{'})$ does not deviate much from $\sum_{\boldsymbol{z}^{'}\in\mathcal{Z}} \frac{1}{|\mathcal{D}_\mathcal{T}|} e^{k(\boldsymbol{z}, \boldsymbol{z}^{'})}$.*

Such an assumption is adopted by Li et al. (2021b) to ensure $k(\boldsymbol{z}, \boldsymbol{z}^{'}) \approx \sum_{\boldsymbol{z}^{'}\in\mathcal{Z}} \frac{1}{|\mathcal{D}_\mathcal{T}|} e^{k(\boldsymbol{z}, \boldsymbol{z}^{'})}$ so that Taylor expansion can be conducted on exponent in the InfoNCE loss (Oord et al., 2018). It can also be adopted in our following analysis results without having to add any other assumption.

Theorem 3.2 (**Lower bound of NCC-based loss**). *Given a set of normalized support representations $\mathcal{Z} = \{\boldsymbol{z}_i\}_{i=1}^{|\mathcal{D}_\mathcal{T}|} = \{h_\theta \circ f_{\phi^*}(\boldsymbol{x}_i)\}_{i=1}^{|\mathcal{D}_\mathcal{T}|}$ and the corresponding labels $\{y_i\}_{i=1}^{|\mathcal{D}_\mathcal{T}|}$ that includes N_C classes from a support set $\mathcal{D}_\mathcal{T}$. Let $k(\cdot, \cdot)$*

be the cosine similarity function. Then, with Assumption 3.1, the NCC-based loss (Eq. (1)) owns a lower bound:

$$\mathcal{L}(\theta) \geq -\frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\frac{1}{|\mathcal{C}|}\sum_{\boldsymbol{z}^+\in\mathcal{C}} k(\boldsymbol{z}_i,\boldsymbol{z}^+) + \frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\sum_{\boldsymbol{z}'\in\mathcal{Z}}\frac{k(\boldsymbol{z}_i,\boldsymbol{z}')}{|\mathcal{D}_\mathcal{T}|} + \mathcal{O}\left(k(\boldsymbol{z},\boldsymbol{z}')\right) + const,$$

where $\boldsymbol{z}^+$ denotes the data samples belonging to the same class as $\boldsymbol{z}_i$, $\mathcal{C}$ denotes the class that $\boldsymbol{z}_i$ belongs to, $\mathcal{O}\left(k(\boldsymbol{z},\boldsymbol{z}')\right)$ denotes a high-order moment term. In addition, $const = \log \alpha_e N_C$, where N_C denotes the number of classes in task, α_e is a constant.

For convenience, we define the distance function as the cosine similarity function, though it was defined as Euclidean or Mahalanobias distance functions in previous works (Snell et al., 2017; Requeima et al., 2019). Cosine similarity has been widely used as distance function (Liu et al., 2021a; Li et al., 2021a; 2022) since it approximates the Euclidean distance function when data are normalized and can be treated as a generalization of Mahalanobis distance computation by decomposing the inverse of the covariance matrix into a lower triangular matrix and its conjugate transpose (Li et al., 2021a). More discussions are available in Appendix A.

The lower bound in Theorem 3.2 reveals two key insights adopted in the NCC-based loss. On the one hand, the similarities among samples belonging to the same class are maximized via the first term, which facilitates learning class-specific representations for each class in the given task. On the other hand, the similarities between arbitrary two data samples are minimized through the second term. Ideally, the second term potentially helps reduce the similarities derived from trivial information, such as common backgrounds. This further contributes to focusing on the discriminative feature areas. Compared to Li et al. (2021b), our analysis is conducted on both samples and prototypes instead of samples only. This introduces difficulty in our analysis work. Thus, we propose Lemma B.1 to overcome such a problem and then provide a more precise lower bound.

3.2. HSIC as a Lower Bound of NCC-based Loss

In this section, we follow Li et al. (2021b) to bridge a connection between NCC-based loss and kernel HSIC measure.

Definition 3.3 (Label kernel (Li et al., 2021b))**.** *Given a support set $\mathcal{D}_\mathcal{T}$ that includes N_C classes and K_n shots for each class $n = \{1, 2, ..., N_C\}$, we assume that the label of each data point is a one-hot vector, and each datapoint belonging to class n is sampled from the given support set with the probability $\frac{1}{|\mathcal{D}_\mathcal{T}|}$. Then, any kernel that is a function of $y_i^\top y_j$ or $||y_i - y_j||$ have the form*

$$l(y_i, y_j) = \begin{cases} l_1, & y_i = y_j, \\ l_0, & otherwise \end{cases} \equiv \Delta l \mathbb{I}(y_i = y_j) + l_0, \quad (4)$$

where $\Delta l = l_1 - l_0$.

Theorem 3.4. *Given a support representation set $\mathcal{Z} = \{\boldsymbol{z}_i\}_{i=1}^{|\mathcal{D}_\mathcal{T}|} = \{h_\theta \circ f_{\phi^*}(\boldsymbol{x}_i)\}_{i=1}^{|\mathcal{D}_\mathcal{T}|}$ where N_C classes are included, let $k(\cdot,\cdot)$ be a linear kernel function on data representations and $l(\cdot,\cdot)$ be a label kernel defined in Eq. (4), then $\mathrm{HSIC}(Z, Y)$ owns a lower bound:*

$$\mathrm{HSIC}(Z,Y) \geq \frac{\lambda\Delta l}{|\mathcal{D}_\mathcal{T}|^2}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\sum_{\boldsymbol{z}^+\in\mathcal{C}} k(\boldsymbol{z}_i,\boldsymbol{z}^+) - \frac{\lambda\Delta l}{|\mathcal{D}_\mathcal{T}|^2}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\sum_{\boldsymbol{z}'\in\mathcal{Z}}\frac{1}{|\mathcal{D}_\mathcal{T}|}k(\boldsymbol{z}_i,\boldsymbol{z}'),$$

where $\boldsymbol{z}^+$ denotes the data samples belonging to the same class as $\boldsymbol{z}_i$, $\mathcal{C}$ denotes the class that $\boldsymbol{z}_i$ belongs to, $\boldsymbol{z}'$ is an independent copy of $\boldsymbol{z}$, λ is a scale constant.

Remark 3.5. Under the setting of few-shot classification task with varied ways and shots, the lower bound of $\mathrm{HSIC}(Z, Y)$ obtained in Theorem 3.4 shares a similar structure to the lower bound of NCC-based loss obtained in Theorem 3.2. Such a phenomenon mainly results from both NCC-based loss and $\mathrm{HSIC}(Z, Y)$ leveraging the label information. Specifically, NCC-based loss implicitly takes advantage of label information in the prototypes while $\mathrm{HSIC}(Z, Y)$ explicitly measures the dependence between representations and labels. In addition, an explicit intuition of $\mathrm{HSIC}(Z, Y)$ is that maximizing $\mathrm{HSIC}(Z, Y)$ is equivalent to guiding models to explore class-specific representations that match the cluster structure indicated by labels.

Since the constant scaling does not affect the optimization and it is easy to obtain that the high-order moment term satisfies $\mathcal{O}(k(\boldsymbol{z},\boldsymbol{z}')) \geq \gamma\mathrm{HSIC}(Z,Z)$, where $\gamma = \frac{|\mathcal{D}_\mathcal{T}|}{2N_C C_{\max}}$, $C_{\max}$ is a constant that satisfies $C_{\max} \geq |\mathcal{C}_c|$ for $\forall c \in \{1, 2, ..., N_C\}$ (see Appendix B.4 for more details), we then can build a connection between NCC-based loss and HSIC measure via omitting the scale constants as following:

$$\mathcal{L}(\theta) \geq -\mathrm{HSIC}(Z,Y) + \gamma\mathrm{HSIC}(Z,Z) + const.$$

The new lower bound of NCC-based loss is further expressed as a combination of $\mathrm{HSIC}(Z, Y)$ and $\mathrm{HSIC}(Z, Z)$. As revealed in Theorem 3.4, the lower bound of $\mathrm{HSIC}(Z, Y)$ owns the similar structure to that of NCC-based loss and plays the same role to simultaneously maximizes the similarities among samples belonging to the same class and minimizes the similarities between samples from different classes. Moreover, the $\mathrm{HSIC}(Z, Z)$ term measures the dependence among all data samples in the given task.

In this case, we can control the dependence between sample representations via scaling $\mathrm{HSIC}(Z, Z)$ term. Based on these, a spontaneous insight into the NCC-based loss is that the lower bound can be adopted as a surrogate loss of NCC-based loss to perform few-shot classification tasks.

3.3. Perform CFC Tasks with HSIC

Comparison to NCC. Compared with NCC-based loss, the surrogate loss mentioned above owns the following two desirable merits. First of all, different from NCC-based loss which implicitly and simply leverages label information via class centroids to learn a data cluster for each class, the first term of the surrogate loss explicitly measures the dependence between support data representations and labels that contain cluster structure information of the given set to explore the class-specific representations for each class. Moreover, due to the scarce labeled data for adaptation in few-shot classification tasks, the model usually overfits the data and in turn obtains poor generalization performance. However, such an undesirable phenomenon can be mitigated through scaling $\mathrm{HSIC}(Z, Z)$ term which measures the dependence among all data samples in a given task.

Challenges When Using HSIC. A challenge of applying HSIC to perform cross-domain few-shot classification tasks is that kernel HSIC may fail to accurately measure the dependence between two data samples. For example, as shown in Fig. 1(a), although the similarities among samples within the same class are evidently higher, there still exist high similarities between two samples from different classes. This phenomenon implies that NCC-based loss may fail to accurately measure the dependence between representations and labels, and in turn, fails to learn a set of class-specific representations that match the cluster structure of the support set. Such a phenomenon may potentially induce undesirable uncertainty and further result in the misclassification of samples. To address this problem, a feasible way is improving the capability of the kernel HSIC such that a set of discriminative class-specific representations can be learned.

4. Maximizing Optimized Kernel Dependence

In order to solve the above problem, we propose *maximizing optimized kernel dependence* (MOKD) to perform cross-domain few-shot classification with kernel HSIC measures in which the kernels are optimized to accurately measure the dependence. Specifically, we first maximize the test power of the kernels used in HSIC to improve its capability in dependence detection, and then optimize the dependence respectively between representations and labels, and among representations with the optimized kernel HSIC measures. Intuitively, test power maximization facilitates increasing the sensitivity of kernel HSIC measures to the dependence. In this way, the dependence among samples can be more accurately measured and effectively optimized. In turn, we can learn a set of class-specific and discriminative representations where the similarities among samples belonging to the same class are maximized while the similarities between two samples belonging to different classes are minimized.

4.1. Backbone Pre-training

In this paper, we follow URL (Li et al., 2021a) and adopt ResNet-18 as the backbone. The pre-training strategy is consistent with that in URL. Specifically, we first pre-train 8 domain-specific backbones respectively for the 8 seen domain datasets. Specifically, in the "train on ImageNet only" setting, the backbone pre-trained on the ImageNet dataset is adopted for embedding extraction. Then, a universal backbone is distilled from all domain-specific backbones. Specifically, assume that $\mathcal{D}^{\mathrm{tr}}_{\mathcal{S}^{\mathrm{seen}}_\tau}$, where $\tau \in \{1, 2, ..., |\mathcal{S}^{\mathrm{seen}}|\}$, denotes the training set of the dataset $\mathcal{S}^{\mathrm{seen}}_\tau$, the distillation of the universal backbone can be formulated as:

$$\min_{\phi,\omega} \sum_{\tau=1}^{|\mathcal{S}^{\mathrm{seen}}|} \frac{1}{|\mathcal{D}^{\mathrm{tr}}_{\mathcal{S}^{\mathrm{seen}}_\tau}|} \sum_{i=1}^{|\mathcal{D}^{\mathrm{tr}}_{\mathcal{S}^{\mathrm{seen}}_\tau}|} \ell\left(g_{\omega_\tau} \circ f_\phi(\boldsymbol{x}_i), y_i\right) + \zeta \mathcal{R}(\phi, \omega_\tau),$$

where f_ϕ is the universal backbone parameterized with ϕ, g_{ω_τ} is the classifier for domain $\mathcal{S}^{\mathrm{seen}}_\tau$ parameterized with ω_τ, $\mathcal{R}(\cdot, \cdot)$ is the regularization and ζ is the coefficient. To be specific, in URL (Li et al., 2021a), the regularization is composed of two losses that respectively aim at minimizing the distance between the predictions and maximizing the similarity of representations between the distilled universal backbone and the corresponding domain-specific backbone.

4.2. Problem Formulation for MOKD

Consider a set of support representations pairs $\{(\boldsymbol{z}_i, y_i)\}_{i=1}^m$, where m denotes the size of the set, the ultimate goal of MOKD is to learn a set of optimal task-specific parameters θ^* from the given data by performing optimization on the optimized kernel HSIC measures where the test power is maximized to increase their sensitivity to dependence. Thus, MOKD is formulated as a bi-level optimization problem:

$$\begin{aligned} &\min_\theta -\mathrm{HSIC}(Z, Y; \sigma^*_{ZY}, \theta) + \gamma \mathrm{HSIC}(Z, Z; \sigma^*_{ZZ}, \theta), \\ s.t. &\max_{\sigma_{ZY}} \frac{\mathrm{HSIC}(Z, Y; \sigma_{ZY}, \theta)}{\sqrt{v_{ZY} + \epsilon}}, \max_{\sigma_{ZZ}} \frac{\mathrm{HSIC}(Z, Z; \sigma_{ZZ}, \theta)}{\sqrt{v_{ZZ} + \epsilon}}, \end{aligned} \tag{5}$$

where σ_{ZY} and σ_{ZZ} are the bandwidths of Gaussian kernels respectively calculated in $\mathrm{HSIC}(Z, Y)$ and $\mathrm{HSIC}(Z, Z)$, v_{ZY} and v_{ZZ} are the variances of estimated $\mathrm{HSIC}(Z, Y)$ and $\mathrm{HSIC}(Z, Z)$, γ is the scalar coefficient of $\mathrm{HSIC}(Z, Z)$ term and ϵ is a scalar that is used to avoid the case $v \leq 0$.

Since the true distribution of support data features is unknown, in this paper, we follow Song et al. (2012) and estimate the kernel HSIC, $\widehat{\mathrm{HSIC}}(Z, Y)$, with a set of finite

Table 1. **Results on Meta-Dataset (Trained on ImageNet Only).** Mean accuracy and 95% confidence interval are reported.

Datasets	Finetune	ProtoNets	ProtoNets(large)	BOHB	FP-MAML	ALFA+FP-MAML	FLUTE	SSL-HSIC	URL	**MOKD(Ours)**
ImageNet	45.8±1.1	50.5±1.1	53.7±1.1	51.9±1.1	49.5±1.1	52.8±1.1	46.9±1.1	55.5±1.1	**57.3±1.1**	**57.3±1.1**
Omniglot	60.9±1.6	60.0±1.4	68.5±1.3	67.6±1.2	63.4±1.3	61.9±1.5	61.6±1.4	66.4±1.2	69.4±1.2	**70.9±1.3**
Aircraft	**68.7±1.3**	53.1±1.0	58.0±1.0	54.1±0.9	56.0±1.0	63.4±1.1	48.5±1.0	49.5±0.9	57.6±1.0	**59.8±1.0**
Birds	57.3±1.3	68.8±1.0	**74.1±0.9**	70.7±0.9	68.7±1.0	69.8±1.1	47.9±1.0	71.6±0.9	72.9±0.9	**73.6±0.9**
Textures	69.0±0.9	66.6±0.8	68.8±0.8	68.3±0.8	66.5±0.8	70.8±0.9	63.8±0.8	72.2±0.7	75.2±0.7	**76.1±0.7**
Quick Draw	42.6±1.2	49.0±1.1	53.3±1.0	50.3±1.0	51.5±1.0	59.2±1.2	57.5±1.0	54.2±1.0	57.9±1.0	**61.2±1.0**
Fungi	38.2±1.0	39.7±1.1	40.7±1.2	41.4±1.1	40.0±1.1	41.5±1.2	31.8±1.0	43.4±1.1	46.2±1.0	**47.0±1.1**
VGG Flower	85.5±0.7	85.3±0.8	87.0±0.7	87.3±0.6	87.2±0.7	86.0±0.8	80.1±0.9	85.5±0.7	86.9±0.6	**88.5±0.6**
Traffic Sign	**66.8±1.3**	47.1±1.1	58.1±1.1	51.8±1.0	48.8±1.1	60.8±1.3	46.5±1.1	50.5±1.1	61.2±1.2	**61.6±1.1**
MSCOCO	34.9±1.0	41.0±1.1	41.7±1.1	48.0±1.0	43.7±1.1	48.1±1.1	41.4±1.0	51.4±1.0	53.0±1.0	**55.3±1.0**
MNIST	-	-	-	-	-	-	80.8±0.8	77.0±0.7	86.2±0.7	**88.3±0.7**
CIFAR-10	-	-	-	-	-	-	65.4±0.8	71.0±0.8	69.5±0.8	**72.2±0.8**
CIFAR-100	-	-	-	-	-	-	52.7±1.1	59.0±1.0	62.0±1.0	**63.1±1.0**
Average Seen	45.8	50.5	53.7	51.9	49.5	52.8	46.9	55.5	**57.3**	**57.3**
Average Unseen	-	-	-	-	-	-	56.5	62.5	66.6	**68.1**
Average All	-	-	-	-	-	-	55.8	62.0	65.9	**67.3**
Average Rank	7.1	8.4	4.6	5.5	6.8	4.4	8.9	4.9	2.8	**1.4**

[1] The results on URL and MOKD are the average of 5 reproductions with different random seeds.

Table 2. **Results on Meta-Dataset (Trained on All Datasets).** Mean accuracy and 95% confidence interval are reported.

Datasets	ProtoMAML	CNAPS	S-CNAPS	SUR	URT	Tri-M	FLUTE	2LM	SSL-HSIC	URL	**MOKD**
ImageNet	46.5± 1.1	50.8±1.1	58.4 ±1.1	56.2 ± 1.0	56.8 ± 1.1	**58.6 ± 1.0**	51.8 ± 1.1	58.0 ± 3.6	56.5 ± 1.2	57.3 ± 1.1	57.3 ± 1.1
Omniglot	82.7± 1.0	91.7±0.5	91.6 ± 0.6	94.1 ± 0.4	94.2 ± 0.4	92.0 ± 0.6	93.2 ± 0.5	**95.3 ± 1.0**	92.0 ± 0.9	94.1 ± 0.4	**94.2 ± 0.5**
Aircraft	75.2± 0.8	83.7±0.6	82.0 ± 0.7	85.5 ± 0.5	85.8 ± 0.5	82.8 ± 0.7	87.2 ± 0.5	88.2 ± 0.5	87.3 ± 0.7	88.2 ± 0.5	**88.4 ± 0.5**
Birds	69.9± 1.0	73.6±0.9	74.8 ± 0.9	71.0 ± 1.0	76.2 ± 0.8	75.3 ± 0.8	79.2 ± 0.8	**81.8 ± 0.6**	78.1 ± 1.1	80.2 ± 0.7	**80.4 ± 0.8**
Textures	68.2± 1.0	59.5±0.7	68.8 ± 0.9	71.0 ± 0.8	71.6 ± 0.7	71.2 ± 0.8	68.8 ± 0.8	76.3 ± 2.4	75.2 ± 0.8	76.2 ± 0.7	**76.5 ± 0.7**
Quick Draw	66.8± 0.9	74.7±0.8	76.5 ±0.8	81.8 ± 0.6	**82.4 ± 0.6**	77.3 ± 0.7	79.5 ± 0.7	78.3 ± 0.7	81.4 ± 0.7	82.2 ± 0.6	82.2 ± 0.6
Fungi	42.0±1.2	50.2±1.1	46.6 ± 1.0	64.3 ± 0.9	64.0 ± 1.0	48.5 ± 1.0	58.1 ± 1.1	**69.6 ± 1.5**	63.5 ± 1.2	68.7 ± 1.0	68.6 ± 1.0
VGG Flower	88.7± 0.7	88.9±0.5	90.5 ± 0.5	82.9 ± 0.8	87.9 ± 0.6	90.5 ± 0.5	91.6 ± 0.6	90.3 ± 0.8	90.9 ± 0.8	91.9 ± 0.5	**92.5 ± 0.5**
Traffic Sign	52.4 ± 1.1	56.5 ±1.1	57.2 ± 1.0	51.0 ± 1.1	48.2 ± 1.1	63.0 ± 1.0	58.4 ± 1.1	63.6 ± 1.5	59.7 ± 1.3	63.3 ± 1.2	**64.5 ± 1.1**
MSCOCO	41.7 ± 1.1	39.4 ±1.0	48.9 ± 1.1	52.0 ± 1.1	51.5 ± 1.1	52.8 ± 1.1	50.0 ± 1.0	**57.0 ± 1.1**	51.4 ± 1.1	54.2 ± 1.0	**55.5 ± 1.0**
MNIST	-	-	94.6 ± 0.4	94.3 ± 0.4	90.6 ± 0.5	**96.2 ± 0.3**	**95.6 ± 0.5**	94.7 ± 0.5	93.4 ± 0.6	94.7 ± 0.4	**95.1 ± 0.4**
CIFAR-10	-	-	74.9 ± 0.7	66.5 ± 0.9	67.0 ± 0.8	75.4 ± 0.8	**78.6 ± 0.7**	71.5 ± 0.9	70.0 ± 1.1	71.9 ± 0.8	**72.8 ± 0.8**
CIFAR-100	-	-	61.3 ± 1.1	56.9 ± 1.1	57.3 ± 1.0	62.0 ± 1.0	**67.1 ± 1.0**	60.0 ± 1.1	61.8 ± 1.1	62.9 ± 1.0	**63.9 ±1.0**
Average Seen	67.5	71.6	73.7	75.9	77.4	76.2	76.2	79.7	76.5	79.9	**80.0**
Average Unseen	-	-	67.4	64.1	62.9	69.9	69.9	69.4	68.2	69.4	**70.3**
Average All	-	-	71.2	71.3	71.8	73.8	73.8	75.7	74.6	75.8	**76.3**
Average Rank	-	-	7.2	7.3	6.4	5.2	5.2	3.4	5.5	3.1	**2.2**

[1] Results of URL are the average of 5 reproductions with different random seeds. The reproductions are consistent with the results reported on their website. The results of our method are the average of 5 random reproduction experiments. The ranks considers all 13 datasets and are calculated only with the methods in the table.

data samples in an unbiased way:

$$\frac{1}{m(m-3)}\left[\operatorname{tr}\left(\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\right)+\frac{\mathbf{1}^\top\tilde{\boldsymbol{K}}\mathbf{1}\mathbf{1}^\top\tilde{\boldsymbol{L}}\mathbf{1}}{(m-1)(m-2)}-\frac{2\mathbf{1}^\top\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\mathbf{1}}{m-2}\right], \tag{6}$$

where $\tilde{\boldsymbol{K}}$ and $\tilde{\boldsymbol{L}}$ are kernel matrices where $\tilde{\boldsymbol{K}}_{i,j}=(1-\delta_{i,j})k(\boldsymbol{z}_i,\boldsymbol{z}_j)$ and $\tilde{\boldsymbol{L}}_{i,j}=(1-\delta_{i,j})l(y_i,y_j)$, m denotes the number of samples in support set. In practice, the time complexity of the calculation of $\widehat{\mathrm{HSIC}}(Z,Y)$ is $\mathcal{O}(m^2)$.

The bi-level optimization problem proposed in Eq. (5) mainly contains two aspects: inner optimization for test power maximization and outer optimization for dependence optimization. To be specific, during the inner optimization phase, MOKD performs optimization on the kernel HSIC to maximize its test power via maximizing $\frac{\mathrm{HSIC}(\cdot,\cdot;\sigma,\theta)}{\sqrt{v+\epsilon}}$. In this way, the optimized kernel HSIC measures are more sensitive to dependence and in turn, the dependence among data samples can be more precisely measured. Moreover, during the outer optimization, with the optimized kernel HSIC measures, MOKD maximizes the dependence between representations and labels to explore a set of class-specific representations, where the similarities among samples within the same class are maximized while the similarities between samples belonging to different classes are minimized, to match the cluster structures of the given support set. Meanwhile, the dependence among all representations is minimized as a regularization to penalize the high-variance representations for alleviating the overfitting derived from scarce data.

Differences between SSL-HSIC and MOKD. We notice that the outer optimization objective in Eq. (5) shares the similar format as that in SSL-HSIC (Li et al., 2021b). From our perspective, in fact, there are two major differences between them. Firstly, the most obvious difference between SSL-HSIC and MOKD is that MOKD takes the test power into consideration. This facilitates increasing the sensitivity of the kernel HSIC to the dependence and further contributes to the dependence optimization. In addition, SSL-HSIC is derived from unsupervised contrastive learning and focuses on learning discriminative features by contrasting two different views of a sample. However, MOKD is derived from supervised few-shot classification and aims at learning a set of class-specific features where similarities among samples within the same class are maximized while similarities between samples from different classes are minimized. More

Table 3. **Comparisons of MOKD with different characteristic kernels.**

Datasets	ImageNet	Omniglot	Aircraft	Birds	DTD	QuickDraw	Fungi	VGG_Flower	Traffic Sign	MSCOCO	MNIST	CIFAR10	CIFAR100
Gaussian	57.3±1.1	94.2±0.5	**88.4±0.5**	80.4±0.8	**76.5±0.7**	82.2±0.6	**68.6±1.0**	**92.5±0.5**	**64.5±1.1**	**55.5±1.0**	95.1±0.4	72.8±0.8	**63.9±1.0**
IMQ	57.3±1.1	94.3±0.5	88.0±0.5	**80.5±0.8**	76.2±0.7	82.3±0.6	67.7±1.0	92.1±0.5	63.8±1.1	54.8±1.0	**95.4±0.4**	72.7±0.8	63.7±1.0

details and empirical results are available in Appendix C.

4.3. Bandwidth Selection for Test Power Maximization

In our proposed MOKD method, since dependence optimization significantly depends on the kernels' capability of precise dependence measuring, we first propose to optimize the kernel HSIC measures to improve their capability of detecting dependence between two variables by maximizing their test power. According to the definition of the test power mentioned above, maximizing the test power of a kernel HSIC measure is equivalent to maximizing the $\frac{\mathrm{HSIC}(\cdot,\cdot;\sigma,\theta)}{\sqrt{v+\epsilon}}$ term. In practice, we adopt a Gaussian kernel, which contains a bandwidth σ, as the kernel function in the Problem (5). Thus, test power maximization can be further reformulated as finding an optimal bandwidth σ^* for the Gaussian kernel function to maximize the $\frac{\mathrm{HSIC}(\cdot,\cdot;\sigma,\theta)}{\sqrt{v+\epsilon}}$ term.

A key step of performing test power maximization is estimating the variance of $\mathrm{HSIC}(\cdot,\cdot,\sigma,\theta)$ with the given bandwidth σ. According to Theorem 5 proposed by Song et al. (2012), $\widehat{\mathrm{HSIC}}$, which is estimated in an unbiased way following Eq. (6), converges in distribution to a Gaussian random variable with the mean HSIC and the estimated variance:

$$v = \frac{16}{m}\left(R - \widehat{\mathrm{HSIC}}^2\right), R = (4m)^{-1}(m-1)_3^{-2}\boldsymbol{h}^\top\boldsymbol{h}, \quad (7)$$

where m denotes the number of samples, $(m-1)_3$ denotes the Pochhammer symbols $\frac{(m-1)!}{(m-4)!}$, and $\boldsymbol{h}$ is a basic vector for the calculation of R, and can be calculated as:

$$\begin{aligned}\boldsymbol{h} =&(m-2)^2\left(\tilde{\boldsymbol{K}} \circ \tilde{\boldsymbol{L}}\right)\mathbf{1} - m(\tilde{\boldsymbol{K}}\mathbf{1}) \circ (\tilde{\boldsymbol{L}}\mathbf{1}) \\ &+ (\mathbf{1}^\top\tilde{\boldsymbol{L}}\mathbf{1})\tilde{\boldsymbol{K}}\mathbf{1} + (\mathbf{1}^\top\tilde{\boldsymbol{K}}\mathbf{1})\tilde{\boldsymbol{L}}\mathbf{1} - (\mathbf{1}^\top\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\mathbf{1})\mathbf{1} \\ &+ (m-2)\left((\mathrm{tr}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}})\mathbf{1} - \tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\mathbf{1} - \tilde{\boldsymbol{L}}\tilde{\boldsymbol{K}}\mathbf{1}\right),\end{aligned} \quad (8)$$

where $\circ$ denotes elementwise multiplication on matrices, $\mathbf{1} \in \mathbb{R}^{m\times 1}$ denotes the vector where all elements are 1.

The maximization of test power can be performed by any optimizer, such as gradient-based optimizers. However, in practice, we perform test power maximization via selecting the optimal bandwidth from a list of candidates in the way of grid search (Jitkrittum et al., 2016) since optimizing the bandwidth with optimizers requires extra hyperparameter selection (e.g., learning rate) and gradient steps, which are time-consuming and may negatively affect the efficiency. To some extent, the grid search can be treated as a variant version of the typical bi-level optimization framework (Eq. (5)). Specifically, selecting bandwidth with grid search resembles performing "stop gradient" on the Problem (5). Although the typical bi-level optimization facilitates exploring higher-order information, such as the Hessian matrix, it will consume more computational resources. Meanwhile, as demonstrated by previous works (e.g. MAML (Finn et al., 2017)) first-order information is sufficient to achieve good performance without hurting the efficiency of the algorithm.

Generally, the bandwidth selection of kernel HSIC mainly includes two steps. Firstly, the bandwidth σ is initialized as the median of the non-zero elements of a kernel matrix. Meanwhile, a list of coefficients, which covers as many potential values as possible, is manually set to scale the median. Then, the bandwidth selection is respectively performed for both $\mathrm{HSIC}(Z,Y)$ and $\mathrm{HSIC}(Z,Z)$ by selecting the optimal scale coefficient to generate a scaled bandwidth that is able to maximize $\frac{\mathrm{HSIC}(\cdot,\cdot;\sigma,\theta)}{\sqrt{v+\epsilon}}$. In practice, we directly apply the optimal coefficient of $\mathrm{HSIC}(Z,Y)$ to $\mathrm{HSIC}(Z,Z)$. A complete process of MOKD is summarized in Algorithm 1.

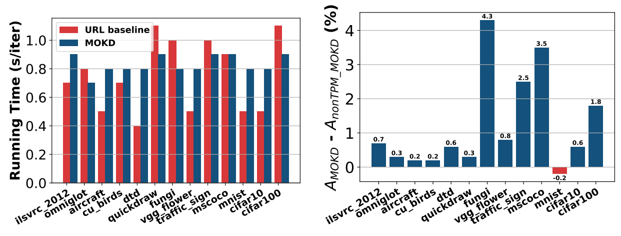
5. Experiments

In this section, we evaluate our proposed MOKD method on the representative mainstream benchmark Meta-Dataset (Triantafillou et al., 2020) under several task settings in order to answer the following questions: (1). Does MOKD achieve better empirical performance on Meta-Dataset under different task settings? (2). What roles do test power maximization and $\mathrm{HSIC}(\boldsymbol{Z},\boldsymbol{Z})$ play in MOKD? (3). Is MOKD efficient? (4). Does MOKD facilitate alleviating the high similarity problem and further learning better clusters?

In this paper, we follow most settings in URL (Li et al., 2021a) to train a simple linear head on top of a pre-trained ResNet-18 backbone by initializing it as an identity matrix for each adaptation episode and optimizing it with Adadelta (Zeiler, 2012). In order to validate the performance of MOKD, we compare MOKD with existing state-of-the-art approaches, including Proto-MAML (Triantafillou et al., 2020), fo-Proto-MAML (Triantafillou et al., 2020), ALFA+fo-Proto-MAML (Baik et al., 2020) CNAPS (Requeima et al., 2019), SimpleCNAPS (S-CNAPS) (Bateni et al., 2020), SUR (Dvornik et al., 2020), URT (Liu et al., 2021a), FLUTE (Triantafillou et al., 2021), Tri-M (Liu et al., 2021b), 2LM (Qin et al., 2023) and URL (Li et al., 2021a). More details are available in Appendix D and E.

5.1. Main Results

In this section, we evaluate MOKD on vary-way vary-shot tasks under both "train on all datasets" and "train on ImageNet only" settings. To be clear, we mark seen domains with green while unseen domains with red. More details



(a) Running time of MOKD (b) Ablation study on TPM

Figure 2. **Results of analysis on MOKD. (a).** Comparison results of running time between MOKD and URL. **(b).** Accuracy gaps between MOKD with and without test power maximization.

about task settings are available in Appendix D.2 and D.4.

Train on ImageNet Only. The results under "train on ImageNet only" settings are reported in Table 1. As shown in the table, MOKD outperforms other baselines on 10 out of 13 datasets and ranks 1.4 on average. Compared with URL, which MOKD is based on, MOKD outperforms on almost all domains with an average improvement of 1.4%. In particular, we find that MOKD performs better on unseen domains (all datasets except ImageNet in this case) compared with the performance on seen domains. MOKD roughly achieves about 1.5% improvements on average on unseen domains. Due to large gaps between seen and unseen domains, performing classification on unseen domains is more challenging. Such a phenomenon further reveals that MOKD can generalize well on previously unseen domains.

Train on All Datasets. The results under the "train on all datasets" settings are reported in Table 2. As shown in the table, MOKD achieves the best performance on average and ranks 2.2 among all methods. Compared with URL, which MOKD is based on, MOKD achieves better generalization performance on 10 out of 13 datasets. In addition, consistent with that under "train on ImageNet only" settings, MOKD performs better on unseen domains as well and achieves nearly 1% improvements on average compared with URL.

However, although MOKD achieves better results under the "train on all datasets" settings, we still notice that there exist some failure cases. For example, MOKD fails to outperform URT on Quick Draw. On the one side, URT contains an attention head that is more powerful in capturing fine-grained features. On the other side, Quick Draw is a dataset composed of doodling created by different individuals, and thus sometimes it is hard to recognize the common features between data within the same class. Besides, we also notice that MOKD also fails on Fungi. According to the curve (Fig. 5(g)), such a failure may be derived from overfitting.

MOKD v.s. SSL-HSIC. Due to the similarity between MOKD and SSL-HSIC, we further evaluate SSL-HSIC on CFC tasks under both task settings to compare the two approaches empirically. To be fair, we estimate HSIC measures in SSL-HSIC in the same way as MOKD and use the same bandwidth. According to the results reported in Table 1 and 2, MOKD outperforms SSL-HSIC under both settings. The complete analysis is available in Appendix C.

We can observe that the main difference between the two objectives is the $\mathrm{HSIC}(Z,Z)$ term. Specifically, in SSL-HSIC loss, the term is $\mathrm{HSIC}(Z,Z)$ is modified to $\sqrt{\mathrm{HSIC}(Z,Z)}$ for achieving better performance in practice. However, according to the original theoretical results of SSL-HSIC, the term should be $\mathrm{HSIC}(Z,Z)$. In this paper, since we propose to maximize the test power through $\mathrm{HSIC}(\cdot,\cdot)$, the modification may result in a mismatch of test power.

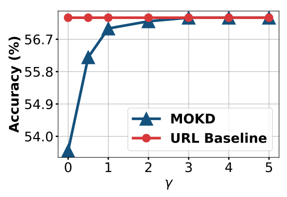
5.2. Experimental Analysis

Effect of Kernel Type. HSIC is a valid statistical measure with characteristic kernels (e.g., Gaussian kernel). In order to study the effect of kernel type, we further run MOKD with inverse multiquadric kernel (IMQ) under "train on all datasets" settings with the same random seeds. Since linear kernels, such as cosine similarity, are not characteristic kernels, we do not consider them in this study. The results are reported in Table 3. According to the table, it is easy to observe that MOKD with Gaussian kernel generally achieves better generalization performance than MOKD with IMQ.

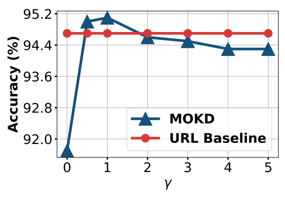
Running Time. To discuss the efficiency of MOKD, we assume that the number of data in the given support set is m, the length of the bandwidth list is k and the number of adaptation steps is s. Then, in each adaptation episode, the time complexity can be roughly expressed as $(4k+2s)\mathcal{O}(m^2)$ according to Algorithm 1. To further quantitatively evaluate the efficiency of MOKD, empirical results are reported in Fig. 2(a) and Table 9. We run the experiment on the same NVIDIA RTX 3090 GPU with the same seeds for fairness.

According to the results, we find that the time that MOKD consumes for each adaptation is acceptable. In some cases, such as datasets like Omniglot and Fungi, MOKD is even more efficient compared with URL. The main reason for such a phenomenon is that MOKD explicitly leverages the label information via label kernel instead of prototypes. In this way, MOKD does not have to calculate the prototypes repeatedly during adaptation. Thus, although MOKD is a bi-level optimization algorithm that includes HSIC and variance estimations, the algorithm is still efficient in total. In addition, we also notice that MOKD fails to consume less time in some cases, such as ImageNet, Aircraft, and CU_Birds. The main reason here is that the estimations of HSIC measures and variances depend on the size of the support data. Since the size of support data varies among tasks during adaptation, the time consumption also changes.

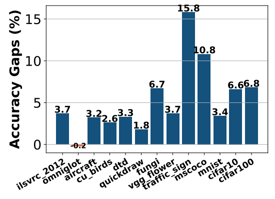
Ablation Study: Test Power. In order to highlight the importance of test power maximization proposed in MOKD, we perform an ablation study on it. Fig. 2(b) shows the



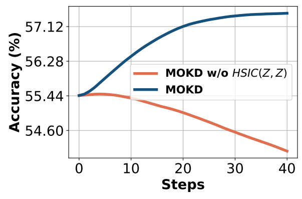
(a) γ on ImageNet



(b) γ on MNIST



(c) Accuracy gaps



(d) Accuracy curves

Figure 3. **Quantitative analysis of γ. (a).** Effect of γ on accuracy of ImageNet dataset; **(b).** Effect of γ on accuracy of MNIST dataset; **(c).** Performance gaps between MOKD w / w.o. $\mathrm{HSIC}(Z, Z)$; **(d).** Test accuracy curves of MOKD w. / w.o. $\mathrm{HSIC}(Z, Z)$ on ImageNet.

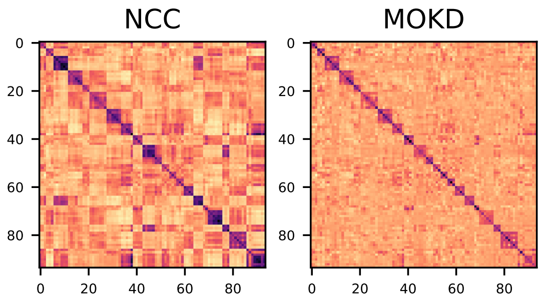


Figure 4. **Heatmap visualization of representation similarity on Omniglot..** The results indicate that MOKD does help learn more discriminative sample clusters than those learned with NCC loss.

performance gaps between MOKD with and without test power maximization (TPM) on Meta-dataset benchmark. As shown in the figure, MOKD with test power maximization performs better. Besides, MOKD with TPM performs better on unseen domains and complicated datasets like Fungi.

Analysis of γ. In order to figure out how datasets in Meta-Dataset react to γ, we propose to run MOKD on all datasets of Meta-Dataset with different γ values under the "train on all datasets" settings. According to the results, we can observe that complicated datasets, such as ImageNet (Fig. 3(a)), prefer large γ while simple datasets, such as MNIST (Fig. 3(b)), prefer small γ. A potential reason for such a phenomenon is the semantic information contained in the images of different datasets. For complicated datasets, due to the huge amounts of semantic information, large γ is required to simultaneously learn discriminative features and penalize high-variance representations. In contrast, since simple datasets, such as MNIST, own evident and definite semantic areas, small γ is enough to focus on the correct semantic areas and achieve better generalization performance.

In addition, an interesting case is that it is equivalent to performing an ablation study on the $\mathrm{HSIC}(Z, Z)$ term when γ is set to zero. According to Fig. 3(c), when $\mathrm{HSIC}(Z, Z)$ is removed, the performance of MOKD drops significantly. According to Fig. 3(d) and 6, it is easy to find that the reason for the performance drop is overfitting. Thus, these results demonstrate that $\mathrm{HSIC}(Z, Z)$ facilitates alleviating overfitting and improving the generalization performance.

Visualization Results. Fig. 4 visualizes the heatmap of the similarity matrices of support data representations respectively learned with NCC-based loss and our proposed MOKD on Omniglot dataset. According to the figure, MOKD learns more definite and clear data clusters compared with those learned with NCC-based loss (URL), which demonstrates that MOKD facilitates capturing the cluster structures of the given support set and learning better class-specific features. Meanwhile, the visualization results on other datasets, such as Fig. 1(b) and Fig. 9(c), further reveal that MOKD is able to alleviate the undesirable high similarities between two samples from different classes.

6. Conclusion

In this paper, we propose an efficient bi-level framework, maximizing optimized kernel dependence, to perform classification on cross-domain few-shot tasks. Specifically, MOKD first maximizes the test power of kernel HSIC to maximize its sensitivity to dependence and then optimizes the optimized kernel HSIC to learn class-specific representations. Extensive experimental results on the Meta-Dataset benchmark demonstrate that MOKD can simultaneously achieve good generalization performance and mitigate the undesirable high similarities for better data clusters.

Impact Statement

In this paper, we provide a new interpretation of NCC-based loss from the perspective of dependence measure and propose to solve cross-domain few-shot classification tasks with the dependence measures where the test power is maximized so that the dependence between samples can be more accurately detected. Two advantages are worth noticing in the MOKD framework. For one thing, such a framework is efficient though it is a bi-level optimization problem. For another thing, by defining an appropriate label kernel, more information, such as cluster structure, can be utilized for model adaptation. Thus, such a framework can be generalized or specified to any other existing pipelines, where cross-entropy loss with softmax function is adopted, to learn more class-specific features where the similarity among samples within the same class is maximized while the similarity between samples from different classes is minimized. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgement

HDT and BH were supported by the NSFC General Program No. 62376235, Guangdong Basic and Applied Basic Research Foundation Nos. 2022A1515011652 and 2024A1515012399, HKBU Faculty Niche Research Areas No. RC-FNRA-IG/22-23/SCI/04, and HKBU CSD Departmental Incentive Grant. FL was supported by the Australian Research Council with grant numbers DP230101540 and DE240101089, and the NSF&CSIRO Responsible AI program with grant number 2303037. BD was supported by the National Natural Science Foundation of China under Grants 62225113, the National Key Research and Development Program of China 2023YFC2705700. YMC was supported in part by NSFC/Research Grants Council (RGC) Joint Research Scheme under Grant: N_HKBU214/21; in part by the General Research Fund of RGC under Grants: 12201321, 12202622, and 12201323; and in part by RGC Senior Research Fellow Scheme under Grant: SRFS2324-2S02. TLL was partially supported by the following Australian Research Council projects: FT220100318, DP220102121, LP220100527, LP220200949, and IC190100031.

References

Bach, F. R. and Jordan, M. I. Kernel independent component analysis. *Journal of Machine Learning Research (JMLR)*, 3(Jul):1–48, 2002.

Baik, S., Choi, M., Choi, J., Kim, H., and Lee, K. M. Meta-learning with adaptive hyperparameters. *NeurIPS*, 2020.

Bateni, P., Goyal, R., Masrani, V., Wood, F., and Sigal, L. Improved few-shot visual classification. In *CVPR*, 2020.

Chi, H., Liu, F., Yang, W., Lan, L., Liu, T., Han, B., Cheung, W., and Kwok, J. Tohan: A one-step approach towards few-shot hypothesis adaptation. *NeurIPS*, 2021.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. Describing textures in the wild. *CVPR*, 2014.

Doersch, C., Gupta, A., and Zisserman, A. Crosstransformers: spatially-aware few-shot transfer. In *NeurIPS*, 2020.

Dvornik, N., Schmid, C., and Mairal, J. Selecting relevant features from a multi-domain representation for few-shot classification. In *ECCV*, 2020.

El Amri, M. R. and Marrel, A. More powerful hsic-based independence tests, extension to space-filling designs and functional data. *International Journal for Uncertainty Quantification*, 14(2), 2024.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Freidling, T., Poignard, B., Climente-González, H., and Yamada, M. Post-selection inference with hsic-lasso. In *ICML*, 2021.

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. Measuring statistical dependence with hilbert-schmidt norms. In *Algorithmic Learning Theory: 16th International Conference, ALT 2005, Singapore, October 8-11, 2005. Proceedings 16*, pp. 63–77. Springer, 2005a.

Gretton, A., Herbrich, R., Smola, A., Bousquet, O., Schölkopf, B., et al. Kernel methods for measuring independence. *Journal of Machine Learning Research (JMLR)*, 2005b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., and Igel, C. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *IJCNN*, 2013.

Jitkrittum, W., Szabó, Z., Chwialkowski, K. P., and Gretton, A. Interpretable distribution features with maximum testing power. *NIPS*, 2016.

Jongejan, J., Henry, R., Takashi, K., Kim, J., and Nick, F.-G. The quick, draw! a.i. experiment. *https://quickdraw.withgoogle.com/*, 2016.

Koyama, K., Kiritoshi, K., Okawachi, T., and Izumitani, T. Effective nonlinear feature selection method based on hsic lasso and with variational inference. In *AISTATS*, 2022.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

Kumagai, A., Iwata, T., Ida, Y., and Fujiwara, Y. Few-shot learning for feature selection with hilbert-schmidt independence criterion. In *NeurIPS*, 2022.

Kuzborskij, I. and Orabona, F. Stability and hypothesis transfer learning. In *ICML*, 2013.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, W.-H., Liu, X., and Bilen, H. Universal representation learning from multiple domains for few-shot classification. In *ICCV*, 2021a.

Li, W.-H., Liu, X., and Bilen, H. Cross-domain few-shot learning with task-specific adapters. In *CVPR*, 2022.

Li, Y., Pogodin, R., Sutherland, D. J., and Gretton, A. Self-supervised learning with kernel dependence maximization. In *NeurIPS*, 2021b.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *ECCV*, 2014.

Liu, L., Hamilton, W., Long, G., Jiang, J., and Larochelle, H. A universal representation transformer layer for few-shot image classification. In *ICLR*, 2021a.

Liu, X., Zhang, J., Hu, T., Cao, H., Yao, Y., and Pan, L. Inducing neural collapse in deep long-tailed learning. 2023.

Liu, Y., Lee, J., Zhu, L., Chen, L., Shi, H., and Yang, Y. A multi-mode modulator for multi-domain few-shot classification. In *CVPR*, 2021b.

Maji, S., Rahtu, E., Kannala, J., Blaschko, M., and Vedaldi, A. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729. IEEE, 2008.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

Qin, X., Song, X., and Jiang, S. Bi-level meta-learning for few-shot domain generalization. In *CVPR*, 2023.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *ICLR*, 2019.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. *ICLR*, 2017.

Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., and Turner, R. E. Fast and flexible multi-task classification using conditional neural adaptive processes. *NeurIPS*, 2019.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

Schroeder, B. and Cui, Y. Fgvcx fungi classification challenge. *github.com/visipedia/fgvcx_fungi_comp*, 2018.

Serfling, R. J. *Approximation theorems of mathematical statistics*. John Wiley & Sons, 2009.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *NIPS*, 2017.

Song, L., Smola, A., Gretton, A., Bedo, J., and Borgwardt, K. Feature selection via dependence maximization. *Journal of Machine Learning Research (JMLR)*, 13:1393–1434, 2012.

Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. Rethinking few-shot image classification: a good embedding is all you need? In *ECCV*, 2020.

Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P.-A., et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *ICLR*, 2020.

Triantafillou, E., Larochelle, H., Zemel, R., and Dumoulin, V. Learning a universal template for few-shot dataset generalization. In *ICML*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *NIPS*, 2017.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. *NIPS*, 2016.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The caltech-ucsd birds-200-2011 dataset. *Technical Report CNS-TR-2011-001*, 2011.

Wang, T. and Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. 2020.

Yamada, M., Jitkrittum, W., Sigal, L., Xing, E. P., and Sugiyama, M. High-dimensional feature selection by feature-wise kernelized lasso. *Neural computation*, 26(1): 185–207, 2014.

Yang, Z., Xu, Q., Bao, S., Cao, X., and Huang, Q. Learning with multiclass auc: Theory and algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (11):7747–7763, 2021.

Zeiler, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Appendix

A. More Related Work

Cross-domain Few-shot Classification. Cross-domain few-shot classification aims to perform few-shot classification on tasks that are sampled from not only previously unseen data sets with the same distribution of seen domains (e.g. test set) but also previously unseen domains with different distributions. Compared with conventional few-shot classification (Vinyals et al., 2016; Ravi & Larochelle, 2017; Finn et al., 2017; Snell et al., 2017; Nichol et al., 2018; Tian et al., 2020), CFC is much more challenging mainly due to two aspects. First of all, the distribution gaps between source and target domains are sometimes quite large. For example, the distribution of ImageNet (i.e. nature scenes) is quite different from that of Omniglot (i.e. handwritings). Thus, since the feature patterns of ImageNet has never been observed by the model, the model may fail to achieve good generalization performance if it was pre-trained on Omniglot dataset (Lake et al., 2015) but evaluated on ImageNet (Russakovsky et al., 2015). On the other side, the task setting of CFC is more difficult (Yang et al., 2021). For example, under the vary-way vary-shot setting, the number of classes and the number of shots for each class are randomly determined at the beginning of each episode.

Generally, although many works have been done in this field, existing methods are built in the way of typical meta-learning frameworks (Finn et al., 2017; Snell et al., 2017) and can be mainly divided into two genres according to the ways of training feature encoders. One of the genres trains the feature encoder and the classifier from scratch. For example, Triantafillou et al. (2020) proposes Proto-MAML which combines Prototypical Nets (Snell et al., 2017) and MAML (Finn et al., 2017) by treating the prototypes obtained from the feature encoder as the parameters of the linear classification layer to perform cross-domain few-shot classification. Based on Proto-MAML, ALFA+fo-Proto-MAML (Baik et al., 2020) further proposes to adaptively generate task-specific hyperparameters, such as learning rate and weight decay, from a small model for each task. Besides, CNAPS (Requeima et al., 2019) proposes to leverage the FiLM (Perez et al., 2018) module to adapt the parameters of both the feature encoder and classifier to new tasks. Further, SimpleCNAPS proposes to substitute the parametric classifier used in CNAPS with a class-covariance-based distance metric to improve the efficiency and generalization performance with fewer parameters. Currently, as far as we know, the state-of-the-art method in this genre is CrossTransformer (Doersch et al., 2020) which proposes to learn spatial correspondence with a self-attention module from tasks sampled from ImageNet and generalize the prior knowledge to other unseen domains.

The other genre in cross-domain few-shot classification aims to transfer the prior knowledge of a pre-trained backbone to tasks sampled from previously unseen domains by further training or fine-tuning a module on top of the frozen backbone. An intuition behind these methods is that a pre-trained backbone is able to extract good representations from the given new tasks sampled from previously unseen domains since it has been well-trained on some large datasets, such as ImageNet (Russakovsky et al., 2015)). Statistically, such an intuition can be further explained by an assumption that the distribution learned by a pre-trained model is similar to the distribution where downstream tasks are sampled. To some extent, this assumption can be demonstrated by Raghu et al. (2019) that the success of meta-learning ought to be attributed to feature reuse. Thus, the semantic feature space embedded in the pre-trained model can partially or fully cover the feature space of target domains and recognize as many feature patterns as possible. Specifically, SUR (Dvornik et al., 2020) proposes to learn a weight vector to select features from the output of several pre-trained backbones and combine these features in a linear way with the corresponding weights. Later, URT (Liu et al., 2021a) proposes to train a universal representation Transformer (Vaswani et al., 2017) to select features from the embeddings extracted from 8 pre-trained domain-specific backbones. Besides, FLUTE (Triantafillou et al., 2021) proposes to treat the convolutional layers of a model as universal templates which will be frozen during the test phase while training specific batch normalization layers for each of the 8 seen domain datasets. Then, a 'Blender' model is trained to learn to combine the statistical information from all specific BN layers. During meta-test, a new specific BN is generated by feeding the support data in the Blender model. However, since the forward passes of several backbones consume too much time during the test phase, Li et al. (2021a) proposes URL to learn a universal multi-domain backbone from the 8 pre-trained backbones via knowledge distillation. During the test phase, URL fine-tunes a simple linear transformation head on top of the distilled backbone for adaptation of the new tasks. As a further version of URL, TSA (Li et al., 2022), which is the state-of-the-art method in this genre, plugs additional learning modules into the backbone so that more task-specific features can be learned.

Distance Function. Distance function is an essential component of approaches that are based on the prototypes. Originally, the distance function adopted is Euclidean distance function (Triantafillou et al., 2020; Requeima et al., 2019). However, Bateni et al. (2020) points out that the Euclidean distance function, which corresponds to the square Mahalanobis distance, implicitly assumes each cluster is distributed according to a unit norm. Based on this, squared Mahalanobias distance is adopted in SimpleCNAPs (Bateni et al., 2020) to consider cluster covariance when computing distances to the cluster centers.

In addition, in most pre-training based pipelines (Liu et al., 2021a; Li et al., 2021a; 2022), the cosine similarity function is adopted. On the one side, when data sampled are normalized to a unit sphere, the cosine similarity is approximately equivalent to the Euclidean distance function and the relationship can be expressed as $(x-y)^\top(x-y) = 2(1-\cos(x,y))$. It also has been demonstrated that directly matching uniformly sampled points on the unit hypersphere contributes to learning good representations (Liu et al., 2023; Wang & Isola, 2020). In addition, as argued by Li et al. (2021a), the cosine similarity function can be treated as a generalization of the Mahalanobis distance computation via decomposing the inverse of the covariance matrix into a product of a lower triangle matrix and its conjugate transpose.

Feature Selection via Dependence Measures Dependence measure has been well studied in statistics, and a series of measures have been proposed in previous works, such as constrained covariance (COCO) (Gretton et al., 2005b), kernel canonical correlation (Bach & Jordan, 2002) and Hilbert-Schmidt Independence Criterion (HSIC) (Gretton et al., 2005a). Recently, these tools, especially HSIC, have been widely applied in deep learning. Specifically, Song et al. (2012) firstly proposed to select features by maximizing the dependence between the selected features and the labels. Kumagai et al. (2022) follows previous works (Yamada et al., 2014; Freidling et al., 2021; Koyama et al., 2022) and propose to select relevant features and remove the redundant features by solving an ℓ_1-regularized regression problem. Besides, Li et al. (2021b) proposes to replace InfoNCE (Oord et al., 2018) with SSL-HSIC to directly optimize statistical dependence.

B. Proof Results

B.1. A Necessary Lemma

Lemma B.1. *Suppose that $a_i \in [0,1]$ for $\forall i \in \{1,2,...,n\}$, then $\frac{1}{n}\sum_{i=1}^n e^{a_i}$ and $e^{\frac{1}{n}\sum_{i=1}^n a_i}$ satisfy that*

$$e^{\frac{1}{n}\sum_{i=1}^n a_i} \geq \frac{1}{n}\sum_{i=1}^n e^{a_i} - (e + (e-1)\log(e-1)).$$

Proof. Since $\exp(\cdot)$ is a convex function, according to Jesen's Inequality, we have $e^{\frac{1}{n}\sum_{i=1}^n a_i} \leq \frac{1}{n}\sum_{i=1}^n e^{a_i}$. In addition, in the interval $[0,1]$, $\exp(x)$ can be approximated to a linear function $(e-1)x+1$. Thus, in this way, we have $\frac{1}{n}\sum_{i=1}^n e^{a_i} = (e-1)(\frac{1}{n}\sum_{i=1}^n a_i) + 1 \geq e^{\frac{1}{n}\sum_{i=1}^n a_i}$.

Then, let $b = \sum_{i=1}^n a_i$, we have a function $f(b) = (e-1)b + 1 - e^x$. By computing $\frac{df}{db} = 0$, we can obtain that $f(b)$ achieves its maximum $e + (e-1)\log(e-1)$ at $b = \log(e-1)$. Thus, we have $e^{\frac{1}{n}\sum_{i=1}^n a_i} \geq \frac{1}{n}\sum_{i=1}^n e^{a_i} - (e + (e-1)\log(e-1))$. □

B.2. Proof of Theorem 3.2

Proof. We first unfold the NCC-based loss as follows:

$$\begin{aligned}
-\frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\log\left(p(y=y_i|\boldsymbol{z}_i,\theta)\right) &= -\frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\log\frac{\exp(k(\boldsymbol{z}_i,\boldsymbol{c}))}{\sum_{j=1}^{N_C}\exp(k(\boldsymbol{z}_i,\boldsymbol{c}_j))} \\
&= -\frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}k(\boldsymbol{z}_i,\boldsymbol{c}) + \frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\log\sum_{j=1}^{N_C}\exp(k(\boldsymbol{z}_i,\boldsymbol{c}_j)) \\
&= -\frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\left(\boldsymbol{z}_i^\top\frac{1}{|\mathcal{C}|}\sum_{\boldsymbol{z}^+\in\mathcal{C}}\boldsymbol{z}^+\right) + \frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\log\sum_{j=1}^{N_C}\exp\left(\boldsymbol{z}_i^\top\frac{1}{|\mathcal{C}_j|}\sum_{\boldsymbol{z}'\in\mathcal{C}_j}\boldsymbol{z}'\right) \\
&\overset{(1)}{\geq} -\frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\frac{1}{|\mathcal{C}|}\sum_{\boldsymbol{z}^+\in\mathcal{C}}\boldsymbol{z}_i^\top\boldsymbol{z}^+ + \frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\log\sum_{j=1}^{N_C}\frac{1}{|\mathcal{C}_j|}\sum_{\boldsymbol{z}'\in\mathcal{C}_j}\exp\left(\boldsymbol{z}_i^\top\boldsymbol{z}'\right) + \log\alpha_e \\
&= -\frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\frac{1}{|\mathcal{C}|}\sum_{\boldsymbol{z}^+\in\mathcal{C}}\boldsymbol{z}_i^\top\boldsymbol{z}^+ + \frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{i=1}^{|\mathcal{D}_\mathcal{T}|}\log\sum_{j=1}^{N_C}\sum_{\boldsymbol{z}'\in\mathcal{C}_j}\frac{1}{N_C|\mathcal{C}_j|}\exp\left(\boldsymbol{z}_i^\top\boldsymbol{z}'\right) + \log\alpha_e N_C,
\end{aligned} \tag{9}$$

where (1) follows Lemma B.1 and $\alpha_e \in \mathbb{R}^+$ is a real constant that satisfies that $\alpha_e \left(\sum_{j=1}^{N_C} \frac{1}{|\mathcal{C}_j|} \sum_{\boldsymbol{z}' \in \mathcal{C}_j} \exp\left(\boldsymbol{z}_i^\top \boldsymbol{z}'\right)\right) \leq \sum_{j=1}^{N_C} \left(\frac{1}{|\mathcal{C}_j|} \sum_{\boldsymbol{z}' \in \mathcal{C}_j} \exp\left(\boldsymbol{z}_i^\top \boldsymbol{z}'\right) - (e + (e-1)\log(e-1))\right)$, $\boldsymbol{c}$ denote the class centroid representations $\boldsymbol{z}_i$ belonging to, $\boldsymbol{c}_j$ denotes the j-th class centroid, $\mathcal{C}_j$ denotes the j-th class sets.

Then, we perform Taylor expansion on the second term of Eq. (9) around $\mu = \sum_{\boldsymbol{z}' \in \mathcal{Z}} \frac{1}{|\mathcal{D}_\mathcal{T}|} k\left(\boldsymbol{z}_i \boldsymbol{z}'\right)$. Based on Assumption 3.1, we can approximately assume that $k(\boldsymbol{z}, \boldsymbol{z}') \approx \mu$. Then, we have:

$$\begin{aligned}
\frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \log \sum_{j=1}^{N_C} \sum_{\boldsymbol{z}' \in \mathcal{C}_j} \frac{1}{N_C |\mathcal{C}_j|} \exp\left(\boldsymbol{z}_i^\top \boldsymbol{z}'\right) &= \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \log \sum_{j=1}^{N_C} \sum_{\boldsymbol{z}' \in \mathcal{C}_j} \frac{e^{\mu} \left[1 + \frac{\left(k(\boldsymbol{z}, \boldsymbol{z}') - \mu\right)^2}{2}\right]}{N_C |\mathcal{C}_j|} \\
&\overset{(2)}{\geq} \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{\boldsymbol{z}' \in \mathcal{Z}} \frac{1}{|\mathcal{D}_\mathcal{T}|} k\left(\boldsymbol{z}_i, \boldsymbol{z}'\right) + \mathcal{O}\left(k(\boldsymbol{z}, \boldsymbol{z}')\right)
\end{aligned}$$

where (2) follows Jesen's Inequality for the convex function $-\log(\cdot)$ the expansion of $\log(1+x)$ around $x = 0$, $\mathcal{O}\left(k(\boldsymbol{z}, \boldsymbol{z}')\right) = \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{j=1}^{N_C} \sum_{\boldsymbol{z}' \in \mathcal{C}_j} \frac{\left[k(\boldsymbol{z}_i, \boldsymbol{z}') - \sum_{\boldsymbol{z}' \in \mathcal{Z}} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}')\right]^2}{2 N_C |\mathcal{C}_j|}$ denotes a high-order moment term regarding $k(\boldsymbol{z}, \boldsymbol{z}')$. In this way, we conclude that Problem 1 owns a lower bound:

$$\mathcal{L}(\theta) \geq -\frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \frac{1}{|\mathcal{C}|} \sum_{\boldsymbol{z}^+ \in \mathcal{C}} k(\boldsymbol{z}_i, \boldsymbol{z}^+) + \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{\boldsymbol{z}' \in \mathcal{Z}} \frac{k(\boldsymbol{z}_i, \boldsymbol{z}')}{|\mathcal{D}_\mathcal{T}|} + \mathcal{O}\left(k(\boldsymbol{z}, \boldsymbol{z}')\right) + \log \alpha_e N_C,$$

where $\mathcal{O}\left(k(\boldsymbol{z}, \boldsymbol{z}')\right) = \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{j=1}^{N_C} \sum_{\boldsymbol{z}' \in \mathcal{C}_j} \frac{\left[k(\boldsymbol{z}_i, \boldsymbol{z}') - \sum_{\boldsymbol{z}' \in \mathcal{Z}} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}')\right]^2}{2 N_C |\mathcal{C}_j|}$ denotes a high-order moment term regarding $k(\boldsymbol{z}, \boldsymbol{z}')$. □

B.3. Proof of Theorem 3.4

Proof. Following (Li et al., 2021b), we compute HSIC by directly calculating the three terms in Eq. (3) under cross-domain few-shot classification setting respectively. Given a set of support representations $\mathcal{Z} = \{\boldsymbol{z}_i\}_{i=1}^{|\mathcal{D}_\mathcal{T}|} = \{h_\theta \circ f_{\phi^*}(\boldsymbol{x}_i)\}_{i=1}^{|\mathcal{D}_\mathcal{T}|}$, each data sample in $\mathcal{Z}$ is randomly sampled with the probability $\frac{1}{|\mathcal{D}_\mathcal{T}|}$, where N_C denotes the number of classes. Then, for the first term, we can obtain

$$\begin{aligned}
\mathbb{E}\left[k(Z, Z') l(Y, Y')\right] &= \mathbb{E}_{Z, Z', Y, Y'}\left[k(Z, Z') l(Y, Y')\right] \\
&= \Delta l \mathbb{E}_{\boldsymbol{Z}, \boldsymbol{Z}', Y, Y'}\left[k(Z, Z') \mathbb{I}(Y, Y')\right] + l_0 \mathbb{E}_{\boldsymbol{Z}, \boldsymbol{Z}'}\left[k(\boldsymbol{Z}, \boldsymbol{Z}')\right] \\
&= \Delta l \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{j=1}^{|\mathcal{D}_\mathcal{T}|} \mathbb{E}_{Z|y_i, Z'|y_j'}\left[\frac{1}{|\mathcal{D}_\mathcal{T}|} \cdot \frac{1}{|\mathcal{D}_\mathcal{T}|} \cdot k(Z, Z') \mathbb{I}(y_i = y_j')\right] + l_0 \mathbb{E}_{Z, Z'}\left[k(Z, Z')\right] \\
&= \Delta l \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \mathbb{E}_{Z|y_i, Z'|y_i}\left[\frac{1}{|\mathcal{D}_\mathcal{T}|} \cdot \frac{|\mathcal{C}_{y_i}|}{|\mathcal{D}_\mathcal{T}|} k(Z, Z')\right] + l_0 \mathbb{E}_{Z, Z'}\left[k(Z, Z')\right] \\
&= \Delta l \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \frac{|\mathcal{C}_{y_i}|}{|\mathcal{D}_\mathcal{T}|^2} \mathbb{E}_{Z|y_i, Z'|y_i}\left[k(Z, Z')\right] + l_0 \mathbb{E}_{Z, Z'}\left[k(Z, Z')\right]
\end{aligned}$$

$$
\begin{aligned}
&= \frac{\Delta l}{|\mathcal{D}_{\mathcal{T}}|} \sum_{i=1}^{|\mathcal{D}_{\mathcal{T}}|} \frac{|\mathcal{C}_{y_i}|}{|\mathcal{D}_{\mathcal{T}}|} \sum_{m=1}^{|\mathcal{C}_{y_i}|} \sum_{n=1}^{|\mathcal{C}_{y_i}|} \frac{1}{|\mathcal{C}_{y_i}|^2} \left[k(\boldsymbol{z}_m, \boldsymbol{z}_n^{'})\right] + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right] \\
&= \frac{\Delta l}{|\mathcal{D}_{\mathcal{T}}|} \sum_{c=1}^{N_C} \frac{|\mathcal{C}_c|^2}{|\mathcal{D}_{\mathcal{T}}|} \sum_{m=1}^{|\mathcal{C}_c|} \sum_{n=1}^{|\mathcal{C}_c|} \frac{1}{|\mathcal{C}_c|^2} k(\boldsymbol{z}_m, \boldsymbol{z}_n^{'}) + l_0 \mathbb{E}_{Z,Z'} \mathbb{E}\left[k(Z, Z^{'})\right] \\
&= \frac{\Delta l}{|\mathcal{D}_{\mathcal{T}}|^2} \sum_{i=1}^{|\mathcal{D}_{\mathcal{T}}|} \sum_{\boldsymbol{z}^+ \in \mathcal{C}} k(\boldsymbol{z}_i, \boldsymbol{z}^+) + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right]
\end{aligned}
$$

where $\mathcal{C}$ denotes the class set that y_i belongs to.

Then, due to the independence between $\boldsymbol{Z}^{'}$ and $Y^{''}$, the second term can be calculated as:

$$
\begin{aligned}
\mathbb{E}\left[k(Z, Z^{'}) l(Y, Y^{''})\right] &= \mathbb{E}_{ZY} \left[\mathbb{E}_{Z'}[k(Z, Z^{'})] \mathbb{E}_{Y''}[l(Y, Y^{''})]\right] \\
&= \sum_{i=1}^{|\mathcal{D}_{\mathcal{T}}|} \mathbb{E}_{Z|y_i} \left[\mathbb{E}_{Z'} k(Z, Z^{'}) \left(\frac{\Delta l |\mathcal{C}_{y_i}|}{|\mathcal{D}_{\mathcal{T}}|^2} + l_0\right)\right] \\
&= \sum_{i=1}^{|\mathcal{D}_{\mathcal{T}}|} \frac{\Delta l |\mathcal{C}_{y_i}|}{|\mathcal{D}_{\mathcal{T}}|^2} \sum_{m=1}^{|\mathcal{C}_{y_i}|} \frac{1}{|\mathcal{C}_{y_i}|} \sum_{n=1}^{|\mathcal{D}_{\mathcal{T}}|} \frac{1}{|\mathcal{D}_{\mathcal{T}}|} \left[k(\boldsymbol{z}_m, \boldsymbol{z}_n^{'})\right] + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right] \\
&= \frac{\Delta l}{|\mathcal{D}_{\mathcal{T}}|} \sum_{c=1}^{N_C} \frac{|\mathcal{C}_c|^2}{|\mathcal{D}_{\mathcal{T}}|} \sum_{m=1}^{|\mathcal{C}_c|} \frac{1}{|\mathcal{C}_c|} \sum_{n=1}^{|\mathcal{D}_{\mathcal{T}}|} \frac{1}{|\mathcal{D}_{\mathcal{T}}|} k(\boldsymbol{z}_m, \boldsymbol{z}_n^{'}) + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right] \\
&= \frac{\Delta l}{|\mathcal{D}_{\mathcal{T}}|^2} \sum_{c=1}^{N_C} \sum_{m=1}^{|\mathcal{C}_c|} \sum_{n=1}^{|\mathcal{D}_{\mathcal{T}}|} \frac{|\mathcal{C}_c|}{|\mathcal{D}_{\mathcal{T}}|} k(\boldsymbol{z}_m, \boldsymbol{z}_n^{'}) + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right] \\
&= \frac{\Delta l}{|\mathcal{D}_{\mathcal{T}}|^2} \sum_{i=1}^{|\mathcal{D}_{\mathcal{T}}|} \sum_{j=1}^{N_C} \sum_{\boldsymbol{z}^{'} \in \mathcal{C}_j} \frac{|\mathcal{C}_j|}{|\mathcal{D}_{\mathcal{T}}|} k(\boldsymbol{z}_i, \boldsymbol{z}^{'}) + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right] \\
&= \frac{\Delta l}{|\mathcal{D}_{\mathcal{T}}|^2} \sum_{i=1}^{|\mathcal{D}_{\mathcal{T}}|} \sum_{\boldsymbol{z}^{'} \in \mathcal{Z}} \frac{1}{|\mathcal{D}_{\mathcal{T}}|} k(\boldsymbol{z}_i, \boldsymbol{z}^{'}) + \frac{\Delta l}{|\mathcal{D}_{\mathcal{T}}|^2} \sum_{i=1}^{|\mathcal{D}_{\mathcal{T}}|} \sum_{j=1}^{N_C} \sum_{\boldsymbol{z}^{'} \in \mathcal{C}_j} \frac{|\mathcal{C}_c| - 1}{|\mathcal{D}_{\mathcal{T}}|} k(\boldsymbol{z}_i, \boldsymbol{z}^{'}) + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right]
\end{aligned}
$$

For the third part, we can obtain:

$$
\begin{aligned}
\mathbb{E}\left[k(Z, Z^{'})\right] \mathbb{E}\left[l(Y, Y^{'})\right] &= \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right] \mathbb{E}_{Y,Y'} \left[l(Y, Y^{'})\right] \\
&= \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right] \left(\sum_{i=1}^{|\mathcal{D}_{\mathcal{T}}|} \sum_{j=1}^{|\mathcal{D}_{\mathcal{T}}|} \frac{1}{|\mathcal{D}_{\mathcal{T}}|} \cdot \frac{1}{|\mathcal{D}_{\mathcal{T}}|} \cdot \Delta l \mathbb{I}(y_i = y_j^{'}) + l_0\right) \\
&= \sum_{i=1}^{|\mathcal{D}_{\mathcal{T}}|} \frac{1}{|\mathcal{D}_{\mathcal{T}}|} \cdot \frac{\Delta l |\mathcal{C}_{y_i}|}{|\mathcal{D}_{\mathcal{T}}|} \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right] + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right] \\
&= \sum_{i=1}^{|\mathcal{D}_{\mathcal{T}}|} \frac{1}{|\mathcal{D}_{\mathcal{T}}|} \cdot \frac{\Delta l |\mathcal{C}_{y_i}|}{|\mathcal{D}_{\mathcal{T}}|} \sum_{m=1}^{|\mathcal{D}_{\mathcal{T}}|} \frac{1}{|\mathcal{D}_{\mathcal{T}}|} \sum_{n=1}^{|\mathcal{D}_{\mathcal{T}}|} \frac{1}{|\mathcal{D}_{\mathcal{T}}|} k(\boldsymbol{z}_m, \boldsymbol{z}_n^{'}) + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right] \\
&= \frac{\Delta l}{|\mathcal{D}_{\mathcal{T}}|^2} \sum_{i=1}^{|\mathcal{D}_{\mathcal{T}}|} \frac{|\mathcal{C}_j|}{|\mathcal{D}_{\mathcal{T}}|^2} \sum_{m=1}^{|\mathcal{D}_{\mathcal{T}}|} \sum_{n=1}^{|\mathcal{D}_{\mathcal{T}}|} k(\boldsymbol{z}_m, \boldsymbol{z}_n^{'}) + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'})\right]
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{\Delta l}{|\mathcal{D}_\mathcal{T}|^2} \frac{\sum_{c=1}^{N_C} |\mathcal{C}_c|}{|\mathcal{D}_\mathcal{T}|^2} \sum_{m=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{n=1}^{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_m, \boldsymbol{z}_n^{'}) + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'}) \right] \\
&= \frac{\Delta l}{|\mathcal{D}_\mathcal{T}|^2} \sum_{m=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{n=1}^{|\mathcal{D}_\mathcal{T}|} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_m, \boldsymbol{z}_n^{'}) + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'}) \right] \\
&= \frac{\Delta l}{|\mathcal{D}_\mathcal{T}|^2} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{\boldsymbol{z}^{'} \in \mathcal{Z}} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}^{'}) + l_0 \mathbb{E}_{Z,Z'} \left[k(Z, Z^{'}) \right],
\end{aligned}
$$

Thus, $\mathrm{HSIC}(Z, Y)$ can be specifically reformulated as:

$$
\begin{aligned}
\mathrm{HSIC}(Z, Y) &= \mathbb{E}[k(Z, Z^{'}) l(Y, Y^{'})] - 2\mathbb{E}[k(Z, Z^{'}) l(Y, Y^{''})] + \mathbb{E}[k(Z, Z^{'})] \mathbb{E}[l(Y, Y^{'})] \\
&= \frac{\Delta l}{|\mathcal{D}_\mathcal{T}|} \left(\frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{\boldsymbol{z}^+ \in \mathcal{C}} k(\boldsymbol{z}_i, \boldsymbol{z}^+) - \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{\boldsymbol{z}^{'} \in \mathcal{Z}} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}^{'}) \right) - \frac{2\Delta l}{|\mathcal{D}_\mathcal{T}|^2} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{j=1}^{N_C} \sum_{\boldsymbol{z}^{'} \in \mathcal{C}_j} \frac{|\mathcal{C}_c| - 1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}^{'}) \\
&\geq \lambda \cdot \frac{\Delta l}{|\mathcal{D}_\mathcal{T}|} \left(\frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{\boldsymbol{z}^+ \in \mathcal{C}} k(\boldsymbol{z}_i, \boldsymbol{z}^+) - \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{\boldsymbol{z}^{'} \in \mathcal{Z}} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}^{'}) \right),
\end{aligned}
$$

where λ is a scale constant. □

B.4. Relation between $\mathrm{HSIC}(Z, Z)$ and High-order Moment Term

We first decompose $\mathrm{HSIC}(Z, Z)$. According to the definition of HSIC and Cauchy-Schwarz Inequality, we can expand $\mathrm{HSIC}(Z, Z)$ under the setting of vary-way vary-shot few-shot classification task as:

$$
\begin{aligned}
\mathrm{HSIC}(Z, Z) &= \mathbb{E}_{Z,Z'} k(Z, Z^{'})^2 - 2\mathbb{E}_Z (\mathbb{E}_{Z'} k(Z, Z^{'}))^2 + (\mathbb{E}_{Z,Z'} k(Z, Z^{'}))^2 \\
&\leq \mathbb{E}_{Z,Z'} k(Z, Z^{'})^2 - \mathbb{E}_Z (\mathbb{E}_{Z'} k(Z, Z^{'}))^2 \\
&= \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{j=1}^{\mathcal{D}_\mathcal{T}} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}_j^{'})^2 - \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \frac{1}{|\mathcal{D}_\mathcal{T}|} \left(\sum_{j=1}^{|\mathcal{D}_\mathcal{T}|} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}_j^{'}) \right)^2 \\
&= \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \frac{1}{|\mathcal{D}_\mathcal{T}|} \left[\sum_{j=1}^{|\mathcal{D}_\mathcal{T}|} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}_j^{'})^2 - \left(\sum_{j=1}^{|\mathcal{D}_\mathcal{T}|} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}_j^{'}) \right)^2 \right].
\end{aligned}
$$

Then, we study the high-order moment term:

$$
\begin{aligned}
\mathcal{O}\left(k(\boldsymbol{z}, \boldsymbol{z}^{'}) \right) &= \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{j=1}^{N_C} \sum_{\boldsymbol{z}^{'} \in \mathcal{C}_j} \frac{\left[k(\boldsymbol{z}_i, \boldsymbol{z}^{'}) - \sum_{\boldsymbol{z}^{'} \in \mathcal{Z}} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}^{'}) \right]^2}{2 N_C |\mathcal{C}_j|} \\
&\geq \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \sum_{\boldsymbol{z}^{'} \in \mathcal{Z}} \frac{\left[k(\boldsymbol{z}_i, \boldsymbol{z}^{'}) - \sum_{\boldsymbol{z}^{'} \in \mathcal{Z}} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}^{'}) \right]^2}{2 N_C C_{\max}} \\
&= \frac{|\mathcal{D}_\mathcal{T}|}{2 N_C C_{\max}} \sum_{i=1}^{|\mathcal{D}_\mathcal{T}|} \frac{1}{|\mathcal{D}_\mathcal{T}|} \left[\sum_{j=1}^{|\mathcal{D}_\mathcal{T}|} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}_j^{'})^2 - \left(\sum_{j=1}^{|\mathcal{D}_\mathcal{T}|} \frac{1}{|\mathcal{D}_\mathcal{T}|} k(\boldsymbol{z}_i, \boldsymbol{z}_j^{'}) \right)^2 \right] \\
&\geq \frac{|\mathcal{D}_\mathcal{T}|}{2 N_C C_{\max}} \mathrm{HSIC}(Z, Z),
\end{aligned}
$$

where $C_{\max} \geq |\mathcal{C}_c|$ for $\forall c \in \{1, 2, ..., N_C\}$.

Algorithm 1 Maximizing Optimized Kernel Dependence Algorithm

```
**Input:** pre-trained backbone $f_{\phi^*}$, number of inner iterations $n$, learning rate $\eta$, linear transformation parameters $h_\theta$, a list of bandwidths $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_T\}$, and $\epsilon = 1e-5$.
**Output:** the optimal parameters for linear transformation head $\theta^*$.
# Sample a task
Sample a new task $\mathcal{T} = \{\{\boldsymbol{X}^{\mathrm{s}}, Y^{\mathrm{s}}\}, \{\boldsymbol{X}^{\mathrm{q}}, Y^{\mathrm{q}}\}\}$;
Obtain the representations: $\mathcal{Z} = \{h_\theta \circ f_{\phi^*}(\boldsymbol{x}_i)\}_{i=1}^{|\boldsymbol{X}^{\mathrm{s}}|}$;
# Inner optimization for test power maximization
Maximize the test power of $\widehat{\mathrm{HSIC}}(Z, Y; \sigma_{ZY}, \theta)$ and $\widehat{\mathrm{HSIC}}(Z, Z; \sigma_{ZZ}, \theta)$ with Eq. (6) and (7):
    $\sigma^*_{ZY} = \max_\Sigma \frac{\widehat{\mathrm{HSIC}}(Z,Y;\sigma_{ZY},\theta)}{\sqrt{v_{ZY}+\epsilon}}$; $\sigma^*_{ZZ} = \max_\Sigma \frac{\widehat{\mathrm{HSIC}}(Z,Z;\sigma_{ZZ},\theta)}{\sqrt{v_{ZZ}+\epsilon}}$
# Outer optimization for dependence optimization
for $i = 1$ to $n$ do
    Obtain the representations: $\mathcal{Z} = \{h_\theta \circ f_{\phi^*}(\boldsymbol{x}_i)\}_{i=1}^{|\boldsymbol{X}^{\mathrm{s}}|}$
    Compute $\widehat{\mathrm{HSIC}}(Z, Y, \sigma^*_{ZY}, \theta)$ and $\widehat{\mathrm{HSIC}}(Z, Z; \sigma^*_{ZZ}, \theta)$ with Eq. (6) for loss:
        $\mathcal{L}(Z, Y; \theta) = -\widehat{\mathrm{HSIC}}(Z, Y, \sigma^*_{ZY}, \theta) + \gamma\widehat{\mathrm{HSIC}}(Z, Z; \sigma^*_{ZZ}, \theta)$
    Update parameters:
        $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}(Z, Y; \theta)$
end for
```

C. Differences between SSL-HSIC and MOKD

In this paper, we propose a bi-level optimization framework MOKD, which is inspired by a new interpretation of NCC-based loss from the perspective of kernel dependence measure. We find that the core insight of NCC-based loss is learning a set of class-specific representations, where the similarities among samples within the same class are maximized while the similarities between samples from different classes are minimized. Our proposed MOKD method achieves the same goal by optimizing the dependence respectively between representations and labels and among all representations based on the optimized HSIC measures where the test power of the kernels used is maximized.

However, we notice that the outer optimization objective in Eq. (5) of our proposed MOKD method shares a similar format as the objective of SSL-HSIC (Li et al., 2021b). From our perspective, such a similar format mainly results from two aspects. On the one side, the outer optimization objectives of NCC-based loss and InfoNCE share the same softmax-like structure. On the other hand, we adopt a similar label kernel as SSL-HSIC by adapting it to few-shot classification settings. Even though, actually, there are two major differences between these two objectives.

Firstly, the most obvious difference between SSL-HSIC and MOKD is that MOKD takes the test power of kernel HSIC measures into consideration. As aforementioned, a challenge of applying HSIC to few-shot classification tasks is that the kernels used may sometimes fail to accurately measure the dependence between the given two data samples. As a result, the transformation model may fail to learn a set of class-specific representations where the similarities among samples belonging to the same class are maximized while the similarities between samples from different classes are minimized. Such a phenomenon may further induce uncertainty and result in misclassification of samples. Thus, by introducing test power maximization in HSIC, kernels' capability of detecting dependence between data samples is improved. This facilitates increasing the sensitivity of kernel HSIC to dependence and further contributes to dependence optimization.

In addition, SSL-HSIC and MOKD are derived from different learning frameworks and are designed for different task settings. To be specific, SSL-HSIC is derived from the InfoNCE loss (Oord et al., 2018) that is designed for unsupervised contrastive learning and focuses on learning robust and discriminative representations of a sample by contrasting two different views of samples. The ultimate goal of SSL-HSIC is to learn a good feature encoder for downstream tasks. However, MOKD is derived from NCC-based loss (a.k.a., Prototypical loss) (Snell et al., 2017) that is designed for supervised few-shot classification and aims at learning a set of class-specific representations where the similarities among samples within the same class are maximized while the similarities between samples from different classes are minimized. The ultimate goal of MOKD is to learn the optimal task-specific parameters (of a linear transformation head) for each task to extract a set of class-specific representations where the data clusters are well learned and the undesirable high similarities are alleviated.

In order to compare SSL-HSIC with MOKD, we further conduct an experiment to reveal the differences between the two

Table 4. **Comparisons of MOKD and SSL-HSIC.**

Datasets	ImageNet	Omniglot	Aircraft	Birds	DTD	QuickDraw	Fungi	VGG_Flower	Traffic Sign	MSCOCO	MNIST	CIFAR10	CIFAR100
MOKD	**57.3±1.1**	**94.2±0.5**	**88.4±0.5**	**80.4±0.8**	**76.5±0.7**	**82.2±0.6**	**68.6±1.0**	**92.5±0.5**	**64.5±1.1**	**55.5±1.0**	**95.1±0.4**	**72.8±0.8**	**63.9±1.0**
SSL-HSIC	56.5±1.2	92.0±0.9	87.3±0.7	78.1±1.1	75.2±0.8	81.4±0.7	63.5±1.2	90.9±0.8	59.7±1.3	51.4±1.1	93.4±0.6	70.0±1.1	61.8±1.1
SSL-HSIC(TPM)	56.9±1.1	92.6±0.9	87.5±0.6	79.8±0.9	75.7±0.7	82.0±0.7	67.1±1.0	91.4±0.6	62.4±1.0	53.6±1.0	94.3±0.5	71.5±0.8	63.5±1.0

learning frameworks. In this experiment, HSIC measures used in SSL-HSIC are estimated in the same unbiased way as MOKD. The results are reported in Table 4. As we can observe, MOKD outperforms SSL-HSIC and SSL-HSIC with Test Power Maximization on all datasets of Meta-Datasets. Moreover, an interesting phenomenon is that SSL-HSIC achieves better performance when applying test power maximization to kernels used in SSL-HSIC. This strongly demonstrates that test power facilitates capturing the dependence between data samples and in turn, learning better representations for each class in the given support set. Compared with MOKD, the main difference between the two objectives is the $\mathrm{HSIC}(Z, Z)$ term. Specifically, in SSL-HSIC loss, the term is $\mathrm{HSIC}(Z, Z)$ is modified to $\sqrt{\mathrm{HSIC}(Z, Z)}$ to achieve better performance in practice. However, in the original theoretical results of SSL-HSIC, the term should be $\mathrm{HSIC}(Z, Z)$. In this paper, since we propose to maximize the test power via $\mathrm{HSIC}(\cdot, \cdot)$, the modification may potentially result in a mismatch of test power.

D. More Settings for CFC

In this section, we provide more details about cross-domain few-shot classification task settings. Specifically, detailed information on Meta-Dataset, experimental settings, data split settings, and vary-way vary-shot settings are introduced.

D.1. Introduction to Meta-Dataset

Meta-Dataset was first proposed by Triantafillou et al. (2020) as a CFC benchmark. The selected datasets are free and easy to obtain and span various visual concepts with different degrees in fine-grain. The original Meta-Dataset is composed of 10 datasets that are ILSVRC_2012 (ImageNet) (Russakovsky et al., 2015), Omniglot (Lake et al., 2015), FGVC_Aircraft (Aircraft) (Maji et al., 2013), CUB_200-2011 (CU_Birds) (Wah et al., 2011), Describable Textures (DTD) (Cimpoi et al., 2014), Quick Draw (Jongejan et al., 2016), FGVCx Fungi (Fungi) (Schroeder & Cui, 2018), VGG_Flower (Flower) (Nilsback & Zisserman, 2008), Traffic Sign (Houben et al., 2013), MSCOCO (Lin et al., 2014). Then, MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky et al., 2009) and CIFAR-100 (Krizhevsky et al., 2009) were added by Requeima et al. (2019).

ILSVRC_2012. ImageNet is a dataset composed of natural images from 1000 categories. In Meta-Dataset, some images that are duplicates of other datasets (e.g. 43 images in CU_Birds) are removed.

Omniglot. Omniglot is a dataset composed of 1623 handwritten characters. The dataset contains 50 classes with 20 examples in each class. The split of Omniglot dataset follows Lake et al. (2015).

Aircraft. Aircraft is a dataset of images of aircraft spanning 102 model variants, and each class contains 100 images. In Meta-Dataset, the images are cropped according to the provided bounding boxes in case of including other aircrafts or the copyright texts.

CU_Birds (CUB-200-2011). CU_Birds is a dataset for fine-grained classification of 200 different bird species. The images in CU_Birds did not use the provided bounding boxes for harder challenges.

Describable Textures. DTD dataset consists of 5640 images of textures, which are organized according to a list of 47 categories inspired from human perception.

Quick Draw. A dataset of 50 million black-and-white drawings across 345 categories.

Fungi. Fungi is a dataset of 1500 wild mushroom species and contains about 100K images.

VGG Flower. VGG Flower is a dataset of natural images of 102 flower categories. Each class contains 40$\sim$258 images.

Traffic Sign. Traffic Sign is a dataset of 50K images of German road signs in 43 classes.

MSCOCO. MSCOCO is a dataset of images collected from Flickr with 1.5 million object instances belonging to 80 categories labeled and localized using bounding boxes. The version adopted in Meta-Dataset is the train2017 split and the images are created images from original images using each object instance's groundtruth bounding box.

MNIST. MNIST is a dataset of handwritten digits. MNIST is composed of 60K training images and 10K test images.

CIFAR-10 & CIFAR-100. CIFAR-10 is a dataset of 60K 32$\times$32 color images in 10 different classes that are airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. CIFAR-100 is a dataset with 100 classes and each class contains 500 training data and 100 test data. Each image in CIFAR-100 owns both a “fine” label and a “coarse” label.

D.2. Task Settings

In this paper, there are two main experimental settings for our main results: “train on all datasets” and “train on ImageNet only”. In “train on all datasets” settings, the backbone we use is the multi-domain backbone which has observed training data of all 8 domains (ImageNet, Omniglot, Aircraft, CU_Birds, DTD, QuickDraw, Fungi and VGG_Flower). In the “train on ImageNet only” settings, the backbone we use is the single-domain backbone which is trained only on the training data of the ImageNet dataset. For both sets of settings, during the meta-test phase, the evaluation is performed on the test data of seen domains and data from unseen domains.

Moreover, for simplicity, the default setting in this paper is “train on all datasets” settings if not any specific clarification.

D.3. Split Settings

The splits of the datasets in this paper are consistent with those in Meta-Dataset. For example, under “train on all datasets” settings, ImageNet, Omniglot, Aircraft, Birds, DTD, QuickDraw, Fungi, and VGG Flower are preserved as ‘seen domains’ where the training set of each dataset are accessible for training the backbone. Each dataset of the seen domain is divided into a training set, a validation set, and a test set roughly with the proportions of 75%, 15%, and 15%. Specifically, for ImageNet, Meta-Dataset constructs a sub-graph of the overall DAG that describes the relationships among all 82115 ‘synsets’ in ILSVRC_2012. Then, the entire graph is cut into three pieces for training, validation, and testing without overlap.

D.4. Vary-way Vary-shot Settings

Vary-way vary-shot task is a popular and basic task setting in cross-domain few-shot classification (Triantafillou et al., 2020; Dvornik et al., 2020; Liu et al., 2021a; Doersch et al., 2020; Li et al., 2021a; 2022). Such task settings stimulate the common daily situations where there exist distribution gaps among tasks and the data in the given task are imbalanced. Compared with conventional few-shot classification task settings where the numbers of ways and shots are fixed and tasks for the test are sampled from unseen data sets with the same distribution, the vary-way vary-shot task is more challenging due to imbalanced data and distributional discrepancies between source and target domains.

In the context of cross-domain few-shot classification, a vary-way vary-shot task is sampled from a single dataset for each learning episode. Generally, the sampling process of a vary-way vary-shot task mainly includes two independent steps: sampling a set of classes and sampling support and query data from the sampled classes. We only provide a brief introduction to the task sampling process, for more details, please refer to the paper of Meta-Dataset (Triantafillou et al., 2020).

Class Sampling Given a dataset, the number of ways (classes) N_C is sampled uniformly from the interval $[5, N_{\max}]$, where $N_{\max}$ denotes the maximum of the number of classes. Usually, $N_{\max}$ is either 50 or as many classes as available.

Support and Query Data Sampling After a set of classes is sampled, the numbers of shots for support and query sets are respectively determined by the following rules.

Compute query set size. In vary-way vary-shot task settings, the number of query data of each class in a task is fixed to the same number. The fixed number should be no more than half of the total number of data in the given class so that there are still roughly 50% of data being used as support data. The process is formulated as:

$$q = \min\left\{10, \left(\min_{c\in\mathcal{C}} \lfloor 0.5 * |c| \rfloor\right)\right\},$$

where $\mathcal{C}$ denotes a set of selected classes, c denotes a single class and $|c|$ denotes the number of images in the given class c. In order to avoid a too large query set, the maximum number of query data of each class is set to 10.

Compute support set size. The computation of support set size is formulated as:

$$s = \min\left\{500, \sum_{c\in\mathcal{C}} \lceil \beta \min\left\{100, |c| - q\right\} \rceil\right\},$$

where β is a coefficient sampled uniformly from $(0, 1]$. In vary-way vary-shot task settings, the total number of data in a support set of a task is no more than 500. For each class in the selected set, the number of shots is determined by its remaining data where query data have been excluded. The maximum number of shots for each class is 100. The coefficient β is used to sample a smaller number of support data and generate a task with an imbalanced number of shots.

Data Sampling for Each Class. After the support set size is determined, the number of shots for each class is calculated. First of all, N_C random scalars $\{\alpha_1, \alpha_2, ..., \alpha_{N_C}\}$ are uniformly sampled from the interval $[\log(0.5), \log(2))$. Then, their 'contributions' to the support set are calculated as:

$$R_c = \frac{\exp(\alpha_c)|c|}{\sum_{c' \in \mathcal{C}} \exp(\alpha_{c'})|c'|}.$$

Then, the number of shots for class c can be calculated by:

$$K_c = \min\left\{\lfloor R_c * (s - |\mathcal{C}|)\rfloor + 1, |c| - q\right\}.$$

The term $R_c * (s - |\mathcal{C}|)\rfloor + 1$ is to guarantee that there is at least one sample being select for the class.

E. More Experimental Settings

E.1. Pre-trained Backbone

In this paper, we directly use both multi-domain and single-domain ResNet-18 (He et al., 2016) backbones provided by URL repository[1] for simplicity and fairness. Two kinds of backbones are respectively applied in our experiments according to different experimental settings. For "train on ImageNet only" settings, the pre-trained backbone applied is a single domain-specific backbone that is trained only on the ImageNet dataset. For "train on all datasets" settings, the pre-trained backbone applied is a multi-domain backbone. The multi-domain backbone is distilled from 8 single domain-specific pre-trained backbones. More details about model distillation are available in (Li et al., 2021a).

For simplicity, except for specific clarification, the experiments are conducted on the multi-domain backbone under the "train on all datasets" settings. In practice, we directly use both multi-domain and single-domain backbones provided in URL repository in order to make fair comparisons.

E.2. More Implementation Details

In this paper, we follow most settings in URL (Li et al., 2021a) to train a simple linear head on top of a pre-trained backbone.

Initialization & Learning rate. For each adaptation episode, we re-initialize the linear transformation layer with an identity matrix and learn a set of task-specific parameters for the given task. The optimizer used in MOKD is Adadelta (Zeiler, 2012). The learning rate is 1.0 for Traffic Sign and MNIST and 0.25 for the remaining datasets. Besides, the weight decay is set to 0.25 for seen domains and 0.0 for unseen domains.

Values of γ. In vary-way vary-shot task settings, we intuitively set γ to 1.0 for Omniglot, Aircraft, CU_Birds, Quick Draw and MNIST while 3.0 for other datasets. Since datasets like Omniglot and Aircraft are simple and the main object of each image is salient, small γ is enough. In contrast, since datasets like ImageNet and Fungi are complex and each image contains too much semantic information, large γ is required to penalize the high-variance representations and alleviate the overfitting.

In addition, in vary-way 5-shot and 5-way 1-shot task settings, we respectively set γ to 1.0 and 0 for all datasets since there are only a few data samples in each task.

Hardware & Seed settings. In this paper, all experiments are performed on an NVIDIA GeForce RTX 3090 GPU. The GPU memory required for running MOKD is about 5 GB. For fairness, all baselines of URL and experiments on MOKD are performed with seeds 41, 42, 43, 44, 45.

[1] https://github.com/VICO-UoE/URL

E.3. Adaptive Bandwidth Selection

Bandwidth is an essential component of a kernel (such as Gaussian kernel) since it closely corresponds to the test power of the kernel as shown in this paper. It is widely believed that kernels with large test power are more sensitive to the dependence among data. As the ultimate goal of this paper is to learn class-specific representations by maximizing dependence among samples belonging to the same class and minimizing the dependence among all samples, a viable way is to select a suitable bandwidth to maximize the test power of the kernel. To this end, we first perform bandwidth selection before optimizing the objective loss to maximize its test power so that the optimized kernel is much more sensitive to the dependence. To be concrete, the test power is maximized by selecting an optimal bandwidth to maximize $\frac{\mathrm{HSIC}(\cdot,\cdot;\sigma,\theta)}{\sqrt{v+\epsilon}}$, where, σ denotes the bandwidth, v denotes the variance of HSIC, and ϵ is a constant that aims to avoid $v \leq 0$.

First of all, bandwidth is initialized as the median of the Gram matrix obtained with the data. Then, we manually set a list of coefficients to scale the median as the new bandwidth. To be concrete, the scale coefficient is selected from the list [0.001, 0.01, 0.1, 0.2, 0.25, 0.5, 0.75, 0.8, 0.9, 1.0, 1.25, 1.5, 2.0, 5.0, 10.0]. Finally, by iteratively calculating $\frac{\mathrm{HSIC}(\cdot,\cdot;\sigma,\theta)}{\sqrt{v+\epsilon}}$ where σ is a scaled median, we select the σ which obtains the largest test value of $\frac{\mathrm{HSIC}(\cdot,\cdot;\sigma,\theta)}{\sqrt{v+\epsilon}}$ as the optimal bandwidth.

The reason that we chose the grid search method for the optimal bandwidth is the efficiency of MOKD. Optimizing bandwidth with auto optimizer requires extra hyperparameter selection and gradient descent steps, and this extra work will make the algorithm complicated and time-consuming.

F. Detailed Experimental Results

F.1. Results Under Vary-way Vary-shot Settings

In this section, we evaluate MOKD on vary-way vary-shot tasks under both “train on all datasets” and “train on ImageNet only” settings. To be clear, we mark seen domains with green while unseen domains with red.

F.1.1. Results under Train on ImageNet Only Settings

The empirical results under “train on ImageNet only” settings are reported in Table 1 with mean accuracy and 95% confidence. Here, we provide a more detailed analysis of the results.

Generally, MOKD achieves the best performance among all approaches on 10 out of 13 datasets, including ImageNet, Omniglot, Textures (DTD), Quick Draw, Fungi, VGG_Flower, MSCOCO, MNIST, CIFAR10, and CIFAR100, and ranks 1.3 on average of all datasets. Compared with URL, where MOKD is based, MOKD outperforms URL on almost all datasets. Specifically, compared with URL, MOKD obtains 1.5%, 0.2%, 0.7%, 0.9%, 3.3%, 0.8%, 1.6%, 0.4%, 2.3%, 2.1%, 1.2% and 1.1% improvements respectively from Omniglot to CIFAR-100. For ImageNet, which is the seen domain, MOKD gets the same results as URL and outperforms other previous works.

An interesting phenomenon is that MOKD performs better than URL on unseen domains compared with the results on seen domains. As we can see from the table, MOKD achieves 1.5% improvements on average on unseen domains. Such a phenomenon indicates that MOKD has better generalization ability than previous works. Due to there exist distribution gaps between seen and unseen domains, it is challenging for a model to perform well on the domains that it has never observed before. We guess the reason for such a phenomenon is that MOKD directly optimizes the dependence respectively between representations and labels and representation themselves with the optimized kernel HSIC where the test power is maximized to be more sensitive to dependence. Thus, it is able to learn a set of better representations where similarities among samples within the same class are maximized while similarities between samples from different classes are minimized via capturing the accurate dependence between representations and labels.

F.1.2. Results Under Train on All datasets Settings

The results under “train on all datasets” settings are reported in Table 2 with mean accuracy and 95% confidence. Here, we intend to provide a more detailed analysis of empirical results.

According to the table, it is easy to observe that MOKD achieves the best performance on average and ranks 1.8 among all baselines. Compared with URL where our proposed MOKD is based, MOKD outperforms URL on 10 out of 13 datasets. Specifically, MOKD achieves 0.1%, 0.2%, 0.2%, 0.3%, 0.6%, 1.2%, 1.1%, 0.4%, 0.9% and 1.0% improvements respectively on Omniglot, Aircraft, CU_Birds, Textures (DTD), VGG_Flower, Traffic Sign, MSCOCO, MNIST, CIFAR10 and CIFAR100 datasets. Besides, compared with 2LM which is a recent new state-of-the-art method in the cross-domain few-shot classification community, MOKD still achieves better performance on 8 out of 13 datasets.

Consistent with the results under "train on ImageNet only" settings, MOKD also obtained better performance on unseen domains (Traffic Sign, MSCOCO, MNIST, CIFAR10, and CIFAR100) under "train on all datasets" settings. Specifically, under "train on all datasets" settings, MOKD achieves 1.2%, 1.3%, 0.4%, 0.9%, and 1.0% improvements on Traffic Sign, MSCOCO, MNIST, CIFAR10 and CIFAR100 datasets. Such a phenomenon consistently demonstrates that MOKD is able to obtain better generalization performance on previously unseen domains with only a few learning adaptation steps.

Although MOKD achieves impressive performance on the Meta-Dataset benchmark, we also notice that slight overfitting happens on the Fungi dataset (see Fig. 5(g)).

F.1.3. Discussion about Why MOKD Generalizes Well on Unseen Domains

According to the results under both "train on all datasets" and "train on ImageNet only" settings, we observe that MOKD achieves better generalization performance on unseen domains than on seen domains. From our perspective, the reasons for this phenomenon are collectively determined by both pre-trained backbones and the optimization objective of MOKD. Specifically, on the one side, the optimization objective of MOKD proposed in Eq. (5) is more powerful in exploring class-specific representations; on the other side, the pre-trained backbones limit feature exploration to some extent.

From the perspective of the optimization objective of MOKD, as we have mentioned in Theorem 3.4 of our paper, maximizing $\mathrm{HSIC}(Z, Y)$ is equivalent to exploring a set of representations that matches the cluster structure of the given task. Meanwhile, since test power maximization is further taken into consideration, MOKD has a more powerful ability to explore such representations compared with NCC-based loss. Our visualization results in Fig. 1, 8, and 9 have demonstrated this.

However, the performance is not only simply determined by the optimization objective in Eq. (5), but also decided by the pre-trained backbones. In our paper, the backbone used under "train on all datasets" settings is pre-trained on 8 datasets, including ILSVRC_2012, Omniglot, Aircraft, CU_Birds, DTD, Quick Draw, Fungi and VGG Flowers. Since the distribution is shared between the training and test sets of a single dataset, it is easy for the pre-trained backbone to extract good features, where the cluster structures are definite, from test data of seen domains. However, such an advantage may somewhat constrain the function space that can be explored by the linear transformation head. An intuitive explanation for this conjecture is that a loss will converge to the local optimal if the initial point is close to that local optimal. In contrast, since data from unseen domains have never been observed by the pre-trained model, the features extracted from the pre-trained backbone are not so good. Thus, it is probable that MOKD can find better results from a relatively bad initialization.

As a simple demonstration, we compare the performance gaps between the initial and final adaptation steps. The results are obtained with random seed 42 from both URL and MOKD methods. Since both URL and MOKD initialize the linear head with an identity matrix, their initial accuracies before performing adaptation are the same (as shown in Fig. 5 in our paper).

Intuitively, a small performance gap means the extracted features from the pre-trained backbone are good enough for direct classification. Otherwise, the extracted features are not so good. According to the table, we notice that the gaps on seen domains are generally smaller than unseen domains for both URL and MOKD, which demonstrates that the extracted features from unseen domain data are not so good. Based on this observation, as MOKD is more powerful in exploring a set of representations that matches the cluster structure of the given task, better improvements are obtained on unseen domains.

F.2. Effect of More Trainable Modules

As demonstrated in previous work (Li et al., 2022), plugging extra trainable modules into the frozen pre-trained backbone contributes to achieving better generalization performance on Meta-Dataset. As MOKD can be seen as a variant case of URL, we also evaluate our proposed MOKD with the TSA strategy.

Specifically, since fine-tuning 4 extra trainable modules consumes more running time (about 30s per iteration), we plug the extra trainable modules only into the second and third resnet block. In addition, in this experiment, we set the learning rate for the trainable module in the backbone to 0.5 for Traffic Sign, MNIST, and CIFAR-100 and 0.05 for the remaining datasets. We also set the learning rate for the transformation head to 1.0 for Traffic Sign, MNIST, and CIFAR-100 and 0.1 for the remaining datasets. The results are reported in Table 5.

According to the results in the table, we can observe that MOKD+TSA fails to outperform TSA in most cases. According

Table 5. **Comparisons of MOKD and MOKD TSA.**

Datasets	ImageNet	Omniglot	Aircraft	Birds	DTD	QuickDraw	Fungi	VGG_Flower	Traffic Sign	MSCOCO	MNIST	CIFAR10	CIFAR100
TSA	**57.7±1.1**	94.7±0.4	**88.9±0.5**	**80.7±0.8**	**77.2±0.7**	**82.3±0.6**	**66.8±1.0**	**92.7±0.5**	83.6±0.9	**55.2±1.0**	96.8±0.3	**80.6±0.8**	**71.2±0.9**
MOKD+TSA	56.2±1.1	**94.9±0.4**	88.1±0.6	78.6±0.8	73.0±0.7	81.2±0.5	63.9±1.0	92.1±0.6	**87.3±0.8**	54.5±1.0	**97.4±0.3**	78.9±0.8	64.7±0.8

Table 6. **Comparisons of performance gaps between initial and final steps.**

Datasets	ImageNet	Omniglot	Aircraft	Birds	DTD	QuickDraw	Fungi	VGG_Flower	Traffic Sign	MSCOCO	MNIST	CIFAR10	CIFAR100
URL	2.00	0.15	1.36	0.27	2.42	0.19	1.40	0.58	13.94	2.30	3.59	2.77	3.52
MOKD	2.00	0.23	1.55	0.46	2.68	0.25	1.22	1.22	14.99	3.70	4.19	3.67	4.64

to our further visualization results of learning curves, we find that severe overfitting happens in the failure cases. Such a phenomenon implies that MOKD strategy may be more suitable for the efficient transformation head adaptation. In the case of more trainable modules in the backbone, MOKD may limit the performance due to overfitting.

F.3. Further Studies on Vary-way 5-shot and 5-way 1-shot

In this section, we further conduct experiments on more challenging 5-way 1-shot and vary-way 5-shot tasks under the "train on all datasets" settings. These tasks are difficult for our proposed MOKD since scarce data have a negative effect on HSIC estimation. For example, less accurate HSIC measures can be estimated with fewer data samples. Besides, maximizing dependence between representations and labels with only extremely few data samples may result in learning biased class-specific representations. The results are reported in Table 7.

Vary-way 5-shot. According to Table 7, MOKD achieves best performance on 6 out of 13 datasets and ranks 1.8 among all baselines on vary-way 5-shot task settings. Generally, MOKD obtains comparable results on Omniglot and Quick Draw compared with the best results but obtains better results on Aircraft, CU_Birds, VGG_Flower, and Traffic Sign with improvements 0.5%, 0.7%, 0.6% and 2.5% respectively. Such results show that MOKD can obtain good performance even if the available data are relatively scarce. However, compared with the results under vary-way vary-shot settings, MOKD fails to achieve impressive performance in general.

Five-way One-shot. The results regarding 5-way 1-shot tasks are reported in Table 7. According to the table, we observe that MOKD fails to achieve the best performance under 5-way 1-shot task settings. We notice that overfitting took place when performing MOKD on 5-way 1-shot tasks. The reason for such a phenomenon is that there is only one sample available for training and MOKD tends to learn excessively biased representations for each class. Even so, MOKD still outperforms other baselines except for URL.

Some remarks regarding the empirical results. As shown in both vary-way 5-shot and 5-way 1-shot task settings, MOKD fails to significantly outperform all baselines. According to the learning curves in our experiments, we find that MOKD tends to overfit the data under these two settings. A reasonable cause for such a phenomenon is that too few data samples negatively affect the estimation of HSIC measures. For example, in 5-way 1-shot task settings, $\mathrm{HSIC}(Z, Y)$ has to explore representations from only 5 data samples to match the cluster structures of the given support set. Thus, it is highly possible that the learned representations will be extremely biased since there is only one reference sample for each class. Thus, the generalization performance drastically drops. From this perspective, we know that the proposed MOKD is unsuitable for tasks with extremely scarce data samples in each class since the estimated HSIC is not reliable enough to learn a set of good representations that match the cluster structures.

F.4. Analyses on Gamma

In our work, γ functions as a coefficient of regularization term $\mathrm{HSIC}(Z, Z)$. Since $\mathrm{HSIC}(Z, Z)$ mainly facilitates penalizing high-variance representations and removing common information shared among samples, γ intuitively determines the power of penalization and suppression imposed to high-variance representations and common features shared across samples. In order to figure out how datasets in Meta-Dataset react to γ, we run MOKD with different γ values under the "train on all datasets" settings. The results are reported in Table 8 and Fig. 6 and 7.

According to the numerical results reported in Table 8, a general conclusion we can summarize is that different datasets prefer different values of γ. To be concrete, for simple datasets, such as Omniglot, Aircraft and MNIST, small γ is preferred since images in these datasets are simple and the main object of each image is evident. However, for complicated datasets, such as ImageNet, Fungi, and MSCOCO, large γ is better since images in these datasets contain abundant objects and semantic information. In some cases, the semantic information, such as the backgrounds, is useless and sometimes may

Table 7. **Results on vary-way 5-shot and 5-way 1-shot task settings (Trained on All Datasets).** Mean accuracy, 95% confidence interval reported.

Datasets	Vary-way 5-shot					5-way 1-shot				
	Sim-CNAPS	SUR	URT	URL	**MOKD**	Sim-CNAPS	SUR	URT	URL	**MOKD**
ImageNet	47.2±1.0	46.7±1.0	**48.6±1.0**	47.8±1.0	47.5±1.0	42.6±0.9	40.7±1.0	**47.4±1.0**	46.5±1.0	46.0±1.0
Omniglot	95.1±0.3	95.8±0.3	**96.0±0.3**	95.8±0.3	**96.0±0.3**	93.1±0.5	93.0±0.7	**95.6±0.5**	95.5±0.5	95.5±0.5
Aircraft	74.6±0.6	82.1±0.6	81.2±0.6	83.9±0.5	**84.4±0.5**	65.8±0.9	67.1±1.4	77.9±0.9	**78.6±0.9**	**78.6±0.9**
Birds	69.6±0.7	62.8±0.9	71.2±0.7	76.1±0.7	**76.8±0.6**	67.9±0.9	59.2±1.0	70.9±0.9	**76.2±0.9**	75.9±0.9
Textures	57.5±0.7	60.2±0.7	65.2±0.7	**66.8±0.6**	66.3±0.6	42.2±0.8	42.5±0.8	49.4±0.9	**52.0±0.9**	51.4±0.9
Quick Draw	70.9±0.6	79.0±0.5	**79.2±0.5**	78.3±0.5	78.9±0.5	70.5±0.9	**79.8±0.9**	79.6±0.9	79.1±0.9	78.9±0.9
Fungi	50.3±1.0	66.5±0.8	66.9±0.9	68.7±0.9	**68.8±0.9**	58.3±0.1	64.8±1.1	71.0±1.0	**71.4±1.0**	71.1±1.0
VGG Flower	86.5±0.4	76.9±0.6	82.4±0.5	88.5±0.4	**89.1±0.4**	79.9±0.7	65.0±1.0	72.7±1.0	**80.3±0.8**	79.8±0.8
Traffic Sign	55.2±0.8	44.9±0.9	45.1±0.9	56.7±0.8	**59.2±0.8**	55.3±0.9	44.6±0.9	52.7±0.9	**57.4±0.9**	57.0±0.9
MSCOCO	49.2±0.8	48.1±0.9	**52.3±0.9**	51.3±0.8	51.8±0.8	48.8±0.9	47.8±1.1	**56.9±1.1**	52.1±1.0	50.9±0.8
MNIST	88.9±0.4	**90.1±0.4**	86.5±0.5	88.5±0.4	89.4±0.3	**80.1±0.9**	77.1±0.9	75.6±0.9	73.3±0.8	72.5±0.9
CIFAR-10	**66.1±0.7**	50.3±1.0	61.4±0.7	59.6±0.7	58.8±0.7	**50.3±0.9**	35.8±0.8	47.3±0.9	48.6±0.8	47.3±0.8
CIFAR-100	53.8±0.9	46.4±0.9	52.5±0.9	**55.8±0.9**	55.3±0.9	53.8±0.9	42.9±1.0	54.9±1.1	**61.5±1.0**	60.2±1.0
Average Seen	69.0	71.2	73.8	75.7	**76.0**	65.0	64.0	70.6	**72.5**	72.2
Average Unseen	62.6	56.0	59.6	62.3	**63.0**	57.7	49.6	57.5	**58.4**	57.5
Average All	66.5	65.4	68.3	70.6	**71.0**	62.2	58.5	65.5	**67.1**	66.5
Average Rank	4.1	3.8	2.8	2.2	**1.8**	3.6	4.2	2.5	**1.7**	2.8

[1] Both the results on URL and MOKD are the average of 5 random seed. The ranks only consider the first 10 datasets.

Table 8. **Analyses on γ (Trained on All Datasets).** Mean accuracy, 95% confidence interval are reported.

Datasets	$\gamma = 0.0$	$\gamma = 0.5$	$\gamma = 1.0$	$\gamma = 2.0$	$\gamma = 3.0$	$\gamma = 4.0$	$\gamma = 5.0$
ImageNet	53.6±1.0	56.2±1.1	57.0±1.1	57.2±1.1	**57.3±1.1**	**57.3±1.1**	**57.3±1.1**
Omniglot	**94.4±0.5**	**94.4±0.5**	94.2±0.5	93.9±0.5	93.7±0.5	93.5±0.5	93.3±0.5
Aircraft	85.2±0.5	87.6±0.5	**88.4±0.5**	88.3±0.5	88.2±0.5	88.0±0.5	87.9±0.5
Birds	77.8±0.7	80.3±0.7	**80.4±0.8**	80.3±0.8	80.1±0.8	79.9±0.8	79.8±0.8
Textures	73.2±0.7	75.4±0.7	76.1±0.7	76.3±0.7	**76.5±0.7**	**76.5±0.7**	**76.5±0.7**
Quick Draw	80.5±0.6	82.1±0.6	**82.3±0.6**	**82.3±0.6**	82.2±0.6	82.1±0.6	82.0±0.6
Fungi	61.9±0.9	65.1±1.0	66.8±1.0	68.1±1.0	68.6±1.0	**68.7±1.0**	**68.7±1.0**
VGG Flower	88.8±0.5	91.5±0.5	92.1±0.5	92.4±0.5	**92.5±0.5**	**92.5±0.5**	92.3±0.5
Traffic Sign	48.7±1.0	62.9±1.1	64.1±1.1	**64.6±1.1**	64.5±1.1	64.2±1.1	64.0±1.1
MSCOCO	44.7±1.0	51.3±1.0	53.4±1.0	55.0±1.0	**55.5±1.0**	55.4±1.0	55.3±1.0
MNIST	91.7±0.5	95.0±0.4	**95.1±0.4**	94.6±0.4	94.5±0.4	94.3±0.4	94.3±0.4
CIFAR-10	66.2±0.8	71.0±0.8	72.3±0.8	72.4±0.8	72.8±0.8	**72.9±0.8**	**72.9±0.8**
CIFAR-100	57.1±1.0	62.5±1.0	63.5±1.0	**64.0±1.0**	63.9±1.0	63.8±1.0	63.6±1.0
Average Seen	76.9	79.1	79.7	79.8	**79.9**	79.8	79.7
Average Unseen	61.7	68.4	69.7	70.1	**70.2**	70.1	70.0
Average All	71.1	75.0	75.8	76.1	**76.2**	76.1	76.0

have a negative effect on the performance.

In addition, a special case is that it is equivalent to performing an ablation study on $\mathrm{HSIC}(Z, Z)$ when γ is set to zero. As shown in the table, the performance drops drastically on most datasets. By further plotting the learning curves (see Fig. 6), we find that overfitting happens on these datasets when the $\mathrm{HSIC}(Z, Z)$ term is removed. Thus, such phenomenon demonstrates that $\mathrm{HSIC}(Z, Z)$ contributes to penalizing high-variance kernelized representations and alleviating the overfitting phenomenon.

Further discussion about γ. According to the results reported in Table 8, an interesting phenomenon is that different datasets achieve their best performance with different gamma values. In our opinion, two aspects are worth noticing here.

On the one hand, as aforementioned, there exist high similarities between samples from different classes when performing classification with NCC-based loss. A reasonable cause for such a phenomenon is the trivial common features shared across samples. On the other side, according to Fig. 6, merely maximizing $\mathrm{HSIC}(Z, Y)$ results in the overfitting phenomenon.

For complicated datasets, such as ImageNet and MSCOCO, there are many objects and abundant semantic information in images. However, most of semantic information is useless and sometimes has a negative effect on representation learning. Thus, when tasks are sampled from these datasets, it is challenging for the model to learn definite and discriminative representations for each class. This will in turn result in uncertainties. For example, Fig. 8(g) and 9(d) have demonstrated

Table 9. **Comparisons of running time between MOKD and URL.** (sec. per task)

Datasets	ImageNet	Omniglot	Aircraft	Birds	DTD	QuickDraw	Fungi	VGG_Flower	Traffic Sign	MSCOCO	MNIST	CIFAR10	CIFAR100
URL	0.7	0.8	0.5	0.7	0.4	1.1	1.0	0.5	1.0	0.9	0.5	0.5	1.1
MOKD	0.9	0.7	0.8	0.8	0.8	0.9	0.8	0.8	0.9	0.9	0.8	0.8	0.9

Table 10. **Ablation study on test power maximization.** Mean accuracy, 95% confidence interval are reported.

Datasets	MOKD w/o Test Power	MOKD w Test Power
ImageNet	56.4±1.1	**57.2±1.1**
Omniglot	93.9±0.5	**94.2±0.5**
Aircraft	88.1±0.5	**88.3±0.5**
Birds	80.1±0.8	**80.2±0.8**
Textures	**75.9±0.7**	75.7±0.7
Quick Draw	82.0±0.6	**82.1±0.6**
Fungi	64.1±1.1	**68.9±1.0**
VGG Flower	**91.9±0.5**	91.8±0.5
Traffic Sign	62.8±1.2	**64.1±1.0**
MSCOCO	52.9±1.1	**55.3±1.0**
MNIST	**95.3±0.4**	95.0±0.4
CIFAR-10	72.1±0.8	**73.0±0.8**
CIFAR-100	62.0±1.0	**63.0±1.0**
Average Seen	79.0	**79.8**
Average Unseen	69.0	**70.1**
Average All	75.2	**76.1**

this. Meanwhile, due to the scarce data in few-shot classification tasks and the strong power of $\mathrm{HSIC}(Z, Y)$, the model tends to learn high-variance representations and overfit the data.

Thus, a large gamma value is essential for these complicated datasets. For one thing, $\mathrm{HSIC}(Z, Z)$ facilitates penalizing high-variance kernelized representations for further alleviating the overfitting phenomenon. For another thing, according to its definition, $\mathrm{HSIC}(Z, Z)$ measures the dependence between two sets of data. Thus, minimizing $\mathrm{HSIC}(Z, Z)$ drives the model to learn discriminative features for each single sample so that samples are "independent" of each other. This further helps remove the trivial common features shared across samples and in turn, alleviates the high similarities among samples. In contrast, since simple datasets, such as Omniglot and Aircraft, own evident and definite semantic area, small γ is enough.

Thus, in our work, we set small γ for those simple datasets, such as Aircraft, Omniglot, and MNIST. However, for those complex datasets, such as ImageNet and MSCOCO, we set large $\gamma = 3$ (inspired by SSL-HSIC (Li et al., 2021b)).

F.5. Comparisons of Different Bandwidth Selection

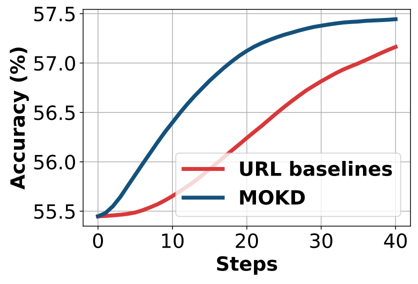
In this paper, we adopt the grid search method to select the optimal bandwidth for the kernel to maximize its test power for dependence detection. In addition to the grid search Jitkrittum et al. (2016), many previous works have been done to optimize the Gaussian kernel in this field. Specifically, El Amri & Marrel (2024) similarly proposed to select the bandwidth in a range of values 2^{β}, where $\beta = \{-15, -13, ..., 4, 5\}$. Li et al. (2021b) proposed to optimize the kernel distance entropy to tune kernel parameters automatically. To figure out the differences among these strategies, we perform MOKD respectively with each of these strategies. To be clear, we denote these invariants respectively as "MOKD + 2^{β}" and "MOKD + entropy". The results are reported in the following table. All results are obtained with random seed 42.

Table 11. **Comparisons of different bandwidth selection strategies.**

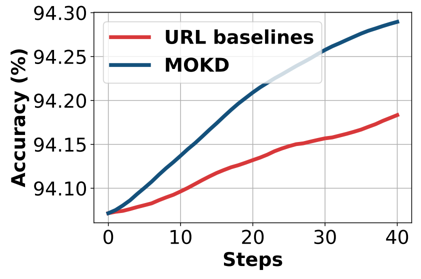
Datasets	ImageNet	Omniglot	Aircraft	Birds	DTD	QuickDraw	Fungi	VGG_Flower	Traffic Sign	MSCOCO	MNIST	CIFAR10	CIFAR100
MOKD	**57.5±1.1**	94.3±0.4	88.3±0.5	80.2±0.8	**76.7±0.7**	82.5±0.6	**67.8±1.0**	**92.8±0.5**	**64.9±1.1**	**55.6±1.0**	95.1±0.4	**73.0±0.8**	**64.5±1.0**
MOKD + 2^{β}	56.9±1.1	**94.4±0.4**	88.3±0.5	80.2±0.8	76.2±0.7	82.5±0.6	65.2±1.0	92.3±0.5	62.7±1.2	54.8±1.0	**95.6±0.4**	72.4±0.8	63.5±1.0
MOKD + entropy	56.8±1.1	94.0±0.5	88.1±0.5	80.0±0.8	76.0±0.7	82.3±0.6	63.6±1.1	91.9±0.6	62.5±1.2	52.0±1.0	95.5±0.4	72.6±0.8	62.6±1.0

As we can observe from the table, it is easy for us to observe that the grid search approaches achieve better performance than the entropy optimization method. Specifically, we can observe evident performance gaps on Fungi, VGG_Flower, Traffic Sign, MSCOCO, and CIFAR-100 datasets. Further, we can observe that the test power maximization strategy adopted in our paper achieves better performance than "MOKD + 2^{β}. The main difference between the two strategies is the original

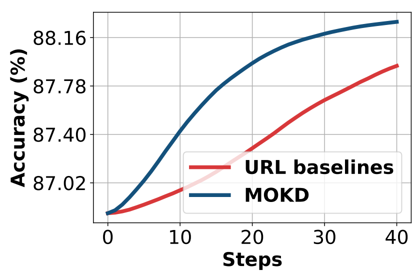
MOKD scales the median of the kernel matrix while "MOKD + 2^{β}" uses 2^{β} as the bandwidth. However, 2^{β} is not an empirically appropriate bandwidth for the kernel.



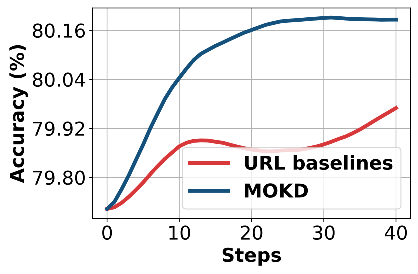
(a) ImageNet



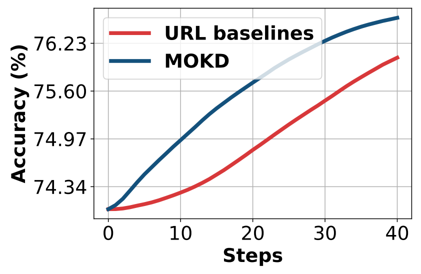
(b) Omniglot



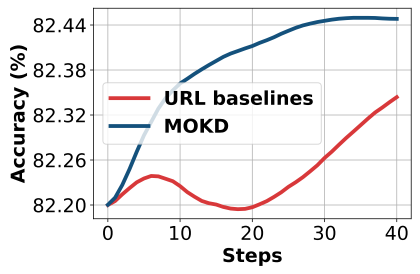
(c) Aircraft



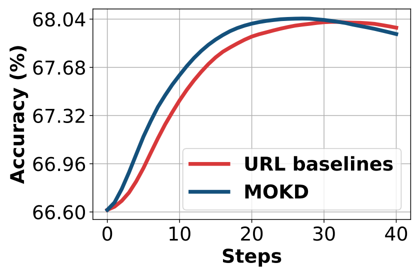
(d) CU_Birds



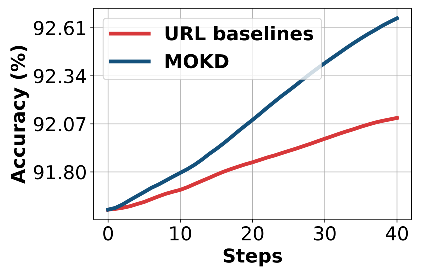
(e) DTD



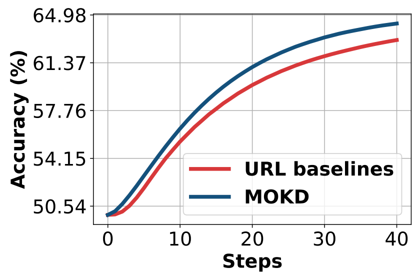
(f) Quick Draw



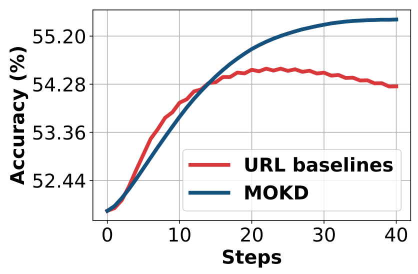
(g) Fungi



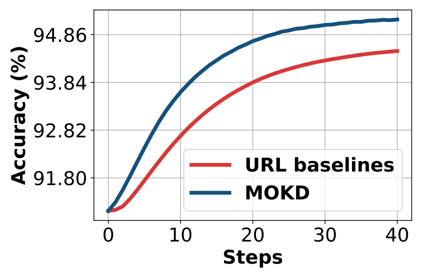
(h) VGG Flower



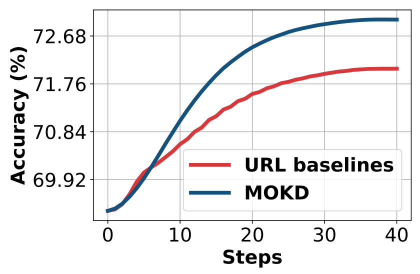
(i) Traffic Sign



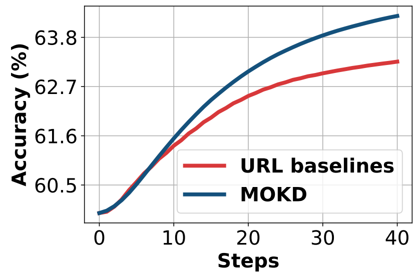
(j) MSCOCO



(k) MNIST

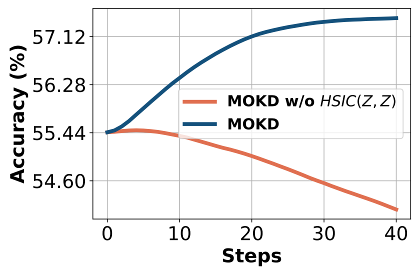


(l) CIFAR 10

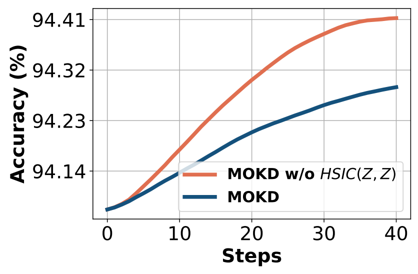


(m) CIFAR 100

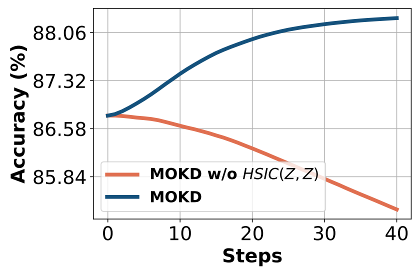
Figure 5. Test accuracy curves of Meta-Dataset with respect to the steps under “train on all datasets” settings. As shown in the figures, MOKD evidently achieves a better learning process and convergence performance compared with URL baseline.



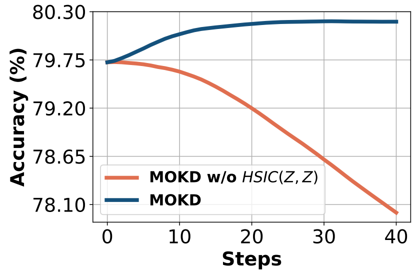
(a) ImageNet



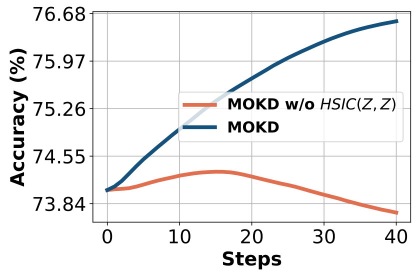
(b) Omniglot



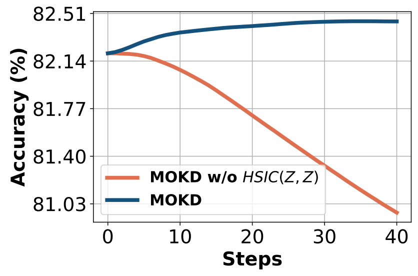
(c) Aircraft



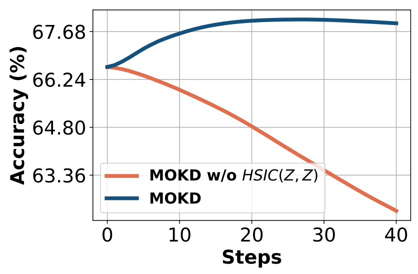
(d) CU_Birds



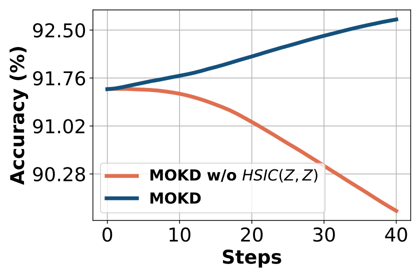
(e) DTD



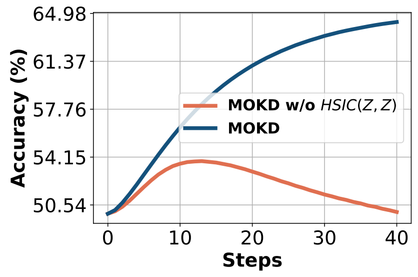
(f) Quick Draw



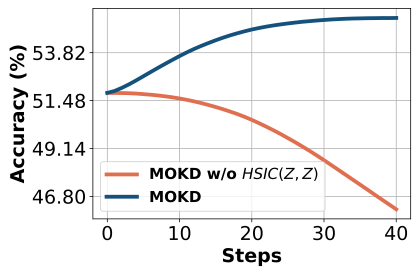
(g) Fungi



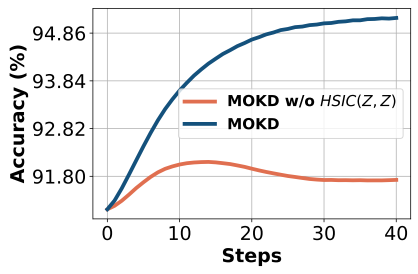
(h) VGG_Flower



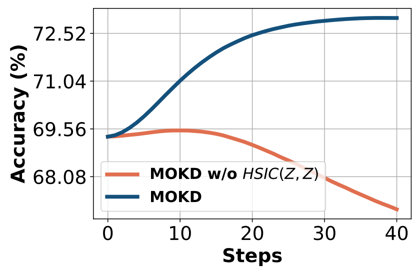
(i) Traffic Sign



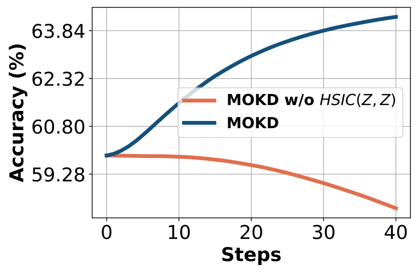
(j) MOCOCO



(k) MNIST

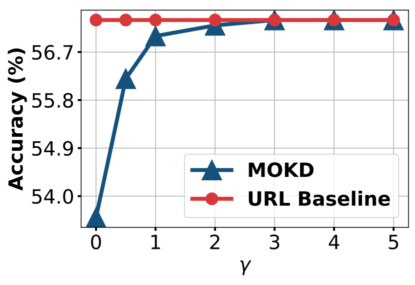


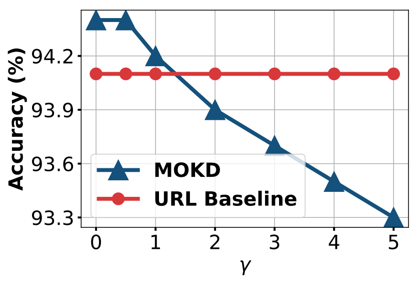
(l) CIFAR 10

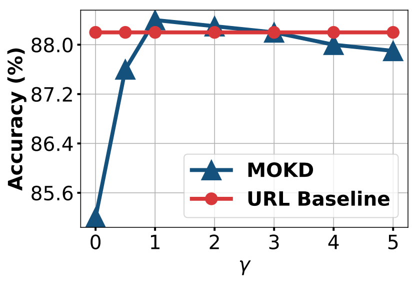


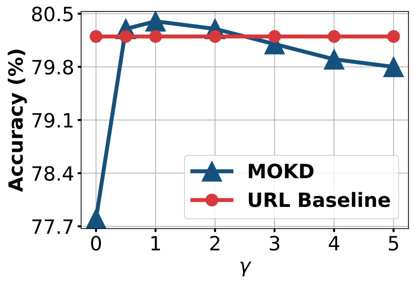
(m) CIFAR 100

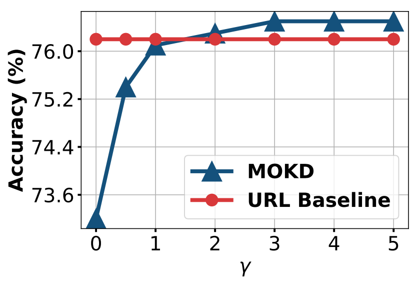
Figure 6. **Part test accuracy curves of datasets in Meta-Dataset.** The learning curves show that when $\mathrm{HSIC}(Z, Z)$ is removed, MOKD tends to overfit the training data and achieves bad generalization performance.

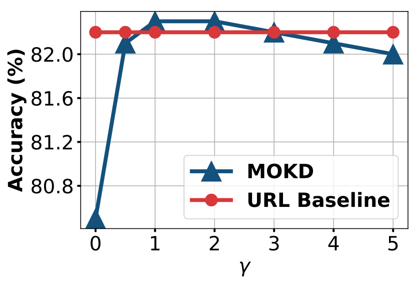

(a) ImageNet

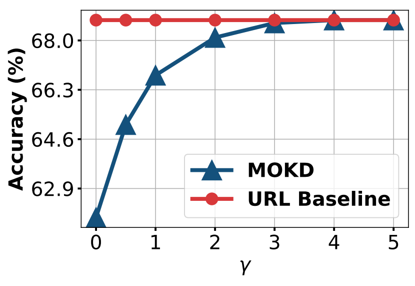

(b) Omniglot

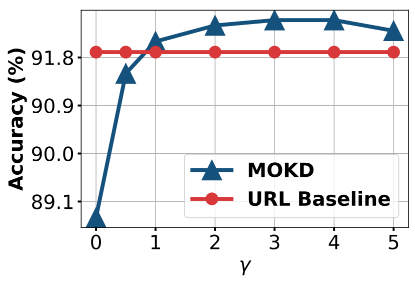

(c) Aircraft

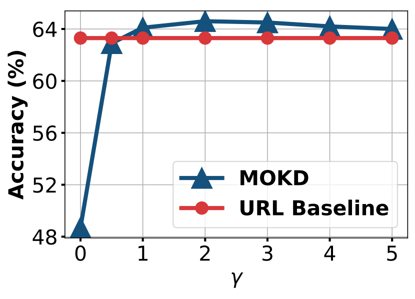

(d) CU_Birds

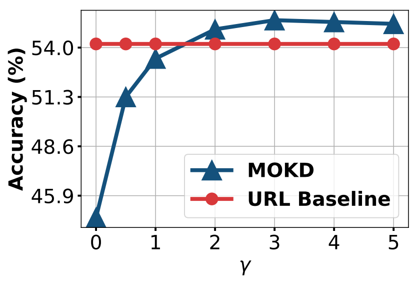

(e) DTD

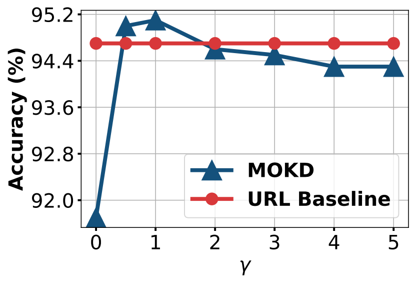

(f) Quick Draw

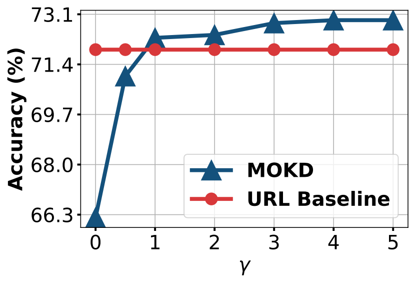

(g) Fungi


(h) VGG Flower


(i) Traffic Sign


(j) MSCOCO


(k) MNIST


(l) CIFAR 10

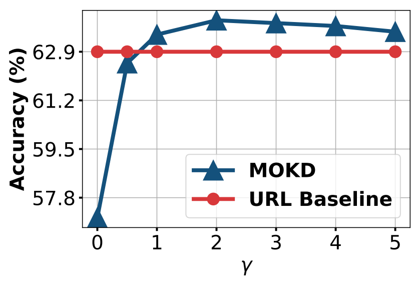
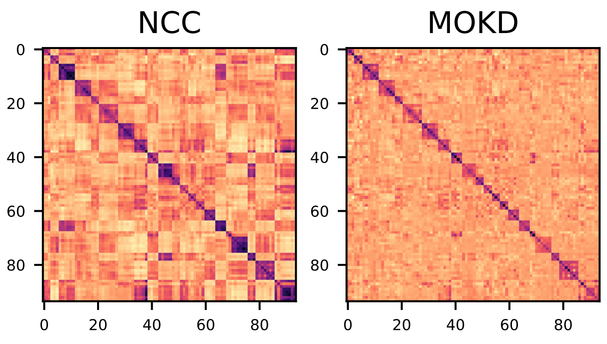
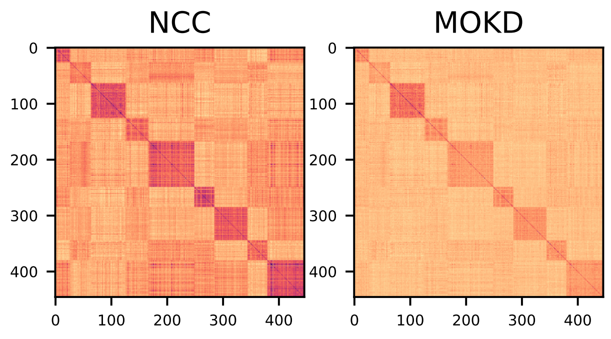

(m) CIFAR 100

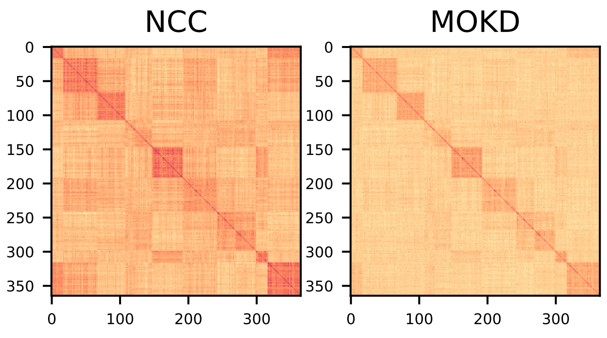
Figure 7. Illustration of effect of γ on all datasets in Meta-Dataset.



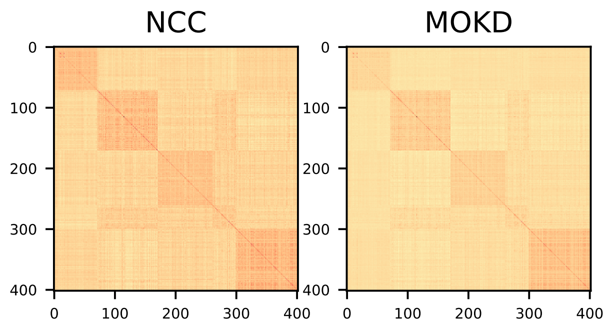
(a) Omniglot (21 classes)



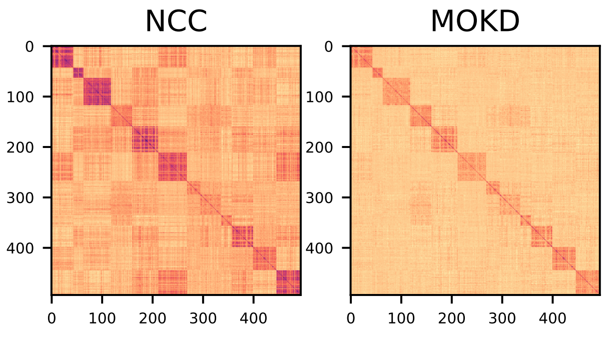
(b) Aircraft (9 classes)



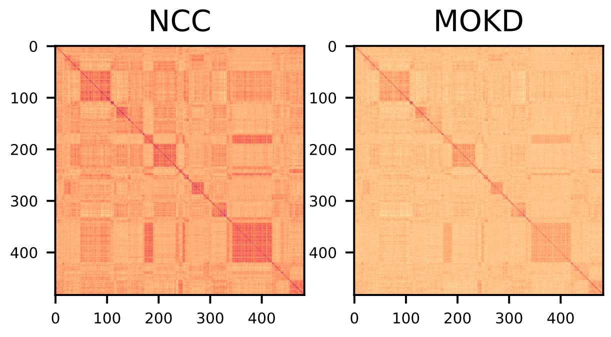
(c) CU_Birds (11 classes)



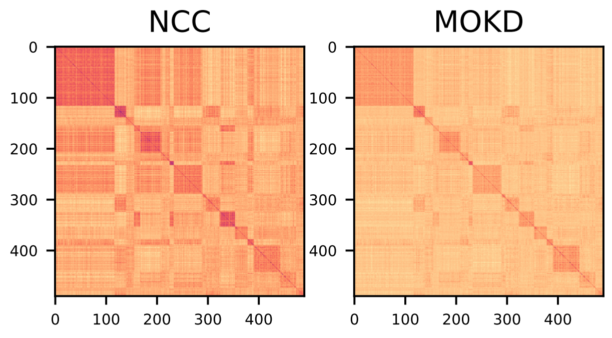
(d) DTD (5 classes)



(e) Quick Draw (12 classes)

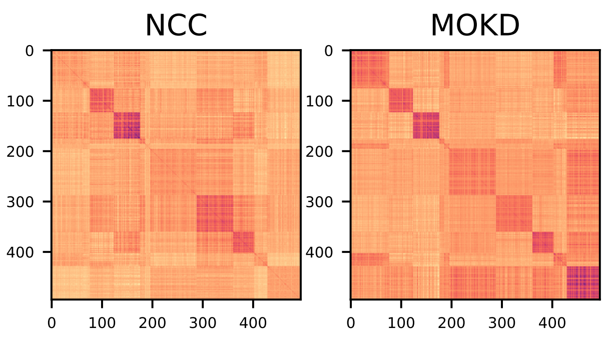


(f) Fungi (33 classes)

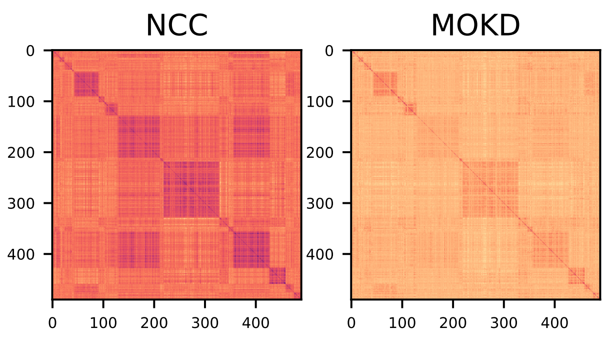


(g) VGG_Flower (16 classes)

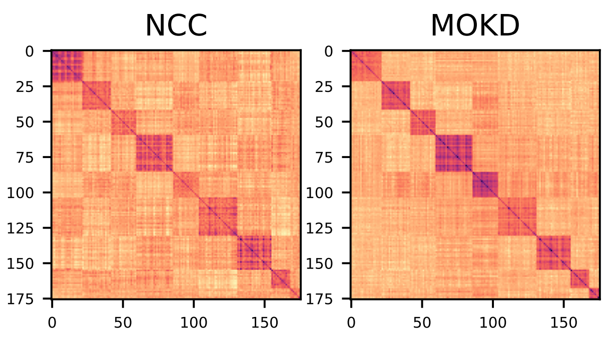
Figure 8. Visualization results of similarity matrices of representations respectively learned with NCC-based loss and MOKD on seen domains.



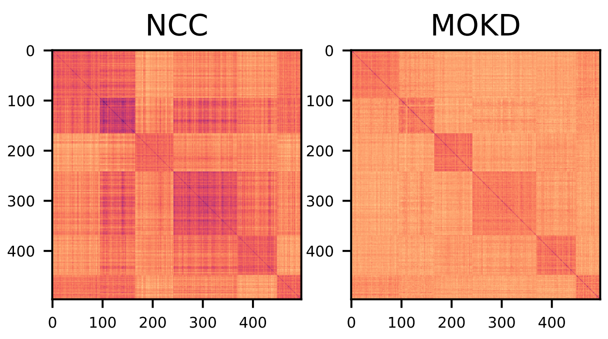
(a) Traffic Sign (12 classes)



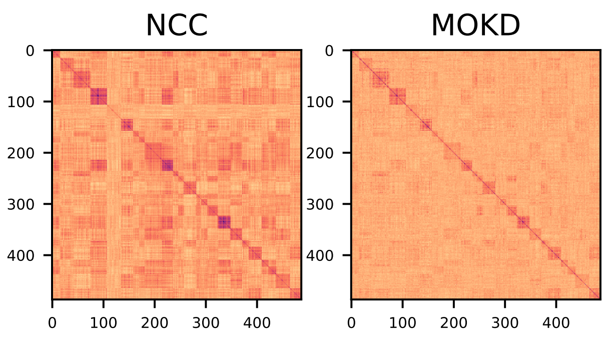
(b) MSCOCO (19 classes)



(c) MNIST (9 classes)



(d) CIFAR 10 (6 classes)



(e) CIFAR 100 (24 classes)

Figure 9. Visualization results of similarity matrices of representations repsectively learned with NCC-based loss and MOKD on unseen domains.