AdaNPC: Exploring Non-Parametric Classifier for Test-Time Adaptation

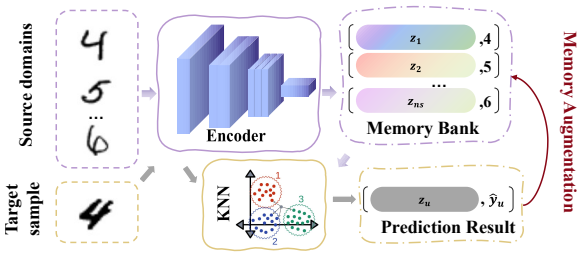
Yi-Fan Zhang [1 2 3] **Xue Wang** [4] **Kexin Jin** [5] **Kun Yuan** [6 7] **Zhang Zhang** [1 2] **Liang Wang** [1 2] **Rong Jin** [8] **Tieniu Tan** [1 2 9]

Abstract

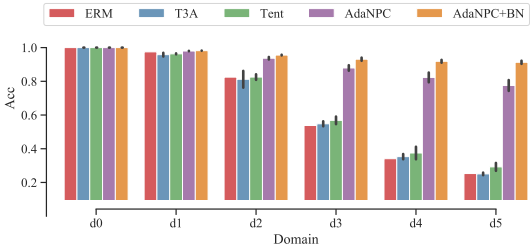
Many recent machine learning tasks focus to develop models that can generalize to unseen distributions. Domain generalization (DG) has become one of the key topics in various fields. Several literatures show that DG can be arbitrarily hard without exploiting target domain information. To address this issue, test-time adaptive (TTA) methods are proposed. Existing TTA methods require offline target data or extra sophisticated optimization procedures during the inference stage. In this work, we adopt **N**on-**P**arametric **C**lassifier to perform the test-time **Ada**ptation (AdaNPC). In particular, we construct a memory that contains the feature and label pairs from training domains. During inference, given a test instance, AdaNPC first recalls k closed samples from the memory to vote for the prediction, and then the test feature and predicted label are added to the memory. In this way, the sample distribution in the memory can be gradually changed from the training distribution towards the test distribution with very little extra computation cost. We theoretically justify the rationality behind the proposed method. Besides, we test our model on extensive numerical experiments. AdaNPC significantly outperforms competitive baselines on various DG benchmarks. In particular, when the adaptation target is a series of domains, the adaptation accuracy of AdaNPC is 50% higher than advanced TTA methods. Code is available at https://github.com/yfzhang114/AdaNPC.

[1]School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS) [2]MAIS, CRIPAC, CASIA. [3]Work done during an internship at Alibaba Group. [4]Machine Intelligence Technology, Alibaba Group. [5] Department of Mathematics at Princeton University. [6]Center for Machine Learning Research, Peking University. [7]AI for Science Institute, Beijing. [8]Work done at Alibaba Group, and now affiliated with Meta. [9]Nanjing University. Correspondence to: Xue Wang <xue.w@alibaba-inc.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).



(a) AdaNPC memorizes features and labels of source domain instances. *During inference*, each arrival target sample will be classified by a KNN classifier, where the nearest neighbors are searched in the memory. *For test-time adaptation*, the target feature and prediction will be further stored in the memory bank.



(b) Strong knowledge expandability of AdaNPC. When our model is trained on a domain d_0 and adapted to target domains $d_1, ..., d_5$ successively, advanced TTA methods only bring margin performance improvement, however, AdaNPC and its variants boost the accuracy significantly.

Figure 1. **An illustration example of AdaNPC that proposes to utilize a non-parametric classifier for test-time adaptation.**

1. Introduction

The classic machine learning models generally suffer from degraded performance when the training and test data are not from the same distribution. Many researchers consider developing out-of-distribution (OOD) generalization approaches (e.g., disentanglement (Zhang et al., 2022a), causal invariance (Arjovsky et al., 2020; Zhang et al., 2022c), and adversarial training (Ganin et al., 2016; Li et al., 2018c).), in which models are trained on multiple source domains/datasets and can be directly deployed on *unseen* target domains.

In recent studies (Zhang et al., 2022d; Dubey et al., 2021),

people show that robustifying a model to any unknown distribution is almost impossible without utilizing target samples during inference. The test-time adaptive (TTA) methods are then proposed to utilize target samples with computationally practical constraints. However, current TTA methods suffer from several drawbacks. (1) **Overhead computation:** existing TTA methods either need batches of target data for gradient updates (Sun et al., 2020; Wang et al., 2020a; Zhang et al., 2021b) and/or an additional model for fine-tuning (Sun et al., 2020; Dubey et al., 2021; Zhang et al., 2022d), which are prohibited when target sample arrives one by one in the *online* manner. (2) **Domain forgetting:** existing TTA methods require making changes in the trained model. The model would gradually lose the prediction ability of the training domains, indicating that some knowledge loss is inevitable. This issue is especially significant when conducting inference for a series of domains. Let us consider a simple test on Rotated MNIST dataset (Ghifary et al., 2015), we perform test-time adaptation to $d_1, d_2, ..., d_5$ serially using the latest TTA methods, T3A (Iwasawa & Matsuo, 2021) and Tent (Wang et al., 2020a). In Figure 1(b), we observe that the generalization ability on d_5 of all existing methods is poor even after adaptation in the first four domains. We also summarize the generalizability in the source domain d_0 in Figure 2 drops significantly. That is, current TTA methods cannot adapt to a series of online domains and easy to forget historical knowledge.

To this end, we propose a *non-parametric adaptation* approach, debuted AdaNPC. In particular, AdaNPC trains the model with a K nearest neighbor (KNN) based loss instead of the cross-entropy loss, which minimizes the influence of outliers / irrelevant samples on the potentially noisy training dataset. After training, AdaNPC constructs a memory bank to maintain the trained feature and label pairs of the training dataset. When switching to the inference stage, the feature of a given test sample is first computed using a forward procedure, and then, based on similarity, the top k closed samples in the memory bank are collected to generate a voting prediction. Finally, the new testing pair (feature and predicted label) is added to the memory bank. We illustrate the whole procedure in Figure 1(a). As dense vectors' searching can be efficiently implemented with logarithmic dependence in total sample size (Johnson et al., 2019), the computation cost of AdaNPC in the inference stage is almost the same as a single forward pass and *is significantly smaller than backward gradient updates.* On the other hand, our approach separates the feature extraction procedure and individual sample memorization. It facilitates us to maintain the information among source and target domains simultaneously. The main contributions of this paper are:

1. We investigate a non-parametric paradigm to perform test-time adaptation by storing features and predicted pseudo-labels of target instances. The proposed AdaNPC can be incorporated with any representation learning models.

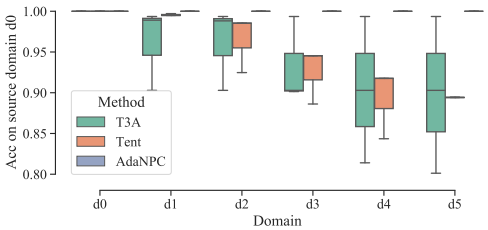


Figure 2. **Re-evaluation on the source domain d_0 after the model is adapted to unseen target domain d_i.** When adapted to more target domains, the performance of existing TTA methods on d_0 drops drastically while AdaNPC always attains a high accuracy.

2. We theoretically derive target domain error bounds under both the covariate shift setting and the posterior shift setting. Our theoretical results show that a non-parametric classifier can explicitly reduce the domain divergence and makes the target error bound tighter; Besides, AdaNPC, which incorporates online target instances into the memory bank, will further reduce the target risk.

3. We perform extensive experiments on 5 popular OOD benchmarks with 3 different backbones, where the results show that AdaNPC (1) achieves competitive generalization performance in both target and source domains; (2) beats most existing test-time adaptation methods by a large margin; (3) using non-parametric classifier attains faster convergence and performs well even without fine tuning, which is promising for large pre-trained models.

4. AdaNPC has some unique benefits, such as *interpretability:* by analyzing the neighborhood samples chosen by AdaNPC, we can interpret how past knowledge is used for inference results, and *knowledge expandability:* AdaNPC stores all the data features that have been seen and avoid catastrophic forgetting.

2. Related Work

Test-Time adaptive methods (Liang et al., 2023) are recently proposed to utilize target samples. The taxonomy of Test-Time adaptive methods is summarized in Appendix Table 7, where Test-Time Training methods design proxy tasks during tests such as self-consistence (Zhang et al., 2021b), rotation prediction (Sun et al., 2020) and need extra models; Test-Time Adaptation methods adjust model parameters based on unsupervised objectives such as entropy minimization (Wang et al., 2020a; Zhang et al., 2022b) or update a prototype for each class (Iwasawa & Matsuo, 2021). Domain adaptive method (Dubey et al., 2021) needs additional models to adapt to target domains. Both the Test-Time adaptation methods and the domain-adaptive methods need batches of target samples. Single sample generalization methods are recently proposed, which need to learn

an adapt strategy from source domains, i.e., (Xiao et al., 2022) formulates the adaptation to the single test sample as a variational Bayesian inference problem and uses the meta-learning pipeline to learn the adaptative strategy; (Zhang et al., 2022d) introduces specific classifiers for different domains and adapt the voting weight for test samples dynamically. All aforementioned approaches either require accessing the batch of data or need gradient computation to update the models, which is computationally inefficient especially in the online setting with steaming data. Our proposed AdaNPC performs Test-Time adaptation by storing the test features and predicted labels and doesn't have those restrictions. See more related work in Appendix A.

3. Methods

Problem setting. In domain generalization (DG), we consider a source domain dataset $D_S = \{(x_s^i, y_s^i)\}_{i=1}^{n_s}$ is collect from different environments or domains, where (x_s^i, y_s^i) is sampled i.i.d. from the distribution $\mathbb{D}_S$ and n_s is the total sample size. The DG aims to train a predictor $\hat{f}$ on source domain dataset D_S and to perform well on a testing unseen dataset $D_U = \{(x_u^i)\}_{i=1}^{n_u}$ that sampled from a distribution $\mathbb{D}_U$, which is inaccessible during training. We formally define the classification error and its variants on $\mathbb{D}_U$, whereas definitions for other domains are the same.

Definition 1. ***(Regression function.)*** *In binary classification setting[a], given a distribution $\mathbb{D}_U$, the regression functions are defined to represent the conditional distributions.*

$$\eta_U(x) = P_{x\sim\mathbb{D}_U}(Y = 1|X = x). \tag{1}$$

(Classification error.) *The error of hypothesis $\hat{f} \in \mathcal{H} : \mathcal{X} \to \{0, 1\}$ under the distribution $\mathbb{D}_U$ is defined as*

$$\epsilon_U(\hat{f}) = \mathbb{E}_{(x,y)\sim\mathbb{D}_U}[|\hat{f}(x) - y|] = P_U(\{(x, y) : \hat{f}(x) \neq y\}) \tag{2}$$

where $P_U(A)$ denotes the probability of an event A in $\mathbb{D}_U$.

[a]Theories and examples in this work consider binary classification for easy understanding and can be easily extended to multi-class setting.

Definition 2. ***(excess error and its dual form.)*** *Given the Bayes classifier under $\mathbb{D}_U$: $f_U^*(x) = \mathbb{I}\{\eta_U(x) \geq 1/2\}$, the excess error of $\hat{f}$ is defined as*

$$\begin{aligned}\mathcal{E}_U(\hat{f}) &= \epsilon_U(\hat{f}) - \epsilon_U(f_U^*) \\ &= 2\mathbb{E}_{x\sim\mathbb{D}_U}\left[\left|\eta_U(x) - \frac{1}{2}\right| \mathbb{I}\{\hat{f}(x) \neq f_U^*(x)\}\right]\end{aligned} \tag{3}$$

We then introduce the proposed method using the following pipelines: representation learning, making predictions, test-time memory augmentation, and some useful tricks.

Learning representative features. Let $h_\theta(\cdot)$ be the encoder parameterized by θ. We denote $B_{\theta,D_S}(x) := \{a^{(i)}\}_{i=1}^{n_S}$ as the ordered index set in the source domain dataset D_S for any x, formally as follows: satisfying $\|h_\theta(x_{a^{(1)}}) - h_\theta(x)\|_2 \leq \|h_\theta(x_{a^{(2)}}) - h_\theta(x)\|_2 \leq \cdots \leq \|h_\theta(x_{a^{(n_s)}}) - h_\theta(x)\|_2$. We denote $B_{k,\theta,D_S} \subseteq B_{\theta,D_S}$ as the subset only contains the first k elements in B_{θ,D_S}.

We consider optimizing the KNN loss function $\mathcal{L}_{KNN}$:

$$\min_\theta \frac{-1}{n_s} \sum_i \log \frac{\sum_{j\in B_{k,\theta,D_S}(x_i)} \exp(w_{ij}/\tau)\, \mathbb{I}\{y_i = y_j\}}{\sum_{j\in B_{k,\theta,D_S}(x_i)} \exp(w_{ij}/\tau)}, \tag{4}$$

where $w_{ij} = \frac{h_\theta(x_i)^\top h_\theta(x_j)}{\|h_\theta(x_i)\|_2 \|h_\theta(x_j)\|_2}$ is the consine similarity between $h_\theta(x_i)$ and $h_\theta(x_j)$ and $\tau > 0$ is the temperature parameter to scale the influence of w_{ij}.

With this configuration, features with the same label are zoomed closer and different labeled features are pushed away. Compared to cross-entropy loss, the Eq.(4) training paradigm leads to better representations, which is verified in (Feng et al., 2022) and our experiments. The optimization of Eq.(4) is highly nontrivial as B_{k,θ,D_S} changes with model parameters θ in a non-differentiable manner. In this paper, we adopt an EM algorithm to solve it approximately. We only periodically update B_{k,θ,D_S} and keep them fixed for the remaining time, in which we can easily apply the standard optimization schemes in PyTorch or TensorFlow. In practice, one may also consider ignoring this step and directly using the pre-trained model with conventional methods (e.g., ERM (Vapnik, 1999), IRM (Arjovsky et al., 2020), and CORAL (Sun & Saenko, 2016)). In the experimental section, we show that we don't even need to fine-tune the pre-trained model in the source domain, and AdaNPC can still achieve good generalization performance.

Making prediction by non-parametric classifier. Given an unseen sample x_u, we denote the prediction from AdaNPC as follows:

$$\eta_k(x_u) = \text{softmax}\Big(\sum_{j\in B_{k,\theta,\mathcal{M}}(x_u)} w_{uj}\mathbf{1}\{y_j\}\Big), \tag{5}$$

where $\mathbf{1}\{y_j\}$ is the one-hot class label of x_j for $j \geq 1$, w_{uj} is the cosine similarity between $h_\theta(x_u)$ and $h_\theta(x_j)$, and $\mathcal{M}$ is the memory bank and will be specified later. Intuitively, the predictor from Eq.(5) can be viewed as a voting procedure among the memorized samples similar to x_u. We first collect the k closest samples to x_u in the memory bank. The cosine similarity w_{uk} is then computed between k selected samples and x_u. The final decision is the label with the largest summed weights. In this predictor, we explicitly use the samples in the memory bank. It gives us better interpretability than conventional approaches.

Non-parametric test-time adaptation by memory augmentation. At the beginning of the evaluation procedure,

all source domain images are embedded to make up the **Memory bank** $\mathcal{M} = \{(h_\theta(x_i), y_i)\}_{i=1}^{n_s}$. During inference, assume each x_u arrives one by one online and we must report the classification result once x_u arrives. After classification, if the prediction confidence of $\eta_k(x_u)$ is greater than a predefined margin, then the memory set will be augmented by $\mathcal{M} = \mathcal{M} + (h_\theta(x_u), \eta_k(x_u))$. The overall pipeline of AdaNPC is shown in Figure 1(a).

Some useful tricks. 1. BN retraining. The performance of a non-parametric classifier highly depends on the model representation, to attain more powerful representations and maintain the simplicity of AdaNPC, we optionally add *one* BN layer before the classifier. Then during the evaluation, only the BN layer parameter will be re-trained by minimizing the prediction entropy of $\eta_k(x_u)$. **2. Efficient memory bank construction $\mathcal{M}$.** Although there are many advanced techniques for memory construction, such as momentum update in MoCo (He et al., 2020) or Faiss (Johnson et al., 2019), for presentation simplicity consideration, we only adopt the simplest method via computation all pair-wise distance here and hold the usage of advance KNN searching algorithm for the future work. Moreover, to facilitate the training efficiency on the large source dataset, we consider to construct $\mathcal{M}$ with a relatively small size for faster training, where $|\mathcal{M}| \ll n_s$. $\mathcal{M}$ during training will be updated by the first-in-first-out (FIFO) strategy by each minibatch representation.

4. Theoretical Analysis

In this section, we theoretically verify that using KNN as our classifier can explicitly reduce domain divergence. Besides, incorporating target instances, namely the non-parametric Test-Time adaptation, will further reduce the unseen target error. Before we discuss the major results, we first state some necessary assumptions and notions. Refer to Appendix B for missing proofs and detailed explanations.

Assumption 1. *(Strong Density Condition) Given parameter $\mu_-, \mu_+, c_\mu, c_\mu^*, r_\mu > 0$, we assume $\mathbb{D}_S$, $\mathbb{D}_U$ are absolutely continuous with respect to the Lebesgue measure in $\mathbb{R}^d$, and $\mathcal{B}(x, r) = \{x' : \| x' - x \| \leq r\}$ is the ball centered on x with radius r. We assume that $\forall x_u \in \mathbb{D}_U$ and $r \in (0, r_\mu]$ we have*

$$\lambda[\mathbb{D}_S \cap \mathcal{B}(x_u, r)] \geq c_\mu \lambda[\mathcal{B}(x_u, r)]$$
$$\lambda[\mathbb{D}_U \cap \mathcal{B}(x_u, r)] \geq c_\mu^* \lambda[\mathcal{B}(x_u, r)]$$
$$\mu_- < \frac{d\mathbb{D}_S}{d\lambda} < \mu_+; \mu_- < \frac{d\mathbb{D}_U}{d\lambda} < \mu_+,$$

where λ is the Lebesgue measure in Euclidean space.

Strong Density Condition is a commonly assumed condition when analyzing KNN classifier (e.g., (Audibert & Tsybakov, 2007; Cai & Wei, 2021)). Intuitively, Assumption 1 requires the divergence between supports of $\mathbb{D}_S$ and $\mathbb{D}_U$ being bounded. When $c_\mu = 1$, for each $x \in \mathbb{D}_U$, its neighbor ball $\mathcal{B}(x, r)$ is completely within $\mathbb{D}_S$. In contrast, when $c_\mu \approx 0$, $\mathcal{B}(x, r)$ and $\mathbb{D}_S$ are nearly disjoint. We then consider two common assumptions that parameterize the behavior of $\eta_U(x)$.

Assumption 2. *(Smoothness) Let η_U be the classification function and C be a positive constant. For all feasible x, x' we have $|\eta_U(x) - \eta_U(x')| \leq C \parallel x - x' \parallel$.*

Assumption 2 describes that η_U is Lipschitz continuous. Our analysis is capable with weaker condition, such as (α, C_α)-Hölder condition (Cai & Wei, 2021) for some $\alpha \in (0, 1]$ and $C_\alpha > 0$. For notation simplicity, we hold it for future work.

Assumption 3. *(Low Noise Condition). Let β, C_β be positive constants and we assume $\mathbb{D}_U$ satisfies $P_{x \sim \mathbb{D}_U}\left(\left|\eta_U(x) - \frac{1}{2}\right| < t\right) \leq C_\beta t^\beta$ for all $t > 0$.*

The low noise condition is first proposed in (Tsybakov, 2004), which is also named margin assumption (Cai & Wei, 2021). The assumption places a constraint on η_U around $\eta_U(x) \approx 1/2$. A larger β pushes η_U far from $1/2$ and then the classification task will be easier.

4.1. KNN classifier reduces domain divergence

We characterize domain divergence reduction of KNN classifier in the following proposition 1.

Proposition 1. *Let $\mathbb{D}_S$ and $\mathbb{D}_U$ be the source and target domain respectively. Per Assumptions 1 and 2, the risk of hypothesis $\hat{f}$ on the unseen target domain is bounded by*

$$\epsilon_U(\hat{f}) \leq \kappa_s + \epsilon_s(\hat{f}) + \mathcal{O}\left(\mathcal{W}(\mathbb{D}_S, \mathbb{D}_U)\right), \quad (6)$$

where $\mathcal{W}(\cdot, \cdot)$ is the Wasserstein 1-distance , $\kappa_S = \min_f \epsilon_U(f) + \epsilon_S(f)$, and we use $\mathcal{O}(\cdot)$ to hide the constant dependence.

Furthermore, if we switch from $\mathbb{D}_S$ to a sampled distribution Ω of $\mathbb{D}_S$ around the neighborhood of $\mathbb{D}_U$, i.e., $\Omega := \bigcup_{x \in \mathbb{D}_U} \mathcal{B}(x, r_x)$ with $r_x \leq r_\mu$ such that each $\mathcal{B}(x, r_x)$ contains exact k elements, and assume that the unseen distribution $\mathbb{D}_U$ is finite with cordiality $n_{\mathbb{D}_u}$. The risk of hypothesis $\hat{f}$ on $\mathbb{D}_U$ is then improved to

$$\epsilon_U(\hat{f}) \leq \kappa_\Omega + \epsilon_\Omega(\hat{f}) + \mathcal{O}\left(\left(\frac{2k}{c_\mu \mu_- \pi_d n_s}\right)^{1/d}\right), \quad (7)$$

with probability at least $1 - \exp(-\frac{k}{4} + \log n_{\mathbb{D}_u})$, where $\kappa_\Omega = \min_f \epsilon_U(f) + \epsilon_\Omega(f)$ and d is the feature representation dimension.

Inequality Eq.(6) is adopted from (Shen et al., 2018), which

indicates the error of a given hypothesis $\hat{f}$ bounded by three terms, the minimized combined error κ_S, the error in the source domain $\epsilon_S(\hat{f})$ and a term on the order of a constant term $\mathcal{W}(\mathbb{D}_S, \mathbb{D}_U)$. When source domain $\mathbb{D}_S$ is far away from the target domain $\mathcal{D}_U$, the $\mathcal{O}(\mathcal{W}(\mathbb{D}_S, \mathbb{D}_U))$ becomes the dominating quantity and leads to loose upper bound.

When switching to the non-parametric classifier, in Eq.(7) we replace $\mathcal{O}(\mathcal{W}(\mathbb{D}_S, \mathbb{D}_U))$ by a quantity explicitly decaying in n_s. Intuitively, by constructing Ω, we only keep the samples in the source domain with enough similarity to the target domain and it would naturally short the distance from $\mathcal{W}(\mathbb{D}_S, \mathbb{D}_U)$ to $\mathcal{W}(\Omega, \mathbb{D}_S)$. We also want to highlight that Proposition 1 should imply the non-parametric classifier may be able to take more benefits from the large pretraining source dataset (Figure 5(c)). See the Appendix for a detailed discussion of the influence of k and c_μ.

4.2. AdaNPC further reduces the target risk

In this section, we develop the target excess error bounds under the covariate-shift and the posterior-shift settings, which further articulate all factors that affect the performance of our algorithm (Proposition 2) and the benefits of using online target data (Proposition 3).

Proposition 2. *We consider η_U and η_S to be the KNN predictor in form of Eq.(5) with all w_i fixed as $\frac{1}{k}$. Per assumptions 1-3, the following results hold with high-probability when choosing $k = \mathcal{O}(\log n_s)$.*

***Under the covariate-shift setting**, we have $\eta_U = \eta_S = \eta$ for the source and target domains, and*

$$\begin{aligned}\mathcal{E}_U(\hat{f}) &\leq \mathcal{O}\left(\left(\frac{1}{k}\right)^{1/4} + \left(\frac{k}{c_\mu n_s}\right)^{1/d}\right)^{1+\beta} \\ &= \mathcal{O}\left(\left(\frac{1}{\log n_s}\right)^{1/4} + \left(\frac{\log n_s}{c_\mu n_s}\right)^{1/d}\right)^{1+\beta}.\end{aligned} \tag{8}$$

***Under the posterior-shift setting**, the regression functions η_U and η_S are different and*

$$\mathcal{E}_U(\hat{f}) \leq \mathcal{O}\left(\left(\frac{1}{\log n_s}\right)^{1/4} + \left(\frac{\log n_s}{c_\mu n_s}\right)^{1/d} + C_{ada}\right)^{1+\beta}, \tag{9}$$

where $\sup_{x_u \in \mathbb{D}_U} |\eta_S - \eta_U| \leq C_{ada}$.

We make a few remarks on the excess risk upper bound.

1. The upper bound is affected by k, c_μ, n_s, which is similar to the discussion of Proposition 1. Differently, when setting $k = \mathcal{O}(\log n_s)$, the excess error bound reduces to 0 under the covariate-shift setting when $n_s \to \infty$ in a high probability manner.

2. Proposition 2 shows a trade-off on the choice of k. Although a small k reduces the domain divergence or representing similarity $(k/c_\mu n_s)^{1/d}$, it is well known that the model will become too specific and fails to generalize well.

3. When regression functions are different, an additional term is introduced in the bound, namely the adaptivity gap $\sup_{x_u \in \mathbb{D}_U} |\eta_S(x_u) - \eta_U(x_u)|$, which measure the difference of two regression functions. The gap can be estimated and reduced by existing methods (Zhang et al., 2022d).

4. For presentation simplification, we use the equal weighted w_{uj} instead of cosine similarity between feature representations. In fact by choosing proper k, all cosine similarity values can be safely assumed lower bounded on $\mathcal{O}(k^{-\delta})$ for some $\delta > 0$. Therefore one may introduce extra assumption on the lower bound of w_{uj} to obtain a extended version of Proposition 2 with adaptive w_{uj}.

As discussed in Section 3, the proposed AdaNPC is a special kind of Test-Time adaptation method that can utilize the online target samples to improve prediction generalization. We next theoretically verify that, by incorporating the online target samples into the KNN memory bank, the excess error bound is further reduced.

Proposition 3. *Denote n_u as the number of target instances in the KNN memory bank and the KNN classifier finds the k_s nearest neighbors in D_S and k_u nearest neighbors in D_U during inference with $k_s + k_u = k$. Per Assumptions 1-3, **under the posterior-shift setting**,*

$$\begin{aligned}\mathcal{E}_U(\hat{f}) \leq \mathcal{O}\Bigg(&\left(\frac{1}{k}\right)^{1/4} + \frac{k_s}{k}\left(\frac{k_s}{c_\mu n_s}\right)^{1/d} + \\ &\frac{k_u}{k}\left(\frac{k_u}{c_\mu^* n_u}\right)^{1/d} + \frac{k_s}{k} C_{ada}\Bigg)^{1+\beta}\end{aligned} \tag{10}$$

where $\sup_{x_u \in \mathbb{D}_U} |\eta_S - \eta_U| \leq C_{ada}$ and similar results also hold under the covariate-shift setting.

We want to highlight that the above error bound is tighter than the case without updating the memory bank. The detailed discussion is deferred to section B.4 in Appendix.

5. Experiments

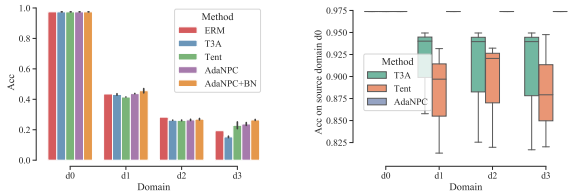
In our experiments, BN retraining and KNN loss are not used by default for fair comparisons. The model name will be AdaNPC +BN when BN retraining is used and the effect of the KNN loss is verified in the ablation study.

5.1. Experimental settings

Domain generalization benchmarks and baselines. We use five popular OOD generalization benchmark datasets: Rotated MNIST (Ghifary et al., 2015), PACS (Li et al., 2017), VLCS (Torralba & Efros, 2011), TerraIncog-

Method	RMNIST	VLCS	PACS	DomainNet	TerraIncognita	Avg
ERM (Vapnik, 1999)	97.8 ± 0.1	77.6 ± 0.3	86.7 ± 0.3	41.3 ± 0.1	53.0 ± 0.3	71.3
IRM (Arjovsky et al., 2020)	97.5 ± 0.2	76.9 ± 0.6	84.5 ± 1.1	28.0 ± 5.1	50.5 ± 0.7	67.5
GDRO (Sagawa et al., 2020)	97.9 ± 0.1	77.4 ± 0.5	87.1 ± 0.1	33.4 ± 0.3	52.4 ± 0.1	69.6
CORAL (Sun & Saenko, 2016)	98.0 ± 0.0	77.7 ± 0.2	87.1 ± 0.5	41.8 ± 0.1	52.8 ± 0.2	71.5
DANN (Ganin et al., 2016)	97.9 ± 0.1	79.7 ± 0.5	85.2 ± 0.2	38.3 ± 0.1	50.6 ± 0.4	70.3
MTL (Blanchard et al., 2021)	97.9 ± 0.1	77.7 ± 0.5	86.7 ± 0.2	40.8 ± 0.1	52.2 ± 0.4	71.1
SagNet (Nam et al., 2021)	97.9 ± 0.0	77.6 ± 0.1	86.4 ± 0.4	40.8 ± 0.2	52.5 ± 0.4	71.1
ARM (Zhang et al., 2021a)	98.1 ± 0.1	77.8 ± 0.3	85.8 ± 0.2	36.0 ± 0.2	51.2 ± 0.5	69.8
VREx (Krueger et al., 2021)	97.9 ± 0.1	78.1 ± 0.2	87.2 ± 0.6	30.1 ± 3.7	51.4 ± 0.5	68.9
Fish (Shi et al., 2022)	97.9 ± 0.1	77.8 ± 0.6	85.8 ± 0.6	**43.4 ± 0.3**	50.8 ± 0.4	71.1
Fishr (Rame et al., 2022)	97.8 ± 0.1	78.2 ± 0.2	86.9 ± 0.2	41.8 ± 0.2	53.6 ± 0.4	71.7
AdaNPC	**98.5 ± 0.1**	79.5 ± 2.4	88.8 ± 0.1	42.9 ± 0.5	**53.9 ± 0.3**	72.7
AdaNPC +BN	98.4 ± 0.1	**80.2 ± 0.2**	**88.9 ± 0.1**	43.1 ± 0.8	**54.0 ± 0.1**	**72.9**

Table 1. Out-of-distribution generalization performance.



(a) (b)

Figure 3. **Successive adaptation results on the TerraIncognita dataset.** (a) Adaptation results on d_1, d_2, d_3. (b) Re-evaluation of the adapted model on the source domain d_0.

nita (Beery et al., 2018) and DomainNet (Peng et al., 2019). We compare our model with ERM (Vapnik, 1999), IRM (Arjovsky et al., 2020), Mixup (Yan et al., 2020), CORAL (Sun & Saenko, 2016), DANN (Ganin et al., 2016), CDANN (Li et al., 2018c), MTL (Blanchard et al., 2021), SagNet (Nam et al., 2021), ARM (Zhang et al., 2021a), VREx (Krueger et al., 2021), RSC (Huang et al., 2020), Fish (Shi et al., 2022), Fishr (Rame et al., 2022). All experimental settings and baselines follow the Domainbed codebase (Gulrajani & Lopez-Paz, 2021). The comparison of AdaNPC with other TTA methods is detailed in the appendix.

See Appendix D for more information, including datasets information, model selection, licensing information, hyperparameter search, and the total amount of computing. See Appendix E for more experimental results and analysis.

5.2. Experimental Results

AdaNPC has strong knowledge expandability. For practical usage, a deployed model should be adapted to a series of domains, where domain partition is unknown and we should guarantee that the model performs well on both any unseen target sample and samples from the training domains. In this case, we propose a setting named *Successive adaptation* that is more practical. As shown in Figure 4, a model (model 0) that is trained on domain d_0 will be adapted to a series of domains. Specifically, model i will be adapted and evaluated on a domain d_{i+1}. The results of successive adaptation results on the Rotated MNIST and TerraIncognita are shown in Figure 1(b) and Figure 3(a). Results show that the latest TTA method, namely T3A (Iwasawa & Matsuo, 2021) and Tent (Wang et al., 2020a) perform marginally above or even worse than the ERM baseline where no test-time adaptation is performed. In contrast, the proposed AdaNPC is superior to these methods, especially in the Rotated MNIST dataset d_5, where the performance gains brought by AdaNPC compared to ERM are greater than 50%. The reason will be that the domain indexes in the Rotated MNIST are continuously changed, namely, the rotation angles are changed from 0° to 75°, which makes the knowledge transfer easier. The continuously changed domain index is also general in real world tasks (Wang et al., 2020b), for example, in medical applications, one needs to adapt disease diagnosis and prognosis across patients of different ages, where the age is continuously changed between domains, verifying the practical utility of AdaNPC.

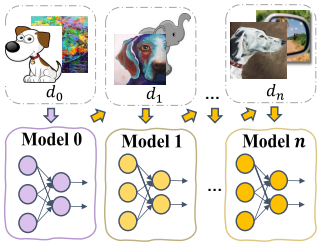


Figure 4. An illustration of successive adaptation setting.

AdaNPC overcomes catastrophic forgetting and reserves superior source performance that is even better than ERM. Figure 2 and Figure 3(b) visualize the source accuracy of model i, where the x-coordinate d_i means the model i is evaluated on the source d_0. Both T3A (Iwasawa & Matsuo, 2021) and Tent (Wang et al., 2020a) forget the knowledge of the source domains as they adapt to more target domains. That is, if these TTA methods are used for a deployed model after a series of adaptations, we cannot expect that the model will still give a correct prediction on the source domain instances. The tradeoff between adaptivity and source domain accuracy is nonexistent for the proposed

Method	A	C	P	S	Avg
ResNet18	79.5	73.0	90.1	77.3	80.0
AdaNPC	82.7	76.8	92.8	77.7	82.5
AdaNPC +BN	83.6	77.7	93.1	77.9	83.1
ResNet50	81.7	82.3	94.3	77.6	84.0
AdaNPC	86.0	81.0	96.4	79.8	85.8
AdaNPC +BN	85.6	80.8	96.5	79.7	85.7
ViT-B16	88.3	82.4	97.9	79.8	87.1
AdaNPC	89.8	86.0	97.9	80.5	88.6
AdaNPC +BN	89.8	86.1	98.3	80.5	88.7

Table 2. OOD accuracy with different backbones on PACS.

AdaNPC, which performs both tasks well. Dur to space limit, we leave the performance over other dataset in the appendix E.2.

AdaNPC achieves a new State-Of-The-Arts on domain generalization benchmarks. The average OOD results on all benchmarks are shown in Table 1. We observe consistent improvements achieved by AdaNPC compared to existing algorithms and **BN retraining** can further boost the generalization capability with few parameter updating. Compared to the advanced DG method Fishr (Rame et al., 2022), which achieves 0.4 higher average accuracy than ERM, the proposed AdaNPC attains a much larger margin (1.4). The results indicate the superiority of AdaNPC in real-world diversity shift datasets.

AdaNPC improves different backbones by a large margin. We conduct experiments with various backbones in Table 2, including ResNet50, ResNet18, and Vision Transformers (ViT-B16). AdaNPC achieves consistent performance improvements compared to ERM. Specifically, AdaNPC improves 2.5%, 1.8%, 2.6%, and 1.5% for ResNet-18, ResNet50, ResNet50-BN, and ViT-B16, respectively.

Without any model fine-tuning on source domains, AdaNPC can perform well. Figure 5(a) shows the results where the pre-trained model is directly evaluated on the target domain and without any fine-tuning on the source domains. The average generalization performance of using an MLP classifier is below 25% even with a strong backbone (ViT-L16). On the contrary, the use of a KNN classifier achieves average generalization accuracy 71.4%. Nowadays, fine-tuning is usually computationally expensive due to the ever-growing size of pre-trained models. The requirement of AdaNPC is not a gradient-based update but external high capacity storage to store knowledge for image classification, such as image feature maps, which provides a new promising direction to utilize pre-trained knowledge. Furthermore, as the number of source instances increases, Figure 5(c) shows that AdaNPC achieves better performance, which validates our theoretical result.

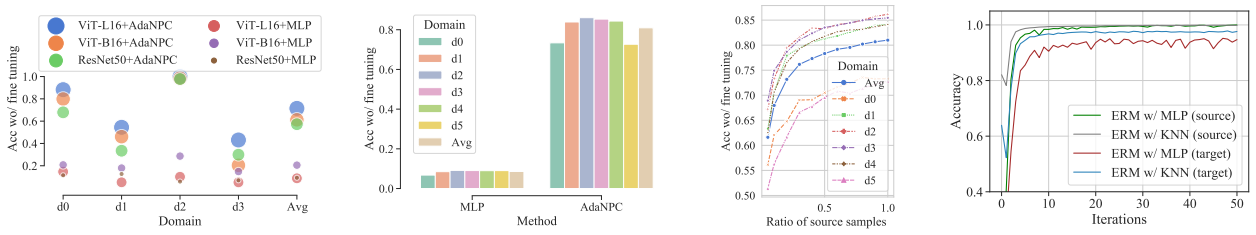
AdaNPC reduces generalization error across various corruption types. To verify robustness to corruptions, we evaluate AdaNPC on the CIFAR-10-C benchmark (Hendrycks & Dietterich, 2018) with a 40-2 Wide ResNet backbone (Zagoruyko & Komodakis, 2016) pre-trained on CIFAR-10. We implement two baselines, where **Linear** denotes that the trained classifier is used directly, and **BN Adapt** will update the batch normalization parameters of the backbone in the target data (Schneider et al., 2020). The results with the highest severity (five) are shown in Figure 6, where AdaNPC is shown to be more robust than Linear and updating the BN statistic of the backbone with AdaNPC brings more benefits than that with a linear classifier. Results with the lowest severity (one), other backbones. **Considering large-scale corruption benchmarks**, we compared the method with Tent and EATA (Niu et al., 2022) on the ImageNet-C dataset. Experimentally, we compared the effectiveness of AdaNPC and EATA separately in Table. 3, where the experimental and baselines are all following EATA (Niu et al., 2022). As we can see, we only need to simply replace the Linear layer with KNN and remember the trustworthy samples during testing, and AdaNPC can achieve much better performance than ResNet50 with a linear head. In addition, AdaNPC can be combined with existing TTA methods such as Tent, ETA, etc., and the final performance surpasses these methods individually. See more analysis in the Appendix E.3.

	Defoc	Glass	Motion	Zoom
R-50 (GN)+JT	88.9	91.7	86.7	81.6
R-50 (BN)	82.1	90.2	85.2	77.5
AdaNPC	83.1	83.0	72.3	60.6
TTT	71.9	92.2	66.8	63.2
TTA	87.5	91.8	87.1	74.2
BN adaptation	80.0	80.0	71.5	60.0
MEMO	80.3	87.0	79.3	72.4
Tent	71.8	72.7	58.6	50.5
Tent (episodic)	85.5	85.4	74.6	62.2
AdaNPC+Tent	71.1	72.0	58.2	49.2
ETA	66.1	67.1	52.2	47.5
EATA	66.3	66.6	52.9	47.2
AdaNPC+ETA	65.2	65.2	51.1	46.5

Table 3. **Comparison with state-of-the-art methods on ImageNet-C** with the highest severity level 5 regarding corruption Error.

5.3. Comparison of AdaNPC with other TTA methods.

Comparison with test-time adaptation methods. For fair comparisons, following (Iwasawa & Matsuo, 2021), the base models (ERM and AdaNPC) are trained only on the default hyperparameters and without the fine-grained parametric search. Because (Gulrajani & Lopez-Paz, 2021)



(a) (b) (c) (d)

Figure 5. **Generalization performance without fine-tuning** with various backbones on the (a) PACS and RotatedMNIST (b) dataset. (c) shows the results wo/ fine-tuning on Rotated MNIST with different ratios of source instances amounts. (d) Convergence comparison.

omits the BN layer from pre-trained ResNet when fine-tuning on source domains, we cannot simply use BN-based methods on the ERM baseline. For these methods, their baselines are additionally trained on ResNet-50 with BN. Models with the highest IID accuracy are selected and all test-time adaptation methods are applied to improve the generalization performance. The baselines include Tent (Wang et al., 2020a), T3A (Iwasawa & Matsuo, 2021), pseudo labeling (PL) (Lee et al., 2013), SHOT (Liang et al., 2020), and SHOT-IM (Liang et al., 2020). For methods that use gradient backpropagation, we implement both update the prediction head (Clf) and full model (Full). Results in Table 4 show that: Different from Tent (Wang et al., 2020a), which is sensitive to batch size, the proposed AdaNPC is not; (ii) The performance of AdaNPC without BN retraining attains comparable results compared to existing methods. (iv) Additionally, when the batch size is very small, updating the model parameters often has a negative impact, whereas the results of AdaNPC are not affected by the batch size.

5.4. Analysis

AdaNPC attains better performance with fewer iterations. We investigate the accuracy dynamics of ERM, which is evaluated by using either a KNN classifier or a Multi-Layer Perception (MLP) classifier on the Rotated MNIST dataset, where the target domain is d_0. The learning curves in Fig. 5(d) show that with the same training process and iteration, using a KNN classifier can attain superior performance on the source and target domains both.

Ablation studies of training loss (Eq.(4)) and the choice of k. Results are shown in Table. 12, where **ERM** means that the model will be trained by cross-entropy loss and only use KNN with parameter k for inference and adaptation. The results show that with $\mathcal{L}_{KNN}$, the representation will be better and the generalization results will be improved. The choice of $|\mathcal{M}|$ depends on the dataset size, a large dataset with more classes generally needs a large memory bank. However, if $|\mathcal{M}| \geq batch\ size$, $\mathcal{M}$ contains many out-of-date features and leads to poor performance. The choice of k depends both on the domain divergence and instances

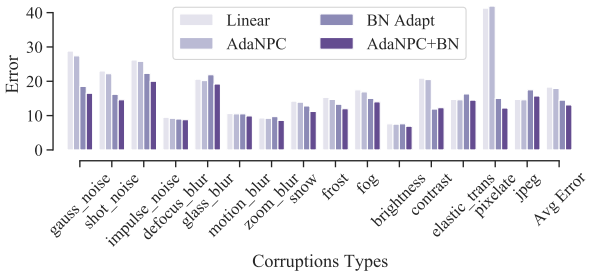


Figure 6. Corruption benchmark on CIFAR-10-C with severity 5.

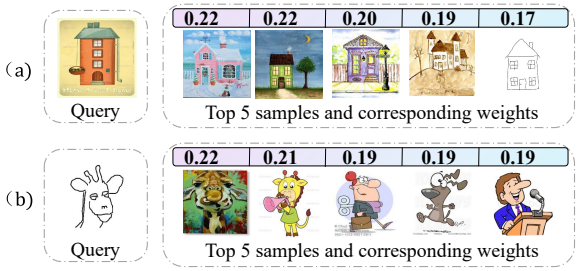


Figure 7. **Visualization of classified results attained by AdaNPC.** (a) A successfully classified instance and (b) a failure case.

numbers of datasets. For example, the best k is around 10 for RMNIST and 50 for PACS (PACS has a greater domain divergence), where PACS has $9,991$ images, which is less than $60,000$ images of RMNIST.

Interpretability and human-model interaction. Figure 7 shows how the source knowledge is used by the KNN classifier. The decision-making process will no longer be a black box. For example, the giraffe in Figure 7(b) is classified with low confidence because its nearest neighbors are most persons or dogs that have similar poses. However, some important characteristics are ignored by the backbone representations, such as the shape of the face. However, these characteristics can be easily identified by humans; therefore, when we get a low confidence prediction, AdaNPC allows us to manually remove some obvious wrong neighbors. In this case, our classification results will be more accurate and

Method	BSZ=32				BSZ=2			
	RMNIST	PACS	VLCS	DomainNet	RMNIST	PACS	VLCS	DomainNet
ResNet50	97.27	86.68	77.75	40.50	97.27	86.68	77.75	40.50
PLClf	98.13	87.73	80.75	40.80	69.78	87.73	80.50	40.67
PLFull	98.30	87.13	77.23	33.80	65.37	73.35	58.55	21.80
SHOT	98.30	88.16	67.83	41.21	92.12	83.55	58.45	36.76
SHOTIM	98.40	88.03	67.68	37.62	92.33	83.20	58.20	35.43
T3A	97.65	87.90	81.38	41.50	97.10	87.90	81.38	41.50
Tent-clf	96.82	87.05	77.78	40.97	96.18	87.03	77.50	40.16
AdaNPC	98.85	88.93	82.45	42.60	98.85	88.93	82.45	42.60

Table 4. Comparison of our method and existing test-time adaptation methods on OOD benchmarks. The reported number is the average generalization performance over all domains.

PACS, 'Test-domain' validation							
Training Loss	k	$\lvert\mathcal{M}\rvert$	**A**	**C**	**P**	**S**	**Avg**
ERM	Linear	/	82.7	82.3	93.1	78.6	84.2
	50	/	83.1	82.1	93.6	78.9	84.4
	75	/	83.2	82.1	93.5	78.8	84.4
	100	/	83.2	81.9	93.5	78.7	84.3
	125	/	84.6	82.3	93.7	81.0	85.4
	150	/	85.2	82.6	94.2	80.4	85.6
$\mathcal{L}_{KNN}$	10	500	84.0	82.6	93.5	80.8	85.2
	50	500	85.2	81.7	92.6	80.8	85.1
	50	1000	**85.8**	81.5	93.8	**83.4**	**86.1**
	50	1500	82.9	**83.0**	91.4	80.5	84.5
	100	1000	85.0	82.5	**94.6**	80.7	85.7

Table 5. Ablation studies of training loss on PACS.

Inference time (ms)					
Method	**RMNIST**	**PACS**	**VLCS**	**DomainNet**	**ImageNet**
PLClf	0.49	5.38	5.57	6.17	5.55
PLFull	123.44	281.90	266.00	267.17	270.68
SHOT	0.66	7.34	7.12	6.96	7.45
SHOTIM	1.58	10.14	10.16	7.86	10.23
T3A	0.21	1.67	1.51	8.11	18.46
Tent-clf	11.24	41.63	27.60	92.37	32.34
AdaNPC	0.13	1.57	1.57	3.16	22.23

Table 6. **Inference time of different TTA methods**, where the AdaNPC is implemented by Faiss (Johnson et al., 2019).

confident, which is promising for high-risk tasks to incorporate expert knowledge for better classification results.

Inference time and Memory usage. During testing, we store all samples from the source domain and do not delete any samples from the memory bank, thus preserving all information from the source domain. This may raise concerns about memory usage and inference time, but our experiments show that the K-nearest neighbor search is very fast. Furthermore, we provide a comparison of inference times for various TTA algorithms on different datasets in Table 6, measured as the average inference time per image in milliseconds. For memory usage. Even for a dataset like ImageNet with 1281167 images, the additional memory cost is only 2.44GB (1281167*2048 B), which can be easily accommodated by current CPUs or memory. It is important to note that the memory bank does not have to be stored on the GPU, and all reported inference times in this paper are based on a CPU-stored memory bank.

6. Concluding Remarks

The paper proposes a new Test-Time adaption method for domain generalization, AdaNPC, which revisits a non-parametric classifier, namely KNN classifier, for prediction and adaptation. Unlike current domain generalization or Test-Time adaptive methods that need model updating and are easy to forget previous knowledge, the proposed method is parameter-free and can memorize all the knowledge, making AdaNPC suitable for practical settings, especially for adapting to a series of domains.

We derive error bounds under both the covariate-shift and the posterior-shift settings, where AdaNPC is theoretically shown able to reduce unseen target error. We empirically show that AdaNPC reduces generalization error on both unseen target domains and corrupted data. Besides, AdaNPC attains faster convergence, better interpretability, and strong knowledge expandability. More importantly, AdaNPC achieves high generalization accuracy without any fine-tuning on source domains, which provides a promising direction for utilizing pretrained models with growing sizes.

Discussion and Limitations. One potential limitation of AdaNPC will be the **computation time of dense vectors' searching** (finding the k-nearest neighbors). However, with efficient nearest neighbor search technique (Johnson et al., 2019), even when the memory bank contains more than 1 million samples, the inference time will around 10ms (Sun et al., 2022), which is significantly faster than backward gradient updates. In the future, we will consider updating the memory bank with advanced methods and try to further reduce the memory cost and inference time.

Acknowledgements

This work was partially funded by the National Key R&D Program of China (2022ZD0117901), and National Natural Science Foundation of China (62236010, and 62141608).

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International conference on machine learning*. PMLR, 2017.

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *stat*, 1050:27, 2020.

Audibert, J.-Y. and Tsybakov, A. B. Fast learning rates for plug-in classifiers. *The Annals of statistics*, 2007.

Beery, S., Van Horn, G., and Perona, P. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, 2018.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 2006.

Blanchard, G., Deshmukh, A. A., Dogan, Ü., Lee, G., and Scott, C. Domain generalization by marginal transfer learning. *J. Mach. Learn. Res.*, 2021.

Cai, T. T. and Wei, H. Transfer learning for nonparametric classification: Minimax rate and adaptive classifier. *The Annals of Statistics*, 2021.

Chernoff, H. A note on an inequality involving the normal distribution. *The Annals of Probability*, 1981.

Chung, F. and Lu, L. Concentration inequalities and martingale inequalities: a survey. *Internet mathematics*, 2006.

David, S. B., Lu, T., Luu, T., and Pál, D. Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 129–136. JMLR Workshop and Conference Proceedings, 2010.

Dubey, A., Ramanathan, V., Pentland, A., and Mahajan, D. Adaptive methods for real-world domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Dudani, S. A. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 1976.

Feng, Y., Jiang, J., Tang, M., Jin, R., and Gao, Y. Rethinking supervised pre-training for better downstream transferring. *ICLR*, 2022.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The journal of machine learning research*, 2016.

Ghifary, M., Kleijn, W. B., Zhang, M., and Balduzzi, D. Domain generalization for object recognition with multi-task autoencoders. In *ICCV*, 2015.

Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization. In *ICLR*, 2021.

Hanneke, S. and Kpotufe, S. On the value of target data in transfer learning. *Advances in Neural Information Processing Systems*, 2019.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

Huang, Z., Wang, H., Xing, E. P., and Huang, D. Self-challenging improves cross-domain generalization. In *ECCV*, 2020.

Iwasawa, Y. and Matsuo, Y. Test-time classifier adjustment module for model-agnostic domain generalization. *Advances in Neural Information Processing Systems*, 2021.

Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 2019.

Kirichenko, P., Izmailov, P., and Wilson, A. G. Last layer re-training is sufficient for robustness to spurious correlations. *arXiv preprint arXiv:2204.02937*, 2022.

Kpotufe, S. and Martinet, G. Marginal singularity, and the benefits of labels in covariate-shift. In *Conference On Learning Theory*, pp. 1882–1886. PMLR, 2018.

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Le Priol, R., and Courville, A. Out-of-distribution generalization via risk extrapolation (rex). In *ICML*, 2021.

Lee, D.-H. et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, pp. 896, 2013.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Deeper, broader and artier domain generalization. In *ICCV*, 2017.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, 2018a.

Li, H., Pan, S. J., Wang, S., and Kot, A. C. Domain generalization with adversarial feature learning. In *CVPR*, 2018b.

Li, Y., Tian, X., Gong, M., Liu, Y., Liu, T., Zhang, K., and Tao, D. Deep domain generalization via conditional invariant adversarial networks. In *ECCV*, 2018c.

Liang, J., Hu, D., and Feng, J. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pp. 6028–6039. PMLR, 2020.

Liang, J., He, R., and Tan, T. A comprehensive survey on test-time adaptation under distribution shifts. *arXiv preprint arXiv:2303.15361*, 2023.

Mansour, Y., Mohri, M., and Rostamizadeh, A. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009.

Mohri, M. and Muñoz Medina, A. New analysis and algorithm for learning with drifting distributions. In *International Conference on Algorithmic Learning Theory*, pp. 124–138. Springer, 2012.

Nam, H., Lee, H., Park, J., Yoon, W., and Yoo, D. Reducing domain gap by reducing style bias. In *CVPR*, 2021.

Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., and Tan, M. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*. PMLR, 2022.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. Moment matching for multi-source domain adaptation. In *ICCV*, 2019.

Quinonero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. *Dataset shift in machine learning*. Mit Press, 2008.

Rame, A., Dancette, C., and Cord, M. Fishr: Invariant gradient variances for out-of-distribution generalization. In *International Conference on Machine Learning*. PMLR, 2022.

Reeve, H. W., Cannings, T. I., and Samworth, R. J. Adaptive transfer learning. *The Annals of Statistics*, 2021.

Rosenfeld, E., Ravikumar, P., and Risteski, A. Domain-adjusted regression or: Erm may already learn features sufficient for out-of-distribution generalization. *arXiv preprint arXiv:2202.06856*, 2022.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International conference on learning representations (ICLR)*, 2020.

Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., and Bethge, M. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33: 11539–11551, 2020.

Shen, J., Qu, Y., Zhang, W., and Yu, Y. Wasserstein distance guided representation learning for domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Shi, Y., Seely, J., Torr, P., N, S., Hannun, A., Usunier, N., and Synnaeve, G. Gradient matching for domain generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vDwBW49HmO.

Sugiyama, M., Suzuki, T., and Kanamori, T. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

Sun, B. and Saenko, K. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, 2016.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., and Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

Sun, Y., Ming, Y., Zhu, X., and Li, Y. Out-of-distribution detection with deep nearest neighbors. *ICML*, 2022.

Torralba, A. and Efros, A. A. Unbiased look at dataset bias. In *CVPR*, 2011.

Tsybakov, A. B. Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 2004.

Vapnik, V. *The nature of statistical learning theory*. Springer science & business media, 1999.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020a.

Wang, H., He, H., and Katabi, D. Continuously indexed domain adaptation. In *ICML*, 2020b.

Xiao, Z., Zhen, X., Shao, L., and Snoek, C. G. Learning to generalize across domains on single test samples. In *International Conference on Learning Representations*, 2022.

Yan, S., Song, H., Li, N., Zou, L., and Ren, L. Improve unsupervised domain adaptation with mixup training. *arXiv preprint arXiv:2001.00677*, 2020.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

Zhang, H., Zhang, Y.-F., Liu, W., Weller, A., Schölkopf, B., and Xing, E. P. Towards principled disentanglement for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8024–8034, 2022a.

Zhang, M., Marklund, H., Dhawan, N., Gupta, A., Levine, S., and Finn, C. Adaptive risk minimization: Learning to adapt to domain shift. *NeurIPS*, 2021a.

Zhang, M., Levine, S., and Finn, C. Memo: Test time robustness via adaptation and augmentation. *Advances in Neural Information Processing Systems*, 2022b.

Zhang, Y., Hooi, B., Hong, L., and Feng, J. Test-agnostic long-tailed recognition by test-time aggregating diverse experts with self-supervision. *arXiv preprint arXiv:2107.09249*, 2021b.

Zhang, Y., Zhang, H., Lipton, Z. C., Li, L. E., and Xing, E. Exploring transformer backbones for heterogeneous treatment effect estimation. In *NeurIPS ML Safety Workshop*, 2022c.

Zhang, Y., Liang, J., Zhang, Z., Wang, L., Jin, R., Tan, T., et al. Free lunch for domain adversarial training: Environment label smoothing. In *ICLR*, 2023.

Zhang, Y.-F., Wang, J., Zhang, Z., Yu, B., Wang, L., Tao, D., and Xie, X. Domain-specific risk minimization. *arXiv preprint arXiv:2208.08661*, 2022d.

Zhang, Y.-F., Zhang, Z., Li, D., Jia, Z., Wang, L., and Tan, T. Learning domain invariant representations for generalizable person re-identification. *IEEE Transactions on Image Processing*, 2022e.

Zhao, H., Des Combes, R. T., Zhang, K., and Gordon, G. On learning invariant representations for domain adaptation. In *ICML*. PMLR, 2019.

AdaNPC: Exploring Non-Parametric Classifier for Test-Time Adaptation

————Appendix————

The structure of Appendix is as follows

- Appendix A contains the extended related work.
- Appendix B contains all missing proofs in the main manuscript.
- Appendix C details the optimization algorithm of the proposed AdaNPC.
- Appendix D details the dataset and implementation Details.
- Appendix E contains additional experimental results.

A. Extended Related work

Transfer learning theory. The first line of work that considers bounding the error on target domains by the source domain classification error and a divergence measure, such as $d_{\mathcal{A}}$ divergence (Ben-David et al., 2006; David et al., 2010) and $\mathcal{Y}$ divergence (Mansour et al., 2009; Mohri & Muñoz Medina, 2012). However, the symmetric differences carry the wrong intuition and most of these bounds depend on the choice of hypothesis (Kpotufe & Martinet, 2018; Zhang et al., 2022d). There are also some studies consider the density ratio between the source and target domain (Quinonero-Candela et al., 2008; Sugiyama et al., 2012; Zhang et al., 2022d), and transfer-exponent for non-parametric transfer learning (Kpotufe & Martinet, 2018; Cai & Wei, 2021; Hanneke & Kpotufe, 2019; Reeve et al., 2021). In this work, we conduct error bounds considering a setting that consists of two different components, namely, the non-parametric classifier and online arrival target samples.

Domain generalization. Previous DG methods mostly focus on representation learning, namely learning domain-invariant representations or only task-relevant representations. However, empirical risk minimization (ERM) has been shown to be able to beat most existing domain generalization methods in average performance (Gulrajani & Lopez-Paz, 2021). Recent work finds that ERM has learned a high-quality representation on datasets with spurious correlations, even when the model relies primarily on spurious features to make predictions (Kirichenko et al., 2022). The current bottleneck to out-of-distribution generalization primarily lies in learning simple and reliable classifiers (Rosenfeld et al., 2022). However, most existing methods have an over-confidence hypothesis space (Zhang et al., 2022d), namely, they assume that the hypothesis performs well on source domains can also perform well on the target domain (Arjovsky et al., 2020; Krueger et al., 2021; Rame et al., 2022; Zhang et al., 2023; 2022e), which cannot hold on any unseen target domains. Our method is orthogonal to most existing DG methods since it replaces the linear classifier with a KNN classifier. AdaNPC retains all information seen in training and has a complex hypothesis space controlled by the parameter k. During inference, *the hypothesis space complicity can be flexibly controlled compared to existing methods that use a frozen classifier*.

Test-Time Adaptative Methods					
	Target Batch	Source Training	Fine-tune	Extra Model	Adaptive
Native DG (Arjovsky et al., 2020)	×	×	×	×	×
Test-Time Training (Sun et al., 2020; Zhang et al., 2021b; 2022b)	×	×	✓	✓	✓
Test-Time Adaptation (Wang et al., 2020a)	✓	×	✓	×	✓
Domain-adaptative method (Dubey et al., 2021; Zhang et al., 2022d)	✓	×	×	✓	✓
Single sample generalization (Xiao et al., 2022)	×	✓	×	×	✓
Non-Parametric Adaptation	×	×	×	×	✓

Table 7. **Test-Time adaptive methods**. The target batch means that the methods need batches of target samples for adaptation. Compared to existing test-time adaptative methods, the proposed AdaNPC imposes no additional parameter (Sun et al., 2020; Dubey et al., 2021; Zhang et al., 2022d), no extra tunning steps (Sun et al., 2020; Wang et al., 2020a; Iwasawa & Matsuo, 2021; Zhang et al., 2021b), and does not need to use source data to learn adaptive strategies (Xiao et al., 2022).

B. Proof of Theoretical Statement

B.1. Non-parametric reduce target-source domain divergence (Proof of Proposition 1)

To complete the proofs, we begin by introducing some necessary definitions and assumptions.

Definition 3. *(**Wasserstein-distance and the dual form** (Arjovsky et al., 2017)). The ρ-th Wasserstein distance between two distributions $\mathbb{D}_S, \mathbb{D}_U$ is defined as*

$$\mathcal{W}_\rho(\mathbb{D}_S, \mathbb{D}_U) = \left(\inf_{\gamma \in \Pi[\mathbb{D}_S, \mathbb{D}_U]} \iint d(x_s, x_u)^\rho d\gamma(x_s, x_u) \right)^{1/\rho} \tag{11}$$

where $\Pi[\mathbb{D}_S, \mathbb{D}_U]$ is the set of all joint distribution on $\mathcal{X} \times \mathcal{X}$ with marginals $\mathbb{D}_S$ and $\mathbb{D}_U$ and $d(x_s, x_u)$ is a distance function for two instances x_s, x_u.

Wasserstein distance can get intuition from the optimal transport problem, where $d(x_s, x_u)^\rho$ is the unit cost for transporting a unit of material from $x_s \in \mathbb{D}_S$ to $x_u \in \mathbb{D}_U$ and $\gamma(x_s, x_u)$ is the transport policy which satisfies the marginal constraint. According to the Kantorovich-Rubinstein theorem, the dual representation of the first Wasserstein distance (Earth-Mover distance) can be written as

$$\mathcal{W}_1(\mathbb{D}_S, \mathbb{D}_U) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x_s \in \mathbb{D}_S}[f(x_s)] - \mathbb{E}_{x_u \in \mathbb{D}_U}[f(x_u)], \tag{12}$$

where $\| f \|_L = \sup |f(x_s) - f(x_u)| / d(x_s, x_u)$ is the Lipschitz semi-norm.

We first use the domain adaptation result, Theorem 1 in (Shen et al., 2018) that considers the Wasserstein distance. On this basis, we can clearly show the effect of AdaNPC on the domain divergence. In this paper, we use $\mathcal{W}_1(\mathbb{D}_S, \mathbb{D}_U)$ as default and ignore subscript 1. For completeness, we present the Theorem 1 in (Shen et al., 2018) as follow:

Proposition 4. *(Theorem 1 in (Shen et al., 2018)) Given two domain distributions $\mathbb{D}_S, \mathbb{D}_U$, denote $f^* = \arg\min_{f \in \mathcal{H}}(\epsilon_U(f) + \epsilon_S(f))$ and $\kappa = \epsilon_U(f^*) + \epsilon_S(f^*)$. Assume all hypotheses h are L-Lipschitz continuous, the risk of hypothesis $\hat{f}$ on the unseen target domain is then bounded by*

$$\epsilon_U(\hat{f}) \leq \kappa + \epsilon_S(\hat{f}) + 2L\mathcal{W}(\mathbb{D}_S, \mathbb{D}_U). \tag{13}$$

Intuitively, by using the non-parametric classifier, during inference, a large number of samples in source domains that are not similar to the target samples are ignored, and thus the domain divergence will be reduced. That is, the source distribution $\mathbb{D}_S$ is replaced by $\Omega := \bigcup_{x \in \mathbb{D}_U} \mathcal{B}(x, r)$, where $\mathcal{B}(x, r) = \{x' : \| x' - x \| \leq r\}$ denotes a ball centered on x with radius r, and With a small r, Ω is intuitively close to $\mathbb{D}_U$ because these dissimilar data points are ignored and the selected source data are all close to the target data. Informally, according to Eq.(11), we have $\mathcal{W}(\Omega, \mathbb{D}_U) = \inf_{\gamma \in \Pi[\Omega, \mathbb{D}_U]} \iint \| x_s - x_u \| d\gamma(x_s, x_u)$, where for each $x_s \in \Omega$, we can find at least one $x_u \in \mathbb{D}_U$ such that $\| x_s - x_u \| \leq r$, the overall distance will then be bounded by r. If r is small enough, $\mathcal{W}(\mathbb{D}_S, \mathbb{D}_U)$ in Proposition 4 is largely reduced. Specifically, we can choose a density function γ^* where $\gamma^*(x_s, x_u) > 0$ only if $x_s \in \mathcal{B}(x_u, r)$ otherwise 0, then we have

$$\mathcal{W}(\Omega, \mathbb{D}_U) = \inf_{\gamma \in \Pi[\Omega, \mathbb{D}_U]} \iint \| x_s - x_u \| d\gamma(x_s, x_u) \leq \iint \| x_s - x_u \| \gamma^*(x_s, x_u) dx_s x_u \leq r \tag{14}$$

Although a small r will reduce the generalization bound, there is no guarantee that each data $x_u \in \mathbb{D}_U$ can find a neighbor $\mathcal{B}(x, r)$ with $|\mathcal{B}(x, r)| > 0$. To this end, we theoretically discuss the choice of r and show *given a choice radius r, what probability that the set of neighbors $\mathcal{B}(x, r)$ of each $x \in \mathbb{D}_U$ is not measuring zero*?

We denote k is the number of neighbors that we prefer to choose, namely the parameter for the KNN classifier, n_s is the total number of data in D_S. With the strong density assumption, for any $x_u \in \mathbb{D}_U, r < r_\mu$, according to Assumption 1, we have

$$\mathbb{D}_S(x_s \in \mathcal{B}(x_u, r)) = \int_{\mathcal{B}(x_u, r) \cap \mathbb{D}_S} \frac{d\mathbb{D}_S}{d\lambda}(x_s) dx_s \geq \mu_- \lambda(\mathcal{B}(x_u, r) \cap \mathbb{D}_S) \geq c_\mu \mu_- \pi_d r^d, \tag{15}$$

where $\pi_d = \lambda(\mathcal{B}(0,1))$ is the volume of the d dimension unit ball and λ is the Lebesgue measure of a set in a Euclidean space. Set $r_0 = (\frac{2k}{c_\mu \mu_- \pi_d n_s})^{1/d}$, with a additional assumption that $\frac{k}{n_s} < \frac{c_\mu \mu_- \pi_d r_\mu^d}{2}$[1], we have $r_0 < r_\mu$. Then for any $x_u \in \mathbb{D}_U$, according to Eq.(15), we have

$$\mathbb{D}_S(x_s \in \mathcal{B}(x_u, r_0)) \geq c_\mu \mu_- \pi_d r_0^d > \frac{2k}{n_s} \tag{16}$$

Denote $\mathbb{I}$ an indicator function and then $\mathbb{I}(x_s \in \mathcal{B}(x_u, r_0))$ are independent and identically Bernuoli variables, which mean is $\mathbb{D}_S(x_s \in \mathcal{B}(x_u, r_0))$. Let $S_n(x_u) = \sum_{i=1}^{n_s} \mathbb{I}(x_s \in \mathcal{B}(x_u, r_0))$ denote the number of data $x_s \in \mathbb{D}_S$ that fall into $\mathcal{B}(x, r_0)$, then $S_n(x_u)$ follows the Binomial distribution. Let $W \sim Binomial(n_s, \frac{2k}{n_s})$, according to the Chernoff inequality (Chernoff, 1981; Chung & Lu, 2006),

$$P(S_n(x_u) < k) \leq P(W < k) = P(W - \mathbb{E}[W] < -k) \leq \exp(-k^2/2\mathbb{E}[W]) = \exp(-k/4), \tag{17}$$

where the second inequality is because $S_n(x)$ has a larger mean than W. We can see the probability that $S_n(x) < k$ is small for any $x_u \in \mathbb{D}_U$, especially when k is large. Denoting $x_s^{(i)}$ the $i-$th nearest data to x_u among $\mathcal{B}(x_u, r_0)$, we have for any $x_u \in \mathbb{D}_U$

$$P(\| x_s^{(k)} - x_u \| \leq r_0) = P(S_n(x) \geq k) \geq 1 - \exp(-k/4) \tag{18}$$

Combine Eq.(18) with the assumption that the distribution $\mathbb{D}_U$ is finite with cardinality $n_{\mathbb{D}_U}$ and the desired probability part is shown by union bound.

$$\begin{aligned}
\bigcap_{x_u \in \mathbb{D}_U} P(\| x_s^{(k)} - x_u \| \leq r_0) &= \bigcap_{x_u \in \mathbb{D}_U} P(S_n(x) \geq k) \\
&= 1 - \bigcup_{x_u \in \mathbb{D}_U} P(S_n(x) < k) \\
&\geq 1 - n_{\mathbb{D}_U} \exp\left(-\frac{k}{4}\right) \\
&= 1 - \exp\left(-\frac{k}{4} + \log n_{\mathbb{D}_U}\right).
\end{aligned} \tag{19}$$

Finally, the following proposition is derived.

Proposition 5. *Given two domain distributions $\mathbb{D}_S, \mathbb{D}_U$, and $\Omega := \bigcup_{x \in \mathbb{D}_U} \mathcal{B}(x, r)$, where $\mathcal{B}(x, r) = \{x' : \| x' - x \| \leq r\}$ denotes a ball centered on x with radius r. Denote $f^* = \arg\min_{f \in \mathcal{H}} (\epsilon_U(f) + \epsilon_\Omega(f))$ and $\kappa = \epsilon_U(f^*) + \epsilon_\Omega(f^*)$. Assume all hypotheses h are L-Lipschitz continuous, the risk of hypothesis $\hat{f}$ on the unseen target domain is then bounded by*

$$\epsilon_U(\hat{f}) \leq \kappa + \epsilon_\Omega(\hat{f}) + 2L \left(\frac{2k}{c_\mu \mu_- \pi_d n_s}\right)^{1/d}. \tag{20}$$

with probability $1 - \exp(-\frac{k}{4} + \log n_{\mathbb{D}_U})$

Remarks. We make the following conclusions (i) with a larger number of source data, the error will be lower; (ii) a large c_μ will reduce the error bound, which is intuitive because $x_u \in \mathbb{D}_U$ will not be so far from $\mathbb{D}_S$ when c_μ is large and the adaptation will be easier; (iii) a smaller parameter k will reduce the domain divergence $\left(\frac{2k}{c_\mu \mu_- \pi_d n_s}\right)^{1/d}$. For example, when $k = 1$, we only choose the closest source data with respect to x_u and the divergence will be the minimum.

Although Proposition 5 provides a nice intuition for using a non-parametric classifier, it highly depends on the risk of the optimal hypothesis κ, that is, the hypothesis space should contain an optimal classifier that performs well on both the

[1]The assumption is rational because, $n_s \gg k$ in general.

source and the target domains. This assumption cannot be guaranteed to hold true under all scenarios, making the bound conservative and loose. Furthermore, the KNN classifier in Proposition 5 can only affect domain divergence, and how KNN affects the prediction results is unknown. To further aid the study of the proposed algorithm, we conduct the following bounds to fully explore the theoretical properties.

B.2. Analysis of the excess error upper-bound under covariate-shift (Proof of Proposition 2)

Per the statement of the Proposition 2, we assume k being of order $\log n_s$. It is quite small number. For example, in the RotatedMNIST dataset, the optimal k is around 10 and the number of total instances from the source domain $n_s \approx 50,000$.

Under the covariate-shift setting, we have $\eta_U = \eta_S = \eta$ for source and target domains. We denote the KNN classifier with k nearest neighbors as $\hat{f}_k = \mathbb{I}\{\hat{\eta}_k \geq \frac{1}{2}\}$. Because we focus on the binary classification setting, then $\hat{f}_k(x_u) \neq f_U^*(x_u)$ implies that $|\hat{\eta}_k(x_u) - \eta(x_u)| \geq \left|\eta(x_u) - \frac{1}{2}\right|$. In this way, we can build the connection between the excess error and the regress error:

$$\mathcal{E}_U(\hat{f}) = 2\mathbb{E}_{x_u \sim \mathbb{D}_U}\left[\left|\eta(x_u) - \frac{1}{2}\right| \mathbb{I}\left\{|\hat{\eta}_k(x_u) - \eta(x_u)| \geq \left|\eta(x_u) - \frac{1}{2}\right|\right\}\right] \tag{21}$$

Let $Z = \left|\eta(x_u) - \frac{1}{2}\right|$, if we can bound $\sup_{x_u} |\hat{\eta}_k(x_u) - \eta(x_u)| \leq t$, then by the marginal assumption in Assumption 3 and the fact that

$$\mathbb{E}\left[Z \cdot \mathbb{I}\{Z \leq t\}\right] \leq tP(Z \leq t), \tag{22}$$

we have $\mathcal{E}_U(\hat{f}) \leq C_\beta t^{\beta+1}$. To bound $|\hat{\eta}_k(x_u) - \eta(x_u)|$, we denote $(x_s^{(i)}, y_s^{(i)})$ as the $i-$th nearest data and the corresponding labels to x_u in $\mathcal{B}(x_u, r_0)$. The KNN classification result will be $\hat{\eta}(x_u) = \sum_{i=1}^k w_i y_s^{(i)}$, where w_i is the weight for the i -th nearest neighbor and $\sum_{i=1}^k w_i = 1$. In this work, we use the cosine similarity as the weight, where the distance-weighted KNN is shown able to reduce the misclassification error (Dudani, 1976). However, for brevity of the proof, we assume $w_i = \frac{1}{k}, \forall i \in [1, ..., k]$, namely all nearest data labels are uniformly mixed. Based on the assumptions and notions above, we have for any $x_u \in \mathbb{D}_U$

$$\begin{aligned}
|\hat{\eta}_k(x_u) - \eta(x_u)| &= \left|\frac{1}{k}\sum_{i=1}^k y_s^{(i)} - \eta(x_u)\right| \\
&\leq \left|\frac{1}{k}\sum_{i=1}^k y_s^{(i)} - \frac{1}{k}\sum_{i=1}^k \eta\left(x_s^{(i)}\right)\right| + \left|\frac{1}{k}\sum_{i=1}^k \eta\left(x_s^{(i)}\right) - \eta(x_u)\right| \\
&\leq \underbrace{\frac{1}{k}\left|\sum_{i=1}^k y_s^{(i)} - \sum_{i=1}^k \eta\left(x_s^{(i)}\right)\right|}_{①} + \underbrace{\frac{1}{k}\sum_{i=1}^k \left|\eta\left(x_s^{(i)}\right) - \eta(x_u)\right|}_{②},
\end{aligned} \tag{23}$$

where ② is easy to bound. According to the assumption that η_U is C-Smoothness, we have

$$\sum_{i=1}^k \frac{1}{k}\left|\eta\left(x_s^{(i)}\right) - \eta(x_u)\right| \leq \sum_{i=1}^k \frac{1}{k} C \cdot \parallel x_s^{(i)} - x_u \parallel \leq C \cdot \parallel x_s^{(k)} - x_u \parallel \tag{24}$$

According to Eq.(18), with probability at least $1 - \exp(-k/4)$, ② $\leq C\left(\frac{2k}{c_\mu \mu_- \pi_d n_s}\right)^{1/d}$. Note that $E_{Y|X}[y_s^{(i)}] = \eta(x_s^{(i)})$, then we use the Hoeffding inequality to obtain the upper bound of ①

$$P_{X,Y}\left(\frac{1}{k}\left|\sum_{i=1}^k y_s^{(i)} - \sum_{i=1}^k \eta\left(x_s^{(i)}\right)\right| > \epsilon\right) = \mathbb{E}_X\left[P_{Y|X}\left(\frac{1}{k}\left|\sum_{i=1}^k y_s^{(i)} - \sum_{i=1}^k \eta\left(x_s^{(i)}\right)\right| > \epsilon\right)\right] \leq 2\exp(-2k\epsilon^2) \tag{25}$$

Set $\epsilon = (1/k)^{1/4}$, we have, with probability, at least $1 - 2\exp(-2\sqrt{k})$, ① $\leq (1/k)^{1/4}$, ② $\leq C\left(\frac{2k}{c_\mu \mu_- \pi_d n_s}\right)^{1/d}$, and then $|\hat{\eta}_k(x_u) - \eta(x_u)| \leq (1/k)^{1/4} + C\left(\frac{2k}{c_\mu \mu_- \pi_d n_s}\right)^{1/d}$. According to Eq.(18) and Eq.(22), the excess error is bounded by

$$\mathcal{E}_U(\hat{f}) \leq 2C_\beta \left(\left(\frac{1}{k}\right)^{1/4} + C \left(\frac{2k}{c_\mu \mu_- \pi_d n_s}\right)^{1/d} \right)^{1+\beta} \approx \left(\left(\frac{1}{k}\right)^{1/4} + C_1 \left(\frac{k}{c_\mu n_s}\right)^{1/d} \right)^{1+\beta}, \tag{26}$$

where C_1 is a newly introduced constant. There is a clear tradeoff between the upper bound of ① and ② with respect to the value of k. A small k will reduce the representation difference in ②, extremely when $k = 1$, only the nearest sample to x_u will be chosen. However, when k is small, there is no guarantee that the nearest selected data will have a confident prediction. Specifically, a smaller ① indicates that the selected k nearest data samples are representative enough and have confidence in the prediction results. Finally, using $k = \mathcal{O}(\log n_s)$, we have

$$\begin{aligned} &\min\{1 - 2\exp(-2\sqrt{k}), 1 - \exp(-k/4)\} \\ &\geq 1 - 2\exp(-2\sqrt{k}) - \exp(-k/4) \\ &\geq 1 - 3\exp(-2\sqrt{k}) = 1 - 3\exp(-2\sqrt{\mathcal{O}(\log n_s)}) \\ &= 1 - 3\exp(-\mathcal{O}(1)\sqrt{\log n_s}) \end{aligned} \tag{27}$$

where the third line is because $k/4 > 2\sqrt{k}$ for large enough k. Namely, with probability at least $1 - 3\exp(-\sqrt{\log n_s})^{\mathcal{O}(1)}$, the following bound holds true.

$$\mathcal{E}_U(\hat{f}) \leq \mathcal{O}\left(\left(\frac{1}{\log n_s}\right)^{1/4} + \left(\frac{\log n_s}{c_\mu n_s}\right)^{1/d} \right)^{1+\beta}, \tag{28}$$

B.3. Analysis of the excess error upper-bound under posterior-shift settings

Under the posterior-shift setting, the support of $\mathbb{D}_S$ and $\mathbb{D}_U$ are the same, i.e., $\text{Supp}(\mathbb{D}_S) = \text{Supp}(\mathbb{D}_U) = \Omega$. The regression functions η_U and η_S are different. Then we have the following.

$$\begin{aligned} |\hat{\eta}_k(x_u) - \eta_U(x_u)| &= \left| \frac{1}{k} \sum_{i=1}^{k} y_s^{(i)} - \eta_U(x_u) \right| \\ &\leq \left| \frac{1}{k} \sum_{i=1}^{k} y_s^{(i)} - \frac{1}{k} \sum_{i=1}^{k} \eta_S\left(x_s^{(i)}\right) \right| + \left| \frac{1}{k} \sum_{i=1}^{k} \eta_S\left(x_s^{(i)}\right) - \eta_U(x_u) \right| \\ &= \frac{1}{k} \left| \sum_{i=1}^{k} y_s^{(i)} - \sum_{i=1}^{k} \eta\left(x_s^{(i)}\right) \right| + \frac{1}{k} \sum_{i=1}^{k} \left| \eta_S\left(x_s^{(i)}\right) - \eta_U(x_u) \right| \\ &= \frac{1}{k} \left| \sum_{i=1}^{k} y_s^{(i)} - \sum_{i=1}^{k} \eta\left(x_s^{(i)}\right) \right| + \frac{1}{k} \sum_{i=1}^{k} \left| \eta_S\left(x_s^{(i)}\right) - \eta_S(x_u) + \eta_S(x_u) - \eta_U(x_u) \right| \\ &\leq \frac{1}{k} \left| \sum_{i=1}^{k} y_s^{(i)} - \sum_{i=1}^{k} \eta\left(x_s^{(i)}\right) \right| + \frac{1}{k} \sum_{i=1}^{k} \left| \eta_S\left(x_s^{(i)}\right) - \eta_S(x_u) \right| + \underbrace{|\eta_S(x_u) - \eta_U(x_u)|}_{\text{Adaptivity gap}} \end{aligned} \tag{29}$$

Compared to Eq.(23), Eq.(29) has an additional term $|\eta_S(x_u) - \eta_U(x_u)|$ (the adaptivity gap (Zhang et al., 2022d)), which measure the difference of two regression functions directly. Although previous work has similar definition, for example, the regression functions difference defined in (Zhao et al., 2019): $\min\{\mathbb{E}_{\mathbb{D}_S}[|\eta_S - \eta_U|], \mathbb{E}_{\mathbb{D}_U}[|\eta_S - \eta_U|]\}$, which care about "how η_U performs on source data". In comparison, our definition is more similar to (Zhang et al., 2022d), which only focuses on the regression difference when evaluated on examples from the target domain and shown to be more practical and intuitive (Kpotufe & Martinet, 2018; Zhang et al., 2022d).

We assume that $|\eta_S - \eta_U|$ is upper bounded by some constant C_{ada}, namely $\sup_{x_u \in \mathbb{D}_U} |\eta_S(x_u) - \eta_U(x_u)| \leq C_{ada}$, under the posterior-shift setting, we have

$$\mathcal{E}_U(\hat{f}) \leq \left(\left(\frac{1}{k}\right)^{1/4} + C_1 \left(\frac{k}{c_\mu n_s}\right)^{1/d} + C_{ada} \right)^{1+\beta} \tag{30}$$

Via the similar duration of inequality Eq.(28), with at least $1 - 3\exp(-\sqrt{\log n_s})^{\mathcal{O}(1)}$ probability we have,

$$\mathcal{E}_U(\hat{f}) \leq \mathcal{O}\left(\left(\frac{1}{\log n_s}\right)^{1/4} + C_1\left(\frac{\log n_s}{c_\mu n_s}\right)^{1/d} + C_{ada}\right)^{1+\beta}. \tag{31}$$

B.4. Effect of utilizing online target samples (Proof of Proposition 3)

Despite the assumptions and notions mentioned above, to study the effect of target data, we denote $\{x_s^{(i)}, y_s^{(i)}\}_{i=1}^{k_s} + \{x_u^{(i)}, y_u^{(i)}\}_{i=1}^{k_u}$ as the nearest data and the corresponding labels to x_u in $\mathcal{B}(x_u, r_0)$, where $k_s + k_u = k$ and $y_u^{(i)}$ is the pseudo-label of $x_u^{(i)}$, that is, $y_u^{(i)} = \mathbb{I}\{\hat{\eta}_k(x_u^{(i)}) \geq 1/2\}$. The KNN classification result will be $\hat{\eta}_k(x_u) = \frac{1}{k}\sum_{i=1}^{k_s} y_s^{(i)} + \frac{1}{k}\sum_{i=1}^{k_u} y_u^{(i)}$. We have the following.

$$\begin{aligned}
|\hat{\eta}_k(x_u) - \eta(x_u)| &= \left|\frac{1}{k}\sum_{i=1}^{k_s} y_s^{(i)} - \frac{1}{k}\sum_{i=1}^{k_s}\eta(x_u) + \frac{1}{k}\sum_{i=1}^{k_u} y_u^{(i)} - \frac{1}{k}\sum_{i=1}^{k_u}\eta(x_u)\right| \\
&\leq \left|\frac{1}{k}\sum_{i=1}^{k_s} y_s^{(i)} - \frac{1}{k}\sum_{i=1}^{k_s}\eta\left(x_s^{(i)}\right)\right| + \left|\frac{1}{k}\sum_{i=1}^{k_s}\eta\left(x_s^{(i)}\right) - \frac{k_s}{k}\eta(x_u)\right| \\
&\quad + \left|\frac{1}{k}\sum_{i=1}^{k_u} y_u^{(i)} - \frac{1}{k}\sum_{i=1}^{k_u}\eta\left(x_u^{(i)}\right)\right| + \left|\frac{1}{k}\sum_{i=1}^{k_u}\eta\left(x_u^{(i)}\right) - \frac{k_u}{k}\eta(x_u)\right| \\
&\leq \underbrace{\frac{1}{k}\left|\sum_{i=1}^{k_s} y_s^{(i)} + \sum_{i=1}^{k_u} y_u^{(i)} - \sum_{i=1}^{k_s}\eta\left(x_s^{(i)}\right) - \sum_{i=1}^{k_u}\eta\left(x_u^{(i)}\right)\right|}_{①} + \underbrace{\frac{1}{k}\sum_{i=1}^{k_s}\left|\eta\left(x_s^{(i)}\right) - \eta(x_u)\right|}_{②} \\
&\quad + \underbrace{\frac{1}{k}\sum_{i=1}^{k_u}\left|\eta\left(x_u^{(i)}\right) - \eta(x_u)\right|}_{③}
\end{aligned} \tag{32}$$

Although the true labels of target samples are unknown, we store the target sample into the KNN query set only when its prediction confidence is large enough. Therefore, it is natural to assume that $\mathbb{E}_{Y|X}[y_u^{(i)}] = \eta(x_u^{(i)})$. According to Eq.(25), we have

$$\begin{aligned}
&P_{X,Y}\left(\frac{1}{k}\left|\sum_{i=1}^{k_s} y_s^{(i)} + \sum_{i=1}^{k_u} y_u^{(i)} - \sum_{i=1}^{k_s}\eta\left(x_s^{(i)}\right) - \sum_{i=1}^{k_u}\eta\left(x_u^{(i)}\right)\right|\right) \\
&= \mathbb{E}_X\left[P_{Y|X}\left(\frac{1}{k}\left|\sum_{i=1}^{k_s} y_s^{(i)} + \sum_{i=1}^{k_u} y_u^{(i)} - \sum_{i=1}^{k_s}\eta\left(x_s^{(i)}\right) - \sum_{i=1}^{k_u}\eta\left(x_u^{(i)}\right)\right|\right)\right] \leq 2\exp(-2k\epsilon^2)
\end{aligned} \tag{33}$$

Set $\epsilon = (1/k)^{1/4}$, we have, with probability, at least $1 - 2\exp(-2\sqrt{k})$, ① $\leq (1/k)^{1/4}$. Then, according to Eq.(24), we have

$$② \leq \frac{k_s}{k} C\left(\frac{2k_s}{c_\mu \mu_- \pi_d n_s}\right)^{1/d}; \; ③ \leq \frac{k_u}{k} C\left(\frac{2k_u}{c_\mu^* \mu_- \pi_d n_u}\right)^{1/d} \tag{34}$$

Finally, the excess error under the covariate shift setting can be bounded by

$$\begin{aligned}
\mathcal{E}_U(\hat{f}) &\leq 2C_\beta\left((1/k)^{1/4} + \frac{k_s}{k} C\left(\frac{2k_s}{c_\mu \mu_- \pi_d n_s}\right)^{1/d} + \frac{k_u}{k} C\left(\frac{2k_u}{c_\mu^* \mu_- \pi_d n_u}\right)^{1/d}\right)^{1+\beta} \\
&\approx \left(\left(\frac{1}{k}\right)^{1/4} + C_1 k_s\left(\frac{k_s}{c_\mu n_s}\right)^{1/d} + C_1 k_u\left(\frac{k_u}{c_\mu^* n_u}\right)^{1/d}\right)^{1+\beta}
\end{aligned} \tag{35}$$

Compared Eq.(35) to Eq.(26), it is easy to verify that

$$\begin{aligned}&\frac{k}{k}C\left(\frac{2k}{c_\mu\mu_-\pi_d n_s}\right)^{1/d}-\frac{k_s}{k}C\left(\frac{2k_s}{c_\mu\mu_-\pi_d n_s}\right)^{1/d}-\frac{k_u}{k}C\left(\frac{2k_u}{c_\mu^*\mu_-\pi_d n_u}\right)^{1/d}\\&\geq\frac{k_u}{k}C\left(\frac{2k_s}{c_\mu\mu_-\pi_d n_s}\right)^{1/d}-\frac{k_u}{k}C\left(\frac{2k_u}{c_\mu^*\mu_-\pi_d n_u}\right)^{1/d}\end{aligned}\tag{36}$$

Because in general, we have $c_\mu^* > c_u$, the difference is then larger than 0, namely incorporating target samples into the KNN memory bank, the excess error can be further reduced. When $\mathbb{D}_S$ is very close to $\mathbb{D}_U$, that is, $c_\mu^* \approx c_\mu$, the two bounds will be similar.

Similar results can be derived under the posterior-shift setting. Under the assumption that $\mathbb{E}_{Y|X}[y_u^{(i)}] = \eta_U(x_u^{(i)})$ and $\sup_{x_u\in\mathbb{D}_U}|\eta_S(x_u)-\eta_U(x_u)|\leq C_{ada}$, we have

$$\begin{aligned}|\hat{\eta}_k(x_u)-\eta_U(x_u)|&=\left|\frac{1}{k}\sum_{i=1}^{k_s}y_s^{(i)}-\frac{1}{k}\sum_{i=1}^{k_s}\eta_U(x_u)+\frac{1}{k}\sum_{i=1}^{k_u}y_u^{(i)}-\frac{1}{k}\sum_{i=1}^{k_u}\eta_U(x_u)\right|\\&\leq\left|\frac{1}{k}\sum_{i=1}^{k_s}y_s^{(i)}-\frac{1}{k}\sum_{i=1}^{k_s}\eta_S\left(x_s^{(i)}\right)\right|+\left|\frac{1}{k}\sum_{i=1}^{k_s}\eta_S\left(x_s^{(i)}\right)-\frac{k_s}{k}\eta_U(x_u)\right|\\&\quad+\left|\frac{1}{k}\sum_{i=1}^{k_u}y_u^{(i)}-\frac{1}{k}\sum_{i=1}^{k_u}\eta_U\left(x_u^{(i)}\right)\right|+\left|\frac{1}{k}\sum_{i=1}^{k_u}\eta_U\left(x_u^{(i)}\right)-\frac{k_u}{k}\eta_U(x_u)\right|\\&\leq\frac{1}{k}\left|\sum_{i=1}^{k_s}y_s^{(i)}+\sum_{i=1}^{k_u}y_u^{(i)}-\sum_{i=1}^{k_s}\eta_S\left(x_s^{(i)}\right)-\sum_{i=1}^{k_u}\eta_U\left(x_u^{(i)}\right)\right|+\frac{1}{k}\sum_{i=1}^{k_s}\left|\eta_S\left(x_s^{(i)}\right)-\eta_S(x_u)\right|\\&\quad+\frac{1}{k}\sum_{i=1}^{k_u}\left|\eta_U\left(x_u^{(i)}\right)-\eta_U(x_u)\right|+\frac{k_s}{k}\left|\eta_S(x_u)-\eta_U(x_u)\right|,\end{aligned}\tag{37}$$

and the conclusion will be

$$\mathcal{E}_U(\hat{f})\leq 2C_\beta\left((1/k)^{1/4}+\frac{k_s}{k}C\left(\frac{2k_s}{c_\mu\mu_-\pi_d n_s}\right)^{1/d}+\frac{k_u}{k}C\left(\frac{2k_u}{c_\mu^*\mu_-\pi_d n_u}\right)^{1/d}+\frac{k_s}{k}C_{ada}\right)^{1+\beta}\tag{38}$$

C. Online Optimization Algorithm for Optimizing the KNN Loss Function $\mathcal{L}_{KNN}$

Algorithm 1 Online optimization algorithm for optimizing the KNN loss function $\mathcal{L}_{KNN}$.

Input:training data D_S, batch size N, learning rate η, training iterations T, Adam hyperparameters β_1, β_2.
Initial: model parameters θ and memory bank $\mathcal{M} = \{h_\theta(x_i), y_i\}_{i=1}^M$ with a predefined size M, where (x_i, y_i) are randomly sampled from D_S.
for t = $1, \ldots, T$ **do**
  $(x_i, y_i)_{i=1}^N \sim D_S$ *//Data sampling*
  $\mathcal{L}_{KNN} = -\frac{1}{N}\sum_i \log\frac{\sum_{j\in B_{k,\theta,\mathcal{M}}(x_i)}\exp(w_{ij}/\tau)\mathbb{I}\{y_i=y_j\}}{\sum_{j\in B_{k,\theta,\mathcal{M}}(x_i)}\exp(w_{ij}/\tau)}$, *//Calculate the loss*
  $\mathcal{M}\leftarrow\mathcal{M}\bigcup\{h_\theta(x_i),y_i\}_{i=1}^N$ *//Update the memory bank by the first-in-first-out (FIFO) strategy*
  $\theta\leftarrow\mathrm{Adam}\left(\mathcal{L}_{KNN},\theta,\eta,\beta_1,\beta_2\right)$ *//Update model parameters*
end for

D. Dataset and Implementation Details

D.1. Dataset details

Rotated MNIST (Ghifary et al., 2015) consists of 10,000 digits in MNIST with different rotated angles where the domain is determined by the degrees $d \in \{0, 15, 30, 45, 60, 75\}$.

#	Layer
1	Conv2D (in=d, out=64)
2	ReLU
3	GroupNorm (groups=8)
4	Conv2D (in=64, out=128, stride=2)
5	ReLU
6	GroupNorm (8 groups)
7	Conv2D (in=128, out=128)
8	ReLU
9	GroupNorm (8 groups)
10	Conv2D (in=128, out=128)
11	ReLU
12	GroupNorm (8 groups)
13	Global average-pooling

Table 8. Details of our MNIST ConvNet architecture. All convolutions use 3×3 kernels and "same" padding

PACS (Li et al., 2017) includes 9, 991 images with 7 classes $y \in \{$ dog, elephant, giraffe, guitar, horse, house, person $\}$ from 4 domains $d \in \{$art, cartoons, photos, sketches$\}$.

VLCS (Torralba & Efros, 2011) is composed of 10,729 images, 5 classes $y \in \{$ bird, car, chair, dog, person $\}$ from domains $d \in \{$Caltech101, LabelMe, SUN09, VOC2007$\}$.

TerraIncognita (Beery et al., 2018) contains photographs of wild animals taken by camera traps at locations $d \in \{L100, L38, L43, L46\}$, with $24,788$ examples of dimension $(3, 224, 224)$ and 10 classes.

DomainNet (Peng et al., 2019) has six domains $d \in \{$clipart, infograph, painting, quickdraw, real, sketch$\}$. This dataset contains $586,575$ examples of sizes $(3, 224, 224)$ and 345 classes.

D.2. Implementation and hyper-parameter details

Hyperparameter search. Following the experimental settings in (Gulrajani & Lopez-Paz, 2021), we conduct a random search of 20 trials over the hyperparameter distribution for each algorithm and test domain. Specifically, we split the data from each domain into 80% and 20% proportions, where the larger split is used for training and evaluation, and the smaller ones are used for select hyperparameters. We repeat the entire experiment twice using three different seeds to reduce the randomness. Finally, we report the mean over these repetitions as well as their estimated standard error.

Model selection. The model selection in domain generalization is intrinsically a learning problem, and we use both the test-domain validation and training domain validation, two of the three selection methods in (Gulrajani & Lopez-Paz, 2021). Test-domain validation chooses the model maximizing the accuracy on a validation set that follows the distribution of the test domain. Training domain validation chooses the model with the highest average source domain accuracy. In the main paper, test domain validation results are presented by default.

Model architectures. Following (Gulrajani & Lopez-Paz, 2021), we use ConvNet (Table.8) as the encoder for RotatedMNIST (detailed in Appendix D.1 in (Gulrajani & Lopez-Paz, 2021)) with MIT License. For other datasets, torch-vision for ResNet18 and ResNet50 (Apache-2.0), *timm* for Vision Transformer (Apache-2.0), and the official repository of T3A (MIT License) are used. We run our experiments mainly on Tesla-V100 (32G)x4 instances.

E. Additional Experimental Results

E.1. Detailed generalization results

Tables 13, 14, 15, 16 contain detailed results for each dataset with 'Test-domain' and 'Training-domain' model selection methods.

Method	PACS					Terr				
	A	C	P	S	Avg	L100	L38	L43	L46	Avg
T3A	94.5	94.2	93.7	91	93.4	89.6	86.7	85.5	83.8	86.4
Tent	94.5	94.2	93.1	92.2	93.5	89.6	89.1	88.6	82.3	87.4
PLFull	94.5	94.1	93.3	90.2	93.0	89.6	81.22	65.8	55.1	72.9
AdaNPC	94.5	94.5	94.5	94.5	94.5	89.6	89.6	89.6	89.6	89.6

Table 9. **Successive adaptation results on the PACS and TerraIncognita datasets.** The metric is the re-evaluation accuracy of the adapted model on the source domain d_0.

DomainNet	**clip**	**info**	**paint**	**quick**	**real**	**sketch**	**Avg**
T3A	63.4	37.7	45.7	45.7	50.0	51.1	48.9
Tent	63.4	50.4	21.2	8.2	6.9	5.3	25.9
PLFull	63.4	61.3	55.5	16.5	10.0	7.0	35.6
AdaNPC	63.4	63.4	63.4	63.4	63.4	63.4	63.4
RMNIST	**0.0**	**15.0**	**30.0**	**45.0**	**60.0**	**75.0**	**Avg**
T3A	100.0	96.1	96.1	93.2	90.3	89.9	94.3
Tent	100.0	99.5	96.5	92.5	89.3	89.4	94.5
PLFull	100.0	98.3	87.5	79.2	67.9	63.5	82.7
AdaNPC	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Table 10. **Successive adaptation results on the DomainNet and RMNIST datasets.** The metric is the re-evaluation accuracy of the adapted model on the source domain d_0.

E.2. AdaNPC can mitigate the issue of domain forgetting

At first, recall our experimental setting is: we first trained our model on d_0 of the dataset as shown in Figure 4. Then, we utilized the TTA algorithm to adapt the model to $d_1, ..., d_n$ one by one. To illustrate this problem more vividly, we conducted experiments on several datasets, and their results are presented in Table 9 and Table 10. These results demonstrate that as the TTA progresses, the model's performance on the source domain declines, indicating that TTA causes the model to forget the knowledge learned from the source domain. In practical scenarios, the model is required to perform for an extended period and may encounter numerous novel data. Under such circumstances, the model's performance on the source domain may suffer greatly, leading to inaccurate predictions on the source domain samples. As can be seen, for most existing baselines, the model forgets the knowledge in the source domain with the progress of TTA on all datasets, while AdaNPC can avoid this situation.

E.3. Extended experiments on Cifar-10-C and ImageNet-C

This paper focus on domain generalization (DG), where TTA is considered as one of the DG methods. Therefore, we emphasized the comparison with existing DG methods in the main text. However, we also conduct additional experiments on corruption datasets to verify the effectiveness of the proposed AdaNPC. Figure 8 shows the results with the lowest severity (one), where AdaNPC performs the best and retraining the BN statistic will not be beneficial. For WRN-28-10 Wide ResNet Backbone, the results are shown in Figure 9 and Figure 10, where the same pattern as results with 40-2 Wide ResNet Backbone are observed.

Considering large-scale corruption benchmarks, we compared the method with Tent and EATA (Niu et al., 2022) on the ImageNet-C dataset. Experimentally, we compared the effectiveness of AdaNPC and EATA separately in Table. 3, where the experimental and baselines are all following EATA (Niu et al., 2022). As we can see, we only need to simply replace the Linear layer with KNN and remember the trustworthy samples during testing, and AdaNPC can achieve much better performance than ResNet50 with a linear head. In addition, AdaNPC can be combined with existing TTA methods such as Tent, ETA, etc., and the final performance surpasses these methods individually.

Existing baselines will be highly affected by the batch size. As can be seen in Table 11 (left), existing methods tend to have very poor performance when the batch size is set to 1, as the gradient noise for individual samples is very high, which

Method	BN Retraining				Clf Retraining			
	Defoc	Glass	Motion	Zoom	Defoc	Glass	Motion	Zoom
R-50 (BN)	82.10	90.20	85.20	77.50	82.10	90.20	85.20	77.50
Tent	99.10	99.10	99.11	99.10	0.90	0.90	0.89	0.90
ETA	99.14	99.14	99.14	99.16	0.86	0.86	0.86	0.84
EATA	99.14	99.14	99.14	99.16	0.86	0.86	0.86	0.84
AdaNPC	83.10	83.00	72.30	60.57	83.10	83.00	72.30	60.57

Table 11. Comparison with state-of-the-art methods on ImageNet-C with the highest severity level 5 regarding corruption Error, where the TTA batch size is set to 1.

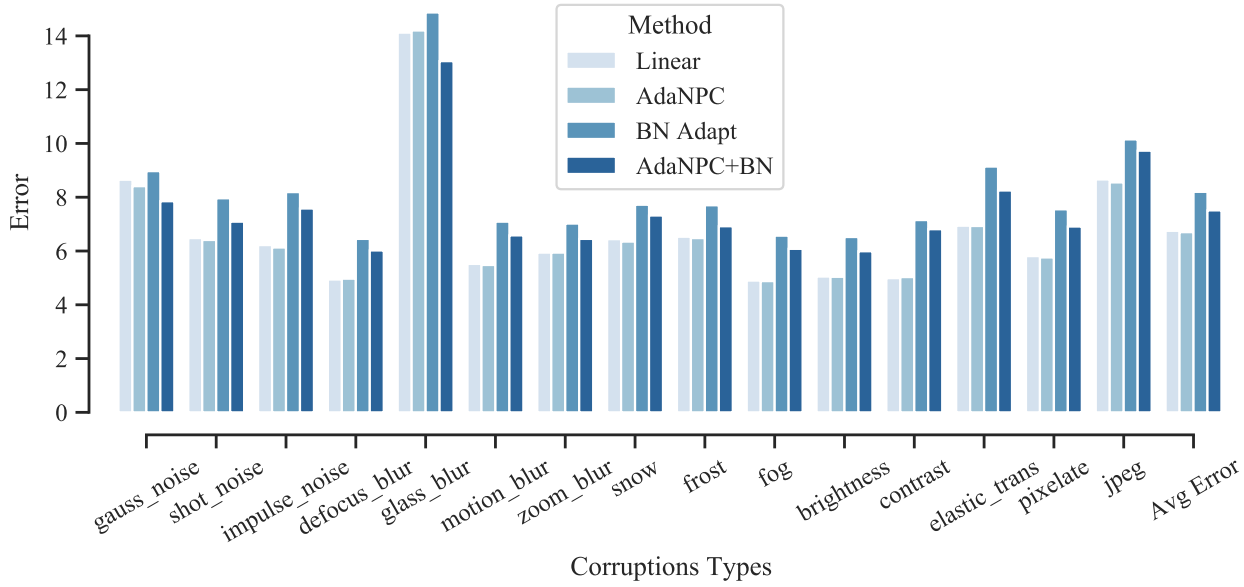


Figure 8. **Corruption benchmark on CIFAR-10-C with the lowest severity (one)**. AdaNPC +BN means that the KNN classifier and BN retraining are both used.

is detrimental to model optimization. However, it should be emphasized that batch data does not align with the setting of online learning, where inference is required on-demand instead of waiting for an incoming batch or when inference is happening on an edge device (such as a mobile phone) where there is no opportunity for batching. Therefore, AdaNPC, a TTA method that is insensitive to batch size, is valuable for the current research field. In addition, the three baselines in the above experiments all suffered from severe collapse due to the influence of the BN layer, which may not be a fair comparison to AdaNPC. We additionally considered a setting where all BN layer parameters were frozen during testing, and the final linear layer was updated using the objective functions of each algorithm (Clf Retraining). The final results are shown in Table 11(right). It can be seen that even if the influence of the BN layer is excluded, high-noisy gradients can still make the models perform very poorly. Although ETA and EATA, as strong baselines, perform much better than using Tent alone, they are still heavily affected by the batch size.

E.4. Extended ablation studies of training algorithms

Results in Table. 12 shows that on the Rotated MNIST dataset, with $\mathcal{L}_{KNN}$, the representation will be better and the generalization results will be improved.

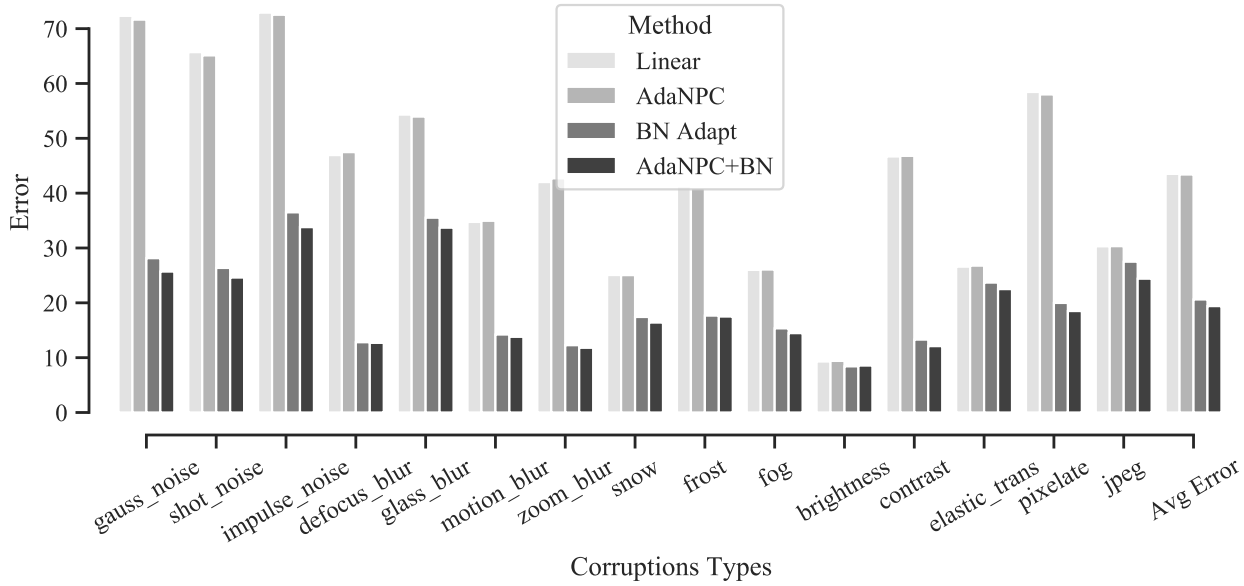


Figure 9. **Corruption benchmark on CIFAR-10-C with the highest severity (five) and a 40-2 Wide ResNet backbone (Zagoruyko & Komodakis, 2016) pre-trained on CIFAR-10.** AdaNPC +BN means that the KNN classifier and BN retraining are both used.

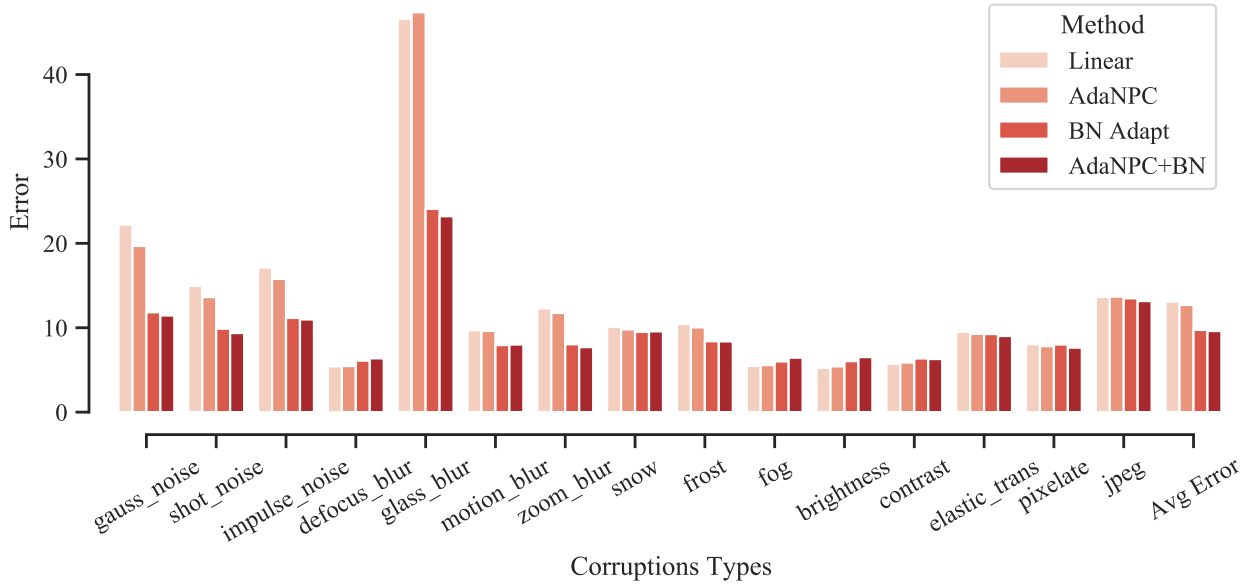


Figure 10. **Corruption benchmark on CIFAR-10-C with the lowest severity (one) and a 40-2 Wide ResNet backbone (Zagoruyko & Komodakis, 2016) pre-trained on CIFAR-10**. AdaNPC +BN means that the KNN classifier and BN retraining are both used.

	d_0	d_1	d_2	d_3	d_4	d_5	Avg
ERM	94.0 ± 0.2	98.1 ± 0.1	98.8 ± 0.4	99.1 ± 0.1	$\mathbf{98.9 \pm 0.1}$	96.4 ± 0.1	97.6
ERM+AdaNPC	96.6 ± 0.4	98.7 ± 0.1	98.9 ± 0.3	99.1 ± 0.1	98.7 ± 0.1	96.7 ± 0.3	98.1
$\mathcal{L}_{KNN}$	96.1 ± 0.9	98.6 ± 0.3	98.9 ± 0.0	99.1 ± 0.2	98.8 ± 0.2	97.0 ± 0.5	98.1
$\mathcal{L}_{KNN}$ + AdaNPC	$\mathbf{96.9} \pm 0.3$	$\mathbf{98.8 \pm 0.1}$	$\mathbf{99.1 \pm 0.2}$	$\mathbf{99.2 \pm 0.1}$	$\mathbf{98.9 \pm 0.1}$	$\mathbf{98.0 \pm 0.1}$	**98.5**

Table 12. Ablation studies of training loss on Rotated MNIST.

E.5. Extended visualization of classification results.

Figure 11 provide more evaluation instances that be prediction correctly and Figure 12 supplies more failure cases.

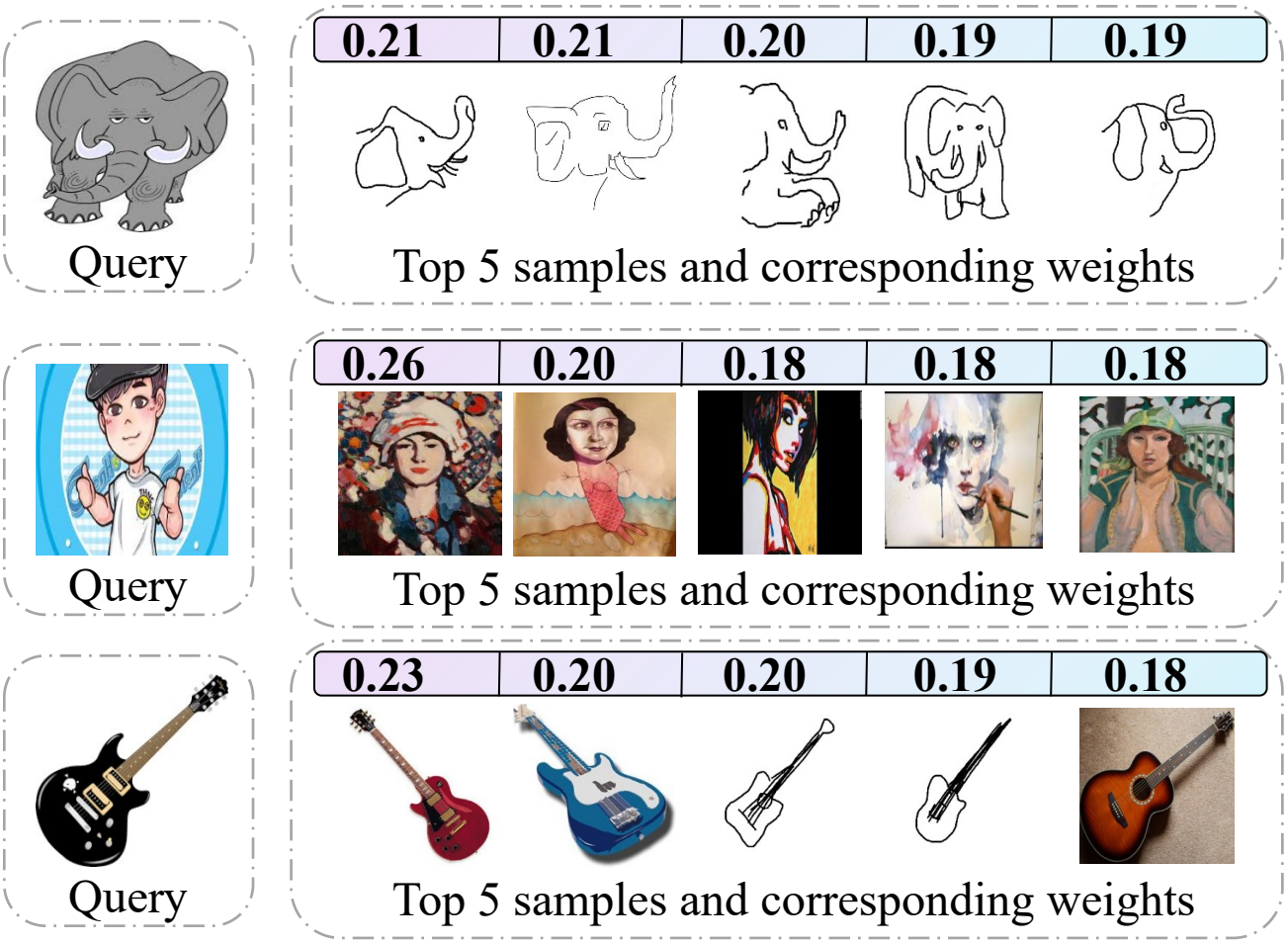


Figure 11. Visualization of successfully classified results attained by AdaNPC.

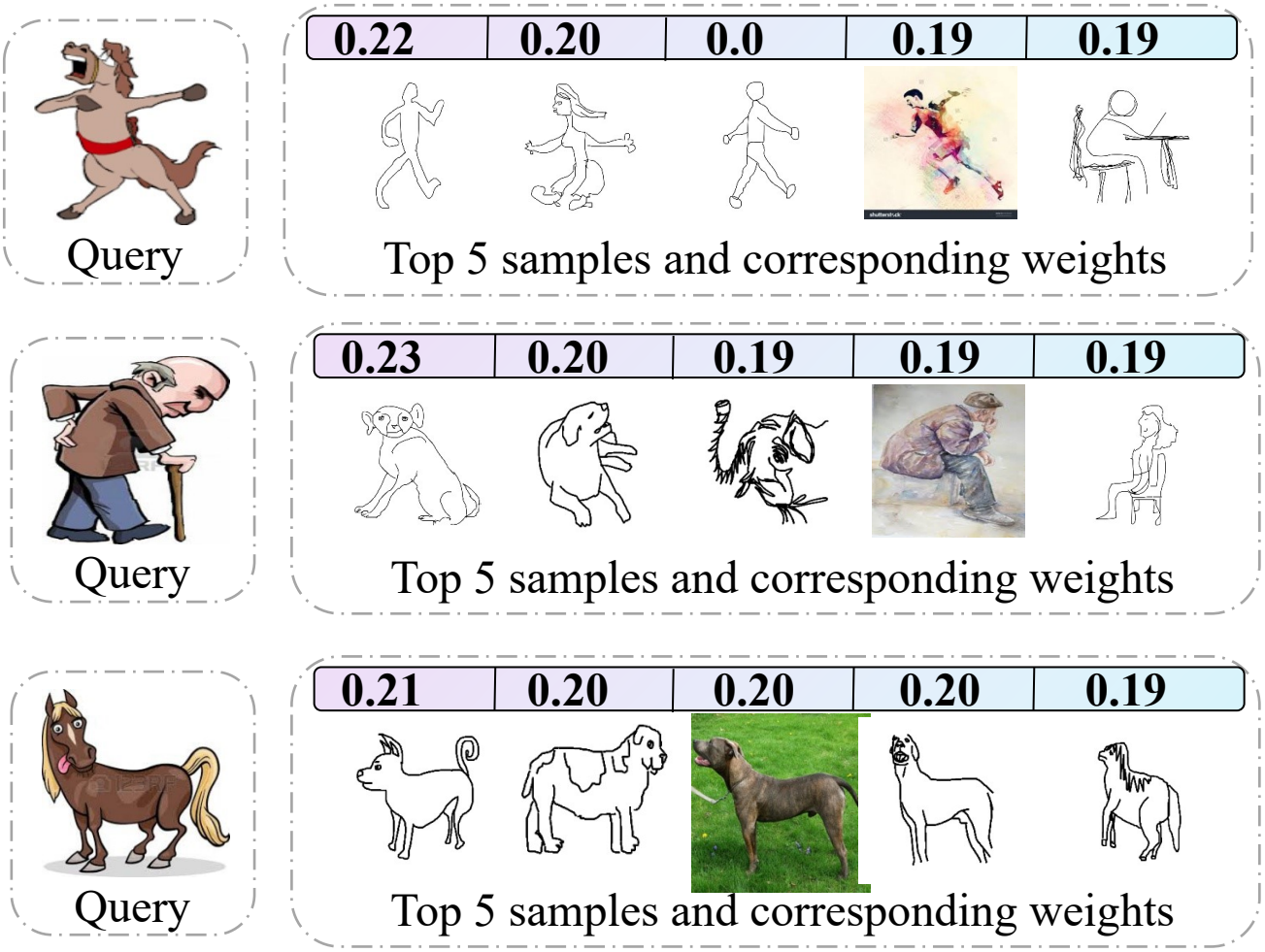


Figure 12. Visualization of misclassified results attained by AdaNPC.

Rotated MNIST, Model selection: 'Test-domain' validation set							
Algorithm	**0**	**15**	**30**	**45**	**60**	**75**	**Avg**
ERM (Vapnik, 1999)	95.3 ± 0.2	98.7 ± 0.1	98.9 ± 0.1	98.7 ± 0.2	98.9 ± 0.0	96.2 ± 0.2	97.8
IRM (Arjovsky et al., 2020)	94.9 ± 0.6	98.7 ± 0.2	98.6 ± 0.1	98.6 ± 0.2	98.7 ± 0.1	95.2 ± 0.3	97.5
GDRO (Sagawa et al., 2020)	95.9 ± 0.1	99.0 ± 0.1	98.9 ± 0.1	98.8 ± 0.1	98.6 ± 0.1	96.3 ± 0.4	97.9
Mixup (Yan et al., 2020)	95.8 ± 0.3	98.7 ± 0.0	99.0 ± 0.1	98.8 ± 0.1	98.8 ± 0.1	96.6 ± 0.2	98.0
MLDG (Li et al., 2018a)	95.7 ± 0.2	98.9 ± 0.1	98.8 ± 0.1	98.9 ± 0.1	98.6 ± 0.1	95.8 ± 0.4	97.8
CORAL (Sun & Saenko, 2016)	96.2 ± 0.2	98.8 ± 0.1	98.8 ± 0.1	98.8 ± 0.1	98.9 ± 0.1	96.4 ± 0.2	98.0
MMD (Li et al., 2018b)	96.1 ± 0.2	98.9 ± 0.0	99.0 ± 0.0	98.8 ± 0.0	98.9 ± 0.0	96.4 ± 0.2	98.0
DANN (Ganin et al., 2016)	95.9 ± 0.1	98.9 ± 0.1	98.6 ± 0.2	98.7 ± 0.1	98.9 ± 0.0	96.3 ± 0.3	97.9
CDANN (Li et al., 2018c)	95.9 ± 0.2	98.8 ± 0.0	98.7 ± 0.1	98.9 ± 0.1	98.8 ± 0.1	96.1 ± 0.3	97.9
MTL (Blanchard et al., 2021)	96.1 ± 0.2	98.9 ± 0.0	99.0 ± 0.0	98.7 ± 0.1	99.0 ± 0.0	95.8 ± 0.3	97.9
SagNet (Nam et al., 2021)	95.9 ± 0.1	99.0 ± 0.1	98.9 ± 0.1	98.6 ± 0.1	98.8 ± 0.1	96.3 ± 0.1	97.9
ARM (Zhang et al., 2021a)	95.9 ± 0.4	99.0 ± 0.1	98.8 ± 0.1	98.9 ± 0.1	99.1 ± 0.1	96.7 ± 0.2	98.1
VREx (Krueger et al., 2021)	95.5 ± 0.2	99.0 ± 0.0	98.7 ± 0.2	98.8 ± 0.1	98.8 ± 0.0	96.4 ± 0.0	97.9
RSC (Huang et al., 2020)	95.4 ± 0.1	98.6 ± 0.1	98.6 ± 0.1	98.9 ± 0.0	98.8 ± 0.1	95.4 ± 0.3	97.6
Fish (Shi et al., 2022)							97.9
Fisher (Rame et al., 2022)	95.8 ± 0.1	98.3 ± 0.1	98.8 ± 0.1	98.6 ± 0.3	98.7 ± 0.1	96.5 ± 0.1	97.8
AdaNPC	97.2 ± 0.1	**99.2 ± 0.0**	**99.1 ± 0.0**	**99.0 ± 0.1**	**99.2 ± 0.0**	**97.6 ± 0.1**	**98.5**
AdaNPC +NP	**97.4 ± 0.1**	99.0 ± 0.0	98.8 ± 0.0	98.8 ± 0.1	99.0 ± 0.1	97.3 ± 0.1	98.4
Rotated MNIST, Model selection: 'Training-domain' validation set							
Algorithm	**0**	**15**	**30**	**45**	**60**	**75**	**Avg**
ERM (Vapnik, 1999)	95.9 ± 0.1	98.9 ± 0.0	98.8 ± 0.0	98.9 ± 0.0	98.9 ± 0.0	96.4 ± 0.0	98.0
IRM (Arjovsky et al., 2020)	95.5 ± 0.1	98.8 ± 0.2	98.7 ± 0.1	98.6 ± 0.1	98.7 ± 0.0	95.9 ± 0.2	97.7
GDRO (Sagawa et al., 2020)	95.6 ± 0.1	98.9 ± 0.1	98.9 ± 0.1	99.0 ± 0.0	98.9 ± 0.0	96.5 ± 0.2	98.0
Mixup (Yan et al., 2020)	95.8 ± 0.3	98.9 ± 0.0	98.9 ± 0.0	98.9 ± 0.0	98.8 ± 0.1	96.5 ± 0.3	98.0
MLDG (Li et al., 2018a)	95.8 ± 0.1	98.9 ± 0.1	99.0 ± 0.0	98.9 ± 0.1	99.0 ± 0.0	95.8 ± 0.3	97.9
CORAL (Sun & Saenko, 2016)	95.8 ± 0.3	98.8 ± 0.0	98.9 ± 0.0	99.0 ± 0.0	98.9 ± 0.1	96.4 ± 0.2	98.0
MMD (Li et al., 2018b)	95.6 ± 0.1	98.9 ± 0.1	99.0 ± 0.0	99.0 ± 0.0	98.9 ± 0.0	96.0 ± 0.2	97.9
DANN (Ganin et al., 2016)	95.0 ± 0.5	98.9 ± 0.1	99.0 ± 0.0	99.0 ± 0.1	98.9 ± 0.0	96.3 ± 0.2	97.8
CDANN (Li et al., 2018c)	95.7 ± 0.2	98.8 ± 0.0	98.9 ± 0.1	98.9 ± 0.1	98.9 ± 0.1	96.1 ± 0.3	97.9
MTL (Blanchard et al., 2021)	95.6 ± 0.1	99.0 ± 0.1	99.0 ± 0.0	98.9 ± 0.1	99.0 ± 0.1	95.8 ± 0.2	97.9
SagNet (Nam et al., 2021)	95.9 ± 0.3	98.9 ± 0.1	99.0 ± 0.1	**99.1 ± 0.0**	99.0 ± 0.1	96.3 ± 0.1	98.0
ARM (Zhang et al., 2021a)	96.7 ± 0.2	99.1 ± 0.0	99.0 ± 0.0	99.0 ± 0.1	99.1 ± 0.1	96.5 ± 0.4	98.2
VREx (Krueger et al., 2021)	95.9 ± 0.2	99.0 ± 0.1	98.9 ± 0.1	98.9 ± 0.1	98.7 ± 0.1	96.2 ± 0.2	97.9
RSC (Huang et al., 2020)	94.8 ± 0.5	98.7 ± 0.1	98.8 ± 0.1	98.8 ± 0.0	98.9 ± 0.1	95.9 ± 0.2	97.6
Fish (Shi et al., 2022)							98.0
Fisher (Rame et al., 2022)	95.0 ± 0.3	98.5 ± 0.0	**99.2 ± 0.1**	98.9 ± 0.0	98.9 ± 0.1	96.5 ± 0.1	97.8
AdaNPC	97.7 ± 0.4	**99.1 ± 0.0**	99.1 ± 0.1	**99.1 ± 0.1**	**99.2 ± 0.0**	97.5 ± 0.2	98.6
AdaNPC +BN	**97.9 ± 0.3**	**99.1 ± 0.1**	**99.2 ± 0.0**	**99.1 ± 0.1**	**99.2 ± 0.0**	**98.0 ± 0.4**	**98.8**

Table 13. Domain generalization accuracy (%) on Rotated MNIST.

VLCS, Model selection: 'Test-domain' validation set					
Algorithm	**C**	**L**	**S**	**V**	**Avg**
ERM (Vapnik, 1999)	97.6 ± 0.3	67.9 ± 0.7	70.9 ± 0.2	74.0 ± 0.6	77.6
IRM (Arjovsky et al., 2020)	97.3 ± 0.2	66.7 ± 0.1	71.0 ± 2.3	72.8 ± 0.4	76.9
GDRO (Sagawa et al., 2020)	97.7 ± 0.2	65.9 ± 0.2	72.8 ± 0.8	73.4 ± 1.3	77.4
Mixup (Yan et al., 2020)	97.8 ± 0.4	67.2 ± 0.4	71.5 ± 0.2	75.7 ± 0.6	78.1
MLDG (Li et al., 2018a)	97.1 ± 0.5	66.6 ± 0.5	71.5 ± 0.1	75.0 ± 0.9	77.5
CORAL (Sun & Saenko, 2016)	97.3 ± 0.2	67.5 ± 0.6	71.6 ± 0.6	74.5 ± 0.0	77.7
MMD (Li et al., 2018b)	98.8 ± 0.0	66.4 ± 0.4	70.8 ± 0.5	75.6 ± 0.4	77.9
DANN (Ganin et al., 2016)	99.0 ± 0.2	66.3 ± 1.2	73.4 ± 1.4	80.1 ± 0.5	79.7
CDANN (Li et al., 2018c)	98.2 ± 0.1	68.8 ± 0.5	74.3 ± 0.6	78.1 ± 0.5	79.9
MTL (Blanchard et al., 2021)	97.9 ± 0.7	66.1 ± 0.7	72.0 ± 0.4	74.9 ± 1.1	77.7
SagNet (Nam et al., 2021)	97.4 ± 0.3	66.4 ± 0.4	71.6 ± 0.1	75.0 ± 0.8	77.6
ARM (Zhang et al., 2021a)	97.6 ± 0.6	66.5 ± 0.3	72.7 ± 0.6	74.4 ± 0.7	77.8
VREx (Krueger et al., 2021)	98.4 ± 0.2	66.4 ± 0.7	72.8 ± 0.1	75.0 ± 1.4	78.1
RSC (Huang et al., 2020)	98.0 ± 0.4	67.2 ± 0.3	70.3 ± 1.3	75.6 ± 0.4	77.8
Fish (Shi et al., 2022)					77.8
Fisher (Rame et al., 2022)	97.6 ± 0.7	67.3 ± 0.5	72.2 ± 0.9	75.7 ± 0.3	78.2
AdaNPC	**98.7 ± 0.2**	66.6 ± 0.2	74.6 ± 0.3	79.6 ± 0.5	79.9
+BN retraining	**98.7 ± 0.2**	**67.4 ± 0.3**	**74.9 ± 0.5**	**79.7 ± 0.5**	**80.2**
VLCS, Model selection: 'Training-domain' validation set					
Algorithm	**C**	**L**	**S**	**V**	**Avg**
ERM (Vapnik, 1999)	97.7 ± 0.4	64.3 ± 0.9	73.4 ± 0.5	74.6 ± 1.3	77.5
IRM (Arjovsky et al., 2020)	98.6 ± 0.1	64.9 ± 0.9	73.4 ± 0.6	77.3 ± 0.9	78.5
GDRO (Sagawa et al., 2020)	97.3 ± 0.3	63.4 ± 0.9	69.5 ± 0.8	76.7 ± 0.7	76.7
Mixup (Yan et al., 2020)	98.3 ± 0.6	64.8 ± 1.0	72.1 ± 0.5	74.3 ± 0.8	77.4
MLDG (Li et al., 2018a)	97.4 ± 0.2	65.2 ± 0.7	71.0 ± 1.4	75.3 ± 1.0	77.2
CORAL (Sun & Saenko, 2016)	98.3 ± 0.1	66.1 ± 1.2	73.4 ± 0.3	77.5 ± 1.2	78.8
MMD (Li et al., 2018b)	97.7 ± 0.1	64.0 ± 1.1	72.8 ± 0.2	75.3 ± 3.3	77.5
DANN (Ganin et al., 2016)	**99.0 ± 0.3**	65.1 ± 1.4	73.1 ± 0.3	77.2 ± 0.6	78.6
CDANN (Li et al., 2018c)	97.1 ± 0.3	65.1 ± 1.2	70.7 ± 0.8	77.1 ± 1.5	77.5
MTL (Blanchard et al., 2021)	97.8 ± 0.4	64.3 ± 0.3	71.5 ± 0.7	75.3 ± 1.7	77.2
SagNet (Nam et al., 2021)	97.9 ± 0.4	64.5 ± 0.5	71.4 ± 1.3	77.5 ± 0.5	77.8
ARM (Zhang et al., 2021a)	98.7 ± 0.2	63.6 ± 0.7	71.3 ± 1.2	76.7 ± 0.6	77.6
VREx (Krueger et al., 2021)	98.4 ± 0.3	64.4 ± 1.4	74.1 ± 0.4	76.2 ± 1.3	78.3
RSC (Huang et al., 2020)	97.9 ± 0.1	62.5 ± 0.7	72.3 ± 1.2	75.6 ± 0.8	77.1
Fish (Shi et al., 2022)					77.8
Fisher (Rame et al., 2022)	98.9 ± 0.3	64.0 ± 0.5	71.5 ± 0.2	76.8 ± 0.7	77.8
AdaNPC	98.9 ± 0.3	64.5 ± 1.0	73.5 ± 0.7	75.6 ± 0.8	78.1
+BN retraining	98.4 ± 0.6	**65.2 ± 1.2**	**74.4 ± 0.3**	**77.4 ± 1.1**	**78.9**

Table 14. Domain generalization accuracy (%) on VLCS.

PACS, Model selection: 'Test-domain' validation set					
Algorithm	**A**	**C**	**P**	**S**	**Avg**
ERM (Vapnik, 1999)	86.5 ± 1.0	81.3 ± 0.6	96.2 ± 0.3	82.7 ± 1.1	86.7
IRM (Arjovsky et al., 2020)	84.2 ± 0.9	79.7 ± 1.5	95.9 ± 0.4	78.3 ± 2.1	84.5
GDRO (Sagawa et al., 2020)	87.5 ± 0.5	82.9 ± 0.6	97.1 ± 0.3	81.1 ± 1.2	87.1
Mixup (Yan et al., 2020)	87.5 ± 0.4	81.6 ± 0.7	97.4 ± 0.2	80.8 ± 0.9	86.8
MLDG (Li et al., 2018a)	87.0 ± 1.2	82.5 ± 0.9	96.7 ± 0.3	81.2 ± 0.6	86.8
CORAL (Sun & Saenko, 2016)	86.6 ± 0.8	81.8 ± 0.9	97.1 ± 0.5	82.7 ± 0.6	87.1
MMD (Li et al., 2018b)	88.1 ± 0.8	82.6 ± 0.7	97.1 ± 0.5	81.2 ± 1.2	87.2
DANN (Ganin et al., 2016)	87.0 ± 0.4	80.3 ± 0.6	96.8 ± 0.3	76.9 ± 1.1	85.2
CDANN (Li et al., 2018c)	87.7 ± 0.6	80.7 ± 1.2	97.3 ± 0.4	77.6 ± 1.5	85.8
MTL (Blanchard et al., 2021)	87.0 ± 0.2	82.7 ± 0.8	96.5 ± 0.7	80.5 ± 0.8	86.7
SagNet (Nam et al., 2021)	87.4 ± 0.5	81.2 ± 1.2	96.3 ± 0.8	80.7 ± 1.1	86.4
ARM (Zhang et al., 2021a)	85.0 ± 1.2	81.4 ± 0.2	95.9 ± 0.3	80.9 ± 0.5	85.8
VREx (Krueger et al., 2021)	87.8 ± 1.2	81.8 ± 0.7	97.4 ± 0.2	82.1 ± 0.7	87.2
RSC (Huang et al., 2020)	86.0 ± 0.7	81.8 ± 0.9	96.8 ± 0.7	80.4 ± 0.5	86.2
Fish (Shi et al., 2022)					85.8
Fisher (Rame et al., 2022)	87.9 ± 0.6	80.8 ± 0.5	97.9 ± 0.4	81.1 ± 0.8	86.9
AdaNPC	89.1 ± 0.3	**84.3 ± 0.1**	**98.1 ± 0.4**	83.7 ± 0.5	**88.8**
AdaNPC +NP	**89.2 ± 0.3**	**84.3 ± 0.1**	98.0 ± 0.4	**83.8 ± 0.4**	**88.9**
PACS, Model selection: 'Training-domain' validation set					
Algorithm	**A**	**C**	**P**	**S**	**Avg**
ERM (Vapnik, 1999)	84.7 ± 0.4	80.8 ± 0.6	97.2 ± 0.3	79.3 ± 1.0	85.5
IRM (Arjovsky et al., 2020)	84.8 ± 1.3	76.4 ± 1.1	96.7 ± 0.6	76.1 ± 1.0	83.5
GDRO (Sagawa et al., 2020)	83.5 ± 0.9	79.1 ± 0.6	96.7 ± 0.3	78.3 ± 2.0	84.4
Mixup (Yan et al., 2020)	86.1 ± 0.5	78.9 ± 0.8	97.6 ± 0.1	75.8 ± 1.8	84.6
MLDG (Li et al., 2018a)	85.5 ± 1.4	80.1 ± 1.7	97.4 ± 0.3	76.6 ± 1.1	84.9
CORAL (Sun & Saenko, 2016)	88.3 ± 0.2	80.0 ± 0.5	97.5 ± 0.3	78.8 ± 1.3	86.2
MMD (Li et al., 2018b)	86.1 ± 1.4	79.4 ± 0.9	96.6 ± 0.2	76.5 ± 0.5	84.6
DANN (Ganin et al., 2016)	86.4 ± 0.8	77.4 ± 0.8	97.3 ± 0.4	73.5 ± 2.3	83.6
CDANN (Li et al., 2018c)	84.6 ± 1.8	75.5 ± 0.9	96.8 ± 0.3	73.5 ± 0.6	82.6
MTL (Blanchard et al., 2021)	87.5 ± 0.8	77.1 ± 0.5	96.4 ± 0.8	77.3 ± 1.8	84.6
SagNet (Nam et al., 2021)	87.4 ± 1.0	80.7 ± 0.6	97.1 ± 0.1	80.0 ± 0.4	86.3
ARM (Zhang et al., 2021a)	86.8 ± 0.6	76.8 ± 0.5	97.4 ± 0.3	79.3 ± 1.2	85.1
VREx (Krueger et al., 2021)	86.0 ± 1.6	79.1 ± 0.6	96.9 ± 0.5	77.7 ± 1.7	84.9
RSC (Huang et al., 2020)	85.4 ± 0.8	79.7 ± 1.8	97.6 ± 0.3	78.2 ± 1.2	85.2
Fish (Shi et al., 2022)					85.5
Fisher (Rame et al., 2022)	**88.4 ± 0.2**	78.7 ± 0.7	97.0 ± 0.1	77.8 ± 2.0	85.5
AdaNPC	87.1 ± 1.3	**82.2 ± 0.6**	97.5 ± 0.4	**81.5 ± 0.8**	**87.1**
AdaNPC +NP	86.2 ± 1.2	**82.2 ± 0.6**	**98.1 ± 0.1**	80.2 ± 1.0	86.7

Table 15. Domain generalization accuracy (%) on PACS.

DomainNet, Model selection: 'Test-domain' validation set							
Algorithm	**clip**	**info**	**paint**	**quick**	**real**	**sketch**	**Avg**
ERM (Vapnik, 1999)	58.1 ± 0.3	18.8 ± 0.3	46.7 ± 0.3	12.2 ± 0.4	59.6 ± 0.1	49.8 ± 0.4	40.9
IRM (Arjovsky et al., 2020)	48.5 ± 2.8	15.0 ± 1.5	38.3 ± 4.3	10.9 ± 0.5	48.2 ± 5.2	42.3 ± 3.1	33.9
GDRO (Sagawa et al., 2020)	47.2 ± 0.5	17.5 ± 0.4	33.8 ± 0.5	9.3 ± 0.3	51.6 ± 0.4	40.1 ± 0.6	33.3
Mixup (Yan et al., 2020)	55.7 ± 0.3	18.5 ± 0.5	44.3 ± 0.5	12.5 ± 0.4	55.8 ± 0.3	48.2 ± 0.5	39.2
MLDG (Li et al., 2018a)	59.1 ± 0.2	19.1 ± 0.3	45.8 ± 0.7	13.4 ± 0.3	59.6 ± 0.2	50.2 ± 0.4	41.2
CORAL (Sun & Saenko, 2016)	59.2 ± 0.1	19.7 ± 0.2	46.6 ± 0.3	13.4 ± 0.4	59.8 ± 0.2	50.1 ± 0.6	41.5
MMD (Li et al., 2018b)	32.1 ± 13.3	11.0 ± 4.6	26.8 ± 11.3	8.7 ± 2.1	32.7 ± 13.8	28.9 ± 11.9	23.4
DANN (Ganin et al., 2016)	53.1 ± 0.2	18.3 ± 0.1	44.2 ± 0.7	11.8 ± 0.1	55.5 ± 0.4	46.8 ± 0.6	38.3
CDANN (Li et al., 2018c)	54.6 ± 0.4	17.3 ± 0.1	43.7 ± 0.9	12.1 ± 0.7	56.2 ± 0.4	45.9 ± 0.5	38.3
MTL (Blanchard et al., 2021)	57.9 ± 0.5	18.5 ± 0.4	46.0 ± 0.1	12.5 ± 0.1	59.5 ± 0.3	49.2 ± 0.1	40.6
SagNet (Nam et al., 2021)	57.7 ± 0.3	19.0 ± 0.2	45.3 ± 0.3	12.7 ± 0.5	58.1 ± 0.5	48.8 ± 0.2	40.3
ARM (Zhang et al., 2021a)	49.7 ± 0.3	16.3 ± 0.5	40.9 ± 1.1	9.4 ± 0.1	53.4 ± 0.4	43.5 ± 0.4	35.5
VREx (Krueger et al., 2021)	47.3 ± 3.5	16.0 ± 1.5	35.8 ± 4.6	10.9 ± 0.3	49.6 ± 4.9	42.0 ± 3.0	33.6
RSC (Huang et al., 2020)	55.0 ± 1.2	18.3 ± 0.5	44.4 ± 0.6	12.2 ± 0.2	55.7 ± 0.7	47.8 ± 0.9	38.9
Fish (Shi et al., 2022)							43.4
Fisher (Rame et al., 2022)	58.3 ± 0.5	20.2 ± 0.2	47.9 ± 0.2	13.6 ± 0.3	60.5 ± 0.3	50.5 ± 0.3	41.8
AdaNPC	**59.5 ± 0.1**	**22.2 ± 0.9**	**48.2 ± 0.9**	**15.3 ± 0.2**	**61.2 ± 0.0**	**51.1 ± 0.1**	42.9
DomainNet, Model selection: 'Training-domain' validation set							
Algorithm	**clip**	**info**	**paint**	**quick**	**real**	**sketch**	**Avg**
ERM (Vapnik, 1999)	58.6 ± 0.3	19.2 ± 0.2	47.0 ± 0.3	13.2 ± 0.2	59.9 ± 0.3	49.8 ± 0.4	41.3
IRM (Arjovsky et al., 2020)	40.4 ± 6.6	12.1 ± 2.7	31.4 ± 5.7	9.8 ± 1.2	37.7 ± 9.0	36.7 ± 5.3	28.0
GDRO (Sagawa et al., 2020)	47.2 ± 0.5	17.5 ± 0.4	34.2 ± 0.3	9.2 ± 0.4	51.9 ± 0.5	40.1 ± 0.6	33.4
Mixup (Yan et al., 2020)	55.6 ± 0.1	18.7 ± 0.4	45.1 ± 0.5	12.8 ± 0.3	57.6 ± 0.5	48.2 ± 0.4	39.6
MLDG (Li et al., 2018a)	59.3 ± 0.1	19.6 ± 0.2	46.8 ± 0.2	13.4 ± 0.2	60.1 ± 0.4	50.4 ± 0.3	41.6
CORAL (Sun & Saenko, 2016)	59.2 ± 0.1	19.9 ± 0.2	47.4 ± 0.2	14.0 ± 0.4	59.8 ± 0.2	50.4 ± 0.4	41.8
MMD (Li et al., 2018b)	32.2 ± 13.3	11.2 ± 4.5	26.8 ± 11.3	8.8 ± 2.2	32.7 ± 13.8	29.0 ± 11.8	23.5
DANN (Ganin et al., 2016)	53.1 ± 0.2	18.3 ± 0.1	44.2 ± 0.7	11.9 ± 0.1	55.5 ± 0.4	46.8 ± 0.6	38.3
CDANN (Li et al., 2018c)	54.6 ± 0.4	17.3 ± 0.1	44.2 ± 0.7	12.8 ± 0.2	56.2 ± 0.4	45.9 ± 0.5	38.5
MTL (Blanchard et al., 2021)	58.0 ± 0.4	19.2 ± 0.2	46.2 ± 0.1	12.7 ± 0.2	59.9 ± 0.1	49.0 ± 0.0	40.8
SagNet (Nam et al., 2021)	57.7 ± 0.3	19.1 ± 0.1	46.3 ± 0.5	13.5 ± 0.4	58.9 ± 0.4	49.5 ± 0.2	40.8
ARM (Zhang et al., 2021a)	49.6 ± 0.4	16.5 ± 0.3	41.5 ± 0.8	10.8 ± 0.1	53.5 ± 0.3	43.9 ± 0.4	36.0
VREx (Krueger et al., 2021)	43.3 ± 4.5	14.1 ± 1.8	32.5 ± 5.0	9.8 ± 1.1	43.5 ± 5.6	37.7 ± 4.5	30.1
RSC (Huang et al., 2020)	55.0 ± 1.2	18.3 ± 0.5	44.4 ± 0.6	12.5 ± 0.1	55.7 ± 0.7	47.8 ± 0.9	38.9
Fish (Shi et al., 2022)							42.7
Fisher (Rame et al., 2022)	58.2 ± 0.5	20.2 ± 0.2	47.7 ± 0.3	12.7 ± 0.2	60.3 ± 0.2	50.8 ± 0.1	41.7
AdaNPC	**59.3 ± 0.0**	**22.2 ± 0.9**	**48.3 ± 0.0**	**14.3 ± 0.0**	**61.0 ± 0.1**	**51.4 ± 0.0**	**42.8**

Table 16. Domain generalization accuracy (%) on DomainNet.