Weight Diffusion for Future: Learn to Generalize in Non-Stationary Environments

Mixue Xie
Beijing Institute of Technology
mxxie@bit.edu.cn

Shuang Li[✉]
Beijing Institute of Technology
shuangli@bit.edu.cn

Binhui Xie
Beijing Institute of Technology
binhuixie@bit.edu.cn

Chi Harold Liu
Beijing Institute of Technology
chiliu@bit.edu.cn

Jian Liang
Kuaishou Technology
liangjian03@kuaishou.com

Zixun Sun
Tencent
sunzixun@126.com

Ke Feng
Tencent
richardfeng@tencent.com

Chengwei Zhu
Tencent
chavezzhu@tencent.com

Abstract

Enabling deep models to generalize in non-stationary environments is vital for real-world machine learning, as data distributions are often found to continually change. Recently, evolving domain generalization (EDG) has emerged to tackle the domain generalization in a time-varying system, where the domain gradually evolves over time in an underlying continuous structure. Nevertheless, it typically assumes multiple source domains simultaneously ready. It still remains an open problem to address EDG in the domain-incremental setting, where source domains are non-static and arrive sequentially to mimic the evolution of training domains. To this end, we propose *Weight Diffusion (W-Diff)*, a novel framework that utilizes the conditional diffusion model in the parameter space to learn the evolving pattern of classifiers during the domain-incremental training process. Specifically, the diffusion model is conditioned on the classifier weights of different historical domain *(regarded as a reference point)* and the prototypes of current domain, to learn the evolution from the reference point to the classifier weights of current domain *(regarded as the anchor point)*. In addition, a domain-shared feature encoder is learned by enforcing prediction consistency among multiple classifiers, so as to mitigate the overfitting problem and restrict the evolving pattern to be reflected in the classifier as much as possible. During inference, we adopt the ensemble manner based on a great number of target domain-customized classifiers, which are cheaply obtained via the conditional diffusion model, for robust prediction. Comprehensive experiments on both synthetic and real-world datasets show the superior generalization performance of W-Diff on unseen domains in the future.

1 Introduction

Domain generalization (DG) deals with a fundamental problem in modern machine learning [13, 46, 47], where performance degeneration often occurs when deep models encounter out-of-distribution (OOD) data [38, 56]. The goal of DG is to learn a model that can perform well on unseen target domains by leveraging labeled data from multiple related but different source domains [23, 19, 57, 22].

[✉] Corresponding author. Code is available at `https://github.com/BIT-DA/W-Diff`.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Despite of abundant works on DG and promising progress so far, they typically embark upon the generalization among stationary and discrete environments [29], where the distribution shift among domains is obvious and remains static over time. In contrast, there have emerged several works on evolving domain generalization (EDG) [24, 29, 51, 2, 44, 50] in recent years, where the data distribution gradually shifts in an underlying continuous structure, e.g., the age-related structural changes in the optic nerve in ocular diseases [12]. But most of EDG methods still assume multiple source domains simultaneously ready, which may be impractical in real world. As the data distribution constantly evolves along time, training data from new data distributions will continue to emerge. Hence, equipping models with lifelong learning ability is crucial for their practical applications.

Despite of the fact that existing researches on continual learning [16, 32, 5] have studied the empowerment of lifelong learning, their focus is on maintaining the performance of seen tasks, instead of generalizing on unseen domains in the future. Therefore, it is still an open problem to achieve evolving domain generalization in the domain-incremental setting, where source domains sequentially arrive to mimic the dynamics of training domains. To this end, previous work EvoS [45] models the features from each domain as a Gaussian distribution and proposes to capture the evolving pattern at the feature level by leveraging self-attention mechanism to generate the feature mean and variance for future domain based on those of historical domains. However, the assumption that features adhere to a Gaussian distribution may not always be applicable. Different from EvoS, we propose to excavate evolving pattern at the parameter level and further achieve domain-customized parameter generation.

Inspired by neural network diffusion [42] that there exist specific parameter patterns in optimized model layers and these patterns can be modeled with diffusion model, we propose to capitalize on the strong modeling ability of diffusion models to capture the evolving pattern of optimized classifiers across domains. To achieve this, we propose a *Weight Diffusion (W-Diff)* approach, which is specialized for EDG in the domain-incremental setting. Specifically, to address the problem of inaccessible historical data, we maintain a first-in-first-out (FIFO) queue to store the optimized classifier weights of historical domains. The stored classifier weights of each historical domain serve as a *reference point* to calculate the change between the classifier weights of current domain *(regarded as the anchor point)* and that of corresponding historical domain. The changes of classifier weights between the anchor point and different reference points provide evolving patterns at different time intervals, which can be utilized to make the modeling of evolving patterns more robust.

In addition, for guidance on how to switch from a reference point to the anchor point, we condition the diffusion model on the class prototypes of current domain along with the reference point. Then, the conditional diffusion model is trained to model the distribution of *residual classifier weights*, i.e., the change of classifier weights between the anchor point and reference point. Meanwhile, to reduce the overfitting of the feature encoder and to restrict the evolutionary pattern to be reflected only in the classifier as much as possible, we learn a domain-shared feature space by enforcing the predictions from different classifiers to be consistent. Finally, during the inference stage, we adopt weights ensemble to give robust predictions based on a great number of generated classifier weights by the diffusion model that is conditioned on current class prototypes and different reference points.

Contributions: **1)** We study the under-explored area of evolving domain generalization in the domain-incremental setting and explore the innovative usage of diffusion model for this problem. **2)** We propose a novel weight diffusion (W-Diff) approach to capture the evolutionary pattern at the parameter level, orthogonal to previous feature level approaches. Capitalizing on the strong generative ability of diffusion model, W-Diff can generate customized parameters by controlling the condition and make robust predictions via weights ensemble. **3)** Comprehensive experiments on both synthetic and real-world datasets verify the effectiveness and superiority of W-Diff on generalization.

2 Related Work

Evolving Domain Generalization (EDG) learns the evolving pattern underlying in multiple source domains to achieve generalization capability on the unseen future target domains over time [2, 50, 24, 52, 44, 29]. To name a few, SDE-EDG [50] introduces stochastic differential equations (SDEs) to model the evolving pattern through individual temporal trajectories. DRAIN [2] builds on the Bayesian framework and leverages LSTM to infer the future status of the whole network. However, most of these methods, except for DRAIN, require multiple source domains to be simultaneously available. Very recently, EvoS [45] focuses on EDG with sequentially arriving domains, considering the low efficiency of training the model from scratch once the accumulated domains are updated. It assumes that features for each domain follow a Gaussian distribution and then models the evolving pattern of the feature distribution, while this assumption is not always suitable. Besides, in the generalization process of multiple consecutive target domains, the statistics generated from previous timestamps are used as inputs to the attention mechanism to generate the statistics at next timestamp, which in turn serve as the input for next generation. This manner is likely to cause error accumulation if previous generation is not accurate. Orthogonal to EvoS [45], we proposes to mine the evolving pattern at the model parameter level and further implement domain-oriented parameter generation via controlling the condition of the diffusion model to avoid the potential error accumulation in EvoS.

Continual Learning (CL) focuses on the scenario where the model is trained on a sequence of tasks, and the model is required to adapt to current task and meanwhile maintain the performance on previous tasks [53, 16, 5, 39, 43, 32, 11]. The literature on this field is abundant. Most of the techniques can be categorized into architecture-based [34, 9], representation-based [4, 43, 10], regularization-based [16, 53, 32] and replay-based [5, 39, 11]. In this work, we also continually train models on sequential domains, but the goal is to generalize well on novel domains in the near future.

Parameter Generation has been gaining great interests with the rise of diffusion models [42, 8, 21, 54]. For example, p-diff [42] directly generates high-performing neural network parameters from random noises with a standard latent diffusion model, which verifies the feasibility of modeling the parameter distribution via diffusion models. Nevertheless, p-diff uses unconditional diffusion model and can only generate parameters for in-distribution data. G.pt [28] collects the loss, error or return of task model checkpoints during training as the condition for the diffusion model. However, it is designed for a single dataset to which the training data belongs, thus struggling with distribution shifts. D2NWG [36] uses CLIP [30] to extract features for each sample and leverages Set Transformer [18] to generate dataset encoding from these features. Then, the diffusion model is conditioned on the dataset encoding. But labeled training set samples of a new dataset are required to obtain the dataset encoding, which is infeasible in unlabeled target domains. In addition, ProtoDiff [8] and MetaDiff [54] generate prototype classifiers for the meta-test stage by conditioning the diffusion model on the information (e.g., the prototypes in [8] and the gradients in [54]) from the support set. Yet, they concentrate on few-shot learning and the support set requires labeled data, which is unavailable in the target domain of EDG. By contrast, we aims at capturing the evolving pattern among source domains and leveraging it to enable generalization on future target domains without any labeled target data.

3 Preliminaries

3.1 Problem Formulation

We consider the evolving domain generalization (EDG) in the domain-incremental setting. Formally, during training phase, we are given T sequentially arriving source (training) domains: $\mathcal{S} = \{\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^T\}$, which are collected at timestamps $t_1 < t_2 < \ldots < t_T$, respectively. Each domain is defined as $\mathcal{D}^t = \{\boldsymbol{x}_i^t, y_i^t\}_{i=1}^{N^t}$, $t = 1, \ldots, T$, where $\boldsymbol{x}_i^t$ is the i-th sample from the t-th domain, $y_i^t \in \{0, 1, \ldots, C-1\}$ is the category label of sample $\boldsymbol{x}_i^t$, and N^t, C are the number of training samples in the t-th domain and the number of categories, respectively. In the domain-incremental setting, we can only access current domain $\mathcal{D}^t$ at timestamp t_t and historical domains $\{\mathcal{D}^1, \ldots, \mathcal{D}^{t-1}\}$ are unavailable. This takes into account the data storage burden, privacy concerns and the dynamic evolution of the source domain. Following previous EDG works [24, 29, 2], the label set is the same among domains, but the data distribution of domains is assumed to continuously evolve over time in some patterns. And our goal is to generalize the model, which is composed of a feature encoder $E_{\boldsymbol{\psi}}$ parameterized with $\boldsymbol{\psi}$ and a classifier $H_{\mathbf{W}}$ parameterized with $\mathbf{W}$, on unseen K target (testing) domains in the future: $\mathcal{T} = \{\mathcal{D}^{T+1}, \ldots, \mathcal{D}^{T+K}\}$. To tackle this problem, we propose to model the evolving pattern at the parameter level via the conditional diffusion model and generate customized parameters for future domains by controlling the condition of the diffusion model.

3.2 Diffusion Model

Diffusion models have achieved tremendous success in computer vision by modeling the probability transformation from a prior Gaussian distribution to the target distribution [14, 40]. They typically comprise a diffusion process to progressively add Gaussian noise to data in a multi-step Markov chain and a denoising process to recover data from the noise via reversing the diffusion process.

Diffusion process. Given a clean data point $\boldsymbol{x}_0$ sampled from a real data distribution $q(\boldsymbol{x})$, i.e., $\boldsymbol{x}_0 \sim q(\boldsymbol{x})$, the diffusion process is characterized as a Markov chain which slowly adds random Gaussian noise to $\boldsymbol{x}_0$ in S steps, obtaining a sequence of noisy samples: $\boldsymbol{x}_1, \dots, \boldsymbol{x}_S$. Formally, this process is expressed as

$$q(\boldsymbol{x}_{1:S}|\boldsymbol{x}_0) = \prod_{s=1}^{S} q(\boldsymbol{x}_s|\boldsymbol{x}_{s-1}), \quad q(\boldsymbol{x}_s|\boldsymbol{x}_{s-1}) = \mathcal{N}(\boldsymbol{x}_s; \sqrt{1-\beta_s}\boldsymbol{x}_{s-1}, \beta_s \mathbf{I}), \tag{1}$$

where $\{\beta_s \in (0,1)\}_{s=1}^{S}$ is a variance schedule, $\mathcal{N}$ represents Gaussian distribution, and $\mathbf{I}$ is the identity matrix. And the forward diffused sample at step s, denoted as $\boldsymbol{x}_s$, can be directly obtained in a single step by Eq. (2) without iteratively adding noise:

$$\boldsymbol{x}_s = \sqrt{\bar{\alpha}_s}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_s}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{2}$$

where $\bar{\alpha}_s = \prod_{s'=1}^{s}(1-\beta_{s'})$. When step size S approaches infinity, $\boldsymbol{x}_S$ is equivalent to a data point from an isotropic Gaussian distribution, i.e., the prior Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Denoising process. Given a start noise $\boldsymbol{x}_S \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the denoising process moves backward on the multi-step Markov chain as s decreases from S to 1 to remove the noise at each step s, finally recovering the clean data. Concretely, the formulation of the denoising process at step s is denoted as

$$\boldsymbol{x}_{s-1} = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_s, s) + \sigma_s \boldsymbol{\epsilon} = \frac{1}{\sqrt{1-\beta_s}} \left(\boldsymbol{x}_s - \frac{\beta_s}{\sqrt{1-\bar{\alpha}_s}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}_s, s) \right) + \sigma_s \boldsymbol{\epsilon}, \tag{3}$$

where $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_s, s) = \frac{1}{\sqrt{1-\beta_s}} \left(\boldsymbol{x}_s - \frac{\beta_s}{\sqrt{1-\bar{\alpha}_s}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}_s, s) \right)$ and $\mathcal{E}_{\boldsymbol{\theta}}(\cdot, \cdot)$ is a denoising model parameterized with $\boldsymbol{\theta}$ to estimate the noise. σ_s is a variance hyperparameter that is theoretically set to $\sigma_s^2 = \beta_s$ in most diffusion works [14, 26]. During the training stage, the denoising model $\mathcal{E}_{\boldsymbol{\theta}}$ is trained by minimizing the following loss $\mathcal{L}_{diff}$ to minimize the noise estimation error:

$$\mathcal{L}_{diff} = \mathbb{E}_{\boldsymbol{x}_0, s, \boldsymbol{\epsilon}} \left[\| \boldsymbol{\epsilon} - \mathcal{E}_{\boldsymbol{\theta}}(\sqrt{\bar{\alpha}_s}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_s}\boldsymbol{\epsilon}, s) \|^2 \right]. \tag{4}$$

Conditional diffusion model. The way of conditional diffusion models to generate samples is analogous to the unconditional one, except for the added conditional information. Specifically, as in most conditional diffusion works [25, 17], the denoising model $\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}_s, s)$ is replaced with $\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}_s, s, \mathfrak{c})$, where $\mathfrak{c}$ denotes the condition, e.g., class labels, texts, images, etc. The matched condition $\mathfrak{c}$ regulates the sample generation in a supervised manner to ensure the desired image content. And inspired by the generating of specific image contents via introducing conditional information to diffusion models, we propose to achieve domain-oriented parameter generation by controlling the diffusion condition.

4 Methodology

With the preliminary knowledge of EDG in the domain-incremental setting and diffusion models, we will present the details of Weight Diffusion (W-Diff) in this section. We begin with how to obtain the data for diffusion model training in Section 4.1 and then model the evolving pattern of parameters via the conditional diffusion model in Section 4.2. Finally, the inference procedure of W-Diff to generate customized classifiers is presented in Section 4.3. The overview of W-Diff is illustrated in Fig. 1.

4.1 Per-domain Parameter Fitting in Domain-Incremental Setting

In our approach, we try to capture the evolving pattern in optimized model parameters across domains and further generate customized parameters for target domain via leveraging the learned pattern. Considering the unaffordable training cost if modeling the whole parameters for relatively large models, we choose to excavate the evolutionary pattern in the task-specific head, e,g., the classifier for classification tasks. Nevertheless, the remaining parts of the task model would overfit to current domain and cause degraded generalization if without any processing. To avoid this, we learn a domain-shared feature encoder for all domains and a domain-specific classifier for each domain during the domain-incremental training. As t increases from 1 to T, once the training stage on

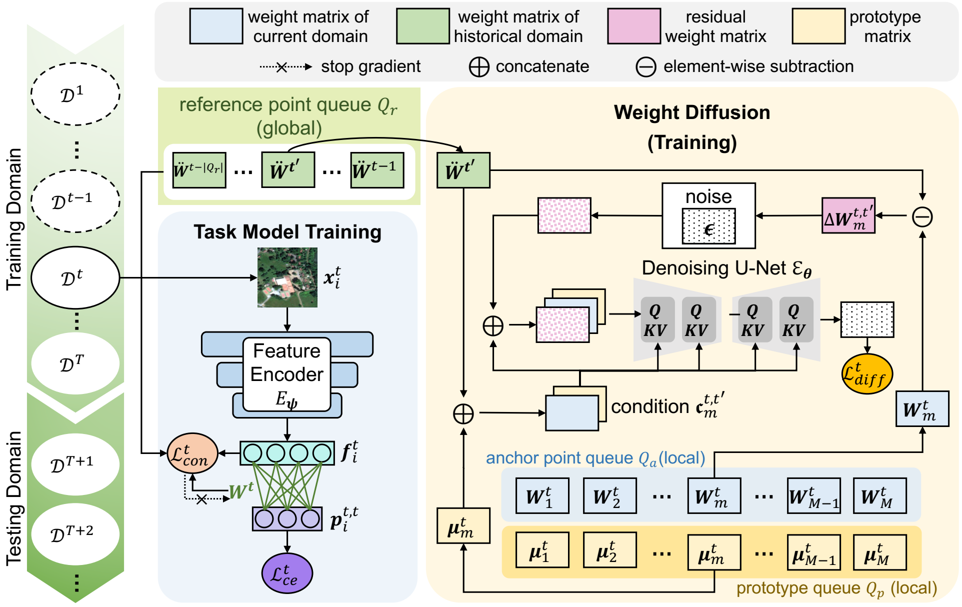


Figure 1: Overview of W-Diff. The reference point queue Q_r stores classifier weights of recent $|Q_r|$ historical domains, and the anchor point queue Q_a and prototype queue Q_p store the updated classifier weights and prototype matrix at each iteration after the warm-up stage on current domain. $\mathcal{L}_{con}^t$ is the prediction consistency loss to learn a domain-shared feature space and $\mathcal{L}_{ce}^t$ is the cross-entropy loss. The conditional diffusion model $\mathcal{E}_{\boldsymbol{\theta}}$ is trained with the noise estimation error loss $\mathcal{L}_{diff}^t$ to model the evolving pattern of classifiers, conditioned on historical reference point and current prototype matrix.

domain $\mathcal{D}^t$ ends, the classifier weights with the best validation performance on the validation set of $\mathcal{D}^t$, denoted as $\ddot{\mathbf{W}}^t \in \mathbb{R}^{C \times d_f}$, is stored in the reference point queue Q_r, where d_f the dimension of deep features output by the feature encoder E_ψ. Q_r is a global FIFO queue with the maximum length L and is used to calculate the change of classifier weights between current domain and a given historical domain in Section 4.2, which reflects the evolving pattern at the parameter level.

Learning Domain-Shared Feature Encoder. In the domain-incremental setting, we can only access the data from current domain, which prohibits us from utilizing conventional DG methods that require to access multiple domains simultaneously to learn domain-invariant feature representations. To tackle this problem, we resort to the stored different classifiers in Q_r. Intuitively, if domain-shared feature representation is learned, classifiers from different domains should give similar predictions for a given data sample. Hence, at timestamp t_t, we train the task model on the t-th domain $\mathcal{D}^t$ by minimizing the consistency loss $\mathcal{L}_{con}^t$ to learn a domain-shared feature space:

$$\mathcal{L}_{con}^t = \frac{1}{1+|Q_r|} \cdot \frac{1}{N^t} \cdot \frac{1}{C} \sum\nolimits_{t'=t-|Q_r|}^{t} \sum\nolimits_{i=1}^{N^t} KL(\bar{\boldsymbol{p}}_i^t \| \boldsymbol{p}_i^{t,t'}),$$

$$\bar{\boldsymbol{p}}_i^t = \frac{1}{1+|Q_r|} \sum\nolimits_{t'=t-|Q_r|}^{t} \boldsymbol{p}_i^{t,t'}, \quad \boldsymbol{p}_i^{t,t'} = \begin{cases} \mathrm{softmax}(\mathrm{sg}(\mathbf{W}^t) \times \boldsymbol{f}_i^t), & t' = t \\ \mathrm{softmax}(\mathrm{sg}(\ddot{\mathbf{W}}^{t'}) \times \boldsymbol{f}_i^t), & t' < t \end{cases}, \tag{5}$$

where $\boldsymbol{f}_i^t = E_\psi(\boldsymbol{x}_i^t) \in \mathbb{R}^{d_f}$, $|\cdot|$ is the length of the object and $\mathrm{sg}(\cdot)$ denotes stopping gradients. $\ddot{\mathbf{W}}^{t'}$ is the stored classifier weights of historical domain $\mathcal{D}^{t'}$ and $\mathbf{W}^t$ is current classifier weights on $\mathcal{D}^t$.

Learning Domain-Specific Classifier. As t increases from 1 to T, domain-specific classifier is directly learned by incrementally training the task model via the supervision loss $\mathcal{L}_{ce}^t$ on domain $\mathcal{D}^t$:

$$\mathcal{L}_{ce}^t = \frac{1}{N^t} \sum\nolimits_{i=1}^{N^t} CrossEntropy\left(\mathrm{softmax}(\mathbf{W}^t \times \boldsymbol{f}_i^t), y_i^t\right). \tag{6}$$

Overall, when training on domain $\mathcal{D}^t$, the task model is optimized by the following total loss $\mathcal{L}_{total}^t$:

$$\mathcal{L}_{total}^t = \mathcal{L}_{ce}^t + \lambda \mathcal{L}_{con}^t, \tag{7}$$

where λ is a tradeoff hyperparameter to balance the two losses.

Collecting Data for Diffusion Model Training. Considering that parameters at the early stage are unstable, we start to collect the training data for diffusion model after the warm-up stage of the task model is over on corresponding domain. Specifically, when training on the t-th domain, the warm-up epochs of the task model account for $\rho \in (0, 1)$ of the total training epochs on domain $\mathcal{D}^t$, where ρ is a hyperparameter. After the warm-up stage, the updated classifier weights $\mathbf{W}^t \in \mathbb{R}^{C \times d_f}$ via back-propagating the gradients of $\mathcal{L}^t_{total}$ and the updated prototype matrix $\boldsymbol{\mu}^t \in \mathbb{R}^{C \times d_f}$ via Eq. (8) are respectively stored into the anchor point queue Q_a and prototype queue Q_p at each iteration.

$$\begin{aligned} \boldsymbol{\mu}^t[c] &:= \frac{n^t \cdot \boldsymbol{\mu}^t[c] + \sum_{i=1}^{B} \boldsymbol{p}_i^{t,t}[c] \cdot \boldsymbol{f}_i^t}{n^t + B}, c = 0, 1, \ldots, C-1, \\ n^t &:= n^t + B. \end{aligned} \tag{8}$$

$\boldsymbol{\mu}^t[c]$ is the c-th row of $\boldsymbol{\mu}^t$, i.e., the prototype of class c on domain $\mathcal{D}^t$ based on all data in the seen batches after the warm-up stage on $\mathcal{D}^t$. B is the batch size of task model. n^t counts the total number of samples in seen batches after the warm-up stage and is initialized to 0 at the start of training on each domain. $\boldsymbol{p}_i^{t,t}[c]$ is the predicted probability of $\boldsymbol{x}_i^t$ belonging to class c via current classifier of domain $\mathcal{D}^t$. Here, using predicted probability instead of ground-truth label to compute prototypes avoids the issue of missing categories in a batch and the inaccessibility of labels in testing domains.

In implementation, Q_a and Q_p are both FIFO queue with the maximum length M, i.e., the training batch size of the diffusion model. When they are full, the training of the conditional diffusion model starts. Note that Q_a and Q_p are only used during the training phase and are locally used on each domain. That is, they are initialized to empty at the beginning of the training stage on each domain.

4.2 Modeling Parameter Evolution Pattern with Conditional Diffusion Model

Having prepared the data for diffusion model training, we can utilize them to learn the evolving pattern of parameters during the domain-incremental training process of the task model. To be specific, we use the difference between the classifier weights of current domain and that of a given historical domain to represent the evolution of parameters:

$$\Delta \mathbf{W}_m^{t,t'} = \mathbf{W}_m^t - \ddot{\mathbf{W}}^{t'}, \tag{9}$$

where $\mathbf{W}_m^t$, $m = 1, 2, \ldots, M$, is the classifier weights of current domain $\mathcal{D}^t$, which is cached in the anchor point queue Q_a when training the task model on $\mathcal{D}^t$. And $\ddot{\mathbf{W}}^{t'}$ is the classifier weights of the historical domain $\mathcal{D}^{t'}$ from the reference point queue Q_r, $t' \in \{t - |Q_r|, \ldots, t-1\}$. Hence, the residual classifier weights $\{\Delta \mathbf{W}_m^{t,t'}\}_{m=1}^{M}$ represent how to evolve from a reference point to the anchor point. Moreover, to guide the evolution, we provide the paired condition for each residual classifier weight matrix $\Delta \mathbf{W}_m^{t,t'}$, where the paired condition is formulated as $\mathfrak{c}_m^{t,t'} = \ddot{\mathbf{W}}^{t'} \oplus \boldsymbol{\mu}_m^t \in \mathbb{R}^{C \times d_f \times 2}$ and $\oplus$ denotes concatenating. The additional condition provides the information about the the starting point and knowledge about the distribution of data in the feature space, which is rightly the anchor point, i.e., the optimized classifier, needs to adapt to. Then, when the task model is incrementally trained on the t-th domain, the conditional diffusion model is also incrementally trained to minimize the following noise estimation error loss $\mathcal{L}^t_{diff}$ in Eq. (10), so as to learn how to generate the desired residual classifier weights when given the reference point and prototype matrix as the condition.

$$\mathcal{L}^t_{diff} = \mathbb{E}_{\ddot{\mathbf{W}}^{t'} \in Q_r, \mathbf{W}_m^t \in Q_a, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), s} \left[\| \boldsymbol{\epsilon} - \mathcal{E}_{\boldsymbol{\theta}}(\sqrt{\bar{\alpha}_s} \cdot \Delta \mathbf{W}_m^{t,t'} + \sqrt{1 - \bar{\alpha}_s} \boldsymbol{\epsilon}, s, \mathfrak{c}_m^{t,t'}) \|^2 \right]. \tag{10}$$

In implementation, the conditional diffusion model adopts the similar U-Net structure as LDM [31] and uses a hybrid conditioning way, i.e., the condition is injected both in the cross-attention and input sides. In Eq. (10), different reference points could enrich the diversity of training data for the conditional diffusion model and provide the evolving pattern at different time intervals.

4.3 Generating Customized Classifiers in Inference Phase

After finishing the training on domain $\mathcal{D}^T$, we can use the conditional diffusion model to generate customized classifiers for a given testing domain $\mathcal{D}^{test}$. Firstly, we calculate the prototype matrix $\boldsymbol{\mu}^{test}$ of domain $\mathcal{D}^{test}$ via $\boldsymbol{\mu}^{test}[c] = \frac{1}{N^{test}} \sum_{i=1}^{N^{test}} \bar{\boldsymbol{p}}_i^{test}[c] \cdot \boldsymbol{f}_i^{test}$, where $\bar{\boldsymbol{p}}_i^{test} = \frac{1}{|Q_r|} \sum_{\ddot{\mathbf{W}}^{t'} \in Q_r} \mathrm{softmax}(\ddot{\mathbf{W}}^{t'} \times \boldsymbol{f}_i^{test})$, $c = 0, 1, \ldots, C-1$. Here, we use the more robust average prediction of multiple classifiers to compute the prototype matrix in the inference phase. Then, given each reference point $\ddot{\mathbf{W}}^{t'}$ in Q_r along with the prototype matrix $\boldsymbol{\mu}^{test}$, we can generate M_g residual classifier weights: $\{\Delta\mathbf{W}_j^{test,t'}\}_{j=1}^{M_g}$ by substituting the denoising net in Eq. (3) with its conditional version and applying the denoising process with condition $\mathbf{c}^{test,t'} = \ddot{\mathbf{W}}^{t'} \oplus \boldsymbol{\mu}^{test}$. Ultimately, we use the following average weight ensemble $\bar{\mathbf{W}}^{test}$ as the final classifier for label predicting on $\mathcal{D}^{test}$:

$$\bar{\mathbf{W}}^{test} = \frac{1}{|Q_r|}\frac{1}{M_g}\sum\nolimits_{\ddot{\mathbf{W}}^{t'}\in Q_r}\sum\nolimits_{j=1}^{M_g}(\ddot{\mathbf{W}}^{t'} + \Delta\mathbf{W}_j^{test,t'}). \tag{11}$$

In this way, we capitalize on the powerful modeling and generating ability of conditional diffusion model to cheaply produce a great number of target-customized classifiers, which offers more robust and accurate predictions. The pseudo codes of training and testing procedures are in Appendix C.

5 Experiments

5.1 Experimental Setup

Benchmark Datasets. We evaluate W-Diff on both synthetic and real-world datasets [2, 48], including two text classification datasets (**Huffpost**, **Arxiv**), three image classification datasets (**Yearbook**, **RMNIST**, **fMoW**) and two multivariate classification datasets (**2-Moons**, **ONP**). Except for synthetic datasets 2-Moons and RMNIST that use the rotation angle as a proxy for time, all other datasets collect real-world data with the distribution shift over time. Following [45], the number of source and target domains is set as Yearbook: ($T = 16, K = 5$), RMNIST: ($T = 6, K = 3$), fMoW: ($T = 13, K = 3$), Huffpost: ($T = 4, K = 3$), Arxiv: ($T = 9, K = 7$), 2-Moons: ($T = 9, K = 1$), ONP: ($T = 5, K = 1$). For each source domain, we randomly divide the data into training and validation sets in the ratio of $9:1$. For more details on datasets, please refer to Appendix D.1.

Network Details. For the task model, we follow the usage in [48, 45]. For the conditional diffusion model, we implement it in a U-Net similar to LDM [31]. Please refer to Appendix D.2 for details.

Training Details. For all datasets, we set the batch size $B = 64$, the loss tradeoff $\lambda = 10$ and the maximum length $L = 8$ for the reference point queue Q_r. To optimize the task model, we adopt the Adam optimizer with momentum 0.9. As for the warm-up hyperparameter ρ, we $\rho = 0.6$ for Huffpost, fMoW and $\rho = 0.2$ for Arxiv, Yearbook, RMNIST, 2-Moons, ONP. For the conditional diffusion model, we set the maximum diffusion step $S = 1000$ and use the AdamW optimizer with batch size $M = 32$, where M is also the maximum length of queue Q_a and Q_p. And the number of generated residual classifier weights based on each reference point is set to $M_g = 32$ in the inference stage. All experiments are conducted using the PyTorch packages and run on a single NVIDIA GeForce RTX 4090 GPU with 24GB memory. Three independent experiments with different random seeds are repeated for each task to report the mean and standard deviation (std) of accuracy, which is denoted in the format of "mean ± std" in the table. Please refer to Appendix D.3 for more details.

Table 1: Accuracy (%) on Huffpost and Arxiv. The best and second best results in the incremental setup are bolded and underlined, respectively. (Huffpost: $K = 3$, Axriv: $K = 7$)

Method	Incremental training	Access multiple domains	Huffpost Accuracy (%) ↑			Arxiv Accuracy (%) ↑		
			$\mathcal{D}^{T+1}$	OOD avg.	OOD worst	$\mathcal{D}^{T+1}$	OOD avg.	OOD worst
Offline	✗	✓	72.74	71.50	69.63	57.49	52.38	49.28
IRM [1]	✗	✓	71.04	70.31	68.97	51.11	45.89	42.86
CORAL [37]	✗	✓	71.34	70.08	68.68	50.98	45.77	42.71
Mixup [55]	✗	✓	73.34	71.16	69.29	57.58	52.77	49.62
LISA [49]	✗	✓	72.19	70.24	68.60	56.53	52.41	49.67
GI [24]	✗	✓	68.06	66.32	64.64	53.43	49.19	46.13
IncFinetune	✓	✗	73.57	71.98	69.80	56.22	52.43	49.37
Mixup [55]	✓	✗	73.07	71.52	69.44	$\underline{56.64}$	52.95	49.97
EWC [16]	✓	✗	$\underline{73.64}$	71.53	68.99	56.60	52.78	49.73
SI [53]	✓	✗	72.58	71.50	69.61	49.98	47.27	44.77
A-GEM [5]	✓	✗	72.23	71.16	69.10	52.02	48.91	46.03
DRAIN [2]	✓	✗	73.42	71.75	69.69	56.04	52.07	48.97
EvoS [45]	✓	✗	73.42	**72.36**	$\underline{70.19}$	56.60	$\underline{53.15}$	$\underline{50.19}$
W-Diff	✓	✗	**73.91**±0.19	$\underline{72.29}$±0.14	**70.40**±0.12	**56.66**±0.11	**53.43**±0.10	**50.70**±0.20

Table 2: Accuracy (%) on Yearbook, RMNIST and fMoW. The best and second best results in the incremental setup are bolded and underlined. (Yearbook: $K = 5$, RMNIST: $K = 3$, fMoW: $K = 3$)

Method	Incremental training	Access multiple domains	Yearbook Accuracy (%) ↑			RMNIST Accuracy (%) ↑			fMoW Accuracy (%) ↑		
			$\mathcal{D}^{T+1}$	OOD avg.	OOD worst	$\mathcal{D}^{T+1}$	OOD avg.	OOD worst	$\mathcal{D}^{T+1}$	OOD avg.	OOD worst
Offline	✗	✓	89.30	88.46	86.81	98.15	92.14	83.89	72.43	59.76	49.85
IRM [1]	✗	✓	97.09	94.52	92.58	95.10	85.05	72.52	64.77	54.92	46.51
CORAL [37]	✗	✓	95.94	91.79	88.84	93.04	79.10	62.96	62.14	51.42	42.19
Mixup [55]	✗	✓	94.98	91.12	88.35	97.11	89.66	79.63	70.27	57.73	48.04
LISA [49]	✗	✓	95.51	92.97	91.29	96.21	87.04	75.15	70.05	55.52	44.61
CDOT [27]	✗	✓	95.17	92.90	91.46	97.96	90.19	79.67	-	-	-
CIDA [41]	✗	✓	92.36	90.67	88.45	97.43	89.19	78.32	-	-	-
GI [24]	✗	✓	97.42	96.37	95.73	97.78	91.00	82.46	61.62	50.83	42.78
LSSAE [29]	✗	✓	93.93	92.12	88.75	96.73	90.36	82.13	59.15	48.66	41.38
IncFinetune	✓	✗	96.61	94.72	93.48	98.62	92.80	84.61	65.52	53.99	45.23
Mixup [55]	✓	✗	90.21	89.83	88.43	98.43	92.38	83.45	64.84	52.00	42.54
SimCLR [6]	✓	✗	95.94	93.07	89.65	98.23	90.98	81.05	64.97	53.20	44.71
SwAV [3]	✓	✗	**97.37**	94.27	91.44	98.08	90.85	80.96	66.47	54.51	45.29
EWC [16]	✓	✗	97.18	95.12	93.64	98.56	92.02	82.80	66.23	54.55	45.80
SI [53]	✓	✗	97.09	94.67	93.48	98.61	93.27	85.65	66.61	54.89	46.46
A-GEM [5]	✓	✗	94.36	90.96	88.88	95.99	86.95	75.45	54.54	47.61	41.13
SGP [32]	✓	✗	95.65	92.92	91.39	97.12	88.97	78.05	-	-	-
DRAIN [2]	✓	✗	96.23	94.71	93.73	98.52	93.09	85.75	67.22	55.05	46.24
EvoS [45]	✓	✗	**97.37**	**95.53**	**94.78**	98.64	93.84	87.04	67.18	54.64	45.86
W-Diff	✓	✗	97.32±0.23	95.03±0.17	94.05±0.31	**98.70**±0.04	**94.12**±0.12	**87.36**±0.20	**68.80**±0.19	**55.86**±0.16	**46.51**±0.23

Table 3: (a): Error rate (%) on 2-Moons and ONP ($K = 1$). (b): Ablation study on RMNIST.

Method	Error rate (%) ↓	
	2-Moons	ONP
Offline	22.4±4.6	33.8±0.6
LastDomain	14.9±0.9	36.0±0.2
IncFinetune	16.7±3.4	34.0±0.3
CDOT [27]	9.3±1.0	34.1±0.0
CIDA [41]	10.8±1.6	34.7±0.6
GI [24]	3.5±1.4	36.4±0.8
DRAIN [2]	3.2±1.2	38.3±1.2
EvoS [45]	2.5±1.0	33.1±0.6
W-Diff	**1.5**±1.0	**32.9**±0.5

(a)

Method	Conditioning way for $\mathcal{E}_{\boldsymbol{\theta}}$	Loss	Computed $\bar{\mathbf{W}}^{test}$ based on $*$			Accuracy (%) ↑		
		$\mathcal{L}_{con}^t$	Q_r	noise-added Q_r	diffusion	$\mathcal{D}^{T+1}$	OOD avg.	OOD worst
variant A	hybrid	-	-	-	✓	98.48	91.69	81.87
variant B	-	✓	-	-	-	98.55	93.70	86.91
variant C	-	✓	✓	-	-	98.47	93.82	87.28
variant D	-	✓	-	✓	-	98.55	93.78	87.13
variant E	cross_attn	✓	-	-	✓	97.98	91.68	83.19
variant F	concat	✓	-	-	✓	98.44	92.93	85.03
W-Diff	hybrid	✓	-	-	✓	**98.70**	**94.12**	**87.36**

(b)

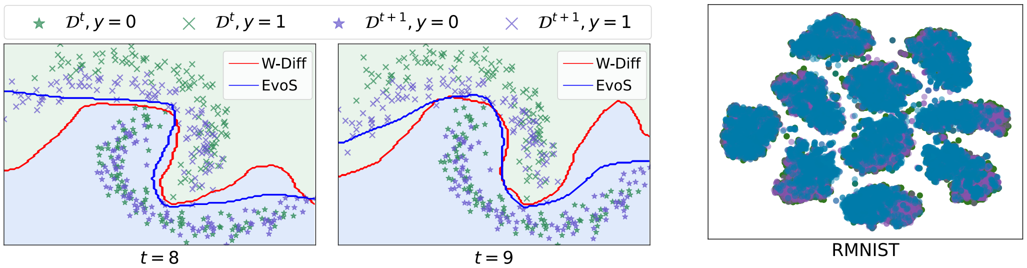
Evaluation Metrics. We report the generalization performance on K target domains in the future, including the average accuracy "OOD avg." ($\frac{1}{K}\sum_{k=1}^{K}\text{Accuracy}(\mathcal{D}^{T+k})$) and the worst accuracy "OOD worst" ($\min_{k\in\{1,\dots,K\}}\text{Accuracy}(\mathcal{D}^{T+k})$) on K target domains and the accuracy on $\mathcal{D}^{T+1}$.

5.2 Main Results

We provide the quantitative results in Table 1, 2, 3(a), where the results of baselines in the non-incremental and incremental scenarios are reported from [45]. For ONP dataset, we notice that previous continuous domain adaptation methods (CDOT and CIDA) and EDG methods (GI and DRAIN) all perform worse than the Offline method that trains the task model on the cumulation of all source domains. In contrast, our W-Diff still obtains superior accuracy to previous state-of-the-art method (EvoS) on this challenging dataset, which validates the superiority of W-Diff. Besides, our W-Diff also achieves the best results on Huffpost, Arxiv, RMNIST and fMoW, in terms of the OOD worst accuracy. These results benefit from the modeling of parameter evolution pattern and the more robust predictions via the weight ensemble based on the conditional diffusion model. For Yearbook dataset, DRAIN, which models the evolution of whole model parameters via LSTM, is inferior to EvoS, which models the evolution of domain-level feature distribution. It suggests that modeling the evolving pattern at the feature level may be more appropriate for Yearbook, which also explains why W-Diff does not obtain the state-of-the-art performance on Yearbook. But our W-Diff still obviously outperforms DRAIN. Overall, we can observe that W-Diff surpasses the baselines in the incremental-training setup on six out of seven datasets, which shows the superiority of W-Diff.

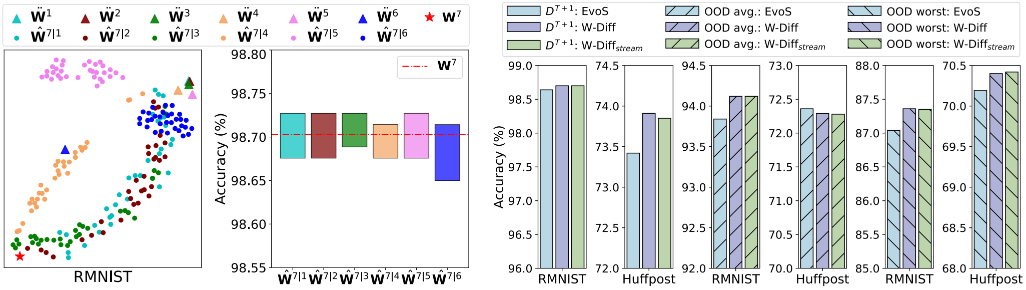
5.3 Analytical Experiments

Ablation Study. Firstly, the significant performance drop of variant A in Table 3(b) suggests that learning domain-invariant feature representations is necessary for EDG in the domain-incremental setting. Otherwise, the feature encoder could easily overfit to current domain, prohibiting the task model from generalization. Then, we try different ways to construct the average weight ensemble $\bar{\mathbf{W}}^{test}$, including variant C which directly uses the historical classifier weights in Q_r, i.e., $\bar{\mathbf{W}}^{test} = \frac{1}{|Q_r|}\sum_{\ddot{\mathbf{W}}^{t'}\in Q_r}\ddot{\mathbf{W}}^{t'}$, and variant D which augments classifier weights by adding small



(a) Visualization of decision boundary on 2-Moons. (b) t-SNE visualization of features.

Figure 2: (a): Decision boundary of EvoS [45] and W-Diff on 2-Moons, where we incrementally train the model until the t-th domain and then visualize the decision boundary for future domain $\mathcal{D}^{t+1}$. (b): visualization of features from target domains, where different colors represent different domains.



(a) Visualization of classifier weights. (b) Results when evaluating in batch-data stream.

Figure 3: (a): Visualization of classifier weights for $\mathcal{D}^{T+1}, T=6$, on RMNIST and their accuracy range. $\ddot{\mathbf{W}}^{t'}, t'=1,\ldots,6$, is the reference point from Q_r, $\hat{\mathbf{W}}^{7|t'}$ is the generated M_g classifier weights based on $\ddot{\mathbf{W}}^{t'}$, and $\mathbf{W}^7$ is the average weights of $\mathcal{D}^7$ fine-tuned classifier weights in the last 200 iterations. (b): Accuracy of EvoS and W-Diff on RMNIST and Huffpost datastes, where W-Diff$_{stream}$ denotes W-Diff is evaluated with batch-data stream for each target domain.

noises to the weights in Q_r, i.e., $\bar{\mathbf{W}}^{test} = \frac{1}{|Q_r|}\frac{1}{M_g}\sum_{\ddot{\mathbf{W}}^{t'}\in Q_r}\sum_{j=1}^{M_g}(\ddot{\mathbf{W}}^{t'} + noise_j), noise_j \sim \mathrm{Uniform}(-0.01, 0.01)$. The inferior results of variant B, C and D indicate that W-Diff benefits from generating meaningful and customized classifier weights via controlling the condition of diffusion model. Finally, different conditioning ways for the diffusion model are explored, including variant E which injects the condition only in the cross-attention, and variant F which injects the condition only in the input by concatenating the condition with diffused residual classifier weights. Results of variant E and F are both unsatisfactory. This is probably due to the large gap between the residual classifier weights in the input side and the full classifier weights in the condition side, which makes the information interaction hard. And injecting the condition only on the input side can lead to insufficient information interaction. Empirically, we find that the hybrid manner works best.

Decision Boundary Visualization on Future Domain. In Fig. 2(a), the model is incrementally trained until the training stage on the t-th domain $\mathcal{D}^t$ finishes. Then we visualize the decision boundary on the next future domain $\mathcal{D}^{t+1}$, $t=8,9$. From the results, we can see that the decision boundary of W-Diff adapts to the evolution of domains better than that of EvoS, which shows the superiority of W-Diff in addressing evolving domain generalization in the domain-incremental setup.

t-SNE Visualization of Features. In this qualitative experiment, we visualize the features of future target domains for RMNIST dataset to show the effectiveness of $\mathcal{L}^t_{con}$. From Fig. 2(b), we can observe that features from different target domains align well. To some extent, it verifies the effectiveness of $\mathcal{L}^t_{con}$ to learn a domain-shared feature space, which contributes to the mitigation of distribution shift.

Visualization of Generated Classifier Weights. In Fig. 3(a), we plot the generated classifier weights for domain $\mathcal{D}^{T+1}, T=6$, on RMNIST, as well as the accuracy range on $\mathcal{D}^{T+1}$ of generated classifier weights based on different reference points. In Fig. 3(a), some generated weights locate close to the average fine-tuned weights $\mathbf{W}^7$, showing that W-Diff generates domain-customized classifiers.

Table 4: Accuracy (%) of W-Diff on RMNIST dataset using different conditions. ($K = 3$)

Method	Condition	Accuracy (%) ↑		
		$\mathcal{D}^{T+1}$	OOD avg.	OOD worst
W-Diff	reference point $\oplus$ prototype matrix	**98.70**	94.12	87.36
W-Diff	scaled reference point[‡] $\oplus$ prototype matrix	98.69	**94.17**	**87.46**

[‡] denotes the reference point is scaled by the factor $\eta = 1.5 - \frac{1}{1+e^{-\Delta t}}$, where Δt is the timestamp difference between the reference point and anchor point.

Besides, $\hat{\mathbf{W}}^{7|1}$, $\hat{\mathbf{W}}^{7|2}$, $\hat{\mathbf{W}}^{7|3}$ are similar, while $\hat{\mathbf{W}}^{7|4}$, $\hat{\mathbf{W}}^{7|5}$, $\hat{\mathbf{W}}^{7|6}$ are different, due to the more pronounced differences among reference points $\ddot{\mathbf{W}}^{4}$, $\ddot{\mathbf{W}}^{5}$, and $\ddot{\mathbf{W}}^{6}$. This implies that the evolution pattern may be different at different time intervals. And these diverse and high-performing generated weights based on different reference points could conduce to more robust predictions.

Evaluating in Batch-Data Stream. In addition to the inference way in Section 4.3, we also provide another version, where the data in target domain arrives batch by batch. Concretely, we use the iterative manner in Eq. (8) along with the average prediction in Section 4.3 to update the prototype matrix $\boldsymbol{\mu}^{test}$, once a batch of data from $\mathcal{D}^{test}$ arrives. Then, we compute the average weight ensemble $\bar{\mathbf{W}}^{test}$ via Eq. 11 for this data batch. Fig. 3(b) shows the results on RMNIST and Huffpost. The two manners present similar results and users can choose appropriate manner based on their scenarios.

Equipping Condition with Timestamp Difference. In this part, we try to explicitly incorporate the timestamp difference between the anchor point and reference point into the condition of diffusion model. Concretely, we scale the reference point $\ddot{\mathbf{W}}^{t'}$ in the condition $\mathfrak{c}^{t,t'}$ by a factor η, where $\eta = 1.5 - \frac{1}{1+e^{-\Delta t}}$ and $\Delta t = t_t - t_{t'}$ is the timestamp difference between reference and anchor points. More distant reference points have larger Δt and are weakened. The results on RMNIST dataset are given in Table 4, where the average and worst accuracies of target domains improve slightly. The insignificant performance improvement may be due to the fact that our approach implicitly takes the timestamp difference into account via the domain-incremental training and residual classifier weights.

Results with Larger Backbones. We try larger backbones on the fMoW dataset by replacing the DenseNet-121 with DenseNet-169/201/161 [15], respectively. The results are provided in Table 5. When applying lagers backbones, our method still works well and further performance improvements are obtained. This benefits from the consideration of only modeling the evolution of classifier weights, instead of the whole network parameters. Otherwise, the training difficulty and huge memory burden from the conditional diffusion model would be unbearable, despite of the greater feature extraction capability of larger backbones.

Table 5: Accuracy (%) of W-Diff on fMoW dataset with different backbones. ($K = 3$)

Backbones	parameters	Accuracy (%) ↑		
		$\mathcal{D}^{T+1}$	OOD avg.	OOD worst
DenseNet-121 (growth rate=32)	64.5MB	68.80	55.86	46.51
DenseNet-169 (growth rate=32)	114.4MB	70.20	56.81	**47.50**
DenseNet-201 (growth rate=32)	161.8MB	70.38	56.28	46.40
DenseNet-161 (growth rate=48)	230.8MB	**71.28**	**57.36**	47.33

6 Conclusion

This work delves into the under-explored problem of evolving domain generalization in the domain-incremental setting, where the source domain is also non-stationary and dynamically evolves. To tackle this, we propose a Weight Diffusion (W-Diff) approach to capture the evolving pattern across domains at the parameter level and further generate customized classifiers for future domains. W-Diff innovatively leverages the conditional diffusion model to learn the evolution of classifiers from historical domain to current domain, conditioned on the historical classifier weights and current prototype matrix. Extensive results on synthetic and real-world datasets verify the efficacy of W-Diff.

Acknowledgements

This paper was supported by National Key R&D Program of China (No. 2021YFB3301503), the National Natural Science Foundation of China (No. 62376026), and also sponsored by Beijing Nova Program (No. 20230484296), CCF-Tencent Rhino-Bird Open Research Fund and KuaiShou.

References

[1] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. *arXiv:1907.02893*, 2019.

[2] G. Bai, C. Ling, and L. Zhao. Temporal domain generalization with drift-aware dynamic neural networks. In *ICLR*, 2023.

[3] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.

[4] H. Cha, J. Lee, and J. Shin. Co^{2}l: Contrastive continual learning. In *ICCV*, pages 9496–9505, 2021.

[5] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with A-GEM. In *ICLR*, 2019.

[6] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607, 2020.

[7] L. Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. *SPM*, 29(6):141–142, 2012.

[8] Y. Du, Z. Xiao, S. Liao, and C. Snoek. Protodiff: Learning to learn prototypical networks by task-guided diffusion. In *NeurIPS*, 2023.

[9] S. Ebrahimi, F. Meier, R. Calandra, T. Darrell, and M. Rohrbach. Adversarial continual learning. In *ECCV*, volume 12356, pages 386–402, 2020.

[10] E. Fini, V. G. T. da Costa, X. Alameda-Pineda, E. Ricci, K. Alahari, and J. Mairal. Self-supervised models are continual learners. In *CVPR*, pages 9611–9620, 2022.

[11] R. Gao and W. Liu. DDGR: continual learning with deep diffusion-based generative replay. In *ICML*, pages 10744–10763, 2023.

[12] H. E. Grossniklaus, J. M. Nickerson, H. F. Edelhauser, L. A. Bergman, and L. Berglin. Anatomic alterations in aging and age-related diseases of the eye. *IOVS*, 54(14):ORSF23–ORSF27, 2013.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[14] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[15] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 2261–2269, 2017.

[16] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *arXiv:1612.00796*, 2016.

[17] G. Kwon and J. C. Ye. Diffusion-based image translation using disentangled style and content representation. In *ICLR*, 2023.

[18] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, volume 97, pages 3744–3753, 2019.

[19] H. Li, S. J. Pan, S. Wang, and A. C. Kot. Domain generalization with adversarial feature learning. In *CVPR*, pages 5400–5409, 2018.

[20] S. Li, C. H. Liu, Q. Lin, Q. Wen, L. Su, G. Huang, and Z. Ding. Deep residual correction network for partial domain adaptation. *TPAMI*, 43(7):2329–2344, 2021.

[21] S. Lutati and L. Wolf. Ocd: Learning to overfit with conditional diffusion models. In *ICML*, pages 23157–23169, 2023.

[22] F. Lv, J. Liang, S. Li, J. Zhang, and D. Liu. Improving generalization with domain convex game. In *CVPR*, pages 24315–24324, 2023.

[23] K. Muandet, D. Balduzzi, and B. Schölkopf. Domain generalization via invariant feature representation. In *ICML*, pages 10–18, 2013.

[24] A. Nasery, S. Thakur, V. Piratla, A. De, and S. Sarawagi. Training for the future: A simple gradient interpolation loss to generalize along time. In *NeurIPS*, pages 19198–19209, 2021.

[25] H. Ni, C. Shi, K. Li, S. X. Huang, and M. R. Min. Conditional image-to-video generation with latent flow diffusion models. In *CVPR*, pages 18444–18455, 2023.

[26] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, pages 8162–8171, 2021.

[27] G. Ortiz-Jiménez, M. E. Gheche, E. Simou, H. P. Maretic, and P. Frossard. CDOT: continuous domain adaptation using optimal transport. *arXiv:1909.11448*, 2019.

[28] W. S. Peebles, I. Radosavovic, T. Brooks, A. A. Efros, and J. Malik. Learning to learn with generative models of neural network checkpoints. *arXiv:2209.12892*, 2022.

[29] T. Qin, S. Wang, and H. Li. Generalizing to evolving domains with latent structure-aware sequential autoencoder. In *ICML*, pages 18062–18082, 2022.

[30] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, volume 139, pages 8748–8763, 2021.

[31] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10674–10685, 2022.

[32] G. Saha and K. Roy. Continual learning with scaled gradient projection. In *AAAI*, pages 9677–9685, 2023.

[33] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv:1910.01108*, 2019.

[34] J. Serrà, D. Suris, M. Miron, and A. Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, volume 80, pages 4555–4564, 2018.

[35] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

[36] B. Soro, B. Andreis, H. Lee, S. Chong, F. Hutter, and S. J. Hwang. Diffusion-based neural network weights generation. *arXiv:2402.18153*, 2024.

[37] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV Workshops*, pages 443–450, 2016.

[38] R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt. Measuring robustness to natural distribution shifts in image classification. In *NeurIPS*, 2020.

[39] R. Tiwari, K. Killamsetty, R. K. Iyer, and P. Shenoy. GCR: gradient coreset based replay buffer selection for continual learning. In *CVPR*, pages 99–108, 2022.

[40] A. Ulhaq, N. Akhtar, and G. Pogrebna. Efficient diffusion models for vision: A survey. *arXiv:2210.09292*, 2022.

[41] H. Wang, H. He, and D. Katabi. Continuously indexed domain adaptation. In *ICML*, volume 119, pages 9898–9907, 2020.

[42] K. Wang, Z. Xu, Y. Zhou, Z. Zang, T. Darrell, Z. Liu, and Y. You. Neural network diffusion. *arXiv:2402.13144*, 2024.

[43] Y. Wang, Z. Huang, and X. Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. In *NeurIPS*, 2022.

[44] B. Xie, Y. Chen, J. Wang, K. Zhou, B. Han, W. Meng, and J. Cheng. Enhancing evolving domain generalization through dynamic latent representations. In *AAAI*, pages 16040–16048, 2024.

[45] M. Xie, S. Li, L. Yuan, C. H. Liu, and Z. Dai. Evolving standardization for continual domain generalization over temporal drift. In *NeurIPS*, 2023.

[46] L. Yang, Y. Han, X. Chen, S. Song, J. Dai, and G. Huang. Resolution adaptive networks for efficient inference. In *CVPR*, pages 2366–2375, 2020.

[47] L. Yang, H. Jiang, R. Cai, Y. Wang, S. Song, G. Huang, and Q. Tian. Condensenet V2: sparse feature reactivation for deep networks. In *CVPR*, pages 3569–3578, 2021.

[48] H. Yao, C. Choi, B. Cao, Y. Lee, P. W. Koh, and C. Finn. Wild-time: A benchmark of in-the-wild distribution shift over time. In *NeurIPS*, 2022.

[49] H. Yao, Y. Wang, S. Li, L. Zhang, W. Liang, J. Zou, and C. Finn. Improving out-of-distribution robustness via selective augmentation. In *ICML*, pages 25407–25437, 2022.

[50] Q. Zeng, C. Shui, L.-K. Huang, P. Liu, X. Chen, C. Ling, and B. Wang. Latent trajectory learning for limited timestamps under distribution shift over time. In *ICLR*, 2023.

[51] Q. Zeng, W. Wang, F. Zhou, C. Ling, and B. Wang. Foresee what you will learn: Data augmentation for domain generalization in non-stationary environments. *arXiv:2301.07845*, 2023.

[52] Q. Zeng, W. Wang, F. Zhou, G. Xu, R. Pu, C. Shui, C. Gagné, S. Yang, C. X. Ling, and B. Wang. Generalizing across temporal domains with koopman operators. In *AAAI*, pages 16651–16659, 2024.

[53] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995, 2017.

[54] B. Zhang, C. Luo, D. Yu, X. Li, H. Lin, Y. Ye, and B. Zhang. Metadiff: Meta-learning with conditional diffusion for few-shot learning. In *AAAI*, pages 16687–16695, 2024.

[55] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[56] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy. Domain generalization: A survey. *TPAMI*, 45(4):4396–4415, 2023.

[57] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang. Domain generalization with mixstyle. In *ICLR*, 2021.

Appendix Contents

A. Broader Impacts & Limitations

B. Notation Table

C. Algorithm of W-Diff

D. Experimental Setup Details

E. More Results

A Broader Impacts & Limitations

Broader Impacts. In this work, we explore the evolving domain generalization in the domain incremental setting. The ability to continually learn from dynamic source domains and leverage the learned evolving pattern to generalize on unseen domains in the future may benefit relevant non-stationary scenarios, e.g., advertisement recommendation with continually emerging new training data and autonomous driving with distribution shift over time or geographical position, etc. It reduces the time and cost for labeling data of target domain and avoids the low efficiency of training the model from scratch with all saved domains once new training domain is available. Yet, for high-security demanding scenarios, the prediction from the model should be adopted with caution to avoid severe accidents, as failures can occur in our method when facing significant distribution shifts.

Limitations. Our work presents a way to capture the evolving pattern at the parameter level via capitalizing on the powerful modeling ability of conditional diffusion model. Yet, like any research, our work is not absolutely perfect. There are indeed some limitations that should be acknowledged. Firstly, the task considered in this paper limits to the classification. In the future, we may extend our method to more diverse tasks, e.g., regression tasks. Besides, considering the training cost, we only model the evolution of classifiers. Perhaps, it is feasible to consider more parameters by mapping them into a low-dimensional latent space, but achieving the accurate encoding and decoding is not easy. We leave this for a future work.

B Notation Table

Given the large number of notations used throughout the paper, we provide an overall notation description in Table 6 to ease the burden on readers.

Table 6: Notation description

Data-related	
T	the number of source domains
K	the number of target domains
C	the number of categories
c	the index of categories ($c \in \{0, 1, \dots, C-1\}$)
t, t'	the index of domains ($t, t' \in \{1, 2, \dots, T+K\}$)
$\mathcal{D}^t$	the t-th domain
t_t	the timestamp which the t-th domain is collected at
N^t	the number of samples in the t-th domain
$\boldsymbol{x}_i^t$	the i-th sample in the t-th domain
y_i^t	the category label of the i-th sample in the t-th domain ($y_i^t \in \{0, 1, \dots, C-1\}$)
$\mathcal{S}$	a sequence of source (training) domains ($\mathcal{S} = \{\mathcal{D}^1, \mathcal{D}^2, \dots, \mathcal{D}^T\}$)
$\mathcal{T}$	a sequence of target (testing) domains ($\mathcal{T} = \{\mathcal{D}^{T+1}, \dots, \mathcal{D}^{T+K}\}$)
$\mathcal{D}^{test}$	a testing domain ($\mathcal{D}^{test} \in \mathcal{T}$)
Task model-related	
B	batch size of task model
$E_{\boldsymbol{\psi}}$	feature extractor
$\boldsymbol{\psi}$	parameters of the feature extractor
d_f	the dimension of of deep features output by the feature encoder
$\boldsymbol{f}_i^t$	the deep features of the i-th sample in the t-th domain ($\boldsymbol{f}_i^t = E_{\boldsymbol{\psi}}(\boldsymbol{x}_i^t) \in \mathbb{R}^{d_f}$)
$H_{\mathbf{W}}$	classifier

$\mathbf{W}$	parameters of the classifier
Q_r	the reference point queue
L	the maximum length of Q_r
$\ddot{\mathbf{W}}^{t'}$	the saved classifier weights in Q_r for the t'-th domain ($\ddot{\mathbf{W}}^{t'} \in \mathbb{R}^{C \times d_f}$)
$\mathbf{W}^t$	the current classifier weights of the t-th domain ($\mathbf{W}^t \in \mathbb{R}^{C \times d_f}$)
$\boldsymbol{\mu}^t$	the current prototype matrix of the t-th domain ($\boldsymbol{\mu}^t \in \mathbb{R}^{C \times d_f}$)
$\boldsymbol{\mu}^t[c]$	the c-th row of $\boldsymbol{\mu}^t$
n^t	the total number of samples in seen batches after the warm-up stage on domain $\mathcal{D}^t$
$\boldsymbol{p}_i^{t,t'}$	the prediction for the i-th sample in the t-th domain by the classifier of domain $\mathcal{D}^{t'}$
$\boldsymbol{p}_i^{t,t'}[c]$	the c-th element of $\boldsymbol{p}_i^{t,t'}$
$\bar{\boldsymbol{p}}_i^t$	the average prediction for the i-th sample in the t-th domain
Q_a	the anchor point queue, which stores the classifier weights of current domain
Q_p	the prototype queue, which stores the prototype matrices of current domain
M	the maximum length of Q_a and Q_p
m	the index of the object in Q_a and Q_p ($m \in \{1, 2, \cdots, M\}$)
$\boldsymbol{\mu}_m^t$	the m-th prototype matrix in the prototype queue Q_p of the t-th domain
$\mathbf{W}_m^t$	the m-th classifier weights in the anchor point queue Q_a of the t-th domain
$\Delta\mathbf{W}_m^{t,t'}$	residual classifier weights ($\Delta\mathbf{W}_m^{t,t'} = \mathbf{W}_m^t - \ddot{\mathbf{W}}^{t'}$)
$\mathfrak{c}_m^{t,t'}$	condition of the conditional diffusion model ($\mathfrak{c}_m^{t,t'} = \ddot{\mathbf{W}}^{t'} \oplus \boldsymbol{\mu}_m^t \in \mathbb{R}^{C \times d_f \times 2}$)
ρ	warm-up hyperparameter ($\rho \in (0, 1)$)
λ	tradeoff hyperparameter
$\mathcal{L}_{con}^t$	the consistency loss on the t-th domain
$\mathcal{L}_{ce}^t$	the cross-entropy loss on the t-th domain
$\mathcal{L}_{total}^t$	the total loss on the t-th domain ($\mathcal{L}_{total}^t = \mathcal{L}_{ce}^t + \lambda\mathcal{L}_{con}^t$)
Diffusion model-related	
S	diffusion steps
s, s'	the index of diffusion steps ($s, s' \in \{1, 2, \ldots, S\}$)
$q(\boldsymbol{x})$	distribution of variable $\boldsymbol{x}$
$\boldsymbol{x}_0$	the original data point from $q(\boldsymbol{x})$
$\boldsymbol{x}_s$	the noisy data point at the s-th diffusion step
β_s	the variance used at the s-th diffusion step ($\beta_s \in (0, 1)$)
$\bar{\alpha}_s$	the product of all $(1 - \beta_{s'})$ until the s-th diffusion step ($\bar{\alpha}_s = \prod_{s'=1}^{s}(1 - \beta_{s'})$)
$\boldsymbol{\epsilon}$	random noise
$\mathcal{E}_{\boldsymbol{\theta}}$	denoising model
$\boldsymbol{\theta}$	parameters of the denoising model
σ_s	a variance hyperparameter
$\mathfrak{c}$	condition of the denoising model
$\mathcal{L}_{diff}^t$	noise estimation error loss on the t-th domain
M_g	the number of generated classifier weights based on each reference point
$\boldsymbol{\mu}^{test}$	the estimated prototype matrix of the testing domain $\mathcal{D}^{test}$ during inference stage
$\mathfrak{c}^{test,t'}$	the condition of diffusion model for domain $\mathcal{D}^{test}$ ($\mathfrak{c}^{test,t'} = \ddot{\mathbf{W}}^{t'} \oplus \boldsymbol{\mu}^{test}$)
$\Delta\mathbf{W}_j^{test,t'}$	the j-th generated residual classifier weights, conditioned on $\mathfrak{c}^{test,t'}$
$\bar{\mathbf{W}}^{test}$	the average ensemble weights for the testing domain $\mathcal{D}^{test}$
Others	
$\mathbf{I}$	identity matrix
$\mathcal{N}(\cdot, \cdot)$	Gaussian distribution
$\mathbb{E}$	mathematic expectation
$\mathbb{R}$	real number
$\vert\vert \cdot \vert\vert$	L2-Norm
$\vert \cdot \vert$	the length of an object
$KL(\cdot\vert\vert\cdot)$	Kullback-Leibler divergence
$\mathrm{sg}(\cdot)$	stopping gradients of an object
$\mathrm{softmax}(\cdot)$	normalized exponential function
$\oplus$	concatenating operation
i, j, k	indices

C Algorithm of W-Diff

The training and testing procedures of W-Diff are presented in Algorithm 1 and 2.

Algorithm 1: Training procedure for W-Diff

```
Input: sequentially arriving source domains $\mathcal{S} = \{\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^T\}$, feature encoder $E_{\boldsymbol{\psi}}$,
    classifier $H_{\mathbf{W}}$, conditional diffusion model $\mathcal{E}_{\boldsymbol{\theta}}$, reference point queue $Q_r$ with length $L$,
    anchor point queue $Q_a$ with length $M$, prototype queue $Q_p$ with length $M$, batch size $B$,
    loss tradeoff hyperparameter $\lambda$, warm-up hyperparameter $\rho$, maximum diffusion step $S$,
    training iterations $I_{TS}$ of task model, inner iterations $I_{DM}$ for updating diffusion model.
Initialize model parameters $\boldsymbol{\psi}$ as $\boldsymbol{\psi}^0$, $\mathbf{W}$ as $\mathbf{W}^0$, $\boldsymbol{\theta}$ as $\boldsymbol{\theta}^0$ and set $Q_r = \emptyset$.
for $t = 1$ to $T$ do
    Set $\boldsymbol{\psi}^t = \boldsymbol{\psi}^{t-1}$, $\mathbf{W}^t = \mathbf{W}^{t-1}$, $\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1}$, $Q_a = \emptyset, Q_p = \emptyset$, $\boldsymbol{\mu}^t = \mathbf{0}$, $n^t = 0$.
    for $iter = 1$ to $I_{TS}$ do
        $\mathcal{L}^t_{total} = 0$.
        Randomly sample a batch of data $\{\boldsymbol{x}^t_i, y^t_i\}^B_{i=1}$ from domain $\mathcal{D}^t$.
        Get the deep features of samples: $\{\boldsymbol{f}^t_i = E_{\boldsymbol{\psi}}(\boldsymbol{x}^t_i)\}^B_{i=1}$.
        Calculate the supervision loss $\mathcal{L}^t_{ce}$ in Eq. 6.
        $\mathcal{L}^t_{total} += \mathcal{L}^t_{ce}$.
        if $t > 1$ then
            Calculate the consistency loss $\mathcal{L}^t_{con}$ in Eq. 5.
            $\mathcal{L}^t_{total} += \mathcal{L}^t_{con}$.
        Update $\boldsymbol{\psi}^t$ and $\mathbf{W}^t$ by backpropagating the gradients of $\mathcal{L}^t_{total}$.
        if $(iter > \rho \cdot I_{TS}) \wedge (t > 1)$ then
            Update prototype matrix $\boldsymbol{\mu}^t$ via Eq. (8).
            Push $\mathbf{W}^t$ into $Q_a$ and $\boldsymbol{\mu}^t$ into $Q_p$: $Q_a \leftarrow \mathbf{W}^t, Q_p \leftarrow \boldsymbol{\mu}^t$.
        if $|Q_a| == M$ then
            for $inner\_iter = 1$ to $\lceil \frac{I_{DM}}{|Q_r|} \rceil$ do
                Sample diffusion step size $s \sim \text{Uniform}(1, \ldots, S)$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
                Calculate the noise estimation error loss $\mathcal{L}^t_{diff}$ via Eq. (10).
                Update $\boldsymbol{\theta}^t$ by backpropagating the gradients of $\mathcal{L}^t_{diff}$.
    Push the classifier weights with the best performance on the validation set of $\mathcal{D}^t$, denoted as
        $\ddot{\mathbf{W}}^t$, into $Q_r$: $Q_r \leftarrow \ddot{\mathbf{W}}^t$.
return Final $\boldsymbol{\psi}^T, \boldsymbol{\theta}^T, Q_r$.
```

D Experimental Setup Details

D.1 Dataset Description

Huffpost (license: CC0: Public Domain) from [48] comprises $63,907$ news headlines from the Huffington Post, with the time span from 2012 to 2018. These news headlines belong to 11 categories: “Black Voices”, “Business”, “Comedy”, “Crime”, “Entertainment”, “Impact”, “Queer Voices”, “Science”, “Sports”, “Tech” and “Travel”. This dataset reflects changes in news content and style over time. Follwoing [45], the first 4 years are used for training ($T = 4$) and the last 3 years are used for testing ($K = 3$). For each training domain, we randomly divide the data into training set and validation set in the ratio of 9 : 1.

Arxiv (license: CC0: Public Domain) in [48] is a large-scale dataset, including $2,057,952$ paper titles from 2007 to 2022. It reflects the change over time as research fields evolve. The task is to classify a research paper into one of 172 categories based solely on its title. For this dataset, we use data from the first 9 years as source domains ($T = 9$) and data from the last 7 years as target domains ($K = 7$). For each source domain, the data is randomly divided into training set and validation set in the ratio of 9 : 1.

Algorithm 2: Testing procedure for W-Diff

```
Input: sequentially arriving target domains $\mathcal{T} = \{\mathcal{D}^{T+1}, \mathcal{D}^{T+2}, \dots, \mathcal{D}^{T+K}\}$, feature encoder
       $E_{\boldsymbol{\psi}^T}$, conditional diffusion model $\mathcal{E}_{\boldsymbol{\theta}^T}$, reference point queue $Q_r$, number of categories
       $C$, batch size $B$, maximum diffusion step $S$, number of generated residual weights $M_g$
       based on each reference point.
for $k = 1$ to $K$ do
    Set $\mathcal{D}^{test} = \mathcal{D}^{T+k}$.
    Calculate the prototype matrix $\boldsymbol{\mu}^{test}$ vai $\boldsymbol{\mu}^{test}[c] = \frac{1}{N^{test}} \sum_{i=1}^{N^{test}} \bar{\boldsymbol{p}}_i^{test}[c] \cdot \boldsymbol{f}_i^{test}$, where
     $\bar{\boldsymbol{p}}_i^{test} = \frac{1}{|Q_r|} \sum_{\ddot{\mathbf{W}}^{t'} \in Q_r} \mathrm{softmax}(\ddot{\mathbf{W}}^{t'} \times \boldsymbol{f}_i^{test})$, $\boldsymbol{f}_i^{test} = E_{\boldsymbol{\psi}^T}(\boldsymbol{x}_i^{test})$, $c = 0, \dots, C-1$.
    for $\ddot{\mathbf{W}}^{t'} \in Q_r$ do
        Generate $M_g$ residual classifier weights: $\{\Delta \mathbf{W}_j^{test,t'}\}_{j=1}^{M_g}$ by substituting the denoising
         net in Eq. (3) with $\mathcal{E}_{\boldsymbol{\theta}^T}$ and applying the denoising process with condition
         $\mathfrak{c}^{test,t'} = \ddot{\mathbf{W}}^{t'} \oplus \boldsymbol{\mu}^{test}$.
    Obtain the average weight $\bar{\mathbf{W}}^{test} = \frac{1}{|Q_r|} \frac{1}{M_g} \sum_{\ddot{\mathbf{W}}^{t'} \in Q_r} \sum_{j=1}^{M_g} (\ddot{\mathbf{W}}^{t'} + \Delta \mathbf{W}_j^{test,t'})$.
    Get the final label predictions on domain $\mathcal{D}^{test}$: $\{\hat{y}_i^{test} = \mathrm{argmax}_c\, \boldsymbol{p}_i^{test}[c]\}_{i=1}^{N^{test}}$, where
     $\boldsymbol{p}_i^{test} = \mathrm{softmax}(\bar{\mathbf{W}}^{test} \times \boldsymbol{f}_i^{test})$.
return Label Predictions $\{\{\hat{y}_i^{T+k}\}_{i=1}^{N^{T+k}}\}_{k=1}^{K}$.
```

Yearbook (MIT license) dataset comes from [48]. It collects $37,189$ grayscale yearbook photos from 128 American high schools, with the time span from 1930 to 2013. The resolution of photos is 32×32. Photos from different years reflect changes in fashion trends and social norms over the decades. The task is to classify the genders from a yearbook photo. It is worth mentioning that we only use this dataset to evaluate the generalization performance of different methods in classification tasks. Following [45], we group the data into domains at four-year intervals, resulting in 21 domains. And the first 16 domains are used as source domains ($T = 16$), with the last 5 domains as target domains ($K = 5$). For each source domain, we randomly select 90% of samples as the training split and 10% of samples as the validation split. And for each target domain, we evaluate on its all data.

RMNIST (license: CC BY-SA 3.0) is constructed from MNIST dataset [7] which contains grayscale images of digits from 0 to 9. The image resolution is 28×28. RMNIST first randomly divides all data in MNIST into 9 groups and then creates 9 domains by rotating the 9 groups by $0°, 10°, \dots, 80°$, respectively. The rotation angle is used to simulate the evolving data distribution over time. Following [45], we use the first 6 domains ($T = 6$) as source domains and the last 3 domains as target domains ($K = 3$). Similarly, we split each source domain into training and validation sets in the ratio of $9:1$.

fMoW (license: `https://github.com/fMoW/dataset/blob/master/LICENSE`) dataset is from [48], which consists of $141,696$ RGB satellite images from 2002 to 2017. The visual features in these satellite images change over time due to human and environmental activities. The image resolution is 224×224 and the task is to classify the functional purpose of the buildings or land in a image into one of 62 categories. For this dataset, we consider each year as a separate domain. And the first 13 domains are used for training ($T = 13$), while the last 3 domains are used for testing ($K = 3$). The ratio of training data to validation data for each source domain is $9:1$.

2-Moons (license: `https://github.com/BaiTheBest/DRAIN/blob/main/LICENSE`) from [2] is constructed from 2-entangled moons dataset, where the lower moon with label 0 and the upper moon with label 1 contain 100 data points, respectively. 2-Moons creates 10 domains by counter-clockwise rotating the 200 data points at an interval of $18°$. Similar to RMNIST, the rotation angle simulates the evolving of data distribution. Following [2], the first 9 ($T = 9$) domains are used as source domain and the last domain is used as target domain ($K = 1$).

Online News Popularity (ONP) (license: CC BY 4.0) in [2] summarizes a heterogeneous set of features related to the articles published by Mashable in a two-year period. This dataset is divided into 6 domains by time, and the goal is to predict whether an article is popular in social networks based on its features. Following [2], we use the first 5 domains for training ($T = 5$) and the last domain for testing ($K = 1$).

Table 7: Configuration of the U-Net $\mathcal{E}_{\boldsymbol{\theta}}$ on different datasets with hybrid conditioning way.

Dataset	Huffpost	Arxiv	Yearbook	RMNIST	fMoW	2-Moons	ONP
Input-shape	$11 \times 128 \times 3$	$172 \times 128 \times 3$	$2 \times 32 \times 3$	$10 \times 128 \times 3$	$62 \times 256 \times 3$	$2 \times 128 \times 3$	$2 \times 128 \times 3$
Diffusion steps	1000	1000	1000	1000	1000	1000	1000
Noise schedule	linear	linear	linear	linear	linear	linear	linear
Channels	64	64	64	64	64	32	64
Depth	1	1	1	1	1	1	1
Channel Multiplier	1,2,4	1,2,2	1	1,2,4	1,2	1,2,4	1,2
Attention Resolutions	4,2,1	4,2,1	1	4,2,1	2,1	4,2,1	2,1
Head Channels	32	32	32	32	32	32	32
Transformer Depth	1	1	2	1	1	1	1
Batch Size	32	32	32	32	32	32	32
Learning Rate	8e-5	8e-5	5e-4	5e-4	8e-5	5e-4	5e-4
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW

Table 8: Training details on different datasets.

Dataset	B	Epochs	ρ	I_{DM}	Optimizer	Learning Rate	λ	L	M	M_g
Huffpost	64	50	0.6	20	Adam	2e-5	10	8	32	32
Arxiv	64	5	0.2	5	Adam	2e-5	10	8	32	32
Yearbook	64	50	0.2	5	Adam	1e-3	10	8	32	32
RMNIST	64	50	0.2	5	Adam	1e-3	10	8	32	32
fMoW	64	25	0.6	30	Adam	2e-4	10	8	32	32
2-Moons	64	150	0.2	10	Adam	1e-3	10	8	32	32
ONP	64	50	0.2	10	Adam	1e-4	10	8	32	32

D.2 Network Details

For the backbone of the task model, Huffpost and Arxiv apply pretrained DistilBERT base model [33] along with a bottleneck layer [20] to reduce the feature dimensions into 128. The bottleneck layer is implemented as the combination of a linaer layer, BatchNorm and ReLU. Yearbook uses the 4-layer convolutional network in [48], RMNIST adopts the ConvNet in [29], and fMoW employs the ImageNet-pretrained DenseNet-121 [15] along with a bottleneck layer [20] to reduce the feature dimensions into 256. Meanwhile, 2-Moons uses a MLP with two hidden layers of hidden size 64 and 128, and ONP adopts a MLP with one hidden layer of hidden size 128.

For the conditional diffusion model, we implement it in a U-Net architecture similar to LDM [31] and make some modifications to better suit our method. Detailed modifications can be found in the code provided in the supplementary material. In Table 7, we provide detailed configurations for U-Net $\mathcal{E}_{\boldsymbol{\theta}}$ on different datasets. Please refer to original paper [31] for the meaning of different hyperparameters.

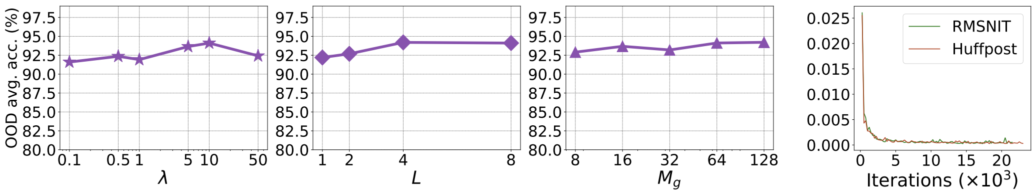
D.3 Training Recipe

Training details on different datasets are given in Table 8, where B is the batch size for the task model, I_{DM} is the inner iterations for updating $\mathcal{E}_{\boldsymbol{\theta}}$, λ is the loss tradeoff hyperparameter, ρ is the warm-up hyperparameter, L is the maximum length of the reference point queue Q_r, M is the maximum length of the anchor point queue Q_a and prototype queue Q_p, and M_g is the number of generated residual classifier weights based on each reference point. All experiments are conducted using the PyTorch packages and run on a single NVIDIA GeForce RTX 4090 GPU with 24GB memory.

E More Results

E.1 Hyperparameter Sensitivity

In Fig. 4(a), we test the sensitivity of W-Diff to the loss tradeoff hyperparameter λ, the maximum length L of the reference point queue Q_r and the number M_g of generated residual classifier weights based on each reference point, where $\lambda \in \{0.1, 0.5, 1.0, 5.0, 10.0, 50.0\}$, $L \in \{1, 2, 4, 8\}$, $M_g \in \{8, 16, 32, 64, 128\}$. We find that larger M_g results in more weights for ensemble and seems to be better. W-Diff is a little bit sensitive to λ and L. Empirically, $\lambda = 10.0$ and larger L work well.



(a) Hyperparameter sensitivity.

(b) Loss curve of $\mathcal{L}^t_{diff}$

Figure 4: (a): Sensitivity of W-Diff to hyperparameters λ, L, M_g on RMNIST. (b): The loss curve of $\mathcal{L}^t_{diff}$ on RMNIST and Huffpost when training conditional diffusion model on the second domain.

Table 9: Memory cost and inference time of diffusion model on different datasets.

Yearbook	RMNIST	fMoW	Huffpost	Arxiv	2-Moons	ONP
Number of parameters (MB) for the conditional diffusion model $\mathcal{E}_{\boldsymbol{\theta}}$						
2.31	41.62	27.23	41.62	20.01	10.52	15.51
Time (s) for generating $M_g = 32$ residual classifier weights in a batch, where denoising step $S = 1000$						
12	23	71	24	181	21	16

E.2 Convergence of Diffusion Model Training

In Fig. 4(b), we plot the loss curve of $\mathcal{L}^t_{diff}$, when the conditional diffusion model is incrementally trained on the second source domain. Form the results, we see that the noise estimation error loss $\mathcal{L}^t_{diff}$ steadily decreases and finally converges, demonstrating that using the FIFO queue to cache the recent M classifier weights and prototype matrices after the warm-up stage is feasible for the diffusion model training. Storing all checkpoints after the warm-up stage provides lost of training data for diffusion model but requires more storage cost. By contrast, using a FIFO queue with a fixed length balances the storage cost and the diversity of training data.

E.3 Memory and Time Cost of Diffusion Model

In Table 9, we list the model size and inference time of the conditional diffusion model on different datasets. Since the diffusion model only models the evolving pattern of the classifier weights which are the parameters of a linear layer, the model size that is measured by the number of parameters is small on all datasets. Besides, the inference time when forwarding the diffusion model for 1000 times to generate a batch of residual classifier weights is moderate. Concretely, forwarding the condition diffusion model for one time requires ≤ 181 ms. Certainly, acceleration techniques, e.g., DDIM [35] can be used to further reduce the inference time.

E.4 Significance test (t-test) of W-Diff.

To comprehensively evaluate the effectiveness of W-Diff, we conduct the significance test (t-test) on Huffpost, Arxiv, Yearbook, RMNIST and fMoW datasets. Concretely, a significance level of 0.05 is applied. If the p-value is less than 0.05, then the accuracy difference between EvoS [45] and W-Diff is statistically significant. For clearer explanation, the -log(p) of each p-value is plotted. In Fig. 5, the majority of the -log(p) of the performance comparison between EvoS [45] and W-Diff are larger than -log(0.05), which means that W-Diff is statistically superior to EvoS [45] at most datasets.

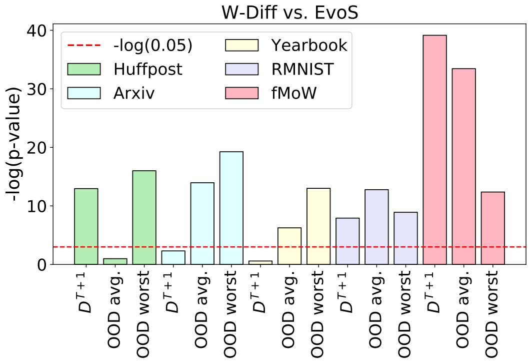


Figure 5: t-test for W-Diff *vs* EvoS [45], where a significance level of 0.05 is adopted.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have stated the contribution of our work in the introduction section.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations of the work in Section A.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: Our work is a methodological level design and does not propose new theories.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided the implementation details in Section D and provide the code in the supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The datasets used in our paper are from previous work and are publicly available. We have provided the dataset details in Section D and the URL for downloading in the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have described the problem setting in Section 3.1 and the experimental details in Section 5.1 and D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We have reported the mean and standard deviation in the main experiments when independently running each task with three random seeds.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have pointed out the used GPU in Section D and the number of parameters for the conditional diffusion model in Section E.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have preserve anonymity in all submitted materials.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have discussed the potential social impacts in Section A.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The datasets and models used in the paper are all public and from previous works.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The datasets and models used in the paper are from previous works and are public. We have cited related papers in our work and provided the license in Section D.1.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets are released.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Neither crowdsourcing nor research with human subjects is used.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.