Revolutionizing Open Set Domain Generalization: Learning to generalize via Conditional Diffusion Without Gradient

Anonymous

Abstract

Open Set Domain Generalization (OSDG) extends traditional Domain Generalization (DG) by addressing the more practical and challenging scenario where the source and unseen target domains not only exhibit distribution shifts but also contain unseen classes in the unseen target domains. Recently, meta-learning methods have shown remarkable performance in OSDG, which can generalize to unseen domains by learning a meta-learner in a second-order optimization manner. However, those methods heavily rely on second-order optimization, which results in huge computational budgets and the risk of vanishing gradients. In addition, those methods suffer from poor generalization to unseen classes, severely limiting OSDG in real-world applications. To overcome these limitations, we propose a novel OSDG framework, **OGDiff**, which reformulates the optimization of base learner parameters as a conditional diffusion process from Gaussian initialization. Unlike conventional gradient-based meta-learners that rely on second-order gradients and incur substantial memory and stability issues, OGDiff learns the weight evolution of base learners via a denoising process, thereby avoiding inner-loop backpropagation. OGDiff can significantly reduce computational budgets due to the low training cost of diffusion models. Furthermore, we introduce a **Multi-Scale Diffusion Feature Fusion (MSDFF)** module, which efficiently aggregates hierarchical latent features across different reverse diffusion steps. Since different steps capture complementary information at varying abstraction levels, MSDFF enables the model to better integrate fine- and coarse-grained patterns, thereby improving recognition of both known and unknown classes. Extensive experiments on standard OSDG benchmarks demonstrate that OGDiff not only achieves state-of-the-art performance but also significantly reduces training cost compared to existing meta-learning approaches.

Introduction

Deep neural networks (DNNs) have achieved remarkable success in numerous computer vision tasks. This success generally relies on the assumption that training and test samples are drawn from the same data distribution and share the same label space. However, in real-world applications, this assumption often fails because the test data distribution can differ from that of the training data, a phenomenon known as domain shift. As a result, model performance can degrade significantly when applied to such shifted domains (Li et al. 2019b). To tackle this, the community has proposed *Domain Generalization (DG)* (Wang et al. 2023a), which aims to train models that can generalize to unseen domains. Techniques such as data augmentation (Zhou et al. 2020b; Li et al. 2021), feature alignment (Dou et al. 2019; Li et al. 2018b), and meta-learning (Li et al. 2018a, 2019a) have shown good progress in this direction.

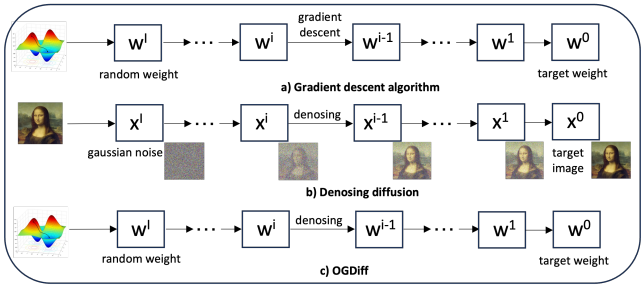

Figure 1: An illustrative comparison highlighting the differences.

However, there's a bigger challenge that DG often overlooks. Most DG methods assume that the categories (classes) in the training and test sets are the same—this is called the *closed-set assumption*. But in practice, this isn't always true. For example, in self-driving cars, a model might encounter a new type of object it has never seen before. Similarly, in medical imaging, some rare diseases may not appear in the training data at all (Scheirer et al. 2013). In such cases, traditional DG models may wrongly classify unknown samples as known ones, which could be dangerous.

To address this, researchers have introduced a more realistic and challenging problem: *Open Set Domain Generalization (OSDG)* (Shu et al. 2021; Katsumata et al. 2021). In OSDG, the model not only needs to deal with domain shifts but also needs to recognize when a sample belongs to an unseen class. This means the model must learn both where the domain boundaries lie, and how to handle class shifts.

Recent works try to solve this using meta-learning (Hospedales et al. 2022), which simulates domain shifts during training to improve generalization. However, these methods usually come with two major drawbacks.

First, they rely heavily on second-order optimization, which requires backpropagation through optimization steps—this is slow, memory-hungry, and sometimes unstable (Li et al. 2018a; Shu et al. 2021). Second, they don't handle unseen classes well, often depending on multi-binary classifiers like one-vs-all, which tend to make biased predictions (Saito and Saenko 2021; Liu et al. 2019). This happens because the classifier learns from many negative samples but only a few positive ones, making the boundary unfairly skewed.

Observation

As shown in Fig. 1, we can find that the optimization of gradient-based meta learning methods tailored for OSGD has the same behavior as the learning process of denoising in conditional diffusion. Motivated by this observation, we explore whether diffusion-based models can serve as a surrogate for traditional gradient-based optimization in meta-learning frameworks, replacing the expensive training process of inner-loop in meta-learning methods for OSDG with second-order optimization.

Contribution

In this paper, we address the challenging and practical problem of *Open Set Domain Generalization (OSDG)*, where models must generalize not only to unseen domains but also to unseen categories within those domains. While recent meta-learning approaches have shown promise in this setting, they still suffer from two major limitations. First, they rely heavily on second-order optimization, which leads to high computational cost and training instability due to vanishing gradients. Second, these methods often struggle to generalize to novel categories, limiting their applicability in real-world open-set scenarios.

To overcome these issues, we propose a novel framework called **OGDiff** (*Open-set Generalization via Diffusion*). Unlike conventional meta-learners that optimize model parameters via second-order gradient descent, OGDiff reformulates the optimization process as a *conditional diffusion process* starting from Gaussian-initialized weights. This denoising-based approach eliminates the need for inner-loop backpropagation and second-order gradients, thereby significantly reducing memory and computation overhead. Moreover, OGDiff naturally captures the evolution of base learner parameters and facilitates both domain-level and category-level generalization. To further enhance representational power, we introduce a **Multi-Scale Diffusion Feature Fusion (MSDFF)** module, which aggregates hierarchical latent features across different reverse diffusion steps. Since different steps capture complementary information at varying abstraction levels, MSDFF enables the model to better integrate fine- and coarse-grained patterns, thereby improving recognition of both known and unknown classes. Extensive experiments on multiple OSDG benchmarks demonstrate that OGDiff achieves state-of-the-art performance while maintaining high computational efficiency. By avoiding the pitfalls of traditional meta-learning methods and introducing a diffusion-based optimization paradigm, OGDiff sets a new standard for open-set generalization. To sum up, our main contributions are:

- We are the first to reformulate the optimization of base learner parameters in open-set domain generalization as a conditional diffusion process. Inspired by the similarity between gradient descent and denoising workflows, we reveal that the denoising process in diffusion models can be interpreted as a generalized and learnable optimizer, which avoids inner-loop backpropagation and inherently models uncertainty in weight evolution.
- Based on this insight, we propose a novel diffusion-based meta-learning framework for OSDG, named **OGDiff**. In particular, a conditional denoising UNet is used to predict the evolution of model weights starting from Gaussian initialization. Compared to traditional second-order meta-learners, OGDiff eliminates the need for expensive inner-loop optimization, alleviating memory burden and vanishing gradient problems, while supporting both domain- and class-level generalization.
- To further enhance representational power, we introduce a **Multi-Scale Diffusion Feature Fusion (MSDFF)** module, which aggregates hierarchical latent features across different reverse diffusion steps. Since different steps capture complementary information at varying abstraction levels, MSDFF enables the model to better integrate fine- and coarse-grained patterns, thereby improving recognition of both known and unknown classes.
- We conduct extensive experiments on several public OSDG benchmarks. Results show that OGDiff not only significantly reduces training cost, but also achieves state-of-the-art performance, outperforming existing meta-learning-based approaches in both closed-set and open-set scenarios.

Revisiting Open Set Domain Generalization from a Diffusion Perspective

Problem Definition

Open Set Domain Generalization (OSDG) aims to learn a model from multiple labeled source domains that can generalize to unseen target domains containing unknown classes. Formally, we are given a set of source domains $\mathcal{S} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_S\}$ with a shared label space $\mathcal{C}$, and a set of unseen target domains $\mathcal{T} = \{\mathcal{D}_{S+1}, \ldots, \mathcal{D}_{S+T}\}$ with an extended label space $\mathcal{C} \cup \mathcal{U}$, where $\mathcal{C} \cap \mathcal{U} = \varnothing$.

Each source domain consists of labeled samples:

$$\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^{N_s}, \quad y_i^s \in \mathcal{C}. \tag{1}$$

Our goal is to utilize all source domains to train a model that can directly generalize to target domains with both known and unknown classes.

Meta-Learning Strategy for OSDG

To improve generalization to unseen domains, OSDG often employs a meta-learning strategy. The source domains $\mathcal{S}$ are split into a meta-train set $\mathcal{S}_\mathcal{F}$ and a meta-test set $\mathcal{S}_\mathcal{G}$ such that:

$$\mathcal{S}_\mathcal{F} \cup \mathcal{S}_\mathcal{G} = \mathcal{S}, \quad \mathcal{S}_\mathcal{F} \cap \mathcal{S}_\mathcal{G} = \varnothing. \tag{2}$$

Let W denote the parameters of a feature extractor. Meta-learning optimizes W via a bi-level gradient-based scheme:

$$\hat{W} = W - \eta \nabla \mathcal{F}(W), \quad W \leftarrow W - \eta' \nabla \mathcal{G}(\hat{W}), \tag{3}$$

where $\mathcal{F}$ and $\mathcal{G}$ denote the training losses on the meta-train and meta-test domains, respectively.

Can Diffusion Replace Meta-Learning? We observe that both meta-learning and OSDG optimization ultimately aim to find an optimal feature extractor parameter W through iterative gradient-based updates. However, traditional meta-learning is computationally expensive and must restart optimization for each new task or target domain. This raises a concern: *Can we leverage the diffusion process to learn an effective W for OSDG, avoiding the need for task-specific meta-learning loops?*

Diffusion Models

Diffusion models define a generative process that gradually transforms a simple prior distribution into a complex data distribution using two stages:

1) Forward Process (Diffusion): Starting from a data point $x_0 \sim p_{\text{target}}$, Gaussian noise is added at each step t:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)\mathbf{I}), \tag{4}$$

where $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. The training objective is to learn a noise predictor $\epsilon_\theta(x_t, t)$:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{x_0,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right]. \tag{5}$$

2) Reverse Process (Denoising): Given a noisy sample $x_T \sim \mathcal{N}(0, \mathbf{I})$, the denoising process recovers x_0 using:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z, \quad z \sim \mathcal{N}(0, \mathbf{I}), \tag{6}$$

where $\sigma_t^2 = \beta_t$.

Connection Between Diffusion and Gradient Descent in OSDG

We find a striking similarity between the denoising update and traditional gradient descent:

$$w_{t+1} = w_t - \eta \nabla L(w_t), \tag{7}$$

which can be aligned with a reformulation of the denoising step:

$$x_{t-1} = x_t - \eta \epsilon_\theta(x_t, t) + (\gamma - 1)x_t + \xi z, \tag{8}$$

where:

$$\gamma = \frac{1}{\sqrt{\alpha_t}}, \quad \eta = \frac{\beta_t}{\sqrt{\alpha_t}\sqrt{1-\overline{\alpha}_t}}, \quad \xi = \sigma_t. \tag{9}$$

Where the first term $x_t - \eta\epsilon_\theta(x_t, t)$ acts like a gradient descent step. The second term $(\gamma - 1)x_t$ is a momentum update. The third term ξz represents uncertainty.

Method

Adaptive Feature Space Diffusion for Open Set Domain Generalization

Inspired by the challenges posed by open set domain generalization, we find that diffusion models, with their generalized and learnable framework, provide a powerful approach for adapting to unseen domains and unknown classes. The inherent advantages of diffusion models—such as their robustness to noise and efficiency in training—make them particularly suitable for this task. Building upon this, we introduce **Adaptive Feature Space Diffusion for Open Set Domain Generalization**. This approach leverages the diffusion process in the feature space, allowing for adaptive refinement of feature representations, enabling the model to effectively handle both known and unknown classes in the target domain. By removing the need for complex inner-loop differentiation, our method alleviates the memory burden and mitigates the risk of vanishing gradients, ultimately improving the model's ability to generalize to new, unseen data while maintaining efficiency during training.

Overall Framework. Our method for Open Set Domain Generalization employs adaptive feature space diffusion with a flexible, multi-scale backbone network. Given an input image x, we extract feature vectors from N layers of the network, denoted as $f_1, f_2, ..., f_N$, where each feature $f_i \in \mathbb{R}^{d_i}$ corresponds to the output of the i-th layer. These features undergo a feature diffusion process to ensure domain-invariant representations that are robust to domain shifts. For each scale feature f_i, we apply forward diffusion by adding noise according to

$$f_{i,t} = \sqrt{1-\beta_t}f_i + \sqrt{\beta_t}\epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, I), \tag{10}$$

where β_t is a time-dependent noise schedule that controls the noise addition rate. This process introduces noise into each feature f_i, making the model more robust to shifts. To recover clean features, we use a learnable denoiser D_θ that operates iteratively at each timestep t, yielding

$$\hat{f}_i = D_\theta(f_{i,t}, t), \tag{11}$$

where D_θ is a small multi-layer perceptron (MLP) trained to reconstruct the original feature from noisy observations. This denoising process helps remove domain-specific noise while maintaining domain-invariant properties of the features. After denoising, the features $\hat{f}_i$ from all N layers are weighted by learnable scalars w_i, and these weights are normalized using the softmax function:

$$w_i = \frac{e^{\mathbf{w}_i}}{\sum_{k=1}^{N} e^{\mathbf{w}_k}}, \quad \forall i \in [1, N], \tag{12}$$

leading to the final fused feature vector.

$$F_{\text{concat}} = \sum_{i=1}^{N} w_i \hat{f}_i \in \mathbb{R}^{\sum_{i=1}^{N} d_i}, \tag{13}$$

where F_{concat} is the concatenated feature vector composed of weighted denoised features. This vector is passed through a linear classifier to produce logits for class prediction

$$\text{logits} = W_{\text{class}} F_{\text{concat}} + b_{\text{class}}, \tag{14}$$

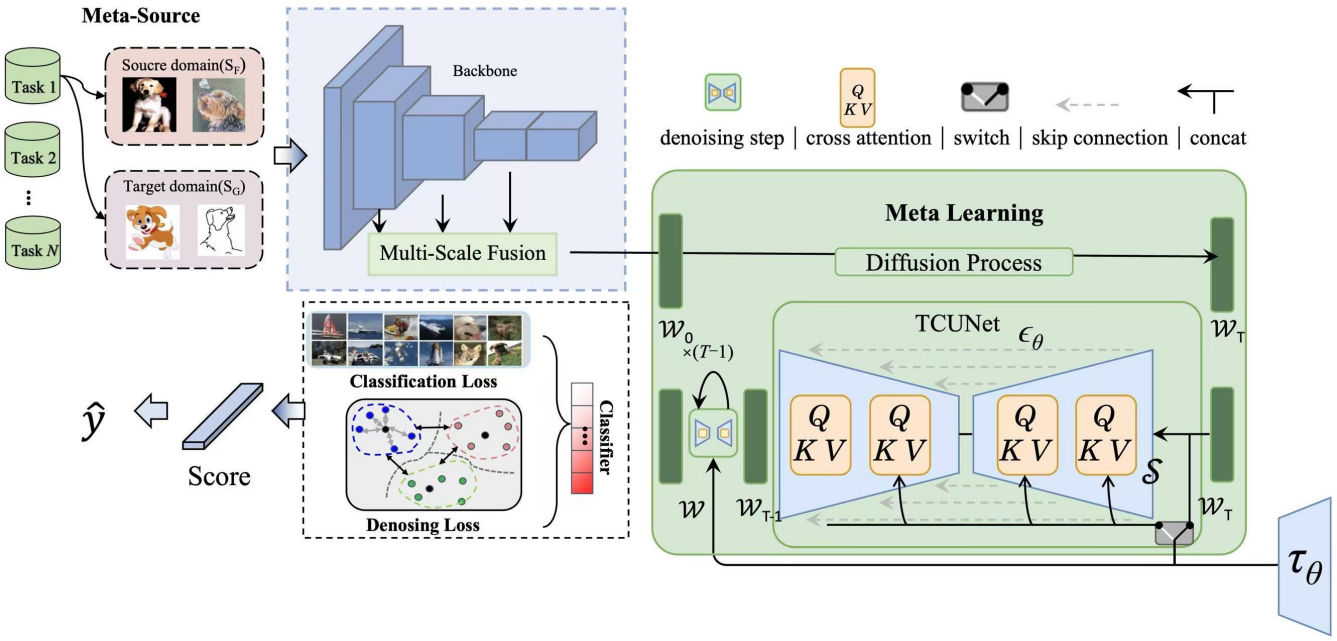

Figure 2: The architecture of our OGDiff framework.

where $W_{\text{class}} \in \mathbb{R}^{C\times D}$ and $b_{\text{class}} \in \mathbb{R}^{C}$ are the weight matrix and bias term of the classifier, and C is the number of classes.

The model is trained using a joint loss function that includes both classification loss and denoising loss

$$\mathcal{L}_{\text{cls}} = -\sum_{k=1}^{C} y_k \log(p(\hat{y}_k|x)), \tag{15}$$

where y_k is the ground truth label for class k, and $p(\hat{y}_k|x)$ is the predicted probability for class k. The denoising loss is defined as

$$\mathcal{L}_{\text{rec}} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{f}_i - f_i\|_2^2, \tag{16}$$

where the reconstruction error is computed as the mean squared error between the denoised feature and the original feature. The total loss function combining both objectives is

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{rec}}, \tag{17}$$

where λ is a hyperparameter controlling the balance between classification and denoising losses.

In addition to these, we introduce an open set loss to deal with unknown classes during inference. This loss is defined as

$$\mathcal{L}_{\text{open}} = \sum_{k=1}^{C} -\log(p(\hat{y}_k|x)) + \sum_{j\neq k} \log(1 - p(\hat{y}_j|x)), \tag{18}$$

where $\hat{y}_k$ represents the predicted class for each known class k, and the rejection of unknown classes is enforced by a multi-binary rejection mechanism. The model is designed to reject unknown samples by using the binary classifier, which increases the accuracy in rejecting out-of-distribution (OOD) samples. The effectiveness of the model is evaluated using metrics such as *H-score* and *OSCR*, which measure the ability to classify known classes accurately and reject unknowns, respectively.

Task-Conditional UNet (TCUNet). The core component of our proposed meta-optimizer is a task-conditional UNet, denoted as $\epsilon_\theta(\cdot)$, which serves as a learnable noise prediction model in the diffusion process. It takes the support set $\mathcal{S} = \{(u_i, y_i)\}$, the current time step t, and the base learner weight w_t as inputs, and predicts the noise to be removed from w_t at each denoising step.

Although one could consider adopting a general conditional UNet for this purpose, we find such generic architectures perform suboptimally in the meta-optimization setting. This observation motivates a deeper investigation into the underlying role of $\epsilon_\theta(\cdot)$.

Gradient-Driven Noise Formulation. To explicitly align with this interpretation, we reformulate the denoising objective of $\epsilon_\theta(\cdot)$ from the perspective of gradient estimation. Given the support set $\mathcal{S}$, we first compute the loss of the base learner $g_{w_t}(\cdot)$ as follows:

$$\mathcal{L}_{w_t}(\mathcal{S}) = \frac{1}{|\mathcal{S}|}\sum_{(u_i,y_i)\in\mathcal{S}} \text{loss\_fun}(g_{w_t}(f_\phi(u_i)), y_i), \tag{19}$$

where $f_\phi(\cdot)$ is the feature extractor, and loss_fun$(\cdot)$ is the task loss function. Rather than using a standard cross-entropy loss, we adopt a prototype-based L2 loss to better suit the few-shot classification setting:

$$\mathcal{L}_{w_t}(\mathcal{S}) = \frac{1}{|\mathcal{S}|}\sum_{(u_i,y_i)\in\mathcal{S}} \|f_\phi(u_i) - w_{t,y_i}\|_2^2, \tag{20}$$

where w_{t,y_i} denotes the class prototype for class y_i at time step t. This design encourages each prototype to move toward the center of the corresponding class samples in the

Algorithm 1: OGDiff

repeat
  Sample a task $\tau = (\mathcal{S}, \mathcal{D}^{\tau}_{base}, \mathcal{Q})$ from dataset $\mathcal{D}_{base}$
  Extract multi-scale features $\{f_1, f_2, ..., f_N\}$ from support set $\mathcal{S}$
  Initialize diffusion states $\{f_{1,T}, f_{2,T}, ..., f_{N,T}\}$ by adding Gaussian noise to each f_i
  for $t = T, \ldots, 1$ **do**
    Cross-scale fusion: $f_t^{\text{fused}} = \text{MSDFF}(\{f_{i,t}\}_{i=1}^{N})$
    for each scale $i = 1$ to N **do**
      Estimate noise: $\hat{\epsilon}_{i,t} = \epsilon_\theta(f_t^{\text{fused}}, i, t)$
      Update state: $f_{i,t-1} = \frac{1}{\sqrt{\alpha_t}}(f_{i,t} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\epsilon}_{i,t})$
    end for
  end for
  Aggregate refined features: $F = \sum_{i=1}^{N} w_i f_{i,0}$, where w_i are learnable weights
  Predict class logits: $\text{logits} = W_{\text{class}} F + b_{\text{class}}$
  Compute total loss $\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{open}}$
  Update parameters θ via gradient descent on $\nabla_\theta \mathcal{L}$
until converged

feature space, aligning with the principle of prototype learning. We compute the gradient of the loss for w_t:

$$\nabla \mathcal{L}_{w_t}(\mathcal{S}), \tag{21}$$

which serves as a coarse noise estimate. However, due to the limited number of labeled samples in $\mathcal{S}$ (e.g., $K = 1$ or 5 per class), this gradient is often noisy and unreliable. To improve estimation accuracy, we feed this initial gradient along with w_t, the time step t, and the support set $\mathcal{S}$ into a learnable task-conditioned denoising network $\epsilon_\theta(\cdot)$, implemented as a structured UNet.

Multi-Scale Diffusion Feature Fusion (MSDFF)

Single-stage features overlook complementary cues available at different reverse-diffusion steps, so we introduce *MSDFF* to merge the entire hierarchy of latent maps $\hat{\mathbf{f}}_i \in \mathbb{R}^{C_i \times H_i \times W_i}$ generated at steps $i = 1, \ldots, N$. Each map follows the transition

$$\hat{\mathbf{f}}_{i-1} = g_\theta^{(i)}(\hat{\mathbf{f}}_i) + \epsilon_{i-1}, \qquad \epsilon_{i-1} \sim \mathcal{N}(\mathbf{0}, \sigma_{i-1}^2 \mathbf{I}),$$

and unrolling the chain produces the telescopic score

$$\nabla_{\hat{\mathbf{f}}_0} \log p_\theta(\hat{\mathbf{f}}_0) = \sum_{i=1}^{N} \Big(\prod_{k=1}^{i-1} \frac{\partial g_\theta^{(k)}}{\partial \hat{\mathbf{f}}_k} \Big) \nabla_{\hat{\mathbf{f}}_i} \log p_\theta(\hat{\mathbf{f}}_i),$$

implying that every intermediate feature provides an independent basis vector. Aggregating all $\hat{\mathbf{f}}_i$ therefore tightens an upper bound on the Bayes risk $\mathcal{R} = \mathbb{E} \|\hat{\mathbf{y}} - \mathbf{y}\|_2^2$. Under early stopping with `patience=20`, we examine three practical fusion rules. After global pooling $\mathbf{z}_i = g(\hat{\mathbf{f}}_i) \in \mathbb{R}^{C_i}$ we set

(a) Plain-Concat	$\mathbf{z} = [\mathbf{z}_1; \ldots; \mathbf{z}_N]$
(b) Weighted-Concat	$\alpha_i = \dfrac{e^{a_i}}{\sum_j e^{a_j}}$, $\mathbf{z} = [\alpha_1 \mathbf{z}_1; \ldots; \alpha_N \mathbf{z}_N]$
(c) SE-Concat	$s_i = \sigma\big(W_2 \delta(W_1 \mathbf{z}_i)\big)$, $\tilde{\mathbf{z}}_i = s_i \odot \mathbf{z}_i$, $\mathbf{z} = [\tilde{\mathbf{z}}_1; \ldots; \tilde{\mathbf{z}}_N]$

where a_i are learnable scalars, δ / σ are ReLU/Sigmoid, and W_1, W_2 are squeeze–excitation weights; $\mathbf{z}$ is finally mapped to logits by $\hat{\mathbf{y}} = W\mathbf{z} + b$.

Plain concatenation (*Muti-OGDiff*) already surpasses the single-stage baseline, and the introduction of learnable soft-max weights (*Muti-OGDiff*$^{+}$) delivers the highest validation accuracy. In contrast, SE-Concat (*Muti-OGDiff-SE*) converges more slowly and plateaus lower, most likely because channel squeezing removes spatial cues while the gating parameters over-fit under domain shift. Consequently, we adopt the weighted-concat variant, whose fusion rule is

$$\mathbf{z} \;=\; (\boldsymbol{\alpha} \otimes \mathbf{1}_C) \;\odot\; \big[\mathbf{z}_1; \ldots; \mathbf{z}_N\big], \qquad \boldsymbol{\alpha} = \text{softmax}(\mathbf{a}),$$

where $\mathbf{a} \in \mathbb{R}^N$ are learnable logits, $\otimes$ replicates each scalar weight across its C channels, $\odot$ denotes element-wise multiplication, and $\sum_{i=1}^{N} \alpha_i = 1$ holds by construction.

Experiments

We evaluate OGDiff on the **PACS**(Li et al. 2017) and **Digits-DG**(Zhou et al. 2020a) benchmarks using backbones including ResNet18(He et al. 2016). Following(Wang et al. 2023b), we report Close-set accuracy (Acc), H-score, and OSCR, with OSCR as the main metric for OSDG. Further training details are provided in App. A.

Experimental Results on PACS Benchmarks

ResNet18 and ResNet50. OGDiff achieves optimal results on PACS, leading in average Acc, H-score, and OSCR. OGDiff-bcls reaches 80.20% OSCR. Improvements are consistent across domains, particularly on the challenging Sketch domain. The diffusion-based framework enables efficient and stable parameter optimization, avoiding issues in second-order methods and enhancing generalization to unseen domains and novel classes. For ResNet50, OGDiff maintains strong performance, with OGDiff-bcls achieving the highest average OSCR of 87.59%, surpassing EBiL-HaDS-bcls (86.12%). It remains competitive across all domains, with notable gains on Cartoon and Sketch, showing that OGDiff scales well to larger backbones and better exploits their capacity for domain generalization.

ViT. With the ViT base model, OGDiff again outperforms all baselines. OGDiff-bcls achieves the top average OSCR of 81.53%, exceeding MEDIC-bcls (79.50%) by over 2%. The method shows strong gains in Art and Photo, confirming

Figure 3: Training (dashed) and validation (solid) accuracy under `patience=20`. OGDiff: single stage; Muti-OGDiff: plain concat; Muti-OGDiff^{+}: weighted concat; Muti-OGDiff-SE: SE-based fusion.

Table 1: Results (%) of PACS on ResNet18 (He et al. 2016). The open-set ratio is 6:1.

Method	Photo (P)			Art (A)			Cartoon (C)			Sketch (S)			Avg		
	Acc	H-score	OSCR	Acc	H-score	OSCR	Acc	H-score	OSCR	Acc	H-score	OSCR	Acc	H-score	OSCR
ARPL (Chen et al. 2022)	94.83	**95.06**	94.63	83.93	67.88	68.82	78.56	62.98	65.30	74.34	61.20	59.80	82.91	71.78	72.14
MMLD (Matsuura and Harada 2020)	94.83	88.80	92.94	84.43	64.83	69.43	77.11	64.21	65.36	75.14	67.70	64.69	82.88	71.38	73.11
RSC (Huang et al. 2020)	94.43	88.37	91.38	83.36	70.27	73.55	78.09	65.13	66.15	77.16	52.98	62.31	83.26	69.19	73.35
DAML (Shu et al. 2021)	91.44	80.87	82.83	83.11	72.05	71.75	79.11	66.26	66.46	82.97	72.63	73.71	84.16	72.95	73.69
MixStyle (Zhou et al. 2020b)	95.23	82.02	88.99	86.18	70.62	72.57	78.92	63.23	63.81	80.34	71.90	72.07	85.17	71.94	74.36
SelfReg (Kim et al. 2021)	95.72	89.34	92.26	86.24	72.45	73.77	80.77	65.75	66.38	78.30	67.06	65.69	85.26	73.65	74.53
MLDG (Li et al. 2018a)	94.99	91.48	93.70	84.12	69.52	72.15	78.45	61.59	64.32	79.99	69.67	68.60	84.39	73.06	74.69
MVDG (Zhang et al. 2022)	94.43	74.07	88.07	87.62	71.98	75.05	81.18	63.95	66.34	82.41	73.55	73.83	86.41	70.89	75.82
ODG-Net (Bose et al. 2023)	93.54	89.39	89.76	85.74	72.36	73.41	81.59	67.04	67.99	79.89	61.57	67.46	85.19	72.59	74.66
MEDIC-cls (Wang et al. 2023b)	94.83	83.68	90.30	86.20	69.35	74.16	**81.94**	63.26	67.43	81.84	69.60	70.85	86.20	71.47	75.69
MEDIC-bcls (Wang et al. 2023b)	94.83	89.49	92.40	86.20	73.82	75.58	**81.94**	66.26	69.04	81.84	74.37	74.52	86.20	75.98	77.89
EBiL-HaDS-bcls (Wang et al. 2024)	95.80	93.10	94.42	87.24	75.66		82.98	67.57	72.22	83.21	78.29	77.52	87.31	78.66	80.34
OGDiff-cls (ours)	**98.75**	93.04	95.89	**91.19**	73.38	75.79	81.24	**70.09**	73.04	**83.16**	76.43	76.06	**88.59**	78.24	80.20
OGDiff-bcls (ours)	**98.75**	94.65	95.66	**91.19**	**77.23**	**79.21**	81.24	69.51	**74.05**	**83.16**	**79.85**	**79.19**	**88.59**	**80.31**	**81.53**

Table 2: Results (%) of PACS on ResNet50 (He et al. 2016). The open-set ratio is 6:1.

Method	Photo (P)			Art (A)			Cartoon (C)			Sketch (S)			Avg		
	Acc	H-score	OSCR	Acc	H-score	OSCR	Acc	H-score	OSCR	Acc	H-score	OSCR	Acc	H-score	OSCR
ARPL (Chen et al. 2022)	97.09	96.81	96.86	88.24	77.48	80.32	82.68	67.19	68.31	78.08	70.04	69.47	86.52	77.88	78.74
MIRO (Cha et al. 2022)	94.85	92.32	93.27	88.51	65.02	79.01	82.98	63.05	73.72	82.22	69.47	70.61	87.14	72.47	79.15
MLDG (Li et al. 2018a)	96.77	95.85	96.33	87.99	77.16	79.93	83.45	68.74	71.32	82.25	73.16	72.27	87.61	78.73	79.96
ERM (Vapnik 1998)	97.09	96.58	96.68	89.99	76.05	82.44	85.10	65.79	70.59	80.31	70.29	70.16	88.12	77.18	79.97
CIRL (Lv et al. 2022)	96.53	87.75	95.40	92.06	70.75	77.44	85.71	68.82	73.71	84.35	66.73	77.24	89.66	73.51	80.95
MixStyle (Zhou et al. 2020b)	96.53	93.57	95.30	90.87	79.15	83.27	86.80	68.08	74.68	84.88	71.57	73.41	89.77	78.09	81.66
CrossMatch (Zhu and Li 2021)	96.53	96.34	96.12	91.37	75.67	82.32	83.92	67.02	74.55	81.61	72.03	73.99	88.37	77.76	81.75
SWAD (Cha et al. 2021)	96.37	84.56	93.24	93.75	68.41	85.00	85.57	58.57	75.90	81.90	74.66	74.65	89.40	71.55	82.20
MVDG (Zhang et al. 2022)	97.17	95.02	96.63	**92.50**	79.47	85.02	86.02	71.05	76.03	83.44	75.24	75.18	89.78	80.20	83.21
ODG-Net (Bose et al. 2023)	96.53	94.93	95.58	89.24	65.22	74.60	83.86	64.32	71.20	84.80	77.58	77.38	88.61	75.51	79.69
MEDIC-cls (Wang et al. 2023b)	96.37	93.80	95.37	91.62	80.80	84.67	86.65	75.85	77.48	84.61	75.80	76.79	89.81	81.56	83.58
MEDIC-bcls (Wang et al. 2023b)	96.37	94.75	95.79	91.62	81.61	85.81	86.65	77.39	78.30	84.61	78.35	79.50	89.81	83.03	84.85
EBiL-HaDS-bcls (Wang et al. 2024)	97.82	96.04	97.14	92.31	82.80	86.17	87.52	78.34	79.85	85.91	78.68	81.32	90.89	83.97	86.12
OGDiff-cls (ours)	**98.39**	95.24	96.97	91.04	82.65	85.75	**89.13**	77.40	80.03	**88.21**	77.63	80.14	**92.19**	83.23	85.72
OGDiff-bcls (ours)	**98.39**	**97.72**	**98.05**	91.04	**88.47**	**87.67**	**89.13**	**80.07**	**81.76**	**88.21**	**80.33**	**82.90**	**92.19**	**86.15**	**87.59**

its effectiveness in leveraging transformer capacity for both domain and open-set generalization.

Experimental Results on DigitsDG Benchmarks

Table4 shows that OGDiff outperforms prior methods including MixStyle, ERM, SWAD, and MEDIC on DigitsDG using ConvNet with an open-set ratio of 6:4. OGDiff consistently leads in both accuracy and OSCR across all datasets and on average. These results highlight its ability to model domain cues effectively, enabling strong generalization to unseen domains and improved open-set recognition.

Table 3: Results (%) of PACS on ViT-B. The open-set ratio is 6:1.

Method	Photo (P)			Art (A)			Cartoon (C)			Sketch (S)			Avg		
	Acc	H-score	OSCR	Acc	H-score	OSCR	Acc	H-score	OSCR	Acc	H-score	OSCR	Acc	H-score	OSCR
ARPL	99.19	95.31	98.61	90.49	85.46	88.59	81.88	72.17	73.34	63.01	29.33	50.59	83.64	70.57	77.78
MLDG	99.19	95.40	98.88	91.87	82.46	89.47	80.56	69.62	74.19	61.66	40.79	43.88	83.32	72.07	76.61
SWAD	98.55	93.19	97.62	90.81	81.34	88.52	83.24	73.03	76.59	57.89	35.83	41.68	82.62	70.85	76.10
ODG-Net	97.58	96.24	95.23	90.49	83.32	87.90	82.36	68.66	75.80	62.59	43.59	50.22	83.26	72.95	77.29
MEDIC-cls	99.03	95.33	98.22	92.06	83.27	87.46	85.62	69.79	75.37	68.40	41.95	56.56	86.28	72.59	79.40
MEDIC-bcls	99.03	96.04	97.55	92.06	82.68	87.73	85.62	69.15	76.80	68.40	39.60	55.92	86.28	71.87	79.50
OGDiff-cls (ours)	**99.54**	93.21	95.85	**94.98**	73.13	75.61	**88.05**	**69.91**	72.94	**70.97**	76.25	75.98	**88.39**	78.13	80.10
OGDiff-bcls (ours)	**99.54**	94.82	95.73	**94.98**	**77.45**	**79.22**	**88.05**	69.73	**73.96**	**70.97**	**80.07**	**79.20**	**88.39**	**80.27**	**81.53**

Table 4: Results (%) of Digits-DG on ConvNet (Zhou et al. 2021). The open-set ratio is 6:4.

Method	MNIST			MNIST-M			SVHN			SYN			Avg		
	Acc	H-score	OSCR	Acc	H-score	OSCR	Acc	H-score	OSCR	Acc	H-score	OSCR	Acc	H-score	OSCR
MixStyle (Zhou et al. 2020b)	97.86	73.25	89.36	74.50	59.30	56.95	69.28	53.24	48.43	85.06	60.22	65.44	81.68	61.50	65.05
ERM (Vapnik 1998)	97.47	80.90	92.60	71.03	53.92	54.04	71.08	54.37	49.86	85.67	51.57	67.63	81.31	60.19	66.03
ARPL (Chen et al. 2022)	97.75	85.74	91.86	69.78	58.08	54.21	71.78	56.98	53.63	85.31	64.04	65.89	81.16	66.21	66.40
MLDG (Li et al. 2018a)	97.83	80.36	94.28	71.11	46.84	55.17	73.64	53.54	53.64	86.08	63.56	70.34	82.16	61.08	68.36
SWAD (Cha et al. 2021)	97.71	84.44	92.65	73.09	53.35	55.94	76.08	59.18	56.25	87.95	51.27	69.03	83.71	62.06	68.47
ODG-Net (Bose et al. 2023)	96.86	71.34	90.93	72.92	58.47	56.98	69.83	55.74	51.55	85.42	67.67	68.12	81.26	63.31	66.90
MEDIC-cls (Wang et al. 2023b)	97.89	67.37	96.17	71.14	48.44	55.37	76.00	51.20	55.58	88.11	64.90	73.62	83.28	57.98	70.19
MEDIC-bcls (Wang et al. 2023b)	97.89	83.20	95.81	71.14	**60.98**	58.28	76.00	58.77	57.60	88.11	62.24	72.91	83.28	66.30	71.15
OGDiff-cls	**99.39**	88.23	97.93	**76.07**	57.68	**61.38**	**80.23**	62.44	**64.79**	**93.86**	**75.22**	78.94	**87.39**	70.89	76.81
OGDiff-bcls	**99.39**	**92.49**	**97.96**	**76.07**	61.89	59.84	**80.23**	**63.44**	63.93	**93.86**	71.17	**79.95**	**87.39**	**72.25**	**76.99**

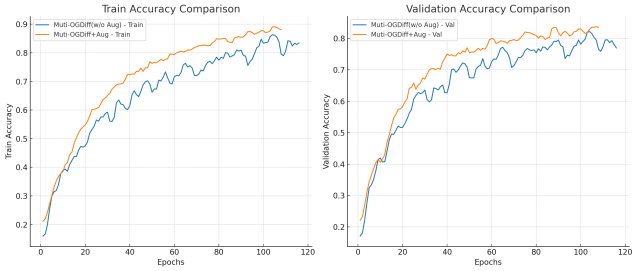

Figure 4: Effect of domain-aware augmentation: training and validation accuracy curves.

Ablation Study

To better understand the impact of each component in OGDiff, we conduct an ablation study on the PACS benchmark using ResNet18 as the backbone. The results are shown in Table 5. We start from a baseline OSDG model without any diffusion-based optimization, then incrementally add key components of OGDiff. Removing data augmentation (DA) leads to a drop in both accuracy and OSCR, confirming that domain-aware augmentation remains a strong prior for generalization. Excluding the MSDFF module also results in performance degradation, indicating the importance of multi-scale feature fusion across diffusion steps. When all components are integrated, OGDiff achieves the highest accuracy and OSCR, demonstrating the effectiveness of the proposed design in OSDG.

Table 5: Ablation study on PACS (ResNet18 backbone).

Method Variant	Avg Acc (%)	OSCR (%)
Baseline OSDG	86.20	75.69
+ OGDiff (w/o DA)	87.92	80.80
+ OGDiff (w/o MSDFF)	87.40	79.52
+ OGDiff	**88.59**	**81.53**

Note: DA denotes data augmentation.

Related Work

Due to space limitations in the main text, a comprehensive discussion of related work is deferred to App. B.

Conclusion

In summary, this work presents the novel OGDiff framework, which models the optimization of base learner parameters as a conditional diffusion process, significantly improving both performance and training efficiency in open-set domain generalization. The introduced Multi-Scale Diffusion Feature Fusion module further enhances the model's representational power. Future work will explore applying OGDiff to more complex scenarios and integrating self-supervised learning to boost generalization.

References

Bose, S.; Jha, A.; Kandala, H.; and Banerjee, B. 2023. Beyond Boundaries: A Novel Data-Augmentation Discourse for Open Domain Generalization. *TMLR*.

Cha, J.; Chun, S.; Lee, K.; Cho, H.-C.; Park, S.; Lee, Y.; and Park, S. 2021. Swad: Domain generalization by seeking flat minima. In *NeurIPS*.

Cha, J.; Lee, K.; Park, S.; and Chun, S. 2022. Domain generalization by mutual-information regularization with pretrained models. In *ECCV*.

Chen, G.; Peng, P.; Wang, X.; and Tian, Y. 2022. Adversarial Reciprocal Points Learning for Open Set Recognition. *TPAMI*.

Dou, Q.; Coelho de Castro, D.; Kamnitsas, K.; and Glocker, B. 2019. Domain generalization via model-agnostic learning of semantic features. In *NeurIPS*.

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

Hospedales, T.; Antoniou, A.; Micaelli, P.; and Storkey, A. 2022. Meta-learning in neural networks: A survey. *TPAMI*.

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

Katsumata, K.; Kishida, I.; Amma, A.; and Nakayama, H. 2021. Open-Set Domain Generalization VIA Metric Learning. In *ICIP*.

Kim, D.; Yoo, Y.; Park, S.; Kim, J.; and Lee, J. 2021. Selfreg: Self-supervised contrastive regularization for domain generalization. In *ICCV*.

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

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

Li, D.; Zhang, J.; Yang, Y.; Liu, C.; Song, Y.-Z.; and Hospedales, T. M. 2019a. Episodic training for domain generalization. In *ICCV*.

Li, H.; Li, J.; Guan, X.; Liang, B.; Lai, Y.; and Luo, X. 2019b. Research on overfitting of deep learning. In *CIS*.

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

Li, P.; Li, D.; Li, W.; Gong, S.; Fu, Y.; and Hospedales, T. M. 2021. A simple feature augmentation for domain generalization. In *ICCV*.

Liu, H.; Cao, Z.; Long, M.; Wang, J.; and Yang, Q. 2019. Separate to adapt: Open set domain adaptation via progressive separation. In *CVPR*.

Lv, F.; Liang, J.; Li, S.; Zang, B.; Liu, C. H.; Wang, Z.; and Liu, D. 2022. Causality inspired representation learning for domain generalization. In *CVPR*.

Matsuura, T.; and Harada, T. 2020. Domain generalization using a mixture of multiple latent domains. In *AAAI*.

Saito, K.; and Saenko, K. 2021. Ovanet: One-vs-all network for universal domain adaptation. In *ICCV*.

Scheirer, W. J.; de Rezende Rocha, A.; Sapkota, A.; and Boult, T. E. 2013. Toward open set recognition. *TPAMI*.

Shu, Y.; Cao, Z.; Wang, C.; Wang, J.; and Long, M. 2021. Open domain generalization with domain-augmented meta-learning. In *CVPR*.

Vapnik, V. 1998. Statistical learning theory.

Wang, J.; Lan, C.; Liu, C.; Ouyang, Y.; Qin, T.; Lu, W.; Chen, Y.; Zeng, W.; and Yu, P. 2023a. Generalizing to unseen domains: A survey on domain generalization. *TKDE*.

Wang, R.; Zhao, R.-W.; Zhang, X.; and Feng, R. 2024. Towards Evidential and Class Separable Open Set Object Detection. In *AAAI*.

Wang, X.; Zhang, J.; Qi, L.; and Shi, Y. 2023b. Generalizable decision boundaries: Dualistic meta-learning for open set domain generalization. In *ICCV*.

Zhang, J.; Qi, L.; Shi, Y.; and Gao, Y. 2022. MVDG: A Unified Multi-view Framework for Domain Generalization. In *ECCV*.

Zhou, K.; Yang, Y.; Hospedales, T.; and Xiang, T. 2020a. Deep domain-adversarial image generation for domain generalisation. In *AAAI*.

Zhou, K.; Yang, Y.; Qiao, Y.; and Xiang, T. 2020b. Domain Generalization with MixStyle. In *ICLR*.

Zhou, K.; Yang, Y.; Qiao, Y.; and Xiang, T. 2021. Domain adaptive ensemble learning. *TIP*.

Zhu, R.; and Li, S. 2021. CrossMatch: Cross-Classifier Consistency Regularization for Open-Set Single Domain Generalization. In *ICLR*.

Reproducibility Checklist

Instructions for Authors:

This document outlines key aspects for assessing reproducibility. Please provide your input by editing this `.tex` file directly.

For each question (that applies), replace the "Type your response here" text with your answer.

Example: If a question appears as

```
\question{Proofs of all novel claims
are included} {(yes/partial/no)}
Type your response here
```

you would change it to:

```
\question{Proofs of all novel claims
are included} {(yes/partial/no)}
yes
```

Please make sure to:

- Replace ONLY the "Type your response here" text and nothing else.
- Use one of the options listed for that question (e.g., **yes**, **no**, **partial**, or **NA**).
- **Not** modify any other part of the `\question` command or any other lines in this document.

You can `\input` this .tex file right before `\end{document}` of your main file or compile it as a stand-alone document. Check the instructions on your conference's website to see if you will be asked to provide this checklist with your paper or separately.

---

1. General Paper Structure

1.1. Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes/partial/no/NA) (yes)

1.2. Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes/no) (yes)

1.3. Provides well-marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper (yes/no) (yes)

2. Theoretical Contributions

2.1. Does this paper make theoretical contributions? (yes/no) (yes)

If yes, please address the following points:

2.2. All assumptions and restrictions are stated clearly and formally (yes/partial/no) Type your response here

2.3. All novel claims are stated formally (e.g., in theorem statements) (yes/partial/no) (yes)

2.4. Proofs of all novel claims are included (yes/partial/no) (yes)

2.5. Proof sketches or intuitions are given for complex and/or novel results (yes/partial/no) (yes)

2.6. Appropriate citations to theoretical tools used are given (yes/partial/no) (yes)

2.7. All theoretical claims are demonstrated empirically to hold (yes/partial/no/NA) (yes)

2.8. All experimental code used to eliminate or disprove claims is included (yes/no/NA) (NA)

3. Dataset Usage

3.1. Does this paper rely on one or more datasets? (yes/no) (yes)

If yes, please address the following points:

3.2. A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA) (yes)

3.3. All novel datasets introduced in this paper are included in a data appendix (yes/partial/no/NA) (NA)

3.4. All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no/NA) (NA)

3.5. All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations (yes/no/NA) (yes)

3.6. All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available (yes/partial/no/NA) (yes)

3.7. All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing (yes/partial/no/NA) (NA)

4. Computational Experiments

4.1. Does this paper include computational experiments? (yes/no) (yes)

If yes, please address the following points:

4.2. This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting (yes/partial/no/NA) (yes)

4.3. Any code required for pre-processing data is included in the appendix (yes/partial/no) (no)

4.4. All source code required for conducting and analyzing the experiments is included in a code appendix (yes/partial/no) (no)

4.5. All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no) (yes)

4.6. All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no) (partial)

4.7. If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results (yes/partial/no/NA) (partial)

4.8. This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks (yes/partial/no) (partial)

4.9. This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics (yes/partial/no) (yes)

4.10. This paper states the number of algorithm runs used to compute each reported result (yes/no) (no)

4.11. Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information (yes/no) (no)

4.12. The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank) (yes/partial/no) (no)

4.13. This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments (yes/partial/no/NA) (yes)