GCCM: ENHANCING GENERATIVE GRAPH PREDICTION VIA CONTRASTIVE CONSISTENCY MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Conditional generative models, particularly diffusion-based methods, have recently been applied to graph prediction by modeling the target as a conditional distribution given the input graph, yielding competitive results compared to deterministic predictor. However, existing diffusion-based prediction methods typically require expensive iterative denoising at inference and often suffer from unstable sampling, which motivates recent efforts to reduce inference denoising steps and enable stable sampling via techniques such as consistency training. Despite this progress, we find that existing consistency training methods for graph prediction could potentially fall into a shortcut solution: the model may attempt to satisfy the self-consistency constraint by ignoring the noisy target (i.e., assigning it negligible weight), ultimately collapsing into a purely deterministic predictor. To mitigate such shortcut solution, we propose GCCM, a graph contrastive consistency model that goes beyond isolated pairwise matching between the same target at different noise levels by introducing negative pairs into a contrastive consistency objective. This adds an additional separation requirement, making the shortcut solution no longer trivially sufficient to satisfy the proposed objective. Moreover, we apply feature perturbation to the input node/edge features to break identical conditioning on the input graph, so that the shortcut no longer yields the same predictions across noise levels and becomes less attractive. Extensive experiments on benchmark datasets demonstrate that GCCM mitigates the shortcut solution and yields consistent performance improvements in graph prediction compared to deterministic predictors.

1 INTRODUCTION

Prediction tasks on graphs (e.g., node/graph-level predictions) are fundamental problems in graph machine learning, aiming to predict target variables defined over graph-structured data. Earlier methods for graph prediction Kreuzer et al. (2021); Ying et al. (2021); Rampášek et al. (2023) mainly focus on a deterministic paradigm, which directly maps an input node or graph to the target. In recent years, generative models, particularly diffusion models, have demonstrated promising performance across a wide range of generation tasks on graphs Sun & Yang (2023); Vignac et al. (2023); Huang et al. (2025b;a). Benefiting from their strong capability in modeling complex data distributions, increasing attention has been paid to extending diffusion-based paradigms to graph prediction tasks. In contrast to deterministic models, diffusion-based approaches reformulate prediction as a conditional generation problem, modeling the full conditional distribution of target variables rather than producing a single point estimate. Despite their potential, applying diffusion-based models to graph prediction tasks remains challenging in two key aspects: (1) **Step-Level Inefficiency:** diffusion-based methods perform inference through iterative denoising, often involving dozens or hundreds of reverse steps, which introduces substantial computational overhead, limiting their inference efficiency; (2) **Sample-Level Inefficiency:** predictions from a single noise sample can exhibit high variance, leading to unstable inference. To obtain reliable predictions, generative methods typically require aggregating predictions from multiple noise samples, which further increases inference cost.

To address the above challenges, a common strategy is to adopt the consistency training Song et al. (2023); Song & Dhariwal (2024) to enhance diffusion-based models. Specifically, PCL Li et al. (2025b) proposes a predictive consistency framework to train the diffusion-based graph prediction model by enforcing consistency mapping across different noise levels, which enables stable sampling and one-step inference from a single noise sample. Despite its efficiency, we observe that applying

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

consistency training to graph prediction admits a shortcut solution. When predictions at different noise levels are conditioned on the same underlying input graph and enforced to be matched, the model can trivially satisfy the consistency objective by ignoring the noisy target and relying solely on the conditioning input. This potential shortcut collapses the conditional denoising network into a purely deterministic predictor, thus undermining the goal of conditional generation.

Our Contributions. Motivated by the above observation, we propose Graph Contrastive Consistency Models (GCCM) to mitigate such shortcut solution in consistency-based prediction. In particular, **(1)** instead of treating self-consistency as a purely matching constraint between two denoising predictions of the same underlying instance (i.e., a node of node-level prediction and a graph for graph-level prediction), we incorporate negative pairs across different instances in the mini-batch. This yields a contrastive consistency objective that aligns the target representations of the positive pair while separating them from negatives in the latent space, making it harder to satisfy the contrastive consistency objective by simply ignoring the noisy target. Nevertheless, the proposed contrastive consistency objective alone does not fully eliminate the shortcut solution. When the conditioning node/edge features remains identical across the two noisy views, the model may still attempt to satisfy the positive-pair agreement by relying solely on the identical conditioning input and ignoring the noisy target (i.e., the shortcut solution). **(2)** To further enhance the proposed contrastive consistency objective, we apply feature perturbation to the input node/edge features to break identical conditioning, making the shortcut solution less attractive. Together, these two components encourage our proposed GCCM to fully leverage the generative formulation to enhance the performance of predictors. Experimental results show that GCCM outperforms the existing diffusion-based graph prediction baselines on wide-range benchmark datasets and prediction tasks. Detailed ablation studies further verify the contribution of each proposed component in mitigating the shortcut solution.

2 Related Works

Generative diffusion models for graph prediction. In recent years, diffusion models Ho et al. (2020); Song et al. (2022); Nichol & Dhariwal (2021); Rombach et al. (2022) have emerged as a powerful framework for a wide range of generation tasks. Motivated by their strong generative capacity, a growing body of work has extended diffusion models to graph-related generation tasks Vignac et al. (2023); Zhou et al. (2024). Beyond graph generation tasks, LGD Zhou et al. (2024) explores diffusion models for graph prediction by applying diffusion in a latent space to generate latent representations and then mapping them to the target space using a learned decoder. Although effective, it still incurs substantial computational overhead due to iterative denoising at inference time. Building upon diffusion models, consistency models Song et al. (2023); Song & Dhariwal (2024) eliminate iterative reverse diffusion by directly learning a mapping from noisy samples to clean data through the self-consistency objective, enabling efficient one-step or few-step generation. Inspired by the consistency mapping principle, PCL Li et al. (2025b) further develops a predictive consistency framework for prediction tasks by enforcing consistent predictions across different noise levels. This formulation enables stable sampling and one-step inference in diffusion-based graph prediction models. Due to the space limitations, an extended related work regarding the classical deterministic graph prediction models can be found in Appendix A.

3 Preliminaries

3.1 Diffusion Models

In our conditional generation setting, $\mathbf{Y}$ denotes the target variable and $\mathbf{C}$ the conditioning input. Diffusion models Sohl-Dickstein et al. (2015); Hoogeboom et al. (2021); Ho et al. (2020) define a generative process by gradually transforming the clean target $\mathbf{Y}_0$ into noise through a forward process, and learning to recover the clean target progressively through a reverse process.

3.1.1 Forward Process

The forward process gradually corrupts the clean target $\mathbf{Y}_0$ into a sequence of noisy latent variables $\mathbf{Y}_{1:T} = \{\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_T\}$, defined as $q(\mathbf{Y}_{1:T} \mid \mathbf{Y}_0) = \prod_{t=1}^{T} q(\mathbf{Y}_t \mid \mathbf{Y}_{t-1})$.

Discrete targets. For discrete targets, the clean target is represented in one-hot form as $\mathbf{Y}_0 = \mathbf{Y} \in \{0,1\}^{n \times K}$, where n is the number of target variables and K is the number of categories. The forward process is defined as $q(\mathbf{Y}_t \mid \mathbf{Y}_{t-1}) = \mathrm{Cat}\left(\mathbf{Y}_t;\ \mathbf{p} = \mathbf{Y}_{t-1}\mathbf{Q}_t\right)$, where $\mathrm{Cat}(\mathbf{Y}_t; \mathbf{p})$ denotes a categorical distribution over K one-hot vectors with probabilities given by $\mathbf{p}$, and $\mathbf{Q}_t =$

$(1-\beta_t)\mathbf{I} + \frac{\beta_t}{K}\mathbf{1}\mathbf{1}^\top \in \mathbb{R}^{K\times K}$, is the transition matrix that determines the corruption introduced at time step t, where $\beta_t \in (0,1)$ is the corruption rate.

Continuous targets. Let $\mathbf{Y}_0 = \mathbf{Y} \in \mathbb{R}^{n\times 1}$ denote the clean target. The forward process $q(\mathbf{Y}_t \mid \mathbf{Y}_{t-1})$ for the continuous targets perturbs $\mathbf{Y}_0$ by progressively injecting standard Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ over time steps, such that $q(\mathbf{Y}_t \mid \mathbf{Y}_{t-1}) = \mathcal{N}\left(\mathbf{Y}_t;\ \sqrt{1-\beta_t}\,\mathbf{Y}_{t-1},\ \beta_t\mathbf{I}\right)$, where $\beta_t \in (0,1)$ controls the noise level at time step t.

3.1.2 Reverse Process

The reverse process reconstructs the clean target by defining a sequence of learned conditional transitions $p_\theta(\mathbf{Y}_{0:T} \mid \mathbf{C}) = p(\mathbf{Y}_T)\prod_{t=1}^{T} p_\theta\left(\mathbf{Y}_{t-1} \mid \mathbf{Y}_t, \mathbf{C}\right)$ that progressively transforms $\mathbf{Y}_T$ toward $\mathbf{Y}_0$. Each transition $p_\theta\left(\mathbf{Y}_{t-1} \mid \mathbf{Y}_t, \mathbf{C}\right)$ is parameterized by the denoising network f_θ. During training, diffusion models sample a time step t and obtain the corresponding noisy target $\mathbf{Y}_t$ via the forward process. During inference, they start from a random noisy target $\mathbf{Y}_T$ and iteratively apply the learned reverse transitions to denoise $\mathbf{Y}_T$ towards the target distribution.

Discrete targets. For discrete targets, a common parameterization is to train a denoising network $f_\theta(\mathbf{Y}_t, t, \mathbf{C})$ to predict the clean-target distribution, and then obtain the reverse distribution by marginalizing over the predicted clean target $\tilde{\mathbf{Y}}_0$, such that $p_\theta(\mathbf{Y}_{t-1} \mid \mathbf{Y}_t, \mathbf{C}) = \sum_{\tilde{\mathbf{Y}}} q(\mathbf{Y}_{t-1} \mid \mathbf{Y}_t, \tilde{\mathbf{Y}}_0)\ f_\theta(\mathbf{Y}_t, t, \mathbf{C})$, where the posterior $q(\mathbf{Y}_{t-1} \mid \mathbf{Y}_t, \mathbf{Y}_0)$ can be obtained by $q(\mathbf{Y}_{t-1} \mid \mathbf{Y}_t, \mathbf{Y}_0) = \frac{q(\mathbf{Y}_t|\mathbf{Y}_{t-1},\mathbf{Y}_0)\ q(\mathbf{Y}_{t-1}|\mathbf{Y}_0)}{q(\mathbf{Y}_t|\mathbf{Y}_0)}$. The training objective minimizes the expected discrepancy between the predicted target and the ground-truth clean target: $\mathcal{L}_{\text{disc}}(\theta) = \mathbb{E}\Big[d\big(f_\theta(\mathbf{Y}_t, t, \mathbf{C}),\ \mathbf{Y}_0\big)\Big]$, where $d(\cdot,\cdot)$ is typically the cross-entropy loss for discrete targets.

Continuous targets. For continuous targets, the reverse Gaussian transition is defined as $p_\theta(\mathbf{Y}_{t-1} \mid \mathbf{Y}_t, \mathbf{C}) = \mathcal{N}\Big(\mathbf{Y}_{t-1}; \mu_\theta(\mathbf{Y}_t, t, \mathbf{C}), \beta_t\mathbf{I}\Big)$, where $\mu_\theta(\mathbf{Y}_t, t, \mathbf{C})$ and $\beta_t\mathbf{I}$ denote the mean and covariance of the reverse transition, respectively. Following the noise-prediction parameterization, the denoising network predicts the injected Gaussian noise $\hat{\boldsymbol{\epsilon}} = (\mathbf{Y}_t - \sqrt{\bar{\alpha}_t}\,\mathbf{Y}_0) \,/\, \sqrt{1-\bar{\alpha}_t} = f_\theta(\mathbf{Y}_t, t, \mathbf{C})$, which induces the mean function $\mu_\theta(\mathbf{Y}_t, t, \mathbf{C}) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{Y}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\, f_\theta(\mathbf{Y}_t, t, \mathbf{C})\Big)$. The training objective minimizes the expected discrepancy between the predicted noise and the injected noise: $\mathcal{L}_{\text{cont}}(\theta) = \mathbb{E}\Big[d\big(f_\theta(\mathbf{Y}_t, t, \mathbf{C}),\ \boldsymbol{\epsilon}\big)\Big]$, where $d(\cdot,\cdot)$ is typically MSE for continuous targets.

3.2 Consistency Models

Consistency models Song et al. (2023); Song & Dhariwal (2024) are a family of generative models that aim to enable stable sampling and one-step inference for the diffusion reverse process by directly predicting the clean target from a randomly sampled noisy target, such that $\hat{\mathbf{Y}}_0^T = f_\theta(\mathbf{Y}_T, T, \mathbf{C})$. To achieve this, unlike diffusion models that sample only one noisy target in the forward process, consistency models sample two noisy targets $\mathbf{Y}_{t_1}$ and $\mathbf{Y}_{t_2}$ by sampling two time steps t_1 and t_2. The denoising network f_θ is then required to satisfy two key properties: **(i) Boundary condition.** At noise level $t = 0$, the output should match the clean target: $f_\theta(\mathbf{Y}_0, 0, \mathbf{C}) = \mathbf{Y}_0$; **(ii) Self-consistency.** Outputs across different noise levels should be consistent: $f_\theta(\mathbf{Y}_{t_1}, t_1, \mathbf{C}) = f_\theta(\mathbf{Y}_{t_2}, t_2, \mathbf{C})$.

To satisfy the boundary condition, consistency models adopt a skip-connection parameterization:

$$f_\theta(\mathbf{Y}_t, t, \mathbf{C}) = c_{\text{skip}}(t)\,\mathbf{Y}_0 + c_{\text{out}}(t)\,F_\theta(\mathbf{Y}_t, t, \mathbf{C}), \tag{1}$$

where F_θ is a free-form neural network, and $c_{\text{skip}}(t)$ and $c_{\text{out}}(t)$ are differentiable functions such that $c_{\text{skip}}(0) = 1$ and $c_{\text{out}}(0) = 0$. And to satisfy the self-consistency property, the training objective of consistency models is defined as

$$\mathcal{L}_{\text{CM}}(\theta) = \mathbb{E}\Big[d\Big(f_\theta(\mathbf{Y}_{t_1}, t_1, \mathbf{C}),\ f_{\theta^-}(\mathbf{Y}_{t_2}, t_2, \mathbf{C})\Big)\Big], \tag{2}$$

where θ^- denotes $\text{stopgrad}(\theta)$.

4 Methodology

4.1 Generative Graph Prediction & Identification of the Shortcut Solution

In this section, we formulate each prediction task on graphs using node embeddings for simplicity. Consider a graph $\mathcal{G} = (\mathbf{X}, \mathbf{A})$ with a set of n nodes, where $\mathbf{X} \in \mathbb{R}^{n\times d_x}$ is the node feature matrix,

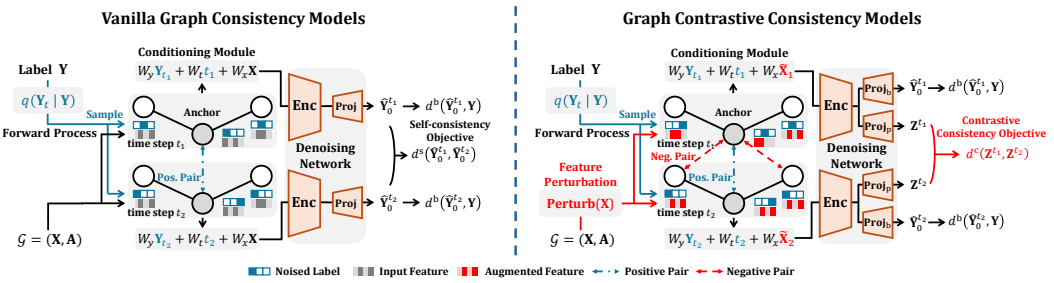


Figure 1: Illustration of Graph Contrastive Consistency Models (GCCM). Compared with vanilla consistency training for graph prediction, which uses d^b to enforce the boundary condition and d^s to enforce the self-consistency constraint, GCCM introduces contrastive consistency via d^c on representations from two independently noisy labels, and further mitigates the shortcut solution via feature perturbation that breaks identical conditioning across consistency passes.

$\mathbf{A} \in \{0,1\}^{n\times n}$ is the adjacency matrix. We denote $\mathbf{Y} \in \mathbb{R}^{n\times d_k}$ as the target node labels for node-level prediction tasks, and $\mathbf{Y} \in \mathbb{R}^{1\times d_k}$ as the target graph label for graph-level prediction tasks. Specifically, d_k denotes the label dimension, where $d_k = K$ for classification tasks with one-hot labels, and $d_k = 1$ for regression tasks.

To formulate prediction tasks on graphs as conditional generation, we adopt a diffusion model and treat the input graph $\mathcal{G} = (\mathbf{X}, \mathbf{A})$ as the conditioning signal and the target node/graph labels $\mathbf{Y}$ as the generation target. We then apply the forward process as described in Section 3 to obtain noisy labels $\mathbf{Y}_t$ at noise level t. Notably, for graph-level prediction, we apply the forward process to the graph label $\mathbf{Y} \in \mathbb{R}^{1\times d_k}$ to obtain the noisy graph label $\mathbf{Y}_t \in \mathbb{R}^{1\times d_k}$, and then broadcast it to all nodes to form $\mathbf{Y}_t \in \mathbb{R}^{n\times d_k}$. Next, the denoising network f_θ takes $(\mathbf{Y}_t, t, \mathcal{G})$ as input and first uses a conditioning module Zhou et al. (2024); Li et al. (2025b) to fuse $(\mathbf{Y}_t, t, \mathbf{X})$, such that:

$$\mathbf{H}_t^{(0)} = W_y\, \mathbf{Y}_t + W_t\, \mathbf{t}_{\text{emb}}(t) + W_x\, \mathbf{X}, \tag{3}$$

Here, $\mathbf{H}^{(0)} \in \mathbb{R}^{n\times d_h}$ denotes the node-level fused initial hidden embedding matrix, $\mathbf{t}_{\text{emb}}(t)$ is the time step embedding, W_y, W_t, and W_x are learnable projection matrices that map $\mathbf{Y}_t$, $\mathbf{t}_{\text{emb}}(t)$, and $\mathbf{X}$ into the same d_h-dimensional space. A graph encoder $\text{Enc}(\cdot)$ is then applied to take $(\mathbf{H}_t^{(0)}, \mathbf{A})$ as inputs and update the node embeddings to $\mathbf{H}_t$. Finally, a predictive projection head $\text{Proj}_p(\cdot)$ is applied to $\mathbf{H}_t$ to estimate the clean node/graph labels $\hat{\mathbf{Y}}_0^t$:

$$\hat{\mathbf{Y}}_0^t = f_\theta(\mathbf{Y}_t, t, \mathcal{G}) = f_\theta(\mathbf{H}_t^{(0)}, \mathbf{A}) = \text{Proj}_p\left(\text{Enc}\left(\mathbf{H}_t^{(0)}, \mathbf{A}\right)\right) \tag{4}$$

To accelerate diffusion models, a common strategy is to adopt consistency models for stable sampling and one-step inference. However, the consistency models in Section 3.2 are not tailored to prediction settings, because their skip-connection parameterization in Eq. 1 assumes that the clean labels $\mathbf{Y}_0$ and the model output lie in the same continuous space and can be combined additively. For graph classification, the model outputs a categorical distribution, whereas the clean labels are discrete class labels. Thus, directly enforcing the boundary condition through additive mixing is less natural and usually requires task-specific adaptations. To address this issue, PCL Li et al. (2025b) enhances consistency models by enforcing the boundary condition through an explicit supervised objective, rather than an additive skip-connection parameterization, which allows the model to accommodate different label types. Specifically, unlike the vanilla consistency objective in Eq. 2, PCL adopts a triadic distance formulation to jointly enforce these two properties, yielding the training objective:

$$\mathcal{L}(\theta) = \mathbb{E}\Big[\lambda_1\Big(d^b\left(\hat{\mathbf{Y}}_0^{t_1}, \mathbf{Y}\right) + d^b\left(\hat{\mathbf{Y}}_0^{t_2}, \mathbf{Y}\right)\Big) + \lambda_2\, d^s\left(\hat{\mathbf{Y}}_0^{t_1}, \hat{\mathbf{Y}}_0^{t_2}\right)\Big]. \tag{5}$$

Here, the boundary condition objective $d^b(\cdot,\cdot)$ is instantiated as a task-specific supervised loss, e.g., cross-entropy for classification tasks and mean absolute error for regression tasks on graphs, while the self-consistency objective $d^s(\cdot,\cdot)$ is typically implemented using the MSE between the two predictions. For notational convenience, throughout this paper we instantiate d^s as the MSE distance unless otherwise specified. The overall framework of PCL for graph prediction is illustrated in Fig. 1.

While the above consistency training framework is simple and effective, we find that minimizing the training objective in Eq. equation 5 can lead to a shortcut solution in graph prediction settings.

The underlying reason is that the self-consistency objective d^s enforces agreement between two predictions $\hat{\mathbf{Y}}_0^{t_1} := f_\theta(\mathbf{Y}_{t_1}, t_1, \mathcal{G})$ and $\hat{\mathbf{Y}}_0^{t_2} := f_\theta(\mathbf{Y}_{t_2}, t_2, \mathcal{G})$, both of which are conditioned on the same input graph $\mathcal{G}$. Under this setup, the only factors that vary across the two passes are the noisy labels $\mathbf{Y}_t$ and the corresponding time step t. Consequently, reducing the influence of $(\mathbf{Y}_t, t)$ on the predictions becomes an easy way to satisfy the agreement requirement, which can be achieved by down-weighting the corresponding fusion terms via W_y and W_t in Eq. equation 3. Moreover, the boundary condition objective d^b does not prevent the model from converging to such a shortcut solution, as we will formalize in the following lemma:

Lemma 1. *Consider the training objective in Eq. equation 5* $\mathcal{L}(\theta) = \mathbb{E}\left[\lambda_1\left(d^b(\hat{\mathbf{Y}}_0^{t_1}, \mathbf{Y}) + d^b(\hat{\mathbf{Y}}_0^{t_2}, \mathbf{Y})\right) + \lambda_2\, d^s(\hat{\mathbf{Y}}_0^{t_1}, \hat{\mathbf{Y}}_0^{t_2})\right]$, *where each predicted label* $\hat{\mathbf{Y}}_0^t$ *can be rewritten as* $\hat{\mathbf{Y}}_0^t = f_\theta(\mathbf{Y}_t, t, \mathcal{G}) = f_\theta(\mathbf{H}_t^{(0)}, \mathbf{A}) = f_\theta\Big(W_y\,\mathbf{Y}_t + W_t\,\mathbf{t}_{\text{emb}}(t) + W_x\,\mathbf{X}, \mathbf{A}\Big)$. *Minimizing the above objective (Eq. equation 5) could potentially lead the denoising network* f_θ *to a shortcut solution set, such that the noisy labels* $\mathbf{Y}_t$ *and time step* t *are ignored:*

$$\mathcal{F}_{\text{sc}} := \{\theta :\ W_y = \mathbf{0},\, W_t = \mathbf{0}\}, \tag{6}$$

where $\mathbf{0}$ *denotes the zero matrix with the same shape as the corresponding projection matrix.*

A formal proof is in Appendix C.1. Under the shortcut solution set $\mathcal{F}_{\text{sc}}$ in Lemma 1, Eq. equation 3 simplifies to $\mathbf{H}_t^{(0)} = W_x\,\mathbf{X}$, and the denoising network collapses into a deterministic predictor $g_\theta(\cdot)$ that depends only on the input graph $\mathcal{G}$, i.e., $f_\theta(\mathbf{Y}_t, t, \mathcal{G}) := g_\theta(\mathcal{G})$. Since both predictions become identical regardless of the noise level, the self-consistency objective attains its minimum value 0, and the training objective in Eq. equation 5 can be simplified as: $\mathcal{L}(\theta) = \mathbb{E}\Big[\lambda_1\Big(d^b\left(g_\theta(\mathcal{G}), \mathbf{Y}\right) + d^b\left(g_\theta(\mathcal{G}), \mathbf{Y}\right)\Big) + \lambda_2 \cdot 0\Big] = 2\lambda_1\,\mathbb{E}\Big[d^b\left(g_\theta(\mathcal{G}), \mathbf{Y}\right)\Big] + \lambda_2 \cdot 0$. The resulting objective is equivalent to a standard deterministic supervised loss, which the model can minimize in the same way as a deterministic predictor to reduce the overall loss. Therefore, the boundary condition objective d^b does not impose any additional restriction that prevents the model from taking the shortcut solution.

4.2 Mitigating Shortcut Solution via Contrastive Consistency Objective

The reason why minimizing the training objective in Eq. equation 5 could lead to the shortcut solution in Lemma 1 is that the self-consistency objective d^s only enforces pairwise matching between two predictions from the same underlying instance (i.e., a node for node-level prediction and a graph for graph-level prediction), and thus can be trivially satisfied by ignoring the noisy labels and the corresponding time step. Since the boundary condition objective d^b does not prevent this shortcut, mitigating it requires modifying the self-consistency objective itself. To mitigate this shortcut, we incorporate negative pairs across different instances and propose a graph contrastive consistency objective to replace the self-consistency objective. This introduces an extra separation requirement beyond positive pairwise matching, making the shortcut solution in Eq. equation 6 no longer trivially sufficient to satisfy the self-consistency constraint.

Specifically, given a mini-batch of N instances with clean labels $\{\mathbf{y}_{0,i}\}_{i=1}^N$, we apply the forward process at two time steps t_1 and t_2 to obtain noisy label sets $\{\mathbf{y}_{t_1,i}\}_{i=1}^N$ and $\{\mathbf{y}_{t_2,i}\}_{i=1}^N$. Taking $\mathbf{y}_{t_1,i}$ as the anchor, we treat $(\mathbf{y}_{t_1,i}, \mathbf{y}_{t_2,i})$ as a positive pair and use the remaining noisy labels in the second set as negatives, i.e., $\{\mathbf{y}_{t_2,j}\}_{j\neq i}$. In this setting, we compute the contrastive consistency objective on the corresponding predicted clean labels $\{\hat{\mathbf{y}}_{0,i}^{t_1}\}_{i=1}^N$ and $\{\hat{\mathbf{y}}_{0,i}^{t_2}\}_{i=1}^N$ in label space, to align the positive pairs while separating the negatives. However, directly computing the contrastive consistency objective in the label space may induce invariance and information compression. Since the prediction outputs in this space are primarily optimized to satisfy the boundary condition objective $d^b(\cdot,\cdot)$, such effects can undesirably reshape the predictions and interfere with the supervision imposed by $d^b(\cdot,\cdot)$.

To alleviate this issue, we compute the contrastive consistency objective in a latent space, allowing alignment and separation of latent representations induced by instances, while avoiding direct constraints on the prediction outputs that are required to satisfy the boundary condition objective. Specifically, we apply an additional projection head $\text{Proj}_c(\cdot)$ to the shared encoder output $\mathbf{h}_t$ to obtain a latent representation $\mathbf{z}_i^t \in \mathbb{R}^{d_z}$ for each noisy label $\mathbf{y}_{t,i}$ in the mini-batch. The contrastive consistency objective $d^c(\cdot,\cdot)$ is then computed on these latent representations, while the boundary condition objective $d^b(\cdot,\cdot)$ is kept unchanged in the label space. With the latent representations

$\{\mathbf{z}_i^{t_1}\}_{i=1}^N$ and $\{\mathbf{z}_i^{t_2}\}_{i=1}^N$, we define the contrastive consistency loss from t_1 to t_2 as

$$\mathcal{L}_{\mathrm{CL}}^{t_1\to t_2} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp\left(\mathrm{sim}\left(\mathbf{z}_i^{t_1},\mathbf{z}_i^{t_2}\right)/\tau\right)}{\sum_{j=1}^{N}\exp\left(\mathrm{sim}\left(\mathbf{z}_i^{t_1},\mathbf{z}_j^{t_2}\right)/\tau\right)}, \text{with } \mathrm{sim}(\mathbf{z}_i^{t_1},\mathbf{z}_j^{t_2}) = \frac{(\mathbf{z}_i^{t_1})^T\mathbf{z}_j^{t_2}}{\|\mathbf{z}_i^{t_1}\|\,\|\mathbf{z}_j^{t_2}\|}. \tag{7}$$

where $\mathrm{sim}(\cdot,\cdot)$ denotes cosine similarity, and τ is a temperature parameter. The reverse direction from t_2 to t_1 is defined analogously. For notational convenience, we stack each latent representation in the mini-batch into a matrix $\mathbf{Z}^t \in \mathbb{R}^{N\times d_z}$, whose i-th row is $\mathbf{z}_i^t$. The overall contrastive consistency objective $d^c(\cdot,\cdot)$ is defined as

$$d^c(\mathbf{Z}^{t_1},\mathbf{Z}^{t_2}) = \frac{1}{2}\Big(\mathcal{L}_{\mathrm{CL}}^{t_1\to t_2} + \mathcal{L}_{\mathrm{CL}}^{t_2\to t_1}\Big). \tag{8}$$

Replacing the self-consistency term $d^s(\cdot,\cdot)$ in Eq. equation 5 with the contrastive consistency objective $d^c(\cdot,\cdot)$, the overall training objective is updated as:

$$\mathcal{L} = \mathbb{E}\Big[\lambda_1\Big(d^b(\hat{\mathbf{Y}}_0^{t_1},\mathbf{Y}) + d^b(\hat{\mathbf{Y}}_0^{t_2},\mathbf{Y})\Big) + \lambda_2\, d^c(\mathbf{Z}^{t_1},\mathbf{Z}^{t_2})\Big]. \tag{9}$$

Here, $\hat{\mathbf{Y}}_0^t \in \mathbb{R}^{N\times d_k}$ denotes the predicted clean labels in the mini-batch, while $\mathbf{Y} \in \mathbb{R}^{N\times d_k}$ denotes the corresponding ground-truth clean labels stacked across the same mini-batch.

Recall from Lemma 1 that, under the original objective in Eq. equation 5, the shortcut solution set $\mathcal{F}_{\mathrm{sc}}$ is sufficient to minimize the self-consistency term d^s, which reduces the joint objective to a standard supervised objective. The following lemma shows that $\mathcal{F}_{\mathrm{sc}}$ is no longer sufficient to minimize the contrastive consistency objective d^c:

Lemma 2. *Consider the contrastive consistency objective defined in Eq. equation 8. Under the shortcut solution set $\mathcal{F}_{\mathrm{sc}}$ defined in Eq. equation 6, i.e., $\mathcal{F}_{\mathrm{sc}} := \{\theta :\ W_y = \mathbf{0},\, W_t = \mathbf{0}\}$. $\mathcal{F}_{\mathrm{sc}}$ is generally no longer sufficient to minimize the contrastive consistency objective d^c.*

A formal proof is provided in Appendix C.2. Lemma 2 follows from the fact that, unlike the self-consistency objective that performs pairwise matching between two predictions, the contrastive consistency objective couples each noisy label in one noisy set with the other noisy set through negative pairs, and thus imposes an additional separation requirement across different instances. As a result, the shortcut solution that ignores $(\mathbf{Y}_t, t)$ by down-weighting W_y and W_t in Eq. equation 3 cannot in general satisfy the positive-agreement and negative-separation requirements simultaneously. Therefore, unlike the self-consistency objective, the contrastive consistency objective discourages the denoising network from collapsing into a purely deterministic predictor that ignores $(\mathbf{Y}_t, t)$.

4.3 Enhancing Contrastive Consistency Objective with Feature Perturbation

Although the proposed contrastive consistency objective $d^c(\cdot,\cdot)$ introduces negative pairs across different instances, making the shortcut solution set $\mathcal{F}_{\mathrm{sc}}$ no longer trivially sufficient to minimize $d^c(\cdot,\cdot)$, the denoising network may still learn to reduce its reliance on $(\mathbf{Y}_t, t)$ by down-weighting W_y and W_t in Eq. equation 3 during training. The underlying reason is that the contrastive consistency objective contains the positive-pair agreement requirement, which the shortcut solution set naturally satisfies. Under the shortcut solution set defined in Eq. equation 6, the fusion in Eq. equation 3 reduces to $\mathbf{H}_t^{(0)} = W_x\mathbf{X}$, so the output latent representation matrix $\mathbf{Z}^t$ is independent of $(\mathbf{Y}_t, t)$:

$$\mathbf{Z}^t = f_\theta\big(\mathbf{Y}_t, t, \mathcal{G}\big) = f_\theta\big(\mathbf{H}_t^{(0)}, \mathbf{A}\big) = \mathrm{Proj}_c\left(\mathrm{Enc}\left(\mathbf{H}_t^{(0)}, \mathbf{A}\right)\right) = \mathrm{Proj}_c\left(\mathrm{Enc}\left(W_x\mathbf{X}, \mathbf{A}\right)\right). \tag{10}$$

Since the graph encoder receives the same input features $\mathbf{X}$ and $\mathbf{A}$ at t_1 and t_2, Eq. equation 10 implies that the resulting latent representations are identical, i.e., $\mathbf{Z}^{t_1} = \mathbf{Z}^{t_2}$. This equality makes positive-pair alignment trivial, since for each instance i in the mini-batch we have $\mathbf{z}_i^{t_1} = \mathbf{z}_i^{t_2}$. Substituting this into Eq. equation 7 yields $\mathcal{L}_{\mathrm{CL}}^{t_1\to t_2} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(1/\tau)}{\sum_{j=1}^{N}\exp\left(\mathrm{sim}\left(\mathbf{z}_i^{t_1},\mathbf{z}_j^{t_2}\right)/\tau\right)}$. In this form, the positive-pair agreement term $\mathrm{sim}\left(\mathbf{z}_i^{t_1},\mathbf{z}_i^{t_2}\right)$ becomes a constant, so the contrastive consistency objective is reduced to separating negative pairs across different instances only. Crucially, this negative-pair separation requirement cannot constrain the model from moving toward the shortcut solution $\mathcal{F}_{\mathrm{sc}}$. In this case, the fused representation in Eq. equation 3 reduces to $\mathbf{H}_t^{(0)} = W_x\mathbf{X}$, so negative pairs can still be separated based on the input features, without relying on $(\mathbf{Y}_t, t)$. Therefore, although the shortcut solution is no longer trivially sufficient to minimize the contrastive consistency

Table 1: Test performance on benchmarks from Dwivedi et al. (2022; 2023). Results for baselines (except GPS, GRIT, GPS-LGD, and GPS-PCL) are taken from Rampášek et al. (2023).

Model	MNIST	CIFAR10	PATTERN	CLUSTER	ZINC	PascalVOC-SP	COCO-SP
	Accuracy ↑	Accuracy ↑	Accuracy ↑	Accuracy ↑	MAE ↓	F1 ↑	F1 ↑
GCN Kipf & Welling (2017)	90.71 ± 0.22	55.71 ± 0.38	71.892 ± 0.334	68.498 ± 0.976	0.367 ± 0.011	0.1268 ± 0.0060	0.0841 ± 0.0010
GIN Xu et al. (2019)	96.49 ± 0.25	55.26 ± 1.53	85.387 ± 0.136	64.716 ± 1.553	0.526 ± 0.051	—	—
GAT Veličković et al. (2018)	95.54 ± 0.21	64.22 ± 0.46	78.271 ± 0.186	70.587 ± 0.447	0.384 ± 0.007	—	—
GatedGCN Bresson & Laurent (2018)	97.34 ± 0.14	67.31 ± 0.31	85.568 ± 0.088	73.840 ± 0.326	0.282 ± 0.015	0.2873 ± 0.0219	0.2641 ± 0.0045
PNA Corso et al. (2020)	97.94 ± 0.12	70.35 ± 0.63	—	—	0.188 ± 0.004	—	—
DGN Beaini et al. (2021)	—	72.84 ± 0.42	86.680 ± 0.034	—	0.168 ± 0.003	—	—
GIN-AK+ Zhao et al. (2022)	—	72.19 ± 0.13	86.850 ± 0.057	—	0.080 ± 0.001	—	—
SAN Kreuzer et al. (2021)	—	—	86.581 ± 0.037	76.691 ± 0.650	0.139 ± 0.006	0.3230 ± 0.0039	0.2592 ± 0.0158
EGT Hussain et al. (2022)	**98.17 ± 0.09**	68.70 ± 0.41	86.821 ± 0.020	79.232 ± 0.348	0.108 ± 0.009	—	—
GRIT Ma et al. (2023)	98.11 ± 0.11	**76.47 ± 0.88**	**87.196 ± 0.076**	**80.026 ± 0.277**	**0.059 ± 0.002**	—	—
GPS Rampášek et al. (2023)	98.08 ± 0.11	72.36 ± 0.32	86.648 ± 0.065	77.780 ± 0.236	0.072 ± 0.004	**0.3793 ± 0.0171**	**0.3453 ± 0.0056**
GPS-LGD Zhou et al. (2024)	95.46 ± 0.61	70.16 ± 0.36	85.662 ± 0.181	76.773 ± 0.301	0.085 ± 0.001	0.3912 ± 0.0207	—
GPS-PCL Li et al. (2025b)	98.11 ± 0.18	71.83 ± 0.22	86.695 ± 0.073	78.171 ± 0.231	**0.069 ± 0.001**	0.4196 ± 0.0086	0.3816 ± 0.0058
GPS-GCCM (ours)	**98.24 ± 0.06**	**72.50 ± 0.29**	**86.772 ± 0.042**	**78.820 ± 0.187**	**0.069 ± 0.002**	**0.4219 ± 0.0102**	**0.3856 ± 0.0021**

objective, the optimization may still move toward it to make positive-pair alignment easier, and this tendency cannot be constrained by the negative-pair separation requirement.

To further prevent positive-pair agreement from being trivially satisfied by ignoring $(\mathbf{Y}_t, t)$, we propose feature perturbation on the input node features to break the identical conditioning input across two time steps t_1 and t_2, such that the shortcut solution set $\mathcal{F}_{\text{sc}}$ can no longer trivially yield identical latent representations $\mathbf{Z}^{t_1} = \mathbf{Z}^{t_2}$. Concretely, we treat the diffusion forward process defined in Section 3 as a generic feature perturbation operator and inject noise into the input features accordingly. Unlike the forward process for labels, which samples the noise level from the full range $\{1, \dots, T\}$, feature perturbation only uses a small noise level controlled by a total perturbation time step T_{per} to avoid overly aggressive perturbations to the conditioning features. By applying this lightweight perturbation twice independently, we obtain $\tilde{\mathbf{X}}_1$ and $\tilde{\mathbf{X}}_2$, which yield two perturbed graph views $\tilde{\mathcal{G}}_1 = (\tilde{\mathbf{X}}_1, \mathbf{A})$ and $\tilde{\mathcal{G}}_2 = (\tilde{\mathbf{X}}_2, \mathbf{A})$. The perturbations are intentionally kept small and mainly serve to introduce non-identical conditioning features across the two views. In this way, the shortcut solution $\mathcal{F}_{\text{sc}}$ is in general no longer sufficient to guarantee the positive-pair agreement $\mathbf{Z}^{t_1} = \mathbf{Z}^{t_2}$.

Lemma 3. *Let t_1 and t_2 be two noise levels sampled in consistency training. Given two inputs $(\mathbf{Y}_{t_1}, t_1, \tilde{\mathcal{G}}_1)$ and $(\mathbf{Y}_{t_2}, t_2, \tilde{\mathcal{G}}_2)$, where $\tilde{\mathcal{G}}_1 = (\tilde{\mathbf{X}}_1, \mathbf{A})$ and $\tilde{\mathcal{G}}_2 = (\tilde{\mathbf{X}}_2, \mathbf{A})$ are independently perturbed inputs with $\tilde{\mathbf{X}}_1 \neq \tilde{\mathbf{X}}_2$, consider the shortcut solution set $\mathcal{F}_{\text{sc}}$ defined in Eq. equation 6, i.e., $\mathcal{F}_{\text{sc}} := \{\theta : W_y = \mathbf{0}, W_t = \mathbf{0}\}$. For any $\theta \in \mathcal{F}_{\text{sc}}$, the latent representation $\mathbf{Z}^t$ produced by the contrastive head at time step t can be written as $\mathbf{Z}^t = \text{Proj}_c\left(\text{Enc}\left(W_x \tilde{\mathbf{X}}, \mathbf{A}\right)\right)$. Then, $\mathcal{F}_{\text{sc}}$ is in general no longer sufficient to guarantee the positive-pair agreement $\mathbf{Z}^{t_1} = \mathbf{Z}^{t_2}$.*

A formal proof is in Appendix C.3. By Lemma 3, positive-pair agreement can no longer be easily achieved by reducing the reliance on $(\mathbf{Y}_t, t)$ through down-weighting W_y and W_t in Eq. equation 3, thereby further reducing the tendency to converge toward the shortcut solution during training.

5 Experiments

5.1 Experimental Settings

Datasets. We conduct extensive experiments covering different prediction tasks on seven benchmark datasets Dwivedi et al. (2022; 2023). These datasets can be divided into three groups based on the prediction tasks: (i) graph-level classification on image-based graph datasets (i.e., MNIST and CIFAR10); (ii) graph-level regression on a molecular graph dataset (i.e., ZINC); and (iii) node-level classification on long-range graph datasets (i.e., PascalVOC-SP and COCO-SP) and synthetic SBM datasets (i.e., PATTERN and CLUSTER). All experiments follow standard dataset splits and evaluation metrics adopted in previous works. More dataset details can be found in Appendix D.1.

Baselines. To examine whether our proposed GCCM can enhance deterministic prediction methods in a generative manner, we compare against existing conditional generative graph prediction frameworks, LGD Zhou et al. (2024) and PCL Li et al. (2025b). Since LGD originally used GPS Rampášek et al. (2023) as the network backbone, we also adopt GPS as the network backbone of both PCL and our GCCM for a fair comparison, which we denote as GPS+PCL and GPS+GCCM. **Our main comparison therefore focuses on LGD, PCL, and GCCM under the same GPS backbone.** For reference, we also include the results of deterministic prediction models.

Table 2: Ablation studies on each component of GCCM.

Model	MNIST	CIFAR10	PATTERN	CLUSTER
	Accuracy ↑	Accuracy ↑	Accuracy ↑	Accuracy ↑
GPS	98.08 ± 0.11	72.36 ± 0.32	86.648 ± 0.065	77.780 ± 0.236
GPS+Diffusion	98.08 ± 0.24	71.93 ± 0.35	86.741 ± 0.158	78.312 ± 0.251
GPS+Consistency	98.11 ± 0.18	71.83 ± 0.22	86.695 ± 0.073	78.171 ± 0.231
GPS+Consistency+Contrastive	98.12 ± 0.06	71.98 ± 0.32	86.676 ± 0.071	78.232 ± 0.159
GPS+GCCM (ours)	$\mathbf{98.24 \pm 0.06}$	$\mathbf{72.50 \pm 0.29}$	$\mathbf{86.772 \pm 0.042}$	$\mathbf{78.820 \pm 0.187}$

5.2 Main Results

We evaluate GCCM on the seven datasets introduced in Section 5.1. For all datasets, we report the mean and standard deviation over five runs with different random seeds. The results are summarized in Table 1. Integrating GCCM into the deterministic base model GPS substantially boosts its performance across most benchmarks. And compared with previous conditional generative approaches for graph prediction (i.e., LGD and PCL), GCCM consistently achieves superior performance on most datasets. In particular, relative to the previous vanilla consistency model for graph prediction (i.e., PCL), by effectively mitigating the shortcut solution in consistency training, GCCM leads to substantial performance improvements. For example, GPS-GCCM improves accuracy over GPS-PCL by 0.67 on CIFAR10 and 0.65 on CLUSTER. On MNIST, where the strongest deterministic baselines EGT, GRIT, and GPS differ by less than 0.1 in accuracy, GPS-GCCM still achieves the best accuracy among the compared methods. Moreover, GCCM is not tied to a specific backbone. As shown in Appendix E.4, GCCM consistently outperforms PCL across different backbones, demonstrating the generalizability of the proposed framework. Overall, these results demonstrate that GCCM can serve as an effective generative enhancement to deterministic backbones.

5.3 Ablation Studies

Beyond demonstrating the overall performance of GCCM, we conduct detailed ablation studies in this section to further verify the effectiveness of each proposed component of our GCCM.

Formulating graph prediction as conditional generation. To evaluate the benefits of formulating graph prediction as conditional generation, we introduce a variant that enhances the deterministic model GPS using a generative diffusion formulation, denoted GPS+Diffusion, which follows the design described in Section 4.1. To obtain stable predictions at inference time, GPS+Diffusion aggregates predictions from 10 independent noise initializations, each generated via 1000 reverse diffusion steps. As shown in Table 2, GPS+Diffusion achieves better performance on MNIST, PATTERN, and CLUSTER compared to the deterministic GPS baseline. These improvements highlight the potential of applying conditional generative models to graph prediction tasks.

Incorporating consistency training. Building on the variant of GPS+Diffusion, we further incorporate vanilla consistency training to enable stable sampling and one-step inference, yielding a consistency-based variant denoted as GPS+Consistency. As shown in Table 2, GPS+Consistency yields more stable performance compared to GPS+Diffusion in terms of the standard deviation. However, it leads to reduced predictive performance, this unsatisfactory performance is mainly due to the fact that vanilla consistency training leads the model toward the shortcut solution and collapses it toward a deterministic predictor. To better illustrate this, we visualize the relative contribution of $W_y\mathbf{Y}_t + W_t\mathbf{t}_{\text{emb}}(t)$ to $\mathbf{H}_t^{(0)}$ in Eq. equation 3 on MNIST and CLUSTER. As shown in Fig. 2, the heatmaps corresponding to case (a) and (d) on both datasets are dominated by dark purple colors, indicating a consistently low contribution from the terms associated with $(\mathbf{Y}_t, t)$ by down-weighting W_y and W_t in Eq. equation 3. Consistent with this observation, Table 3 shows that retaining $(\mathbf{Y}_t, t)$ at inference has only a limited effect on the prediction accuracy of GPS+Consistency compared with the other two variants, particularly on MNIST and CLUSTER. These observations are consistent with our analysis in Section 4.1: the model might satisfy the self-consistency constraint by reducing its reliance on the noisy labels $\mathbf{Y}_t$ and the corresponding time step t (i.e., the shortcut), and thus collapse toward a deterministic predictor whose performance is close to GPS.

Mitigating the shortcut solution via the contrastive consistency objective. To evaluate the effectiveness of the proposed contrastive consistency objective in mitigating the shortcut solution, we replace the self-consistency objective in GPS+Consistency with our contrastive consistency objective, yielding the variant GPS+Consistency+Contrastive. From Table 2, GPS+Consistency+Contrastive generally improves upon GPS+Consistency, but only to a limited extent. Specifically, as illustrated

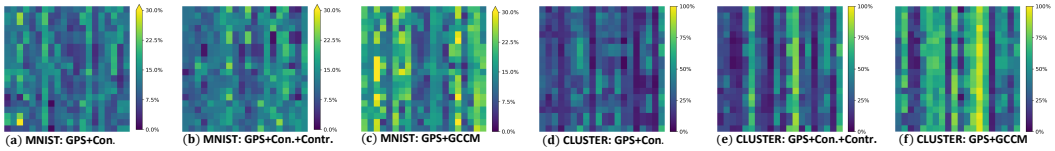


Figure 2: Heatmap visualizations of the contributions of $W_y\mathbf{Y}_t$ and $W_t\mathbf{t}_{\text{emb}}(t)$ to $\mathbf{H}_t^{(0)}$ on MNIST ((a)–(c)) and CLUSTER ((d)–(f)), computed as $\frac{|W_y\mathbf{Y}_t|+|W_t\mathbf{t}_{\text{emb}}(t)|}{|W_y\mathbf{Y}_t|+|W_t\mathbf{t}_{\text{emb}}(t)|+|W_x\mathbf{X}|}$. The x-axis denotes node indices and the y-axis hidden dimensions.

Table 3: Effect of utilizing $(\mathbf{Y}_t, t)$ on prediction accuracy. For each trained model, we evaluate two inference settings: "w/o $(\mathbf{Y}_t, t)$" removes the term $W_y\mathbf{Y}_t + W_t\mathbf{t}_{\text{emb}}(t)$ from $\mathbf{H}_t^{(0)}$ in Eq. equation 3, whereas "w/ $(\mathbf{Y}_t, t)$" retains this term. Gain denotes the accuracy difference between the two settings.

Dataset	GPS+Consistency			GPS+Consistency+Contrastive			GPS+GCCM (ours)		
	w/o $(\mathbf{Y}_t, t)$	w/ $(\mathbf{Y}_t, t)$	Gain	w/o $(\mathbf{Y}_t, t)$	w/ $(\mathbf{Y}_t, t)$	Gain	w/o $(\mathbf{Y}_t, t)$	w/ $(\mathbf{Y}_t, t)$	Gain
MNIST (Accuracy ↑)	97.99	98.11	+0.12	97.75	98.12	+0.37	97.80	98.24	**+0.44**
CIFAR10 (Accuracy ↑)	69.95	71.83	+1.88	68.69	71.98	+3.29	67.99	72.50	**+4.51**
PATTERN (Accuracy ↑)	82.743	86.695	+3.952	81.949	86.676	+4.727	79.098	86.772	**+7.674**
CLUSTER (Accuracy ↑)	78.055	78.171	+0.116	78.051	78.232	+0.181	78.314	78.820	**+0.506**

in Fig. 2, the corresponding heatmap (e) on CLUSTER shows only a mild mitigation effect, while the heatmap (b) on MNIST remains largely unchanged. Consistent with this observation, Table 3 shows that GPS+Consistency+Contrastive consistently increases the gains from retaining $(\mathbf{Y}_t, t)$ at inference over GPS+Consistency across all evaluated datasets, although the improvements remain limited. Overall, these results indicate that although the contrastive consistency objective has the potential to alleviate the shortcut tendency toward a deterministic predictor, itself alone may not be sufficient to fully prevent the denoising network from down-weighting W_y and W_t in Eq. equation 3 to reduce its reliance on $(\mathbf{Y}_t, t)$ to satisfy positive-pair agreement discussed in Section 4.3, thereby necessitating additional training strategies.

Enhancing the contrastive consistency objective with feature perturbation. To further mitigate the shortcut solution in GPS+Consistency+Contrastive, we incorporate feature perturbation to break the identical feature inputs across different time steps, yielding the full GPS+GCCM model. As shown in Table 2, compared with GPS+Consistency+Contrastive, GPS+GCCM improves performance across all evaluated datasets. Moreover, as illustrated in Fig. 2, the corresponding heatmaps (c) and (f) on MNIST and CLUSTER become noticeably brighter and more evenly distributed, indicating that the denoising network balances its reliance on $(\mathbf{Y}_t, t)$ and $\mathbf{X}$, rather than satisfying the consistency constraint by down-weighting W_y and W_t. This is further supported by Table 3, where GPS+GCCM achieves the largest gains from retaining $(\mathbf{Y}_t, t)$ at inference across all evaluated datasets. These larger gains suggest stronger utilization of $(\mathbf{Y}_t, t)$ through feature perturbation compared with GPS+Consistency and GPS+Consistency+Contrastive. Taken together, these results show that GCCM mitigates the shortcut solution in vanilla consistency training by enabling the denoising network to better utilize $(\mathbf{Y}_t, t)$, thereby better leveraging the generative formulation.

More experimental results, including additional ablation studies, further empirical evidence that GCCM mitigates the shortcut solution, and parameter analyses, are provided in Appendix E.

6 Conclusion

In this paper, we show that consistency training for diffusion-based graph prediction can admit a shortcut solution: under identical conditioning, the denoising network may ignore noisy labels and collapse into a deterministic predictor. To address this, we propose a Graph Contrastive Consistency Model (GCCM) with two training strategies: (1) a contrastive consistency objective with mini-batch negatives to enforce separation between different instances, and (2) feature perturbation to break identical conditioning on input node/edge features across different noise levels. Extensive experiments show that GCCM can effectively mitigate the shortcut solution and consistently outperforms existing diffusion-based graph prediction baselines on most evaluated benchmarks across a broad range of datasets and prediction tasks.

AI USE STATEMENT

We used generative AI tools to assist with language polishing and improve the clarity and readability of the manuscript. All AI-assisted text was reviewed and revised by the authors. We take full responsibility for the final content of this work.

REFERENCES

Dominique Beaini, Saro Passaro, Vincent Létourneau, William L. Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks, 2021. URL https://arxiv.org/abs/2010.02863.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yu Guang Wang, Pietro Liò, Guido Montúfar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks, 2022. URL https://arxiv.org/abs/2106.12575.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets, 2018. URL https://arxiv.org/abs/1711.07553.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets, 2020. URL https://arxiv.org/abs/2004.05718.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks, 2022. URL https://arxiv.org/abs/2003.00982.

Vijay Prakash Dwivedi, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark, 2023. URL https://arxiv.org/abs/2206.08164.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions, 2021. URL https://arxiv.org/abs/2102.05379.

Wei Huang, Hanchen Wang, Dong Wen, Shaozhen Ma, Wenjie Zhang, and Xuemin Lin. Towards unsupervised training of matching-based graph edit distance solver via preference-aware gan, 2025a. URL https://arxiv.org/abs/2506.01977.

Wei Huang, Hanchen Wang, Dong Wen, Wenjie Zhang, Ying Zhang, and Xuemin Lin. Diffged: Computing graph edit distance via diffusion-based graph matching, 2025b. URL https://arxiv.org/abs/2503.18245.

Md Shamim Hussain, Mohammed J. Zaki, and Dharmashankar Subramanian. Global self-attention as a replacement for graph convolution. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pp. 655–665. ACM, August 2022. doi: 10.1145/3534678.3539296. URL http://dx.doi.org/10.1145/3534678.3539296.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017. URL https://arxiv.org/abs/1609.02907.

Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention, 2021. URL https://arxiv.org/abs/2106.03893.

Yang Li, Jinpei Guo, Runzhong Wang, Hongyuan Zha, and Junchi Yan. Fast t2t: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization, 2025a. URL https://arxiv.org/abs/2502.02941.

Yang Li, Jiale Ma, Yebin Yang, Qitian Wu, Hongyuan Zha, and Junchi Yan. Generative modeling reinvents supervised learning: Label repurposing with predictive consistency learning. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=FO2fu3daSL.

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K. Dokania, Mark Coates, Philip H.S. Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021. URL https://arxiv.org/abs/2102.09672.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer, 2023. URL https://arxiv.org/abs/2205.12454.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015. URL https://arxiv.org/abs/1503.03585.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=WNzy9bRDvG.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models, 2023. URL https://arxiv.org/abs/2303.01469.

Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization, 2023. URL https://arxiv.org/abs/2302.08224.

Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Walking out of the weisfeiler leman hierarchy: Graph learning beyond message passing, 2023. URL https://arxiv.org/abs/2102.08786.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018. URL https://arxiv.org/abs/1710.10903.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation, 2023. URL https://arxiv.org/abs/2209.14734.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019. URL https://arxiv.org/abs/1810.00826.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation?, 2021. URL https://arxiv.org/abs/2106.05234.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations, 2021. URL https://arxiv.org/abs/2010.13902.

Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any gnn with local structure awareness, 2022. URL https://arxiv.org/abs/2110.03753.

Cai Zhou, Xiyuan Wang, and Muhan Zhang. Unifying generation and prediction on graphs with latent graph diffusion, 2024. URL https://arxiv.org/abs/2402.02518.

A Extended Related Works

Deterministic models for graph prediction. Deterministic models for graph prediction typically learn a direct mapping from input graph data to target values in a supervised manner via Graph Neural Networks, where model parameters are optimized by minimizing a task-specific loss between predictions and ground-truth. Graph neural networks perform prediction by iteratively aggregating information from local neighborhoods through message-passing mechanisms, and have given rise to a wide range of popular architectures, including GCN Kipf & Welling (2017), GIN Xu et al. (2019), GatedGCN Bresson & Laurent (2018), and PNA Corso et al. (2020). Later studies further improve the expressive power of GNNs by moving beyond standard message passing, leading to models such as CIN Bodnar et al. (2022), CRaWI Tönshoff et al. (2023), and GIN-AK+ Zhao et al. (2022).

More recently, motivated by the strong global modeling capability of Transformers, an increasing number of works have extended Transformer-based architectures to graph prediction. Spectral Attention Networks (SAN) Kreuzer et al. (2021) employ a dual-attention design that combines global attention with edge-restricted attention, while incorporating Laplacian positional encodings to inject spectral structural information into node representations. Graphormer Ying et al. (2021) adopts global attention and augments it with structural biases such as centrality and spatial encodings. More recently, GraphGPS Rampášek et al. (2023) provides a general framework that integrates positional and structural encodings alongside local message passing and global attention, resulting in competitive performance across a wide range of prediction tasks on graphs.

Generative diffusion models for graph prediction. In recent years, diffusion models Ho et al. (2020); Song et al. (2022); Nichol & Dhariwal (2021); Rombach et al. (2022) have emerged as a powerful framework for generative modeling. They define a forward process that gradually perturbs clean data with noise, and train neural networks to reverse this process by progressively denoising. At inference time, new samples are generated by iteratively transforming noise drawn from a predefined prior into data samples. Motivated by their strong generative capacity, a growing body of work has extended diffusion models to graph-related generation tasks, adapting the diffusion process to structured graph data. Representative applications include graph edit distance (GED) computation Huang et al. (2025b;a), combinatorial optimization Sun & Yang (2023), and graph generation Vignac et al. (2023); Zhou et al. (2024). Beyond graph generation tasks, LGD Zhou et al. (2024) explores diffusion models for graph prediction by applying diffusion in a latent space to generate latent representations and then mapping them to the target space using a learned decoder. Although effective, it still incurs substantial computational overhead due to iterative denoising at inference time. Building upon diffusion models, consistency models Song et al. (2023); Song & Dhariwal (2024) eliminate iterative reverse diffusion by directly learning a mapping from noisy samples to clean data through the self-consistency objective, enabling efficient one-step or few-step generation. Inspired by the consistency mapping principle, PCL Li et al. (2025b) further develops a predictive consistency framework for prediction tasks by enforcing consistent predictions across different noise levels. This formulation enables stable sampling and one-step inference in diffusion-based graph prediction models.

B Further Details of the Proposed GCCM

B.1 Training Strategy

The detailed training process of GCCM is presented in Algorithm 1.

B.2 One-step Prediction

The one-step prediction procedure of GCCM is summarized in Algorithm 2.

C Proofs

C.1 Proof of Lemma 1

Recall Lemma 1:

Algorithm 1 Training Process of GCCM

Input: Dataset $\mathcal{D}$; forward process $q(\cdot)$; feature perturbation $\mathrm{Perturb}(\cdot)$; losses $d^b(\cdot,\cdot)$ and $d^c(\cdot,\cdot)$; weights λ_1, λ_2
Initialize model parameters θ and optimizer
while not converged **do**
  Sample $(\mathcal{G} = (\mathbf{X}, \mathbf{A}), \mathbf{Y}) \sim \mathcal{D}$
  Sample $t_1 \sim \mathcal{U}\{1, \ldots, T\}$ and set $t_2 \leftarrow \max(1, \lfloor 0.2 t_1 \rfloor)$
  Sample $\mathbf{Y}_{t_1} \sim q(\mathbf{Y}_{t_1} \mid \mathbf{Y})$ and $\mathbf{Y}_{t_2} \sim q(\mathbf{Y}_{t_2} \mid \mathbf{Y})$
  Sample $\tilde{\mathbf{X}}_1 \sim \mathrm{Perturb}(\mathbf{X})$ and $\tilde{\mathbf{X}}_2 \sim \mathrm{Perturb}(\mathbf{X})$
  Set $\tilde{\mathcal{G}}_1 \leftarrow (\tilde{\mathbf{X}}_1, \mathbf{A})$ and $\tilde{\mathcal{G}}_2 \leftarrow (\tilde{\mathbf{X}}_2, \mathbf{A})$
  $\hat{\mathbf{Y}}_0^{t_1}, \mathbf{Z}^{t_1} \leftarrow f_\theta(\mathbf{Y}_{t_1}, t_1, \tilde{\mathcal{G}}_1)$
  $\hat{\mathbf{Y}}_0^{t_2}, \mathbf{Z}^{t_2} \leftarrow f_\theta(\mathbf{Y}_{t_2}, t_2, \tilde{\mathcal{G}}_2)$
  $\mathcal{L} \leftarrow \lambda_1 \Big(d^b(\hat{\mathbf{Y}}_0^{t_1}, \mathbf{Y}) + d^b(\hat{\mathbf{Y}}_0^{t_2}, \mathbf{Y}) \Big) + \lambda_2 \, d^c(\mathbf{Z}^{t_1}, \mathbf{Z}^{t_2})$
  optimizer.zero_grad()
  $\mathcal{L}$.backward()
  optimizer.step()
end while

Algorithm 2 One-step Prediction in GCCM

Input: Trained denoising network $f_\theta(\cdot)$; input graph $\mathcal{G}$
Output: Predicted labels $\hat{\mathbf{Y}}_0$
Sample initial noise $\mathbf{Y}_T$
$\hat{\mathbf{Y}}_0 \leftarrow f_\theta(\mathbf{Y}_T, T, \mathcal{G})$

Consider the training objective

$$\mathcal{L}(\theta) = \mathbb{E}\left[\lambda_1 \left(d^b(\hat{\mathbf{Y}}_0^{t_1}, \mathbf{Y}) + d^b(\hat{\mathbf{Y}}_0^{t_2}, \mathbf{Y}) \right) + \lambda_2 \, d^s(\hat{\mathbf{Y}}_0^{t_1}, \hat{\mathbf{Y}}_0^{t_2}) \right], \tag{11}$$

where $d^b(\cdot,\cdot)$ denotes a task-specific supervised loss and $d^s(\hat{\mathbf{Y}}_0^{t_1}, \hat{\mathbf{Y}}_0^{t_2}) := \|\hat{\mathbf{Y}}_0^{t_1} - \hat{\mathbf{Y}}_0^{t_2}\|_2^2$, and each predicted label $\hat{\mathbf{Y}}_0^t$ can be rewritten as

$$\begin{aligned} \hat{\mathbf{Y}}_0^t &= f_\theta\big(\mathbf{Y}_t, t, \mathcal{G}\big) = f_\theta\big(\mathbf{H}_t^{(0)}, \mathbf{A}\big) \\ &= f_\theta\Big(W_y \, \mathbf{Y}_t + W_t \, \mathbf{t}_{\mathrm{emb}}(t) + W_x \, \mathbf{X}, \mathbf{A}\Big). \end{aligned} \tag{12}$$

Minimizing the above objective could lead the denoising network f_θ to a shortcut solution set

$$\mathcal{F}_{\mathrm{sc}} := \{\theta : \; W_y = \mathbf{0}, \, W_t = \mathbf{0}\}, \tag{13}$$

where d_k denotes the dimension of the input label, such that the noisy labels $\mathbf{Y}_t$ and time step t are ignored.

Proof. The training objective is

$$\mathcal{L}(\theta) = \mathbb{E}\left[\lambda_1 \left(d^b(\hat{\mathbf{Y}}_0^{t_1}, \mathbf{Y}) + d^b(\hat{\mathbf{Y}}_0^{t_2}, \mathbf{Y}) \right) + \lambda_2 \, d^s(\hat{\mathbf{Y}}_0^{t_1}, \hat{\mathbf{Y}}_0^{t_2}) \right], \tag{14}$$

where $d^b(\cdot,\cdot)$ denotes a task-specific supervised loss and $d^s(\hat{\mathbf{Y}}_0^{t_1}, \hat{\mathbf{Y}}_0^{t_2}) := \|\hat{\mathbf{Y}}_0^{t_1} - \hat{\mathbf{Y}}_0^{t_2}\|_2^2$. Under the additive fusion in Eq. equation 3, the predictions can be expressed as

$$\hat{\mathbf{Y}}_0^t \;=\; f_\theta\big(W_y \, \mathbf{Y}_t + W_t \, \mathbf{t}_{\mathrm{emb}}(t) + W_x \, \mathbf{X}, \mathbf{A}\big). \tag{15}$$

Accordingly, the training objective can be written as

$$\begin{aligned} \mathcal{L}(\theta) = \mathbb{E}\Big[& \lambda_1 \Big(d^b\big(f_\theta(W_y \, \mathbf{Y}_{t_1} + W_t \, \mathbf{t}_{\mathrm{emb}}(t_1) + W_x \, \mathbf{X}, \mathbf{A}), \; \mathbf{Y}\big) \\ & + d^b\big(f_\theta(W_y \, \mathbf{Y}_{t_2} + W_t \, \mathbf{t}_{\mathrm{emb}}(t_2) + W_x \, \mathbf{X}, \mathbf{A}), \; \mathbf{Y}\big) \Big) \\ & + \lambda_2 \Big\| f_\theta\big(W_y \, \mathbf{Y}_{t_1} + W_t \, \mathbf{t}_{\mathrm{emb}}(t_1) + W_x \, \mathbf{X}, \mathbf{A}\big) \\ & - f_\theta\big(W_y \, \mathbf{Y}_{t_2} + W_t \, \mathbf{t}_{\mathrm{emb}}(t_2) + W_x \, \mathbf{X}, \mathbf{A}\big) \Big\|_2^2 \Big]. \end{aligned} \tag{16}$$

Minimizing the training objective requires both terms to be minimized jointly.

The boundary condition objective is minimized when

$$\begin{aligned} f_\theta\big(W_y\,\mathbf{Y}_{t_1} + W_t\,\mathbf{t}_{\mathrm{emb}}(t_1) + W_x\,\mathbf{X}, \mathbf{A}\big) &= \mathbf{Y}, \\ f_\theta\big(W_y\,\mathbf{Y}_{t_2} + W_t\,\mathbf{t}_{\mathrm{emb}}(t_2) + W_x\,\mathbf{X}, \mathbf{A}\big) &= \mathbf{Y}. \end{aligned} \tag{17}$$

The self-consistency objective is minimized when

$$f_\theta\big(W_y\,\mathbf{Y}_{t_1} + W_t\,\mathbf{t}_{\mathrm{emb}}(t_1) + W_x\,\mathbf{X}, \mathbf{A}\big) = f_\theta\big(W_y\,\mathbf{Y}_{t_2} + W_t\,\mathbf{t}_{\mathrm{emb}}(t_2) + W_x\,\mathbf{X}, \mathbf{A}\big). \tag{18}$$

Combining Eq. equation 17 and Eq. equation 18, the joint minimization requires

$$f_\theta\big(W_y\,\mathbf{Y}_{t_1} + W_t\,\mathbf{t}_{\mathrm{emb}}(t_1) + W_x\,\mathbf{X}, \mathbf{A}\big) = f_\theta\big(W_y\,\mathbf{Y}_{t_2} + W_t\,\mathbf{t}_{\mathrm{emb}}(t_2) + W_x\,\mathbf{X}, \mathbf{A}\big) = \mathbf{Y}. \tag{19}$$

This requires f_θ to map different inputs to the same output $\mathbf{Y}$ for arbitrary $(\mathbf{Y}_{t_1}, t_1)$ and $(\mathbf{Y}_{t_2}, t_2)$. Since the two predictions share the same $\mathbf{A}$, a sufficient way to satisfy Eq. equation 19 is to make the input to f_θ independent of $(\mathbf{Y}_t, t)$ by enforcing

$$W_y\,\mathbf{Y}_{t_1} + W_t\,\mathbf{t}_{\mathrm{emb}}(t_1) + W_x\,\mathbf{X} = W_y\,\mathbf{Y}_{t_2} + W_t\,\mathbf{t}_{\mathrm{emb}}(t_2) + W_x\,\mathbf{X}. \tag{20}$$

In general, $\mathbf{Y}_{t_1} \neq \mathbf{Y}_{t_2}$ and $\mathbf{t}_{\mathrm{emb}}(t_1) \neq \mathbf{t}_{\mathrm{emb}}(t_2)$. A sufficient solution to satisfy Eq. equation 20 for arbitrary $(\mathbf{Y}_{t_1}, t_1)$ and $(\mathbf{Y}_{t_2}, t_2)$ is

$$W_y = \mathbf{0}, \qquad W_t = \mathbf{0}. \tag{21}$$

We denote the corresponding shortcut solution set $\mathcal{F}_{\mathrm{sc}} := \{\theta :\ W_y = \mathbf{0},\ W_t = \mathbf{0}\}$. Under $\mathcal{F}_{\mathrm{sc}}$, the joint minimization condition in Eq. equation 19 reduces to $f_\theta(W_x\,\mathbf{X}, \mathbf{A}) = \mathbf{Y}$, which is exactly the supervised objective of a standard deterministic predictor. Therefore, the full training objective under $\mathcal{F}_{\mathrm{sc}}$ reduces to

$$\mathcal{L}(\theta) = 2\lambda_1\,\mathbb{E}\left[d^b(f_\theta(W_x\,\mathbf{X}, \mathbf{A}), \mathbf{Y})\right], \tag{22}$$

and the model can minimize this objective in the same way as a deterministic predictor. □

C.2 Proof of Lemma 2

Recall Lemma 2:

Consider the contrastive consistency objective defined as

$$d^c(\mathbf{Z}^{t_1}, \mathbf{Z}^{t_2}) = \frac{1}{2}\Big(\mathcal{L}_{\mathrm{CL}}^{t_1 \to t_2} + \mathcal{L}_{\mathrm{CL}}^{t_2 \to t_1}\Big). \tag{23}$$

where the $t_1 \to t_2$ term of the contrastive consistency objective is defined as

$$\begin{aligned} \mathcal{L}_{\mathrm{CL}}^{t_1 \to t_2} &= -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\left(\mathrm{sim}\left(\mathbf{z}_i^{t_1}, \mathbf{z}_i^{t_2}\right)/\tau\right)}{\sum_{j=1}^{N} \exp\left(\mathrm{sim}\left(\mathbf{z}_i^{t_1}, \mathbf{z}_j^{t_2}\right)/\tau\right)}, \\ &\text{with}\ \ \mathrm{sim}(\mathbf{z}_i^{t_1}, \mathbf{z}_j^{t_2}) = \frac{(\mathbf{z}_i^{t_1})^T \mathbf{z}_j^{t_2}}{\|\mathbf{z}_i^{t_1}\|\,\|\mathbf{z}_j^{t_2}\|}. \end{aligned} \tag{24}$$

Under the shortcut solution set $\mathcal{F}_{\mathrm{sc}}$ defined in Eq. equation 6,

$$\mathcal{F}_{\mathrm{sc}} := \{\theta :\ W_y = \mathbf{0},\ W_t = \mathbf{0}\}.$$

$\mathcal{F}_{\mathrm{sc}}$ is generally no longer sufficient to minimize the contrastive consistency objective.

Proof. Under the shortcut solution set defined in Eq. equation 6,

$$\mathcal{F}_{\mathrm{sc}} := \{\theta :\ W_y = \mathbf{0},\ W_t = \mathbf{0}\}, \tag{25}$$

the additive fusion in Eq. equation 3,

$$\mathbf{H}_t^{(0)} = W_y\,\mathbf{Y}_t + W_t\,\mathbf{t}_{\mathrm{emb}}(t) + W_x\,\mathbf{X}, \tag{26}$$

reduces to $\mathbf{H}_t^{(0)} = W_x\mathbf{X}$. Therefore, the output latent representation matrix $\mathbf{Z}^t$ becomes independent of $(\mathbf{Y}_t, t)$:

$$\begin{aligned}\mathbf{Z}^t &= f_\theta\big(\mathbf{Y}_t, t, \mathcal{G}\big) = f_\theta\big(\mathbf{H}_t^{(0)}, \mathbf{A}\big) \\ &= \mathrm{Proj}_c\left(\mathrm{Enc}\left(\mathbf{H}_t^{(0)}, \mathbf{A}\right)\right) \\ &= \mathrm{Proj}_c\left(\mathrm{Enc}\left(W_x\mathbf{X}, \mathbf{A}\right)\right).\end{aligned} \tag{27}$$

In particular, for two time steps t_1 and t_2 with the same input graph $\mathcal{G} = (\mathbf{X}, \mathbf{A})$, we have

$$\mathbf{Z}^{t_1} = \mathrm{Proj}_c\left(\mathrm{Enc}\left(W_x\mathbf{X}, \mathbf{A}\right)\right) = \mathbf{Z}^{t_2}. \tag{28}$$

Let $\mathbf{z}_i^t$ denote the i-th row of $\mathbf{Z}^t$. Eq. equation 28 implies that, for each instance i,

$$\mathbf{z}_i^{t_1} = \mathbf{z}_i^{t_2}. \tag{29}$$

In the following, we consider the $t_1 \to t_2$ term of the contrastive consistency objective; the $t_2 \to t_1$ term is defined analogously. Recall that

$$\mathcal{L}_{\mathrm{CL}}^{t_1\to t_2} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp\left(\mathrm{sim}\left(\mathbf{z}_i^{t_1}, \mathbf{z}_i^{t_2}\right)/\tau\right)}{\sum_{j=1}^{N}\exp\left(\mathrm{sim}\left(\mathbf{z}_i^{t_1}, \mathbf{z}_j^{t_2}\right)/\tau\right)}. \tag{30}$$

From Eq. equation 29, for each instance i we have

$$\mathrm{sim}\left(\mathbf{z}_i^{t_1}, \mathbf{z}_i^{t_2}\right) = 1. \tag{31}$$

Substituting this into the above expression yields

$$\mathcal{L}_{\mathrm{CL}}^{t_1\to t_2} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(1/\tau)}{\exp(1/\tau) + \sum_{j\neq i}\exp\left(\mathrm{sim}\left(\mathbf{z}_i, \mathbf{z}_j\right)/\tau\right)}. \tag{32}$$

In this form, the contribution of the positive pairs becomes constant. Consequently, minimizing the contrastive consistency objective reduces to separating negative pairs across different instances.

We now analyze the negative-pair terms in the contrastive consistency objective. For each anchor instance i, define the corresponding contrastive consistency loss in Eq. equation 32 as

$$\ell_i = -\log\frac{\exp(1/\tau)}{\exp(1/\tau) + \sum_{j\neq i}\exp\left(\mathrm{sim}\left(\mathbf{z}_i, \mathbf{z}_j\right)/\tau\right)}. \tag{33}$$

Since the numerator is fixed, minimizing ℓ_i requires the denominator to be as small as possible, which in turn encourages $\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j)$ to be small for all $j \neq i$.

However, consider a particular parameter choice within $\mathcal{F}_{\mathrm{sc}}$ by additionally setting $W_x = \mathbf{0}$. Then $\mathbf{H}_{t_1}^{(0)} = \mathbf{H}_{t_2}^{(0)}$, and the resulting latent representations collapse to the same vector for all instances, i.e.,

$$\mathbf{z}_1 = \mathbf{z}_2 = \cdots = \mathbf{z}_N. \tag{34}$$

Consequently, $\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j) = 1$ for all $i \neq j$, and Eq. equation 33 yields

$$\ell_i = -\log\frac{\exp(1/\tau)}{N\exp(1/\tau)} = \log N > 0. \tag{35}$$

Therefore, using only $W_y = \mathbf{0}$ and $W_t = \mathbf{0}$ cannot in general minimize the contrastive consistency objective, which completes the proof. □

C.3 Proof of Lemma 3

Recall Lemma 3:

Let t_1 and t_2 be two noise levels sampled in consistency training. Given two inputs $(\mathbf{Y}_{t_1}, t_1, \tilde{\mathcal{G}}_1)$ and $(\mathbf{Y}_{t_2}, t_2, \tilde{\mathcal{G}}_2)$, where $\tilde{\mathcal{G}}_1 = (\tilde{\mathbf{X}}_1, \mathbf{A})$ and $\tilde{\mathcal{G}}_2 = (\tilde{\mathbf{X}}_2, \mathbf{A})$ are independently perturbed inputs with $\tilde{\mathbf{X}}_1 \neq \tilde{\mathbf{X}}_2$, consider the shortcut solution set $\mathcal{F}_{\mathrm{sc}}$ defined in Eq. equation 6,

$$\mathcal{F}_{\mathrm{sc}} := \{\theta :\ W_y = \mathbf{0},\ W_t = \mathbf{0}\}.$$

For any $\theta \in \mathcal{F}_{\mathrm{sc}}$, the latent representation $\mathbf{Z}^t$ produced by the contrastive head at time step t can be written as

$$\mathbf{Z}^t = \mathrm{Proj}_c\left(\mathrm{Enc}\left(W_x\tilde{\mathbf{X}}, \mathbf{A}\right)\right). \tag{36}$$

Then, $\mathcal{F}_{\mathrm{sc}}$ is in general no longer sufficient to guarantee the positive-pair agreement $\mathbf{Z}^{t_1} = \mathbf{Z}^{t_2}$.

Proof. For any $\theta \in \mathcal{F}_{\mathrm{sc}}$, the denoising network ignores $(\mathbf{Y}_t, t)$, and thus the latent representation produced by the contrastive head at time step t can be written as

$$\mathbf{Z}^t = \mathrm{Proj}_c\left(\mathrm{Enc}\left(W_x\mathbf{X}, \mathbf{A}\right)\right). \tag{37}$$

Under the two perturbed conditioning graphs $\tilde{\mathcal{G}}_1 = (\tilde{\mathbf{X}}_1, \mathbf{A})$ and $\tilde{\mathcal{G}}_2 = (\tilde{\mathbf{X}}_2, \mathbf{A})$, the corresponding latent representations become

$$\mathbf{Z}^{t_1} = \mathrm{Proj}_c\left(\mathrm{Enc}\left(W_x\tilde{\mathbf{X}}_1, \mathbf{A}\right)\right), \qquad \mathbf{Z}^{t_2} = \mathrm{Proj}_c\left(\mathrm{Enc}\left(W_x\tilde{\mathbf{X}}_2, \mathbf{A}\right)\right). \tag{38}$$

Since $\tilde{\mathbf{X}}_1 \neq \tilde{\mathbf{X}}_2$, the conditioning inputs to the network are no longer identical under $\theta \in \mathcal{F}_{\mathrm{sc}}$. As a result, the shortcut solution set can no longer guarantee $\mathbf{Z}^{t_1} = \mathbf{Z}^{t_2}$. □

D Experimental Settings

D.1 Datasets Descriptions

Table 4: Dataset statistics.

Dataset	# Graphs	Avg. # nodes	Avg. # edges	Prediction level	Prediction task	Metric
MNIST	70,000	70.6	564.5	graph	10-class classif.	Accuracy
CIFAR10	60,000	117.6	941.1	graph	10-class classif.	Accuracy
PATTERN	14,000	118.9	3,039.3	inductive node	binary classif.	Accuracy
CLUSTER	12,000	117.2	2,150.9	inductive node	6-class classif.	Accuracy
ZINC	12,000	23.2	24.9	graph	regression	Mean Abs. Error
PascalVOC-SP	11,355	479.4	2,710.5	inductive node	21-class classif.	F1 score
COCO-SP	123,286	476.9	2,693.7	inductive node	81-class classif.	F1 score

ZINC Dwivedi et al. (2022). ZINC contains 12K molecular graphs from the ZINC database. Nodes represent heavy atoms (28 atom types) and edges represent chemical bonds (3 bond types). The task is graph-level regression of constrained solubility (logP), evaluated by MAE. The predefined 10K/1K/1K train/validation/test split is used.

MNIST-SP and CIFAR10-SP Dwivedi et al. (2022). MNIST-SP and CIFAR10-SP are graph equivalents of the MNIST and CIFAR10 image datasets, constructed from SLIC superpixels. Each image is converted into a graph by connecting superpixel nodes via an 8-nearest-neighbor graph. Both are 10-class graph-level classification tasks evaluated by accuracy, using the standard splits (MNIST: 55K/5K/10K; CIFAR10: 45K/5K/10K for train/validation/test graphs).

PATTERN and CLUSTER Dwivedi et al. (2022). PATTERN and CLUSTER are synthetic datasets sampled from a Stochastic Block Model (SBM) and are designed for inductive node classification. PATTERN requires identifying nodes that belong to one of 100 predefined subgraph patterns. CLUSTER requires classifying nodes into 6 SBM-generated clusters (cluster IDs). Accuracy is reported following the standard benchmark splits.

PascalVOC-SP and COCO-SP Dwivedi et al. (2023). PascalVOC-SP and COCO-SP are superpixel-graph datasets derived from Pascal VOC and MS COCO images via SLIC superpixelization. Both are inductive node classification tasks that assign each superpixel node to an object category (semantic segmentation at the superpixel level), evaluated by accuracy using the official benchmark splits.

D.2 Implementation Details

In this section, we summarize the GCCM-specific hyperparameters used across all datasets in Table 5. We adopt GPS Rampášek et al. (2023) as the denoising backbone; therefore, the denoising network architecture in GCCM follows the standard GPS configuration and is not repeated here.

Table 5: GCCM hyperparameters for seven datasets from Dwivedi et al. (2022) and Dwivedi et al. (2023).

Hyperparameter	MNIST	CIFAR10	PATTERN	CLUSTER	ZINC	PascalVOC-SP	COCO-SP
# GPS Layer	3	3	6	16	10	4	4
# Epochs	100	100	100	100	2000	300	300
Hidden dim d_h	52	52	64	48	64	96	96
Batch size	16	16	32	16	32	32	32
Total diffusion steps T	1000	1000	1000	1000	1000	1000	1000
Time-decay ratio α	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Boundary weight λ_1	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Consistency weight λ_2	0.1	0.1	0.1	0.1	0.3	0.1	0.1
Latent dim d_z	52	52	64	48	64	96	96
Temperature parameter τ	0.5	0.5	0.4	0.5	0.3	0.3	0.3
Perturbation type	Continuous	Continuous	Discrete	Discrete	Discrete	Continuous	Continuous
Total perturbation time step T_{per}	10	10	50	50	10	10	10

Table 6: Extended ablation studies on all seven benchmarks.

Model	MNIST	CIFAR10	PATTERN	CLUSTER	ZINC	PascalVOC-SP	COCO-SP
	Accuracy ↑	Accuracy ↑	Accuracy ↑	Accuracy ↑	MAE ↓	F1 ↑	F1 ↑
GPS	98.08 ± 0.11	72.36 ± 0.32	86.648 ± 0.065	77.780 ± 0.236	0.072 ± 0.004	0.3793 ± 0.0171	0.3453 ± 0.0056
GPS+Diffusion	98.08 ± 0.24	71.93 ± 0.35	86.741 ± 0.158	78.312 ± 0.251	0.073 ± 0.007	0.4012 ± 0.0220	0.3654 ± 0.0177
GPS+Consistency	98.11 ± 0.18	71.83 ± 0.22	86.695 ± 0.073	78.171 ± 0.231	0.069 ± 0.001	0.4196 ± 0.0086	0.3816 ± 0.0058
GPS+Consistency+Contrastive	98.12 ± 0.06	71.98 ± 0.32	86.676 ± 0.071	78.232 ± 0.159	**0.068 ± 0.003**	0.4207 ± 0.0084	0.3841 ± 0.0019
GPS+Consistency+Perturbation	98.15 ± 0.07	**73.01 ± 0.47**	86.711 ± 0.089	77.836 ± 0.486	0.075 ± 0.002	0.4189 ± 0.0108	0.3825 ± 0.0064
GPS+GCCM (ours)	**98.24 ± 0.06**	72.50 ± 0.29	**86.772 ± 0.042**	**78.820 ± 0.187**	0.069 ± 0.002	**0.4219 ± 0.0102**	**0.3856 ± 0.0021**

For all datasets, we set the total number of diffusion steps to $T = 1000$. During the forward process on node/graph labels, we sample t_1 uniformly at random for each graph in a mini-batch from $\{1, \ldots, T\}$, and set $t_2 = \max(1, \lfloor \alpha t_1 \rfloor)$ with a fixed time-decay ratio $\alpha = 0.2$. For consistency training, we set $\lambda_1 = 1.0$ and $\lambda_2 = 0.1$ in Eq. equation 9 to balance the contributions of different objective terms. For a fair comparison, the same λ_1 and λ_2 are used for all consistency-based ablation variants in Section 5.3. We compute the contrastive consistency objective in a latent space by applying an additional projection head $\mathrm{Proj}_c(\cdot)$, and set the latent dimension to match the hidden dimension, i.e., $d_z = d_h$. Feature perturbation is selected according to the input feature type of each dataset. For both discrete and continuous features, we construct perturbations using the corresponding forward noising process described in Section 3.1. For continuous features, we set $T_{\mathrm{per}} = 10$, while for the discrete features in PATTERN and CLUSTER, we set $T_{\mathrm{per}} = 50$. For ZINC, although its features are discrete, we use $T_{\mathrm{per}} = 10$ because its node and edge features represent atom and bond types with strict chemical semantics, where perturbing these attributes may alter the molecular structure and properties You et al. (2021). Therefore, we adopt a milder perturbation for ZINC to preserve chemically meaningful information.

D.3 Computational Resources

All experiments are conducted on a single NVIDIA RTX A5000 GPU (24GB) and an Intel Xeon Silver 4314 CPU @ 2.40GHz.

D.4 Source Code

The source code is available at `https://anonymous.4open.science/r/GCCM`.

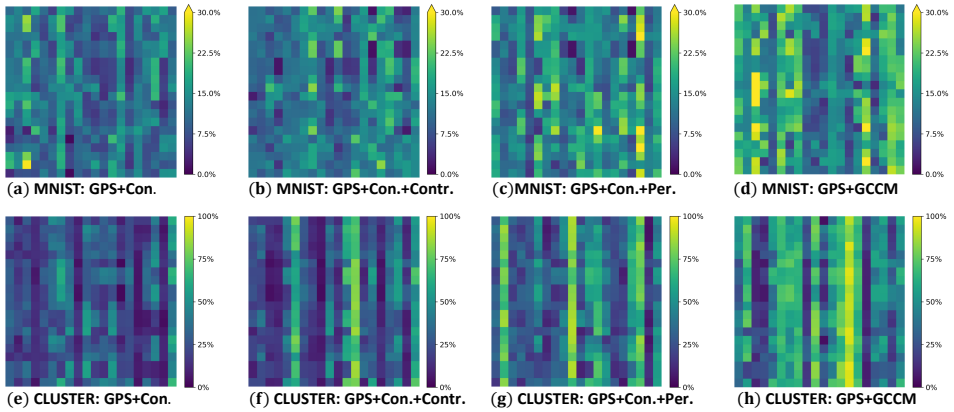
E Additional Experimental Results

E.1 Isolating the Effect of Feature Perturbation

As discussed in Section 4.3, although the shortcut solution is no longer trivially sufficient to minimize the contrastive consistency objective, the optimization may still move toward it to make positive-pair alignment easier when the conditioning inputs remain identical. Feature perturbation is therefore introduced to break the identical conditioning across the two time steps and further reduce this tendency. To isolate its effect, we incorporate feature perturbation into GPS+Consistency without the contrastive consistency objective, yielding the variant GPS+Consistency+Perturbation.

Table 7: Extended analysis of the utilization of $(\mathbf{Y}_t, t)$ across different ablation variants. For each trained model, "w/o $(\mathbf{Y}_t, t)$" removes the term $W_y\mathbf{Y}_t + W_t\mathbf{t}_{\text{emb}}(t)$ from $\mathbf{H}_t^{(0)}$ in Eq. equation 3, whereas "w/ $(\mathbf{Y}_t, t)$" retains this term. Gain denotes the accuracy difference between the two settings.

Dataset	GPS+Consistency			GPS+Consistency+Contrastive			GPS+Consistency+Perturbation			GPS+GCCM (ours)		
	w/o $(\mathbf{Y}_t, t)$	w/ $(\mathbf{Y}_t, t)$	Gain	w/o $(\mathbf{Y}_t, t)$	w/ $(\mathbf{Y}_t, t)$	Gain	w/o $(\mathbf{Y}_t, t)$	w/ $(\mathbf{Y}_t, t)$	Gain	w/o $(\mathbf{Y}_t, t)$	w/ $(\mathbf{Y}_t, t)$	Gain
MNIST (Accuracy ↑)	97.99	98.11	+0.12	97.75	98.12	+0.37	97.90	98.15	+0.25	97.80	98.24	**+0.44**
CIFAR10 (Accuracy ↑)	69.95	71.83	+1.88	68.69	71.98	+3.29	70.44	73.01	+2.57	67.99	72.50	**+4.51**
PATTERN (Accuracy ↑)	82.743	86.695	+3.952	81.949	86.676	+4.727	86.589	86.711	+0.122	79.098	86.772	**+7.674**
CLUSTER (Accuracy ↑)	78.055	78.171	+0.116	78.051	78.232	+0.181	77.466	77.836	+0.370	78.314	78.820	**+0.506**



(a) MNIST: GPS+Con. (b) MNIST: GPS+Con.+Contr. (c)MNIST: GPS+Con.+Per. (d) MNIST: GPS+GCCM

(e) CLUSTER: GPS+Con. (f) CLUSTER: GPS+Con.+Contr. (g) CLUSTER: GPS+Con.+Per. (h) CLUSTER: GPS+GCCM

Figure 3: Heatmap visualizations of the contribution terms of $W_y\mathbf{Y}_t$ and $W_t\mathbf{t}_{\text{emb}}(t)$ to $\mathbf{H}_t^{(0)}$ on MNIST (top) and CLUSTER (bottom), computed as $\frac{|W_y\mathbf{Y}_t|+|W_t\mathbf{t}_{\text{emb}}(t)|}{|W_y\mathbf{Y}_t|+|W_t\mathbf{t}_{\text{emb}}(t)|+|W_x\mathbf{X}|}$. The x-axis corresponds to node indices and the y-axis corresponds to hidden dimensions.

The results are reported in Table 6. Compared with GPS+Consistency, GPS+Consistency+Perturbation improves prediction performance on several benchmarks, indicating that feature perturbation can contribute to prediction performance when applied separately. At the same time, the complete GPS+GCCM achieves stronger overall performance than GPS+Consistency+Perturbation across most benchmarks. These performance results suggest that the improvements of GCCM cannot be attributed to feature perturbation alone.

To directly examine whether the same conclusion holds for the shortcut solution, we compare the utilization of $(\mathbf{Y}_t, t)$ between GPS+Consistency+Perturbation and GPS+GCCM. As shown in Fig. 3, GPS+Consistency+Perturbation exhibits an overall lower relative contribution from the terms associated with $(\mathbf{Y}_t, t)$ than GPS+GCCM on both MNIST and CLUSTER. This is further supported by Table 7, where GPS+GCCM achieves a larger gain from retaining $(\mathbf{Y}_t, t)$ than GPS+Consistency+Perturbation on all four classification benchmarks. These results indicate that the stronger utilization of $(\mathbf{Y}_t, t)$ in GCCM cannot be attributed to feature perturbation alone. These findings are consistent with our design, where feature perturbation is introduced after the contrastive consistency objective to further reduce the shortcut tendency by breaking identical conditioning.

E.2 Further Evidence that GCCM Mitigates the Shortcut Solution

To further evaluate the effectiveness of GCCM in mitigating the shortcut solution, we conduct a direct diagnostic experiment. Under the conditional generative formulation, the denoising network aims to reconstruct the clean label from a noisy label $\mathbf{Y}_t$. Therefore, given a noisy label $\mathbf{Y}_t$ generated from the ground-truth label $\mathbf{Y}$ through the forward process, a conditional generative predictor that effectively utilizes $\mathbf{Y}_t$ should benefit from the label information retained in $\mathbf{Y}_t$ and achieve better prediction performance when $\mathbf{Y}_t$ remains informative than at the high-noise endpoint $t = T$. In contrast, if the model falls into the shortcut solution characterized in Lemma 1 and collapses toward a deterministic predictor, its prediction becomes largely independent of $\mathbf{Y}_t$ and t, and thus even providing a highly informative noisy label would bring little improvement to the prediction performance.

The intermediate noise levels are particularly informative for distinguishing these two behaviors. At the smallest evaluated time steps, the injected label $\mathbf{Y}_t$ remains close to the clean label $\mathbf{Y}$ and retains

substantial label information. At the high-noise endpoint $t = T$, $\mathbf{Y}_t$ carries essentially no information about the clean label and therefore provides little useful information for prediction. In contrast, at intermediate noise levels, $\mathbf{Y}_t$ is partially corrupted while still retaining useful label information, such that the model needs to effectively denoise and utilize $\mathbf{Y}_t$ in order to benefit from it.

Based on this observation, we conduct the diagnostic experiment on four classification benchmarks: MNIST, CIFAR10, PATTERN, and CLUSTER. For each test instance, we sample $\mathbf{Y}_t \sim q(\mathbf{Y}_t \mid \mathbf{Y})$ through the forward process, feed the sampled $\mathbf{Y}_t$ together with the corresponding time step t into the trained model, and report the prediction accuracy as a function of t. This experiment serves only as a diagnostic rather than an inference protocol, since the ground-truth label $\mathbf{Y}$ is unavailable at test time. As a reference, we also report the accuracy obtained by directly decoding $\mathbf{Y}_t$, which provides an indication of how much label information is retained at each time step.

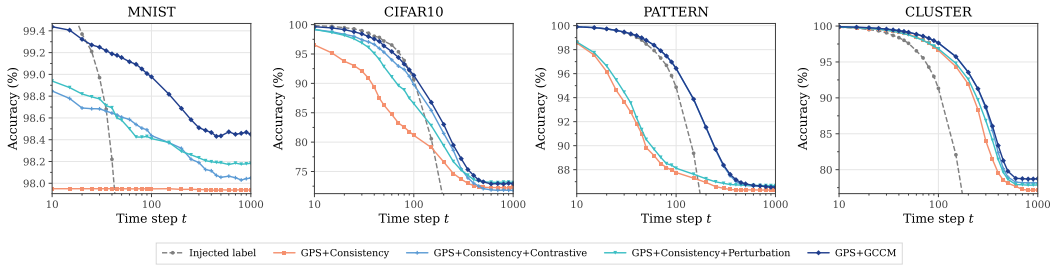


Figure 4: Prediction accuracy under different noise levels of the injected label $\mathbf{Y}_t$ on MNIST, CIFAR10, PATTERN, and CLUSTER.

As shown in Fig. 4, GPS+Consistency+Contrastive generally achieves higher prediction accuracy than GPS+Consistency at intermediate noise levels, suggesting that the contrastive consistency objective enables the model to better utilize the label information retained in $\mathbf{Y}_t$ and alleviates the shortcut tendency in vanilla consistency training. GPS+Consistency+Perturbation also benefits from informative noisy labels, showing that feature perturbation can likewise improve the utilization of $\mathbf{Y}_t$. Together, these observations show that both proposed components contribute to strengthening the utilization of the noisy label. When combined in GPS+GCCM, the model generally achieves the highest prediction accuracy among the model variants at low and intermediate noise levels across the four benchmarks.

The behavior on MNIST is particularly clear. The prediction accuracy of GPS+Consistency changes only slightly across the evaluated noise levels, such that even providing a highly informative $\mathbf{Y}_t$ brings only limited improvement. In contrast, both GPS+Consistency+Contrastive and GPS+Consistency+Perturbation become more responsive to the information retained in $\mathbf{Y}_t$, while GPS+GCCM shows the clearest dependence on its informativeness. As t increases and the injected label becomes less informative, the performance of GPS+GCCM gradually approaches its high-noise prediction level. Together, these results suggest that the contrastive consistency objective and feature perturbation play complementary roles in strengthening the utilization of $(\mathbf{Y}_t, t)$, with their combination in GCCM providing the strongest overall mitigation of the shortcut solution.

E.3 Parameter Analysis

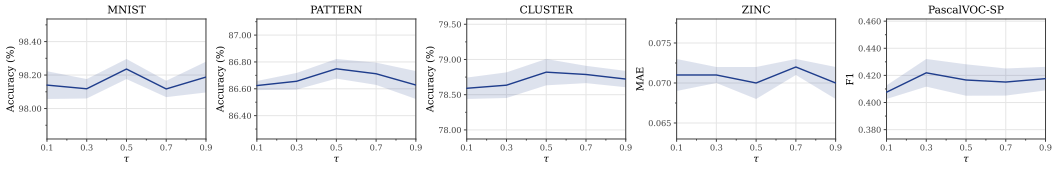


Figure 5: Performance of GCCM under different values of the temperature parameter τ on MNIST, PATTERN, CLUSTER, ZINC, and PascalVOC-SP.

Temperature parameter τ. To analyze the sensitivity of GCCM to the temperature parameter τ in the contrastive consistency objective defined in Eq. equation 7, we vary τ from 0.1 to 0.9 on

MNIST, PATTERN, CLUSTER, ZINC, and PascalVOC-SP, and report the corresponding prediction performance in Fig. 5. As shown in the figure, the performance of GCCM changes only slightly across different values of τ on all five benchmarks. These results show that GCCM is not sensitive to the choice of τ and can maintain stable performance over a wide range of values. Therefore, GCCM does not require careful tuning of the temperature parameter τ.

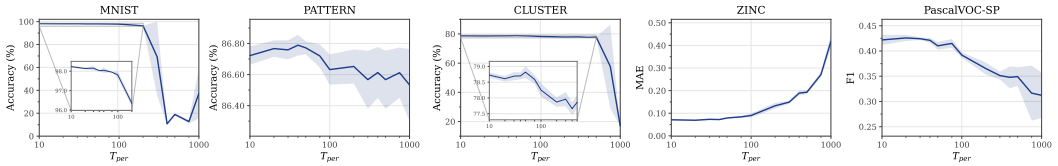


Figure 6: Performance of GCCM with different feature perturbation time steps T_{per} on MNIST, PATTERN, CLUSTER, ZINC, and PascalVOC-SP.

Feature perturbation time step T_{per}. To analyze the sensitivity of GCCM to the feature perturbation time step T_{per}, we vary T_{per} on MNIST, PATTERN, CLUSTER, ZINC, and PascalVOC-SP, and report the corresponding prediction performance in Fig. 6. As shown in the figure, the performance of GCCM remains stable within a relatively small range of T_{per} across all five benchmarks. When T_{per} becomes large, the performance consistently degrades on all five benchmarks. This observation is consistent with the role of feature perturbation in GCCM. As discussed in Section 4.3 and Lemma 3, a small perturbation is already sufficient to break the identical conditioning features across different noise levels, thereby reducing the tendency toward the shortcut solution. In contrast, a large T_{per} introduces overly strong perturbations to the conditioning features and consequently degrades the prediction performance. Overall, these results show that GCCM does not require careful tuning of T_{per}, and a relatively small feature perturbation is sufficient to achieve stable performance across different datasets and feature types.

E.4 Generalization Across Different Backbones

Table 8: Test performance on five benchmarks from Dwivedi et al. (2022). Shown is the mean ± std of five runs with different seeds.

Model	MNIST	CIFAR10	PATTERN	CLUSTER	ZINC
	Accuracy ↑	Accuracy ↑	Accuracy ↑	Accuracy ↑	MAE ↓
SAN	—	—	86.581 ± 0.037	76.691 ± 0.650	0.139 ± 0.006
SAN-LGD	—	—	80.742 ± 0.315	68.711 ± 0.676	0.236 ± 0.012
SAN-PCL	—	—	86.768 ± 0.064	77.461 ± 0.427	0.140 ± 0.002
SAN-GCCM (ours)	—	—	**86.811 ± 0.021**	**78.481 ± 0.176**	**0.131 ± 0.003**
GPS	98.08 ± 0.11	72.36 ± 0.32	86.648 ± 0.065	77.780 ± 0.236	0.072 ± 0.004
GPS-LGD	95.46 ± 0.61	70.16 ± 0.36	85.662 ± 0.181	76.773 ± 0.301	0.085 ± 0.001
GPS-PCL	98.11 ± 0.18	71.83 ± 0.22	86.695 ± 0.073	78.171 ± 0.231	**0.069 ± 0.001**
GPS-GCCM (ours)	**98.24 ± 0.06**	**72.50 ± 0.29**	**86.772 ± 0.042**	**78.820 ± 0.187**	**0.069 ± 0.002**
GRIT	98.11 ± 0.11	**76.47 ± 0.88**	87.196 ± 0.076	**80.026 ± 0.277**	**0.059 ± 0.002**
GRIT-LGD	96.54 ± 0.62	71.17 ± 1.31	84.600 ± 1.841	64.913 ± 1.639	0.118 ± 0.011
GRIT-PCL	98.00 ± 0.13	75.72 ± 0.73	86.873 ± 0.060	78.643 ± 0.836	0.063 ± 0.005
GRIT-GCCM (ours)	**98.47 ± 0.10**	75.91 ± 0.84	**87.256 ± 0.084**	79.816 ± 0.137	**0.059 ± 0.004**

The main results in Section 5.2 demonstrate the effectiveness of GCCM on the GPS backbone. A natural question is whether the proposed framework generalizes to other graph neural network architectures. To investigate this, we apply GCCM and the recent conditional generative baselines (i.e., LGD Zhou et al. (2024) and PCL Li et al. (2025b)) on top of two additional Graph Transformer backbones, SAN Kreuzer et al. (2021) and GRIT Ma et al. (2023). The results are summarized in Table 8. As shown in the table, GCCM consistently outperforms the original backbones and the baseline generative prediction frameworks LGD and PCL across all datasets under the same denoising backbone, confirming its generalizability and effectiveness. This consistent improvement suggests that GCCM's benefit stems from mitigating the shortcut solution in consistency training itself, rather than from interactions with any particular backbone architecture, indicating that GCCM provides a general training framework applicable to diverse graph neural networks.

Table 9: Efficiency comparison of GCCM against baselines across five benchmarks from Dwivedi et al. (2022).

Metric	Model	MNIST	CIFAR10	PATTERN	CLUSTER	ZINC
Time/Epoch (s)	GPS	162.22	129.68	35.63	99.71	19.33
	GPS-Diffusion	171.85	144.79	54.45	152.66	20.84
	GPS-PCL	286.99	248.24	98.70	277.24	45.70
	GPS-GCCM (ours)	302.71	243.92	97.72	276.06	46.67
GPU Memory (MiB)	GPS	418	506	14,490	9,878	408
	GPS-Diffusion	462	532	14,640	10,788	418
	GPS-PCL	538	696	20,454	15,862	484
	GPS-GCCM (ours)	584	700	20,580	16,224	486
Inference Time (s)	GPS	6.21	5.36	2.37	2.91	0.72
	GPS-Diffusion	5,756.25	5,087.50	1,868.58	2,398.31	492.65
	GPS-PCL	6.32	6.35	2.81	3.43	0.83
	GPS-GCCM (ours)	6.69	6.41	3.01	3.28	0.84

E.5 Runtime and Cost Analysis

We report training cost from three aspects: per-epoch training time, GPU memory usage, and inference time. The complete results across all five benchmarks from Dwivedi et al. (2022) are summarized in Table 9. Compared to GPS-PCL, GCCM introduces only negligible overhead, with similar training time and GPU memory usage. The additional cost mainly comes from feature perturbation and the contrastive consistency objective. For inference, compared to GPS-Diffusion which requires iterative denoising, GCCM is substantially faster due to one-step inference, achieving inference speed comparable to the deterministic GPS baseline while retaining the generative formulation.

F Discussion of Broader Impacts & Limitations

GCCM has demonstrated superior performance on general graph prediction tasks. Moreover, the proposed framework is applicable to a broader applications of prediction tasks on graphs, such as combinatorial optimization Sun & Yang (2023); Li et al. (2025a) and graph edit distance Huang et al. (2025b;a). In these settings, directly applying consistency training to accelerate diffusion-based models may similarly converge to the shortcut solution and collapse into a deterministic predictor, leading to unsatisfactory performance. In such cases, GCCM remains applicable for mitigating the shortcut solution, leading to improved performance. However, due to limited GPU computational resources, we are not able to evaluate our framework on broader applications, and we will leave this to future works.