Gradient-Free Editing for Attribute Invariance in Graph Neural Networks

Aastha Verma*
Dept. of Computer Sc. & Eng.
Indian Institute of Technology Delhi
aastha.v1411@gmail.com

Vishakha Agarwal*
Dept. of Computer Sc. & Eng.
Indian Institute of Technology Delhi
vishakha.cse.iitdelhi@gmail.com

Sahil Manchanda
Mastercard AI Garage
sahil.manchanda@mastercard.com

Sayan Ranu
Dept. of Computer Sc. & Eng.
Indian Institute of Technology Delhi
sayanranu@cse.iitd.ac.in

Abstract

Deployed Graph Neural Networks are increasingly subject to post-deployment interventions: a sensitive attribute is flagged for removal, or a spurious correlation is discovered after the model is already in production. In such scenarios, retraining is often infeasible due to a variety of reasons including scale of modern graphs, inaccessible training data, or strict privacy mandates. The challenge is further compounded in graph-structured data, where attribute dependencies propagate through message-passing, spreading unwanted correlations across the entire network. We propose GRAFT, a framework for surgical, post-hoc attribute invariance in GNNs via gradient-free model editing. Rather than retraining or fine-tuning, our method formulates the edit as a subspace-constrained, closed-form optimization: we identify structurally influential nodes that exhibit high attribute sensitivity, isolate the low-rank manifold in activation space where sensitive variation is concentrated, and solve directly for a weight update that decouples the model's decision logic from the target attribute. The update requires no access to ground truth labels, introduces no additional parameters, and is computed in a single pass. Empirical results demonstrate that our method enforces attribute invariance while preserving predictive utility, while offering up to 3 orders of magnitude speed-up over existing GNN editing approaches. Our code is available at https://github.com/idea-iitd/graft.

1 Introduction and Related Works

Graph Neural Networks (GNNs) [23, 44, 25, 47, 34, 21] and Graph Transformers [49, 35, 45, 3, 12, 22] have become the dominant paradigm for learning on relational data, achieving strong predictive performance across high-stakes domains including financial fraud detection, health diagnosis, traffic monitoring, modeling physical systems, and discovery of drugs and molecules [46, 51, 7, 18, 20, 19, 29, 24, 28, 36, 8, 6, 40]. At the core of these architectures is the recursive aggregation of neighbor information [38], which couples learned representations tightly to the input feature set.

Yet in real-world deployment, predictive performance alone is insufficient. Models are increasingly subject to post-deployment interventions such as regulatory audit flags on sensitive demographic attribute [2], or spurious correlations discovered after launch rendering a feature unreliable for robust inference[39, 16, 30]. Consider a loan-approval GNN trained on a financial network that

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

uses an applicant's ZIP code as a node feature. The model may inadvertently exploit correlations between geographic location and loan default rates that reflect historical socioeconomic disparities and act as proxies for socioeconomic status , rather than causal determinants of creditworthiness. When regulatory guidelines subsequently mandate that Zip codes cannot be used as a basis for financial decisions, the model must be corrected. Critically, in graph-structured data, attribute dependencies propagate through the message-passing mechanism [23, 50, 10] and further amplifying the discriminatory dependency across the entire network.

The discovery of such spurious dependencies in a deployed model necessitates a recalibration of its parameters to facilitate a *post-hoc transfer* of its decision logic [43]. The objective is to enforce ***attribute invariance***, mathematically decoupling the model's decision manifold from the latent signals that propagate these dependencies, while preserving-or potentially augmenting-the model's global functional utility. In this regime, traditional interventions such as retraining are typically infeasible [13], precluded by the prohibitive computational scale of modern graphs [5] or by stringent data-privacy mandates that render the original training distribution inaccessible [15]. This raises a fundamental question: *can we enforce attribute invariance in a deployed* GNN *without full retraining and without access to the original training data?*

1.1 Related Works

To address the need for targeted interventions, *model editing* has emerged as a vital research area focusing on the efficient "patching" of specific behaviors while minimizing disruption to unrelated model knowledge [31, 32, 48]. While post-hoc editing is well-established for language and vision, GNN editing remains in its infancy. Current works like EGNN[27] and SEED[53] have been proposed to perform editing on GNNs; however, these methods often necessitate architectural modifications or rely on the availability of original training samples to mitigate catastrophic forgetting. Similarly, while there exists body of works in fair graph learning to reduce bias, these methods often target different intervention regimes : Works such as FAIRGNN and NIFTY are restricted to training-time objectives [11, 1], while post-hoc approaches like GRAPHPAR [52] require architectural modifications and iterative optimization similar to EGNN and SEED-GNN.

Despite initial progress, existing works for GNN editing suffer from four critical bottlenecks:

1. **Label Correction vs. Attribute Invariance:** Existing GNN editing methods are designed to correct *point-wise* mispredictions; given a set of nodes whose labels have changed or were incorrectly assigned, they patch the model to predict the right label for those specific nodes. This is fundamentally different from enforcing *attribute invariance*; the issue is not the predictive quality of the model, but rather its reliance on spurious or sensitive attribute.

2. **Computational Overhead:** Current GNN editing works, namely EGNN and SEED-GNN, involve the addition of trainable layers and rely on costly, iterative gradient-based fine-tuning for parameter updates. These modifications increase the complexity of the deployed model and introduce significant latency during the update phase. A core **desideratum** of model editing is **efficiency**; however, as we will show in § 4, existing methods incur massive computational overhead, making them impractical.

3. **Structural Perturbation and Edit Composability:** EGNN and SEED-GNN address editing by appending additional trainable layers to a frozen GNN backbone. Furthermore, this architectural expansion compounds with each successive edit: every new round of editing appends yet another layer, progressively inflating model complexity and drifting further from the original weight space. This makes continual or repeated editing, a practical necessity in dynamic deployment environments, architecturally unsustainable. Furthermore, since the backbone remains frozen, the expanded model cannot cleanly absorb new training data should it become available, as the added layers and the frozen backbone were never jointly optimized. Finally, the addition of layers slows inference times. A principled editing framework should instead operate entirely within the original weight space, performing surgical updates that are both composable across edit rounds and compatible with future adaptation.

4. **Dependency on Supervised Training Data:** Both SEED-GNN and EGNN assume access to the original *training data labels* during the post-hoc editing phase. As discussed, access to training data may be infeasible due to privacy constraints.

1.2 Contributions

We address these limitations through GRAFT, **GRA**dient-**F**ree GNN Edi**t**ing for Attribute Invariance . Our core contributions are as follows.

- **Problem formulation:** We formulate the novel problem of attribute masking in GNNs, which has not been studied thus far.
- **Gradient-Free Editing:** We introduce a closed-form, least-squares framework for GNN editing that eliminates the need for costly iterative fine-tuning, adds **zero additional parameters** and does not depend on the original training data.
- **Utility-Preserving Subspace Constraints:** To prevent catastrophic forgetting, GRAFT extracts the specific low-rank subspace where the majority of the sensitive variation is concentrated. By mathematically constraining the weight update to this subspace, the edit remains orthogonal to the model's general behavior and preserves functional integrity.
- **Empirical Evaluation:** We benchmark GRAFT across six real-world datasets and demonstrate that GRAFT is effective in decoupling model logic from target attributes without compromising model performance and is up to 3 orders of magnitudes faster than state-of-the-art baselines.

2 Preliminaries and Problem Formulation

In this section, we introduce the preliminary concepts central to our work and formulate the problem of GNN editing for attribute invariance.

Definition 1 (Graph) *A graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}$ is the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. The node feature matrix is denoted by $\mathbf{X} \in \mathbb{R}^{N \times d}$, where d is the number of features from the feature set $\mathcal{F}$. For each node $v \in \mathcal{V}$, $\mathbf{x}_v \in \mathbb{R}^d$ represents its corresponding feature vector.*

Graph learning architectures-including GNNs and Graph Transformers-learn node representations by iteratively aggregating information through L layers. For a node $v \in \mathcal{V}$, the 0-th layer is initialized as $\mathbf{h}_v^{(0)} = \mathbf{x}_v$. At each layer $\ell \in \{1, \ldots, L\}$, the model summarizes relational context $\mathbf{a}_v^{(\ell)}$ from its scope $\mathcal{S}_v$ (e.g., local neighbors or global nodes) and updates the representation via a transformation parameterized by a weight matrix for layer ℓ as $\mathbf{W}^{(\ell)} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$:

$$\mathbf{h}_v^{(\ell)} = \sigma\left(\text{COMBINE}^{(\ell)}(\mathbf{h}_v^{(\ell-1)}, \mathbf{a}_v^{(\ell)}, \mathbf{W}^{(\ell)})\right) \tag{1}$$

where σ is a non-linear activation.

Definition 2 (Inductive Graph Learning) *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be a graph with node features $\mathbf{X} \in \mathbb{R}^{N \times d}$. We assume a hidden target function $Y : \mathcal{V} \to \mathbb{R}$. Given a training subgraph induced by $\mathcal{V}_{tr} \subset \mathcal{V}$ and its corresponding ground-truth labels $\{y_v\}_{v \in \mathcal{V}_{tr}}$, the goal is to optimize the parameters Θ of a GNN f_Θ to minimize a task-specific loss. The resulting model learns an inductive mapping from the joint feature-topological space to a representation manifold to predict $Y(v)$ for any $v \in \mathcal{V}$.*

Evaluation is conducted on a disjoint set of unseen test nodes $\mathcal{V}_{test}$ using a utility metric $Q(f_\Theta, \mathcal{V}_{test})$, such as Accuracy, AUROC, RMSE etc. which quantifies the model's predictive performance.

Definition 3 (Post-hoc Model Editing) *Given a pre-trained GNN f_Θ and a set of editing samples $\mathcal{V}_{edit} \subset \mathcal{V}_{val}$ defining a target behavior, post-hoc model editing is the process of deriving a parameter shift $\Theta \to \Theta'$ to align the model with the new behavior. The update must maintain the model's functional integrity, ensuring that the predictive utility on unseen test nodes $\mathcal{V}_{test}$ is preserved or enhanced: $Q(f_{\Theta'}, \mathcal{V}_{test}) \geq Q(f_\Theta, \mathcal{V}_{test}) - \epsilon$, where $\epsilon \geq 0$ is a small fidelity margin.*

Problem 1 (Post-hoc Attribute Invariance) *Given a pre-trained GNN f_Θ and a sensitive attribute $s \in \mathcal{F}$, the task is to derive an edit $\Theta \to \Theta'$ using a subset of nodes $\mathcal{V}_{val}$ to enforce attribute invariance. Let $\tilde{\mathbf{X}}^{(s)}$ be a counterfactual feature matrix where attribute s is perturbed. The updated parameters Θ' must ensure the model's decision manifold is functionally decoupled from the sensitive signal:*

$$\forall v \in \mathcal{V}_{test}, \quad f_{\Theta'}(\mathcal{G}, \mathbf{X})_v \approx f_{\Theta'}(\mathcal{G}, \tilde{\mathbf{X}}^{(s)})_v \tag{2}$$

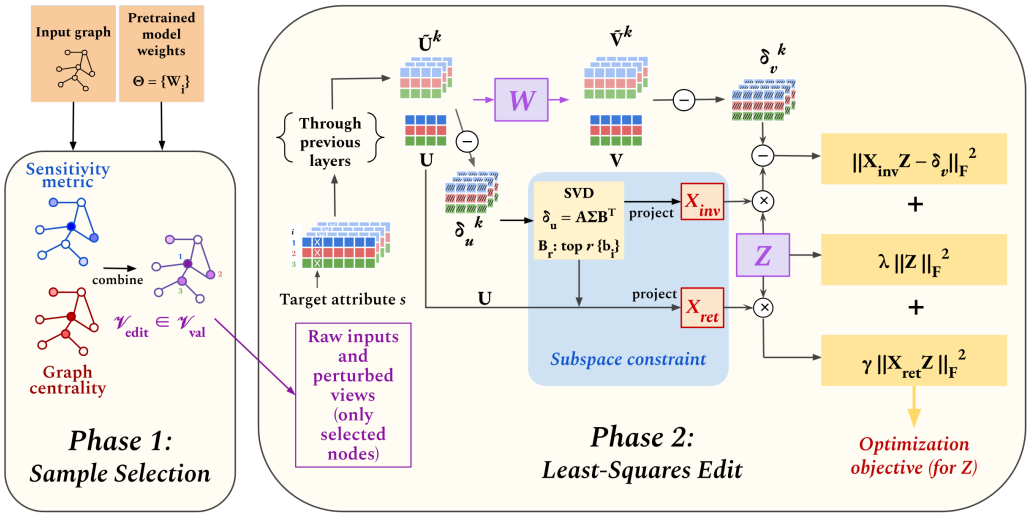


Figure 1: The pipeline of the proposed methodology GRAFT.

3 GRAFT: Proposed Methodology

Without loss of generality, let the parameter set $\Theta = \{\mathbf{W}_1, \cdots, \mathbf{W}_w\}$ be a collection of weight matrices in the GNN. Revisiting Problem 1, our objective is to derive a parameter shift $\mathbf{S}_i$ for each $\mathbf{W}_i \in \Theta$, such that the updated weights $\Theta' = \{\mathbf{W}_i' = \mathbf{W}_i + \mathbf{S}_i \mid \mathbf{W}_i \in \Theta\}$ satisfies Eq. 2. These weight shifts are identified in two sequential phases, as outlined in Fig. 1:

1. **Editing Sample Selection:** To derive $\mathbf{S}_i$, and therefore Θ', we identify a restricted calibration set of representative nodes $\mathcal{V}_{\text{edit}} \subseteq \mathcal{V}_{\text{val}}$ that are both sensitive to attribute perturbations and structurally influential within the graph.
2. **Subspace-Constrained Weight Update:** Subsequently, each $\mathbf{S}_i$ is derived through a closed-form optimization problem over $\mathcal{V}_{\text{edit}}$. The optimization framework restricts the shift within a subspace orthogonal to task-relevant pathways. This ensures that the edit leaves the model's broader predictive behavior intact.

We next detail each of these phases.

3.1 Phase 1 : Sensitivity and Topology-Aware Editing Sample Selection

We select nodes based on two criteria: their functional sensitivity to attribute changes and their topological importance within the graph.

Attribute Sensitivity Estimation: Let the original prediction for node v_i be $\mathbf{p}_i = f_\Theta(\mathbf{X}, A)_i$[2] To evaluate the model's dependence on a target attribute s, we construct a perturbed feature matrix $\widetilde{\mathbf{X}}$ identical to $\mathbf{X}$ except for the target column of the validation nodes. The perturbed value $\widetilde{\mathbf{X}}_{i,s}$ is sampled based on the nature of the attribute.

$$\widetilde{\mathbf{X}}_{i,s} \sim \begin{cases} \text{Uniform}\{a_1, \ldots, a_m\} & \text{if discrete} \\ \mathcal{U}\left[\min_{\forall v_j \in \mathcal{V}_{val}}\{\mathbf{X}_{j,s}\},\ \max_{\forall v_j \in \mathcal{V}_{val}}\{\mathbf{X}_{j,s}\}\right] & \text{otherwise} \end{cases} \tag{3}$$

Here, $\{a_1, \ldots, a_m\}$ represents the domain of a target attribute s (e.g., specific gender). For continuous features, the perturbation is sampled from a continuous uniform distribution $\mathcal{U}$ spanning the observed range in the validation set. Let the perturbed output be $\widetilde{\mathbf{p}}_i = f_\Theta(\widetilde{\mathbf{X}}, A)_i$. To obtain a robust estimate of attribute sensitivity, we sample K independent perturbations of the target attribute for each node $v_i \in \mathcal{V}_{\text{val}}$, yielding perturbed feature matrices $\{\widetilde{\mathbf{X}}_{i,s}^{(1)}, \cdots, \widetilde{\mathbf{X}}_{i,s}^{(K)}\}$. The sensitivity score Δ_i is defined as the average L_1 shift under attribute perturbation:

$$\Delta_i = \frac{1}{K} \sum_{k=1}^{K} \left\| \mathbf{p}_i - \widetilde{\mathbf{p}}_i^{(k)} \right\|_1 \tag{4}$$

[2] A denotes the adjacency matrix. The prediction corresponds to the final layer logits.

Incorporating structure: The attribute sensitivity propagates through the graph via the message-passing mechanism. The impact of this propagation is proportional to the corresponding message-passing weights. We capture this structural influence via *Personalized PageRank* (PPR) [9], which models the steady-state influence of each node over the rest of the graph through iterative propagation. To faithfully reflect the message-passing dynamics of the underlying architecture, we construct a biased transition matrix using model-specific edge weights w_{ij}. For a GCN [25] backbone, these are defined as $w_{ij} = (\tilde{\mathbf{D}}^{-1/2}\tilde{A}\tilde{\mathbf{D}}^{-1/2})_{ij}$, where $\tilde{A} = A + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is its corresponding degree matrix.[3] The transition probabilities are proportional to these weights:

$$P_{ij} = \frac{w_{ij}}{\sum_{k\in\mathcal{N}_i} w_{ik}}, \tag{5}$$

where $\mathcal{N}_i$ denotes the neighborhood of node v_i. The sensitivity-personalized restart distribution is:

$$\mathbf{e}_{=}\frac{\Delta_i}{\sum_{\forall v_j\in\mathcal{V}_{val}} \Delta_j}, \tag{6}$$

concentrating restart probability on nodes that are most sensitive to attribute perturbations. The PPR scores $\boldsymbol{\pi}$ are then obtained via power iteration, where $\boldsymbol{\pi}^{(t+1)} = (1-\alpha)\mathbf{P}^\top\boldsymbol{\pi}^{(t)} + \alpha\mathbf{e}$. $\alpha \in (0,1)$ is the damping factor. To align the propagation horizon with the GNN's computational graph, we set $\alpha = \frac{1}{L}$, where L is the number of GNN layers. Under this parameterization, the expected number of propagation steps before restart is $\frac{1-\alpha}{\alpha} = L-1$, approximately restricting structural influence to within the receptive field of each node. The geometric decay induced by repeated multiplication by $(1-\alpha) = \frac{L-1}{L}$ ensures that contributions from nodes beyond L hops diminish exponentially with hop distance, effectively enforcing a soft boundary at the receptive field of the GNN. The final representative edit set $\mathcal{V}_{edit}$ is then chosen as the top b nodes with the highest PPR score π_i, where b is a user-defined budget controlling the number of editing samples. Through the fusion of attribution sensitivity with the topology-induces message-passing dynamic, we ensure that the subsequent weight update is guided by the most informative and topologically impactful signals in the graph.

3.2 Phase 2: Subspace-Constrained Least-Squares Edit

Our next task is to derive the shift matrix $\mathbf{S}_i$ for each weight matrix $\mathbf{W}_i \in \Theta = \{\mathbf{W}_1, \cdots, \mathbf{W}_w\}$ to construct $\mathbf{W}_i' = \mathbf{W}_i + \mathbf{S}_i$. Since the same procedure is applied to each $\mathbf{W}_i$, we drop the subscript and use $\mathbf{S}$ and $\mathbf{W}$ hereon for simplicity. Generally, each $\mathbf{W}$ applies a linear transformation $\mathbf{v} = \mathbf{W}\cdot\mathbf{u}$ to an *input vector* $\mathbf{u} \in \mathbb{R}^{d_{in}}$ producing an *output vector* $\mathbf{v} \in \mathbb{R}^{d_{out}}$. In the context of GNNs, $\mathbf{u}$ typically represents an aggregation of neighboring node embeddings at a given layer.

The shift $\mathbf{S}$ needs to ensure two properties: (i) *invariance* to the target attribute s, and (ii) *retention of knowledge* embedded in the original weight matrix $\mathbf{W}$. We next mathematically formalize each of these properties and how the shift $\mathbf{S}$ is derived to respect these.

Invariance Objective: For each node $v_i \in \mathcal{V}_{edit}$, let $\mathbf{u}_i$ represent the input vector under original features $\mathbf{X}$, and $\widetilde{\mathbf{u}}_i^k$ represent the input vector corresponding to the k-th counterfactual augmentation $\widetilde{\mathbf{X}_{i,s}^k}$ (Eq. 3). Invariance to attribute s demands:

$$\forall k, i,\ (\mathbf{W}+\mathbf{S})\widetilde{\mathbf{u}}_i^k \approx (\mathbf{W}+\mathbf{S})\mathbf{u}_i \tag{7}$$

Since $\widetilde{\mathbf{v}}_i^k = \mathbf{W}\widetilde{\mathbf{u}}_i^k$ and $\mathbf{v}_i = \mathbf{W}\mathbf{u}_i$, Eq. 7 is equivalent to:

$$\forall k, i,\ \mathbf{S}(\mathbf{u}_i - \widetilde{\mathbf{u}}_i^k) \approx \widetilde{\mathbf{v}}_i^k - \mathbf{v}_i \tag{8}$$

We capture this with the following minimization objective:

$$\min_{\mathbf{S}} \sum_{i=1}^{b}\sum_{k=1}^{K} \|\mathbf{S}(\mathbf{u}_i - \widetilde{\mathbf{u}}_i^k) - (\widetilde{\mathbf{v}}_i^k - \mathbf{v}_i)\|_2^2 \tag{9}$$

Knowledge Retention: Targeting only invariance may inadvertently lead to *catastrophic forgetting* [26], where the model loses task-relevant intelligence. To ensure that the model's predictions for original activations remain stable, we formulate the below constraint:

$$(\mathbf{W}+\mathbf{S})\mathbf{u}_i \approx \mathbf{W}\mathbf{u}_i \implies \mathbf{S}\mathbf{u}_i \approx \mathbf{0}, \tag{10}$$

[3]Weighting schemes for other architectures (e.g., SAGE, GIN) are detailed in Appendix A.1.

leading to the minimization objective: $$\min_{\mathbf{S}} \sum_{i=1}^{b} \|\mathbf{S}\mathbf{u}_i\|_2^2 \tag{11}$$

Full minimization objective: Accounting for both invariance and knowledge retention, the full minimization objective is expressed as:

$$\min_{\mathbf{S}} \underbrace{\sum_{i=1}^{b}\sum_{k=1}^{K} \left\| \mathbf{S}(\mathbf{u}_i - \widetilde{\mathbf{u}}_i^{(k)}) - (\widetilde{\mathbf{v}}_i^{(k)} - \mathbf{v}_i) \right\|_2^2}_{\text{Attribute Invariance}} + \underbrace{\gamma \sum_{i=1}^{b} \|\mathbf{S}\mathbf{u}_i\|_2^2}_{\text{Knowledge Retention}} + \underbrace{\lambda \|\mathbf{S}\|_F^2}_{\text{Regularization}} \tag{12}$$

where $\lambda > 0$ and $\gamma > 0$ are weighting hyperparameters. The regularization term discourages large shifts. Crucially, Eq. 12 is an instance of regularized least-squares, which is convex and therefore admits a closed-form solution, eliminating the need for iterative gradient-based optimization entirely. This is the key property that makes GRAFT both computationally efficient and gradient-agnostic.

3.2.1 Subspace-constrained Minimization

While the least-squares objective in Eq. 12 provides a principled framework for enforcing attribute invariance, it offers no guidance on *where* in the weight space the shift $\mathcal{S}$ should be applied. Without such guidance, the optimization diffuses the update across all $d_{\text{out}} \times d_{\text{in}}$ weight directions indiscriminately, the majority of which encode task-relevant computation and potentially unrelated to the sensitive attribute.

To address this, we adopt a subspace-constrained approach [14] that first identifies *where* in the activation space the sensitive attribute exerts its influence, and then restricts the weight update exclusively to that region. By construction, any update within this subspace is orthogonal to task-relevant weight directions, preserving the model's broader predictive behavior. We identify this *sensitive subspace* by analyzing the directions along which counterfactual inputs systematically deviate from the originals. Specifically, we construct the perturbation matrix $\boldsymbol{\delta_u} \in \mathbb{R}^{Kb \times d_{\text{in}}}$ by stacking the input perturbation vectors $\delta_{\mathbf{u}_i}^{(k)} = \mathbf{u}_i - \widetilde{\mathbf{u}}_i^{(k)}$, iterating over K counterfactual views for each node before moving to the next:

$$\boldsymbol{\delta_u} = \left[\delta_{\mathbf{u}_1}^{(1)}, \cdots, \delta_{\mathbf{u}_1}^{(K)}, \delta_{\mathbf{u}_2}^{(1)}, \cdots, \delta_{\mathbf{u}_2}^{(K)}, \cdots, \delta_{\mathbf{u}_b}^{(1)}, \cdots, \delta_{\mathbf{u}_b}^{(K)} \right]^\top \tag{13}$$

Next, we compute its Singular Value Decomposition (SVD) [17]:

$$\boldsymbol{\delta_u} = \sum_{i=1}^{m} \sigma_i \mathbf{a}_i \mathbf{b}_i^\top, \tag{14}$$

where $m = \text{rank}(\boldsymbol{\delta_u}) \leq \min(Kb, d_{\text{in}})$ denotes the number of non-zero singular values. The right singular vectors $\mathbf{b}_i$ identify the principal directions in activation space along which the counterfactual views systematically deviate from the original inputs. We retain the top r components, selected to account for at least $\rho\%$ of the total variance,i.e., $\frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{m} \sigma_i^2} \geq \rho$, forming the projection basis matrix $\mathbf{B}_r = [\mathbf{b}_1, \ldots, \mathbf{b}_r] \in \mathbb{R}^{d_{\text{in}} \times r}$. We parameterize the surgical weight shift as a projection onto this basis:

$$\mathbf{S}_{sub} = (\mathbf{B}_r \mathbf{Z})^\top = \mathbf{Z}^\top \mathbf{B}_r^\top, \tag{15}$$

where $\mathbf{Z} \in \mathbb{R}^{r \times d_{\text{out}}}$ represents the adjustment coordinates within the sensitive subspace to be determined. This parameterization acts as a mathematical filter: for any input vector $\mathbf{u}$ that lies outside the sensitive subspace, i.e., whose attribute-induced variation has no component along $\mathbf{b}_1, \ldots, \mathbf{b}_r$, we have $\mathbf{B}_r^\top \mathbf{u} = \mathbf{0}$, which directly implies $\mathbf{S}_{sub}\mathbf{u} = \mathbf{Z}^\top \mathbf{B}_r^\top \mathbf{u} = \mathbf{0}$. Hence, it guarantees that nodes whose representations are unaffected by the target attribute remain strictly unperturbed under the edit.

To find the optimal coordinates $\mathbf{Z}$, we substitute the subspace parameterization $\mathbf{S}_{sub} = \mathbf{Z}^\top \mathbf{B}_r^\top$ into Eq. 12, projecting the inputs onto the sensitive subspace via $\mathbf{X}_{\text{inv}} = \boldsymbol{\delta_u} \mathbf{B}_r \in \mathbb{R}^{Kb \times r}$ and $\mathbf{X}_{\text{ret}} = \bar{\mathbf{U}} \mathbf{B}_r \in \mathbb{R}^{b \times r}$, where $\bar{\mathbf{U}} = [\mathbf{u}_1, \cdots, \mathbf{u}_b]^\top$ is the stacked matrix of original node activations. This reduces the optimization from the full $d_{\text{out}} \times d_{\text{in}}$ weight space to a compact $r \times d_{\text{out}}$ problem, where $r \ll d_{\text{in}}$. The steering residual matrix $\boldsymbol{\delta_v} \in \mathbb{R}^{Kb \times d_{\text{out}}}$ is defined analogously to $\boldsymbol{\delta_u}$, stacking the residuals $\delta_{\mathbf{v}_i}^{(k)} = \widetilde{\mathbf{v}}_i^k - \mathbf{v}_i$ across all nodes and counterfactual views. The resulting subspace-constrained objective is:

$$\min_{\mathbf{Z}} \underbrace{\|\mathbf{X}_{\text{inv}}\mathbf{Z} - \boldsymbol{\delta_v}\|_F^2}_{\text{Attribute Invariance}} + \gamma \underbrace{\|\mathbf{X}_{\text{ret}}\mathbf{Z}\|_F^2}_{\text{Functional Integrity}} + \lambda \underbrace{\|\mathbf{Z}\|_F^2}_{\text{Regularization}} \tag{16}$$

Table 1: Dataset details.

Dataset	Task	#Nodes	#Edges	#Features	Prediction Goal
Bail [1]	Classification	18,876	321,308	18	Bail/No-Bail
T-Finance [42]	Classification	39,357	21,222,543	10	Anomaly
Pokec-n [41]	Classification	66,569	1,100,663	266	Occupation
YelpChi [37]	Classification	45,954	3,846,979	32	Anomaly
Twitch-views [4]	Regression	168,114	6,797,557	4	Total View Count
Artnet-views [4]	Regression	50,405	280,348	50	Artist View Count

After computing the optimal $\mathbf{Z}$, we lift the solution from the low-rank subspace back into the full parameter space. The surgical weight shift $\mathbf{S}_{sub} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is reconstructed and applied as:

$$\mathbf{S}_{sub} = (\mathbf{B}_r \mathbf{Z})^\top, \quad \mathbf{W}' = \mathbf{W} + \mathbf{S}_{sub} \tag{17}$$

This procedure is applied layerwise, sequentially from layer $\ell = 1$ to layer $\ell = L$. At each layer, all weight matrices $\{\mathbf{W}_1^{(\ell)}, \cdots, \mathbf{W}_w^{(\ell)}\}$ are updated before proceeding to the next layer, using the updated representations from the preceding layer as inputs. Once all L layers have been edited, the updated parameter set Θ' constitutes the final attribute-invariant model. Appendix A.2 provides proofs of the uniqueness of the layerwise regularized solution, exact preservation of layer outputs for inputs orthogonal to the sensitive subspace, and a computable bound on residual sensitivity over the calibration perturbations.

Theorem 1 (Guarantees of the GRAFT edit) *For $\gamma, \lambda \geq 0$, the subspace-constrained objective admits a unique minimizer $\mathbf{Z}^*$, whose update $\mathbf{S}_{\text{sub}} = (\mathbf{Z}^*)^\top \mathbf{B}_r^\top$ preserves all activations orthogonal to the sensitive subspace and has computable residual attribute sensitivity; see Appendix A.2.*

4 Experiments

In this section, we benchmark GRAFT to investigate the following core research questions:

- **Utility vs. Invariance:** How well does GRAFT masks the impact of sensitive attributes while preserving global predictive performance in comparison to state-of-art editing baselines?
- **Computational Efficiency**: How does the closed-form formulation of GRAFT scale in terms of running time compared to existing iterative gradient-based GNN editing methods across different editing budgets and architectures with varying complexities?
- **Robustness to Budget:** Does GRAFT maintain effectiveness under diverse editing budgets?
- **Ablation Studies:** What is the impact of subspace constraint and graph topology towards reducing attribute variance while maintaining predictive performance?

The code to reproduce our experiments is present at `https://github.com/idea-iitd/graft`

4.1 Experimental Setup

All experiments are conducted across three independent runs with different seeds. We report the mean values in the main text, while results along with standard deviations are provided in App. A.4. Details on hyper-parameters, system configuration, train-val-test splits are provided in App. A.3.
Datasets: We evaluate GRAFT across the six diverse real-world datasets listed in Table 1 spanning social networks, financial transactions, and legal. These include: **Bail** [1], a criminal justice dataset for bail prediction; **Pokec-n** social network [41] ; **T-Finance** [42] financial anomaly detection dataset; **YelpChi** [37], a restaurant review dataset for anomaly detection; and **Twitch-views** and **Artnet-views** [4], which track popularity metrics for content creators and artists, respectively.

Sensitive features: To evaluate editing and identify sensitive features, we utilize explicit demographic metadata where available, such as *Race* in *Bail* and *Age* in *Pokec-n*. In datasets lacking clear sensitive labels, namely *YelpChi*, *T-Finance*, *Twitch-views*, and *Artnet-views*, we identify the feature exhibiting the highest correlation with the target variable. This strategy ensures that an editing model must decouple non-trivial signals from the decision manifold.

Baselines: We compare GRAFT against state-of-the-art GNN editing methods, EGNN [27] and SEED-GNN [53], as well as *Model Fine-tuning*-a baseline that updates the original model parameters via supervised tuning on validation data within the edit budget b, using samples perturbed on the target attribute s with their labels held fixed. To provide a reference point for the attribute

invariance in Problem 1, we include an *Oracle* baseline retrained on the full training set without the sensitive attribute s. This serves as a reference point for the best achievable invariance with respect to the specified attribute. Note that unlike GRAFT, all these baselines require access to original training labels during editing. In addition to GNN editing baselines, we also provide an additional comparison with fairness-aware baselines in Appendix A.6, including FAIRGNN [11], NIFTY [1], and GRAPHPAR [52].

Metrics: We have three primary objectives:

1. **Invariance to Target Attribute (Variance Drop)** $\uparrow$: Let $\mathcal{V}_{\text{test}}$ be the set of test nodes. The variance in the predicted logits under K independent perturbations of the sensitive attribute s (Eq. 3) is:

$$\textbf{Var}(\Theta) = \frac{1}{|\mathcal{V}_{\text{test}}|} \sum_{v \in \mathcal{V}_{\text{test}}} \frac{1}{K} \sum_{k=1}^{K} \left\| f_\Theta(\mathcal{G}, \mathbf{X})_v - f_\Theta(\mathcal{G}, \tilde{\mathbf{X}}^{(k,s)})_v \right\|_2^2 \tag{18}$$

   The drop in variance post editing is therefore $\textbf{Var}(\Theta) - \textbf{Var}(\Theta')$.

2. **Utility Preservation (Performance Drop)** $\downarrow$**:** This metric measures the percentage change in predictive performance on the test samples $\mathcal{V}_{test}$, following the edit. For node classification, we measure this via the percentage drop in *AUC-PR*, and for node regression, we use R^2 [33].

3. **Efficiency (Time)**$\downarrow$**:** We also benchmark the time taken to edit.

Architectures: We evaluate performance on **GCN** [25], **GIN** [47], and **GAT** [44], and a state-of-the-art Graph Transformer **Polynormer**[12].

Experimental Protocol: Unless specified, we fix the default editing budget at $\text{b} = 0.25 \times |\mathcal{V}_{val}|$, utilizing only 25% of the validation nodes for editing. We further vary this budget to investigate the sensitivity of each method regarding both predictive quality and computational efficiency.

4.2 Benchmarking

Utility vs. Invariance Trade-off: Fig. 2 presents the utility vs. invariance tradeoff. Each technique is represented by a two-dimensional point. The ideal region combines zero or negative AUC-PR change with maximum variance reduction. Across the evaluated datasets and GNN architectures, GRAFT consistently provides a favorable trade-off, achieving strong variance reduction with minimal utility degradation. In contrast, state-of-the-art GNN editing baselines such as SEED-GNN (blue square) and EGNN (red circle) exhibit significant attribute sensitivity or catastrophic forgetting, often suffering from extreme performance degradation to achieve comparable levels of invariance. Fine-tuning suffers large accuracy drops as well since gradient signals from the perturbed samples used to induce invariance conflicts with the original weights.

Editing Efficiency: Fig. 3 (a) studies the time taken to edit by GRAFT and other baselines against the number of samples used to edit. GRAFT is 3 orders of magnitude faster over iterative, gradient-based methods across different budgets due to admitting a closed-form minimization objective. Further, `EGNN` and `SeedGNN` scales sharply with the budget-often exceeding 10^2 to 10^3 seconds on datasets like *Pokec-n*, while, GRAFT maintains a milder growth rate. The conclusion holds across diverse architectures. Fine-tuning is the only baseline that is at times faster than or comparable to GRAFT. However, as showing in Fig. 2 shows, this speed comes at the cost of solving the task, as fine-tuning incurs substantial AUC-PR drops, whereas GRAFT preserves utility.

Robustness to Editing Budget: Fig. 3(b) visualizes the trade-off between utility (AUC-PR Drop) and invariance (Variance Reduction) across diverse budget thresholds ranging from 0.01 to 0.25. We observe that across different budget thresholds from 0.25 to even low fractions of 0.01 i.e 1% of the validation set, GRAFT consistently outperforms all methods. Further, even at the 1% budget level, GRAFT outperforms other methods even while using with an order of magnitude less editing points.

Table 2: Ablation study: Starting from the full method (**GRAFT**), we progressively remove the subspace-constraint (§ 3.2.1) and topology-aware selection (§ 3.1) components to measure their contribution to the invariance–utility trade-off, measured through the Variance-to-Utility Ratio (VUR). Higher VUR is better. Results with GAT are present in App. A.7

	Bail		Pokec-n		YelpChi	
Method	**GCN**	**GIN**	**GCN**	**GIN**	**GCN**	**GIN**
GRAFT	**1.01e6**	**2.13e5**	**2.24e3**	**3.22e3**	**3.16e4**	**4.93e4**
w/o Subspace	3.09e4	3.42e3	1.76e3	4.71e2	2.16e4	1.64e4
w/o Topology	2.64e4	2.92e3	1.63e3	4.97e2	1.75e4	3.07e3

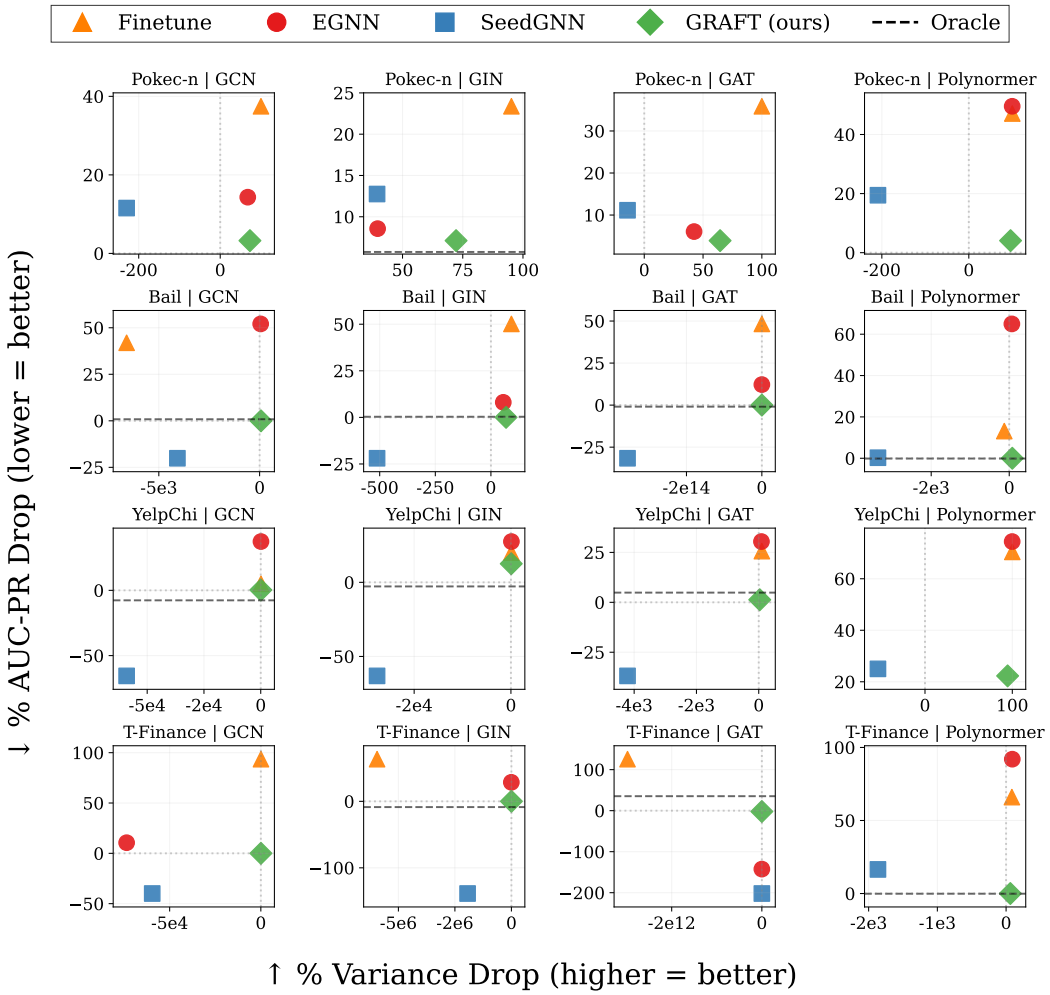


Figure 2: **Utility vs. Invariance** We visualize the trade-off between model utility (AUC-PR Drop) and attribute invariance (Variance Drop) for node classification. The **Oracle line** (dashed) represents the performance of a model trained on original training data ($\mathcal{V}_{tr}$) without using sensitive attributes.

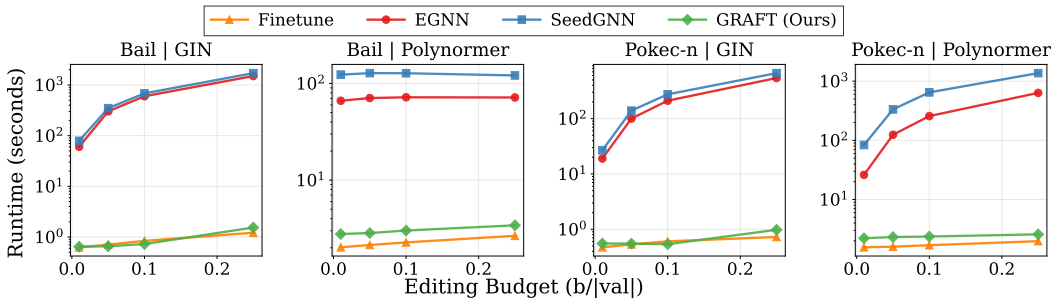
Ablation Studies: We study the marginal impact of (a) topology-aware editing sample selection and (b) the subspace constraint on performance. To quantify the invariance-utility trade-off, we use Variance to Utility Ratio $\mathbf{VUR} = \frac{|\text{Var}(\Theta) - \text{Var}(\Theta')|/\text{Var}(\Theta)}{|\text{AUC-PR}(\Theta) - \text{AUC-PR}(\Theta')|/\text{AUC-PR}(\Theta) + \epsilon}$, the percentage reduction in sensitivity variance relative to the normalized utility degradation, where ϵ is a small positive constant.

We observe that removing the *subspace constraint* leads to a drastic decline in VUR (i.e., replacing Eq. 16 with Eq. 12). Similarly, excluding *topology-aware selection* consistently yields low performance (e.g., 1.75×10^4 in *YelpChi*), demonstrating that leveraging the graph structure via PageRank is essential for ensuring the edit generalizes effectively across the network topology.

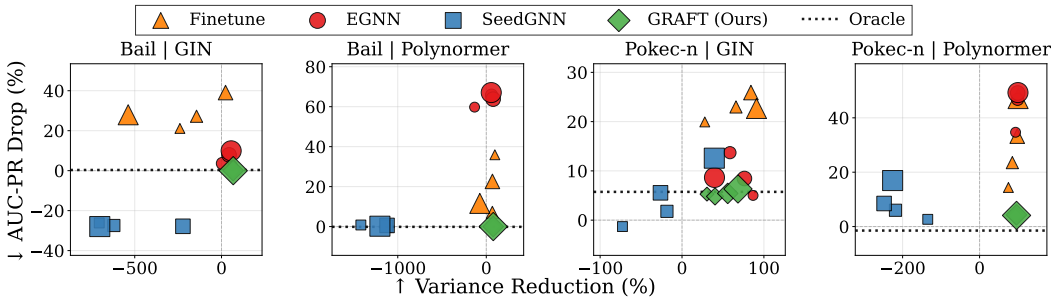
Impact of increasing model parameters on GRAFT Table 3 studies the impact of increasing model parameters on the editing time and utility-invariance trade-off on the largest dataset T-finance. Editing time grows sub-linearly with parameter count for both architectures. This is attributable to the fact that Phase 1-sample selection via PPR-dominates the overall runtime and is independent of model parameter count. The VUR metric exhibits only minor

Table 3: Effect of increasing model parameters on the T-Finance dataset.

#Params (GCN / GIN)	GCN		GIN	
	Time (s) ↓	VUR ↑	Time (s) ↓	VUR ↑
418 / 9001	9.94 ± 1.32	1.06e5	9.33 ± 0.11	1.79e3
1474 / 13580	11.35 ± 1.24	3.64e4	12.37 ± 4.27	1.57e4
3586 / 22738	12.14 ± 0.66	2.79e4	14.56 ± 1.58	2.21e5



(a) Running time across different editing budgets



(b) Performance across different editing budgets. A bigger size indicates higher editing budget. The budget thresholds used for the experiment are: $\{0.01, 0.05, 0.1, 0.25\}$.

Figure 3: **Impact of budget on (a) editing time and (b) knowledge retention Vs. Invariance trade-off.**

fluctuations across parameter scales for both architectures, suggesting that GRAFT's utility-invariance trade-off is robust to model size and does not degrade as the parameter count increases.

Impact of weighting factors: We study the impact of weighting factors specific to GRAFT, namely γ and λ of Eq. 16 in App. A.5. γ controls the importance of knowledge retention. Hence, higher γ leads to reduced accuracy drop, but reduced invariance as well (smaller variance drop). λ, on the other hand, controls the importance of keeping the shift sparse. Sparse shifts mean the edited model is closer to the original model. Hence, increasing λ preserves utility, but induces lower invariance (variance drop).

5 Conclusions, Limitations and Future Directions

Post-deployment correction of GNN behavior is an open practical challenge. Existing methods either require full retraining, depend on inaccessible training labels, or structurally perturb the model with every successive edit, making them fundamentally unsuitable for real-world deployment pipelines. GRAFT breaks this deadlock by establishing that attribute invariance in GNNs can be enforced as a single-pass, closed-form operation entirely within the original weight space. The strong empirical performance across six datasets, four architectures, and diverse sensitive attributes, at a fraction of the computational cost of existing baselines, suggests that gradient-free, subspace-constrained editing is a principled and practical choice for post-deployment GNN adaptation.

Limitations and Future Works: The current work focuses exclusively on attribute invariance. However, spurious correlations in graph-structured data can also arise from topological properties, such as a model developing undue reliance on node degree, or on connectivity patterns to specific influential nodes. Extending GRAFT to enforce invariance with respect to such structural properties is a natural and important direction that we leave for future work.

Broader Impact: GRAFT is designed to reduce harmful reliance on sensitive attributes in deployed GNNs, with direct positive applications in high-stakes domains such as financial lending, criminal justice, and health systems. However, the same capability could potentially be misused to launch adversarial attacks to selectively remove a model's sensitivity to attributes that are *legitimately* predictive. This, however, requires access to model weights, which can be guarded from attackers.

References

[1] Chirag Agarwal, Himabindu Lakkaraju, and Marinka Zitnik. Towards a unified framework for fair and stable graph representation learning. In *Uncertainty in artificial intelligence*, pages 2114–2124. PMLR, 2021.

[2] Rob Ashmore, Radu Calinescu, and Colin Paterson. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *ACM computing surveys (CSUR)*, 54(5):1–39, 2021.

[3] Guy Bar-Shalom, Beatrice Bevilacqua, and Haggai Maron. Subgraphormer: Unifying subgraph gnns and graph transformers via graph products. In *International Conference on Machine Learning*, pages 2959–2989. PMLR, 2024.

[4] Gleb Bazhenov, Oleg Platonov, and Liudmila Prokhorenkova. Graphland: Evaluating graph machine learning models on diverse industrial data. *Advances in Neural Information Processing Systems*, 38, 2025.

[5] Maya Bechler-Speicher, Yoel Gottlieb, Andrey Isakov, David Abensur, Ami Tavory, Daniel Haimovich, Ido Guy, and Udi Weinsberg. Billion-scale graph foundation models. *arXiv preprint arXiv:2602.04768*, 2026.

[6] Ravinder Bhattoo, Sayan Ranu, and NM Krishnan. Learning articulated rigid body dynamics with lagrangian graph neural network. *Advances in Neural Information Processing Systems*, 35:29789–29800, 2022.

[7] Vaibhav Bihani, Sahil Manchanda, Srikanth Sastry, Sayan Ranu, and NM Krishnan. Stridernet: A graph reinforcement learning approach to optimize atomic structures on rough energy landscapes. In *ICML*, 2023.

[8] Suresh Bishnoi, Ravinder Bhattoo, Jayadeva Jayadeva, Sayan Ranu, and N M Anoop Krishnan. Enhancing the inductive biases of graph neural ODE for modeling physical systems. In *The Eleventh International Conference on Learning Representations*, 2023.

[9] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[10] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3438–3445, 2020.

[11] Enyan Dai and Suhang Wang. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 680–688, 2021.

[12] Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-expressive graph transformer in linear time. In *International Conference on Learning Representations*, volume 2024, pages 18323–18348, 2024.

[13] Keyu Duan, Zirui Liu, Peihao Wang, Wenqing Zheng, Kaixiong Zhou, Tianlong Chen, Xia Hu, and Zhangyang Wang. A comprehensive study on large-scale graph training: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 35:5376–5389, 2022.

[14] Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. In *International Conference on Learning Representations*, volume 2025, pages 16366–16396, 2025.

[15] Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 12043–12051, 2024.

[16] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

[17] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[18] Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: a scalable approach to domain-agnostic labeled graph generation. In *Proceedings of The Web Conference 2020*, pages 1253–1263, 2020.

[19] Mridul Gupta, Hariprasad Kodamana, and Sayan Ranu. FRIGATE: Frugal spatio-temporal forecasting on road networks. In *29th SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

[20] Shubham Gupta, Sahil Manchanda, Srikanta Bedathur, and Sayan Ranu. Tigger: Scalable generative modelling for temporal interaction graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6819–6828, 2022.

[21] Shubham Gupta, Sahil Manchanda, Sayan Ranu, and Srikanta Bedathur. GRAFENNE: Learning on graphs with heterogeneous and dynamic feature sets. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 12165–12181. PMLR, 2023.

[22] Shubham Gupta, Sayan Ranu, and Srikanta Bedathur. NEUTAG: Graph transformer for attributed graphs. *Transactions on Machine Learning Research*, 2026.

[23] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[24] Jayant Jain, Vrittika Bagadia, Sahil Manchanda, and Sayan Ranu. Neuromlr: Robust & reliable route recommendation on road networks. *Advances in Neural Information Processing Systems*, 34:22070–22082, 2021.

[25] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[26] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[27] Zirui Liu, Zhimeng Jiang, Shaochen Zhong, Kaixiong Zhou, Li Li, Rui Chen, Soo-Hyun Choi, and Xia Hu. Editable graph neural network for node classifications. *arXiv preprint arXiv:2305.15529*, 2023.

[28] Sahil Manchanda, Akash Mittal, Anuj Dhawan, Sourav Medya, Sayan Ranu, and Ambuj Singh. Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs. *Advances in Neural Information Processing Systems*, 33:20000–20011, 2020.

[29] Sahil Manchanda and Sayan Ranu. Lifelong learning for neural powered mixed integer programming. *AAAI*, 2023.

[30] Sahil Manchanda and Pannaga Shivaswamy. What is in a name? mitigating name bias in text embedding similarity via anonymization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17759–17781, 2025.

[31] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

[32] Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023.

[33] David S Moore. *The Basic Practice of Statistics*. W.H. Freeman, 1995.

[34] Sunil Nishad, Shubhangi Agarwal, Arnab Bhattacharya, and Sayan Ranu. Graphreach: Position-aware graph neural network using reachability estimations. *IJCAI*, 2021.

[35] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable Graph Transformer. *Advances in Neural Information Processing Systems*, 35, 2022.

[36] Rishabh Ranjan, Siddharth Grover, Sourav Medya, Venkatesan Chakaravarthy, Yogish Sabharwal, and Sayan Ranu. Greed: A neural framework for learning graph distance functions. In *Advances in Neural Information Processing Systems*, 2022.

[37] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 985–994, New York, NY, USA, 2015. Association for Computing Machinery.

[38] Emanuele Rossi, Henry Kenlay, Maria I Gorinova, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. In *Learning on graphs conference*, pages 11–1. PMLR, 2022.

[39] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020.

[40] Rishi Shah, Krishnanshu Jain, Sahil Manchanda, Sourav Medya, and Sayan Ranu. Neurocut: A neural approach for robust graph partitioning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 2584–2595, New York, NY, USA, 2024. Association for Computing Machinery.

[41] Lubos Takac and Michal Zabovsky. Data analysis in public social networks. In *International scientific conference and international workshop present day trends of innovations*, volume 1, 2012.

[42] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. Rethinking graph neural networks for anomaly detection. In *International conference on machine learning*, pages 21076–21089. PMLR, 2022.

[43] Ryutaro Tanno, Melanie F Pradier, Aditya Nori, and Yingzhen Li. Repairing neural networks by leaving the right past behind. *Advances in Neural Information Processing Systems*, 35:13132–13145, 2022.

[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

[45] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Sgformer: Simplifying and empowering transformers for large-graph representations. *Advances in Neural Information Processing Systems*, 36:64753–64773, 2023.

[46] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[47] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[48] Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, 2023.

[49] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[50] Guixian Zhang, Debo Cheng, Guan Yuan, and Shichao Zhang. Learning fair representations via rebalancing graph structure. *Information Processing & Management*, 61(1):103570, 2024.

[51] Xiao-Meng Zhang, Li Liang, Lin Liu, and Ming-Jing Tang. Graph neural networks and their current applications in bioinformatics. *Frontiers in genetics*, 12:690049, 2021.

[52] Zhongjian Zhang, Mengmei Zhang, Yue Yu, Cheng Yang, Jiawei Liu, and Chuan Shi. Endowing pre-trained graph models with provable fairness. In *Proceedings of the ACM Web Conference 2024*, pages 1045–1056, 2024.

[53] Shaochen Zhong, Duy Le, Zirui Liu, Zhimeng Jiang, Andrew Ye, Jiamu Zhang, Jiayi Yuan, Kaixiong Zhou, Zhaozhuo Xu, Jing Ma, et al. Gnns also deserve editing, and they need it more than once. In *Forty-first International Conference on Machine Learning*, 2024.

A Appendix

A.1 Architecture-Specific Model-Aware PageRank Weighting

To ensure the structural selection of representative nodes aligns with the message-passing dynamics of the specific GNN backbone, we define the transition matrix P using architecture-specific edge weights w_{ij}. This approximates how information propagates through the model during inference.

- **Graph Convolutional Networks (GCN):** Weights follow the standard symmetric normalized adjacency logic. Self-loops are added to the induced subgraph, and weights are calculated as $w_{ij} \propto d_i^{-1/2} d_j^{-1/2}$ using degrees strictly from that local subgraph.
- **Graph Attention Networks (GAT):** We recover hidden activations U_{final}. Attention coefficients α are computed on the full graph with. Multi-head attention is mean-pooled and restricted to subgraph edges via with the same boolean mask as the induced subgraph. Finally, mass leaving each source is scaled by $1/\text{deg}_{out}(\text{row})$.
- **Graph Isomorphism Networks (GIN):** These architectures utilize uniform outgoing weights $w_{ij} = 1/\text{deg}_{out}(\text{row})$ on the induced subgraph. This approximates the uniform neighbors weighted like mean aggregation over outgoing edges on the subgraph.

A.2 Theoretical Guarantees

We below establish the theoretical properties of the subspace-constrained update used by GRAFT. We show that the layerwise optimization problem has a unique solution, that the edit acts only on the sensitive subspace, and that the residual attribute sensitivity is computable before deployment.

Setup. Let $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ be the weight matrix of a pretrained layer, and let

$$\mathbf{W}' = \mathbf{W} + \mathbf{S}_{\text{sub}}$$

be the edited weight matrix. Let $\mathbf{B}_r \in \mathbb{R}^{d_{\text{in}} \times r}$ contain an orthonormal basis for the sensitive subspace. We parameterize the weight shift as

$$\mathbf{S}_{\text{sub}} = (\mathbf{B}_r \mathbf{Z})^\top = \mathbf{Z}^\top \mathbf{B}_r^\top, \qquad \mathbf{Z} \in \mathbb{R}^{r \times d_{\text{out}}}.$$

For any activation $\mathbf{u}$ orthogonal to the sensitive subspace, $\mathbf{B}_r^\top \mathbf{u} = \mathbf{0}$, and therefore

$$\mathbf{S}_{\text{sub}} \mathbf{u} = \mathbf{Z}^\top \mathbf{B}_r^\top \mathbf{u} = \mathbf{0}.$$

Thus, the update leaves unaffected directions unchanged.
Let $\boldsymbol{\delta}_u \in \mathbb{R}^{Kb \times d_{\text{in}}}$ denote the stacked input perturbations induced by the target attribute, and let $\boldsymbol{\delta}_v \in \mathbb{R}^{Kb \times d_{\text{out}}}$ denote the corresponding desired output corrections. Define

$$\mathbf{X}_{\text{inv}} = \boldsymbol{\delta}_u \mathbf{B}_r \in \mathbb{R}^{Kb \times r}, \qquad \mathbf{X}_{\text{ret}} = \overline{\mathbf{U}} \mathbf{B}_r \in \mathbb{R}^{b \times r},$$

where

$$\overline{\mathbf{U}} = [\mathbf{u}_1, \ldots, \mathbf{u}_b]^\top$$

contains the stacked original node activations. Substitution of $\mathbf{S}_{\text{sub}} = \mathbf{Z}^\top \mathbf{B}_r^\top$ reduces the optimization from the full weight space to an $r \times d_{\text{out}}$ problem:

$$\min_{\mathbf{Z}} \ \|\mathbf{X}_{\text{inv}} \mathbf{Z} - \boldsymbol{\delta}_v\|_F^2 + \gamma \, \|\mathbf{X}_{\text{ret}} \mathbf{Z}\|_F^2 + \lambda \|\mathbf{Z}\|_F^2, \qquad \gamma, \lambda > 0. \tag{19}$$

After solving for $\mathbf{Z}$, the update is lifted as

$$\mathbf{S}_{\text{sub}} = (\mathbf{B}_r \mathbf{Z})^\top, \qquad \mathbf{W}' = \mathbf{W} + \mathbf{S}_{\text{sub}}.$$

Lemma 1 (Uniqueness of the optimal subspace update) *For $\gamma, \lambda > 0$, the objective in Eq. (19) is strictly convex and has the unique minimizer*

$$\mathbf{Z}^* = \left(\mathbf{X}_{\text{inv}}^\top \mathbf{X}_{\text{inv}} + \gamma \mathbf{X}_{\text{ret}}^\top \mathbf{X}_{\text{ret}} + \lambda \mathbf{I}_r \right)^{-1} \mathbf{X}_{\text{inv}}^\top \boldsymbol{\delta}_v.$$

The gradient is

$$\nabla_{\mathbf{Z}} F = 2\mathbf{X}_{\text{inv}}^\top (\mathbf{X}_{\text{inv}} \mathbf{Z} - \boldsymbol{\delta}_v) + 2\gamma \mathbf{X}_{\text{ret}}^\top \mathbf{X}_{\text{ret}} \mathbf{Z} + 2\lambda \mathbf{Z}.$$

The Hessian with respect to $\text{vec}(\mathbf{Z})$ is

$$\mathbf{H} = 2\mathbf{I}_{d_{\text{out}}} \otimes \left(\mathbf{X}_{\text{inv}}^\top \mathbf{X}_{\text{inv}} + \gamma \mathbf{X}_{\text{ret}}^\top \mathbf{X}_{\text{ret}} + \lambda \mathbf{I}_r \right) \succ \mathbf{0},$$

because $\lambda > 0$. Hence, F is strictly convex. Setting the gradient to zero gives

$$\left(\mathbf{X}_{\text{inv}}^\top \mathbf{X}_{\text{inv}} + \gamma \mathbf{X}_{\text{ret}}^\top \mathbf{X}_{\text{ret}} + \lambda \mathbf{I}_r \right) \mathbf{Z} = \mathbf{X}_{\text{inv}}^\top \boldsymbol{\delta}_v,$$

which yields the stated unique minimizer.

Lemma 2 (Exact preservation outside the sensitive subspace) *For any activation $\mathbf{u}$ satisfying $\mathbf{B}_r^\top \mathbf{u} = \mathbf{0}$,*

$$(\mathbf{W} + \mathbf{S}_{\text{sub}}) \mathbf{u} = \mathbf{W} \mathbf{u}.$$

Using the optimal update,

$$\mathbf{S}_{\text{sub}}\mathbf{u} = (\mathbf{Z}^*)^\top \mathbf{B}_r^\top \mathbf{u} = \mathbf{0}.$$

Therefore,

$$(\mathbf{W} + \mathbf{S}_{\text{sub}})\mathbf{u} = \mathbf{W}\mathbf{u}.$$

Lemma 3 (Computable residual sensitivity) *The residual attribute sensitivity after editing is*

$$R^* = \|\mathbf{X}_{\text{inv}}\mathbf{Z}^* - \boldsymbol{\delta}_v\|_F^2,$$

and can be computed before deployment.

By Lemma 1, $\mathbf{Z}^*$ is available in closed form. Substituting it into the expression for R^* computes the residual using only quantities obtained during the editing procedure.
Together, these results show that GRAFT computes a unique regularized update within the selected sensitive subspace, preserves directions orthogonal to that subspace, and provides an a-priori measure of the residual attribute sensitivity.

A.3 Experimental Details

Train/Val/Test Splits: We use standard stratified splits across all datasets. Bail, Artnet-views, and Twitch-views follow a 50:25:25 split, YelpChi uses a 70:10:20 split, T-Finance uses a 40:20:40 split. For Pokec-n, we adopt a (50:30:20) stratified split.

Parameters: By default, we standardize the configurations for GCN, GAT, and GIN backbones to utilize 2 message passing layers with a hidden embedding dimension of 32 across all classification datasets and 512 across regression datasets. For *Polynormer*, we use the default architecture consisting of 7 local attention layers and 2 global attention layers with hidden dimension 32, dropout rates $(0.15, 0.5, 0.5)$ for input, local, and global attention respectively, and aggregation coefficient $\beta = 0.9$.

For GRAFT, we maintain consistent hyperparameters across all datasets: the regularization weight is fixed at $\lambda = 0.1$, the subspace retention parameter at $\gamma = 1.0$ for classification datasets and $\gamma = 10.0$ for regression datasets. We set the rank r in SVD to be the minimum rank required to capture 95% of the variance. The number of perturbation samples is fixed at $K = 6$.

For baselines EGNN and SEED-GNN, we use the default settings provided by the original authors.

Adapting baselines to our task: For our experimental setup, the baseline methods (finetuning, EGNN, SEED-GNN) were trained with the objective of steering perturbed inputs to match the outputs of the original unperturbed inputs. Additionally, the neighbourhood sampling procedure in SEEDGNN was constrained to be exclusively within the validation set.

A.4 Complete Editing Results for GRAFT and baselines

Dataset	Model	Base	Oracle	Strategy	$\Delta R^2 \downarrow$	Variance Drop (%) ↑	Runtime ↓ (s)
Node Regression							
Artnet-views	GCN	0.477 ± 0.001	0.473 ± 0.001	Finetune	0.103 ± 0.018	0.8 ± 19.7	2.19 ± 0.09
				GRAFT (ours)	0.046 ± 0.005	57.8 ± 1.7	3.88 ± 0.03
	Polynormer	0.503 ± 0.026	0.511 ± 0.026	Finetune	0.075 ± 0.044	86.3 ± 3.8	3.14 ± 0.02
				GRAFT (ours)	-0.028 ± 0.018	54.8 ± 4.6	7.96 ± 0.15
Twitch-views	GCN	0.545 ± 0.001	0.531 ± 0.001	Finetune	0.010 ± 0.000	8.2 ± 4.3	333.92 ± 10.44
				GRAFT (ours)	0.006 ± 0.001	13.3 ± 11.7	12.86 ± 0.29
	Polynormer	0.708 ± 0.029	0.591 ± 0.130	Finetune	0.146 ± 0.039	71.8 ± 20.6	26.84 ± 0.04
				GRAFT (ours)	0.030 ± 0.062	40.5 ± 11.1	47.86 ± 3.35

Table 4: Editing performance across node regression datasets. Values are reported as mean ± standard deviation across 3 seeds. Variance Drop (%) indicates variance reduction (higher is better). Oracle is the utility of the model trained after dropping the focused attribute.

Dataset	Model	Base	Gold	Strategy	ΔAUC-PR ↓	Variance Drop (%) ↑	Runtime ↓ (s)
				Node Classification			
Bail	GAT	0.953 ± 0.020	0.945 ± 0.014	Finetune	0.460 ± 0.160	100.0 ± 0.0	3.27 ± 0.09
				EGNN	0.116 ± 0.406	$< -10^6$	3370.26 ± 75.10
				SEEDGNN	-0.301 ± 0.227	$< -10^6$	1881.89 ± 241.60
				GRAFT (ours)	0.000 ± 0.000	67.0 ± 0.3	1.74 ± 0.15
	GCN	0.918 ± 0.001	0.925 ± 0.006	Finetune	0.385 ± 0.174	-6597.1 ± 9471.1	1.22 ± 0.06
				EGNN	0.478 ± 0.015	52.5 ± 16.2	1709.96 ± 243.05
				SEEDGNN	-0.185 ± 0.009	-4078.7 ± 5460.7	1973.48 ± 108.84
				GRAFT (ours)	-0.000 ± 0.000	67.0 ± 0.5	1.49 ± 0.32
	GIN	0.963 ± 0.004	0.966 ± 0.008	Finetune	0.483 ± 0.183	92.1 ± 6.7	1.34 ± 0.03
				EGNN	0.078 ± 0.213	55.6 ± 38.5	1997.39 ± 143.88
				SEEDGNN	-0.211 ± 0.077	-511.8 ± 348.9	2094.48 ± 37.97
				GRAFT (ours)	0.000 ± 0.000	68.1 ± 1.2	1.81 ± 0.09
	Polynormer	0.997 ± 0.000	0.996 ± 0.000	Finetune	0.131 ± 0.041	-126.6 ± 316.1	2.68 ± 0.01
				EGNN	0.648 ± 0.115	67.7 ± 40.1	81.08 ± 13.61
				SEEDGNN	0.003 ± 0.001	-3364.4 ± 4614.8	164.71 ± 6.47
				GRAFT (ours)	0.000 ± 0.000	78.8 ± 14.5	4.44 ± 0.07
Pokec-n	GAT	0.761 ± 0.010	0.749 ± 0.007	Finetune	0.273 ± 0.061	100.0 ± 0.0	1.44 ± 0.32
				EGNN	0.046 ± 0.025	42.4 ± 61.2	718.27 ± 111.06
				SEEDGNN	0.085 ± 0.009	-14.2 ± 18.5	1045.49 ± 148.80
				GRAFT (ours)	0.030 ± 0.004	64.5 ± 14.2	1.22 ± 0.08
	GCN	0.780 ± 0.009	0.775 ± 0.009	Finetune	0.292 ± 0.006	100.0 ± 0.0	0.57 ± 0.02
				EGNN	0.112 ± 0.083	67.6 ± 39.5	766.59 ± 134.51
				SEEDGNN	0.090 ± 0.044	-229.7 ± 246.0	814.60 ± 200.63
				GRAFT (ours)	0.025 ± 0.001	72.9 ± 9.1	4.33 ± 1.83
	GIN	0.737 ± 0.022	0.779 ± 0.014	Finetune	0.172 ± 0.098	95.2 ± 1.1	0.88 ± 0.02
				EGNN	0.063 ± 0.046	39.7 ± 56.5	956.56 ± 41.53
				SEEDGNN	0.094 ± 0.027	39.4 ± 11.8	861.99 ± 73.67
				GRAFT (ours)	0.052 ± 0.040	72.2 ± 9.8	1.89 ± 1.15
	Polynormer	0.784 ± 0.013	0.773 ± 0.008	Finetune	0.369 ± 0.019	100.0 ± 0.0	1.89 ± 0.01
				EGNN	0.389 ± 0.020	99.7 ± 0.2	904.42 ± 83.47
				SEEDGNN	0.153 ± 0.036	-208.9 ± 82.2	2001.14 ± 289.95
				GRAFT (ours)	0.032 ± 0.009	96.0 ± 0.6	6.41 ± 2.64
YelpChi	GAT	0.194 ± 0.018	0.203 ± 0.008	Finetune	0.050 ± 0.009	100.0 ± 0.0	20.18 ± 0.03
				EGNN	0.059 ± 0.032	87.2 ± 12.7	2443.57 ± 106.33
				SEEDGNN	-0.071 ± 0.026	-4212.9 ± 1915.9	19375.70 ± 112.70
				GRAFT (ours)	0.002 ± 0.001	32.5 ± 8.5	3.56 ± 0.11
	GCN	0.178 ± 0.016	0.164 ± 0.002	Finetune	0.009 ± 0.015	100.0 ± 0.0	1.39 ± 0.01
				EGNN	0.066 ± 0.013	100.0 ± 0.0	2645.32 ± 107.18
				SEEDGNN	-0.116 ± 0.012	$< -10^4$	3038.57 ± 100.10
				GRAFT (ours)	0.000 ± 0.000	37.0 ± 10.2	1.65 ± 0.03
	GIN	0.195 ± 0.042	0.190 ± 0.010	Finetune	0.039 ± 0.040	98.1 ± 1.8	1.62 ± 0.00
				EGNN	0.054 ± 0.038	100.0 ± 0.0	2606.60 ± 16.49
				SEEDGNN	-0.123 ± 0.055	$< -10^4$	3299.04 ± 27.60
				GRAFT (ours)	0.024 ± 0.024	63.2 ± 18.3	2.09 ± 0.15
	Polynormer	0.584 ± 0.013	0.522 ± 0.004	Finetune	0.411 ± 0.024	100.0 ± 0.0	13.39 ± 0.01
				EGNN	0.435 ± 0.014	99.9 ± 0.1	3459.25 ± 264.95
				SEEDGNN	0.146 ± 0.046	-53.9 ± 69.3	9536.33 ± 668.83
				GRAFT (ours)	0.130 ± 0.037	94.6 ± 1.4	20.15 ± 0.05
T-Finance	GAT	0.085 ± 0.059	0.115 ± 0.099	Finetune	0.107 ± 0.150	$< -10^6$	6.30 ± 0.10
				EGNN	-0.121 ± 0.174	-48.3 ± 103.7	4197.42 ± 2331.58
				SEEDGNN	-0.172 ± 0.230	-397.3 ± 499.4	10457.73 ± 9497.89
				GRAFT (ours)	-0.002 ± 0.002	14.7 ± 20.6	24.84 ± 2.02
	GCN	0.719 ± 0.026	-	Finetune	0.672 ± 0.025	67.3 ± 46.3	18.08 ± 16.31
				EGNN	0.076 ± 0.304	$< -10^4$	96628.43 ± 160067.44
				SEEDGNN	-0.287 ± 0.128	$< -10^4$	12426.35 ± 4659.60
				GRAFT (ours)	0.000 ± 0.000	49.7 ± 8.6	5.46 ± 0.03
	GIN	0.487 ± 0.237	0.445 ± 0.284	Finetune	0.309 ± 0.436	$< -10^6$	13.40 ± 12.35
				EGNN	0.139 ± 0.270	-3353.4 ± 4231.1	7203.70 ± 5511.78
				SEEDGNN	-0.675 ± 0.051	$< -10^6$	9953.57 ± 310.24
				GRAFT (ours)	0.000 ± 0.000	54.8 ± 13.8	5.62 ± 0.02
	Polynormer	0.906 ± 0.004	0.905 ± 0.003	Finetune	0.598 ± 0.395	84.4 ± 22.1	65.60 ± 0.02
				EGNN	0.833 ± 0.071	90.5 ± 5.4	1489.81 ± 161.79
				SEEDGNN	0.149 ± 0.020	-1864.2 ± 910.6	13046.24 ± 5940.72
				GRAFT (ours)	0.000 ± 0.000	62.0 ± 15.2	4903.01 ± 85.22

Table 5: Editing performance across node classification datasets. Values are reported as mean ± standard deviation across 3 seeds. Variance Drop (%) indicates variance reduction (higher is better). Oracle is the utility of the model trained after dropping the focused attribute.

A.5 Ablation Studies and Hyperparameter Sensitivity

A.5.1 Subspace retention parameter (γ)

The parameter γ (`gamma_retain`) governs the trade-off between invariance and utility.

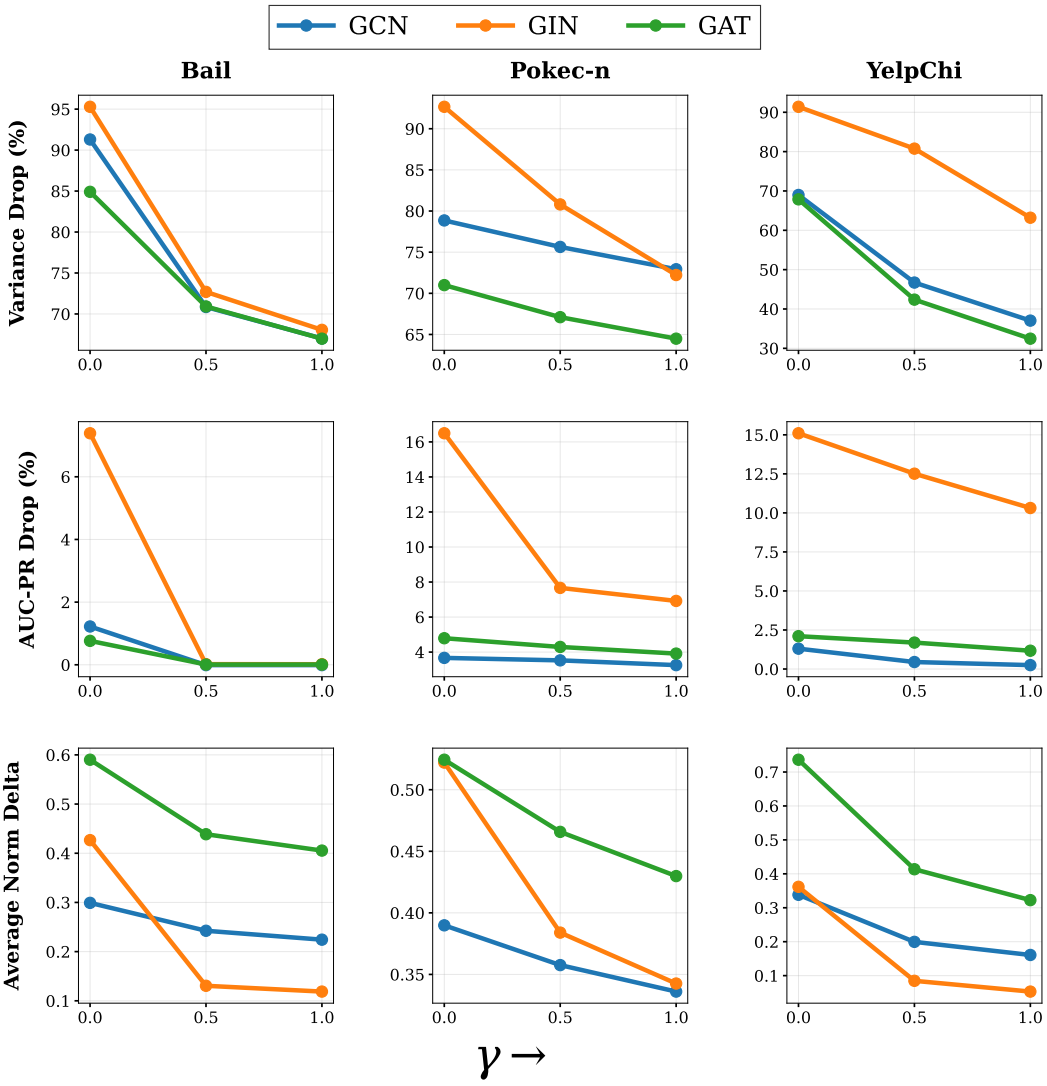


Figure 4: Effect of the subspace retention parameter γ on variance reduction, AUCPR drop, and average weight-update norm across the Bail, Pokec-n, and YelpChi datasets for GCN, GIN, and GAT models. Increasing γ enforces stronger representation retention during editing, resulting in smaller model weight updates and reduced edit magnitude across architectures and datasets.

Impact of γ on Model Utility and Attribute Suppression. The hyperparameter γ governs the trade-off between information erasure and functional stability.

- **Aggressive Removal ($\gamma \to 0$):** Results in the highest *Variance Drop*, effectively scrubbing target attributes but risking catastrophic utility loss (maximized *AUC-PR Drop*).
- **Preservation of Integrity ($\gamma \geq 1.0$):** Minimizes the *AUC-PR Drop* by orthogonalizing the update relative to the original activations U_{orig}. This allows for precise information excision while maintaining the model's original predictive capabilities.

- **Geometric Stability:** The reduction in *Average Norm Delta* as γ increases indicates that the anchored edit produces representations that remain closer to the original embedding space across GCN, GIN, and GAT architectures.

A.5.2 Regularization and Stability (λ)

Sensitivity Analysis of the Regularization Parameter λ: The parameter λ serves as a constraint on the magnitude of the modification applied to the model.

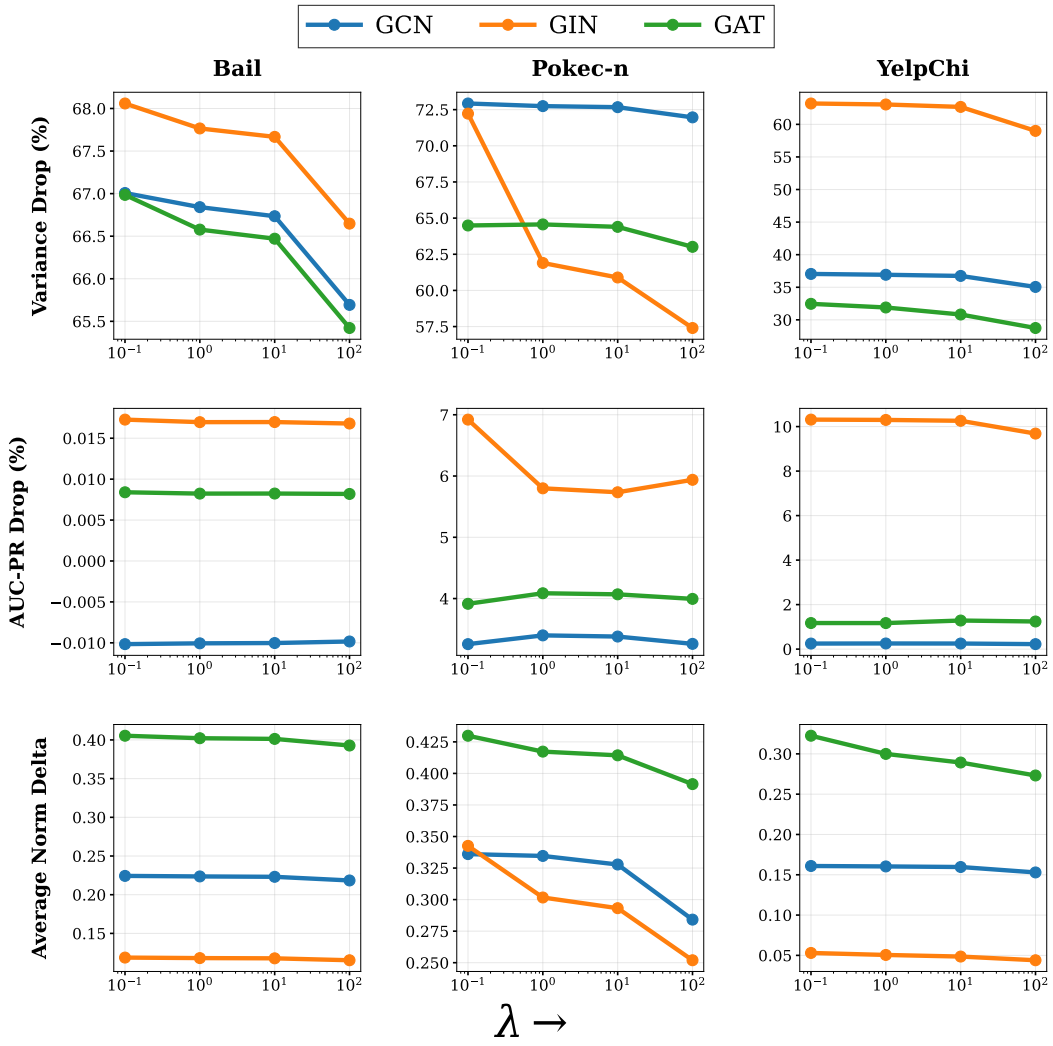


Figure 5: Effect of the regularization parameter λ on variance reduction, AUCPR drop, and average weight-update norm across the Bail, Pokec-n, and YelpChi datasets for GCN, GIN, and GAT models. Increasing λ imposes stronger constraints on the editing update, leading to smaller parameter modifications and more conservative edits while preserving greater representation retention across architectures and datasets.

Sensitivity Analysis of the Regularization Parameter λ. The parameter λ acts as a penalty on the magnitude of the model update, governing the balance between parameter stability and editing effectiveness.

- **Weight and Representation Stability:** Increasing λ leads to a monotonic decrease in the *Average Norm Delta*. This confirms that λ successfully penalizes the weight shift ($\|\Delta W\|/\|W\|$), forcing the edited model to remain geometrically close to the original.

- **Constraint on Attribute Suppression:** As λ increases, the penalty on the weight shift becomes more restrictive, which in turn reduces the *Variance Drop*. This indicates that over-regularization can hinder the model's ability to effectively scrub sensitive information.

A.6 Comparison with fairness-based baselines

We compare GRAFT against fairness-aware baselines, including FAIRGNN [11], NIFTY [1], and GRAPHPAR [52]. As shown in Fig. 6a, GRAFT achieves a more favorable utility–invariance trade-off, consistently lying closer to the oracle while preserving predictive utility. In addition, Fig. 6b shows that GRAFT is substantially more efficient than these baselines. This improvement is attributed to the closed-form, post-hoc edit, which directly removes attribute dependence from the deployed model without introducing additional parameters or iterative tuning.

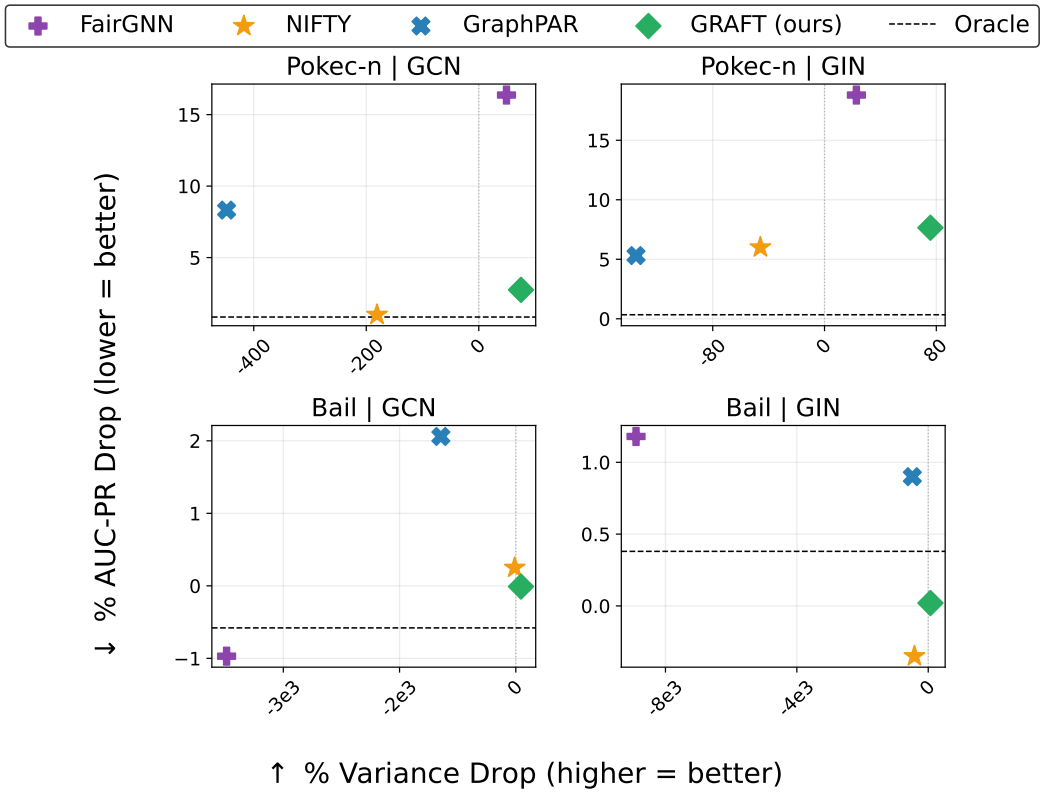
A.7 Ablation Study

Table 6: Ablation study: Starting from the full method (**GRAFT**), we progressively remove the subspace-constraint (§ 3.2.1) and topology-aware selection (§ 3.1) components to measure their contribution to the invariance–utility trade-off, measured through the Variance-to-Utility Ratio (VUR). Higher VUR is better.

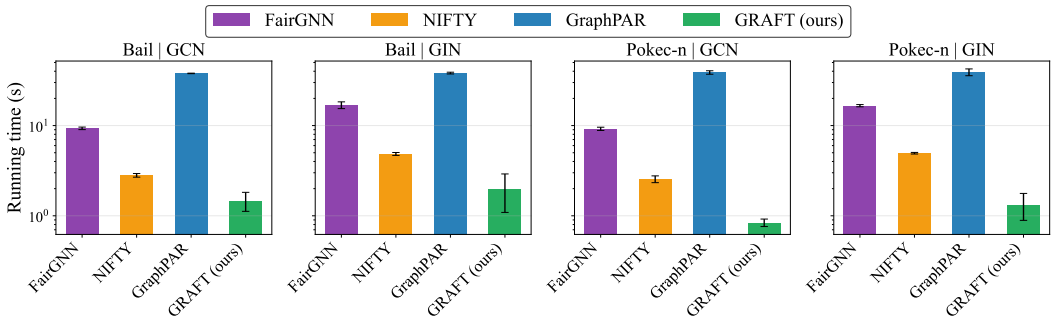
	Bail			**Pokec-n**			**YelpChi**		
Method	**GCN**	**GIN**	**GAT**	**GCN**	**GIN**	**GAT**	**GCN**	**GIN**	**GAT**
GRAFT	**1.01e6**	**2.13e5**	**4.91e6**	**2.24e3**	**3.22e3**	**1.68e3**	**3.16e4**	**4.93e4**	**4.07e3**
w/o Subspace	3.09e4	3.42e3	9.80e4	1.76e3	4.71e2	1.23e3	2.16e4	1.64e4	4.02e3
w/o Topology	2.64e4	2.92e3	2.26e4	1.63e3	4.97e2	8.45e2	1.75e4	3.07e3	3.96e3

A.8 System Configuration

We use a system running on Intel Xeon 6248 processor with 96 cores and 1 NVIDIA A100 GPU with 40GB memory for our experiments. The software environment utilized is Python 3.10, PyTorch 2.1, and PyTorch-Geometric 2.4. All reported metrics are averaged over three seeds (0, 10, 42).



(a) **Comparison with Fairness based methods : Utility vs. Invariance.** We visualize the trade-off between model utility (AUC-PR Drop) and attribute invariance (Variance Drop) for node classification.



(b) Running time comparison against fairness-based methods.

Figure 6: **Comparison with Fairness based methods :** Fig. 6a shows the utility-vs-invariance trade-off and Fig 6b shows running time of different methods.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction accurately summarize the proposed feature-invariant model editing framework, including its theoretical formulation and empirical evaluation across multiple graph learning benchmarks. The stated claims regarding reduced attribute dependence,

preservation of predictive utility, and improved stability are supported by the experimental and ablation results presented in the paper.

Guidelines:

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

2. **Limitations**

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

Answer: [Yes]

Justification: The paper explicitly discusses limitations in a dedicated "Limitations and Future Works" section. In particular, it notes that the current method focuses only on attribute invariance and does not address structural sources of spurious correlations, such as node degree or graph connectivity patterns, which are identified as important directions for future work.

Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: Yes, we have supported each theoretical statement in our method with a complete and correct derivation, stating all details. Particularly we have made sure that our method is completely and correctly theoretically backed.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.

- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Yes, we have disclosed all the experimental setups, hyperparameter configurations, random seeds, and system details for ensuring reproducibility.

   Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The paper provides access to the implementation code, training configurations, and experimental details necessary to reproduce the reported results. Dataset splits, model hyperparameters, optimization settings, and evaluation protocols are described in the main paper and appendix to support faithful reproducibility of the experiments.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper specifies the dataset splits, backbone architectures, hidden dimensions, optimization settings, perturbation parameters, and model editing hyperparameters used in all experiments. Additional implementation details, including the feature masking evaluation protocol and reproducibility settings, are provided in the appendix to ensure the experimental setup is fully understandable and reproducible.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: The paper reports the mean and standard deviation across multiple independent runs for the primary experimental results. All ablations and other claims are made for statistically significant sample size, over multiple datasets and models, enabling assessment of the stability and statistical reliability of the reported performance improvements.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper specifies the compute environment used for the experiments, including an Intel Xeon 6248 CPU with 96 cores, a 40GB NVIDIA A100 GPU, and the software stack consisting of Python 3.10, PyTorch 2.1, and PyTorch-Geometric 2.4. The paper also reports that all experimental results are averaged across three random seeds (0, 10, 42) to ensure robustness and reproducibility.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research complies with the NeurIPS Code of Ethics. The work focuses on improving robustness to sensitive attributes in graph neural networks, uses publicly available benchmark datasets, and does not involve harmful interventions, deceptive practices, or collection of private user data beyond standard research benchmarks.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The paper includes a dedicated "Broader Impact" discussion that addresses both the positive and negative societal implications of the proposed method. Specifically, it highlights beneficial applications in fairness-sensitive domains such as lending, criminal justice, and healthcare, while also acknowledging the potential misuse of the editing framework to suppress legitimately predictive attributes in adversarial settings.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: We use datasets available publicly. Our models do not have any high-risk usage.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The paper cites and credits the original sources of all datasets, backbone architectures, and baseline methods used in the experiments. Publicly available assets are used in accordance with their respective licenses and standard academic research usage policies.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: Our codebases are accompanied with a detailed README.md with detailed documentation.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [N/A]

    Justification: Our paper did not involve crowdsourcing or human subjects.

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [N/A]

    Justification: Our paper did not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A].

Justification: LLM used only to improve articulation.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.