Differentiable Influence Minimization via Continuous Relaxation

Mihil Sreenilayam

December 2025

AI Disclosure

Generative AI tools (including large language models) were used in this project to assist with code structuring, debugging, documentation drafting, and conceptual clarification. All conceptual work, experimental design, implementation decisions, and analyses were performed by the author. All generated material was reviewed, verified, and edited to ensure correctness and academic integrity.

1 Introduction

The diffusion of information, behaviors, and pathogens across complex networks is a fundamental process underlying many real-world phenomena. From the viral spread of misinformation on social media platforms to the propagation of infectious diseases through contact networks, understanding and controlling these diffusion dynamics is a central challenge in network science. While much of the early literature focused on *Influence Maximization* (IM)—identifying a set of seed nodes to maximize spread, as motivated by viral marketing [YCH22]—recent events have shifted the focus toward its inverse: *Influence Minimization* (IMIN).

Influence Minimization is the task of dampening the spread of an undesirable diffusion process by intervening on the network structure. This intervention typically takes the form of blocking nodes (e.g., vaccination) or removing edges (e.g., social distancing, quarantine, or censoring misinformation channels). In this work, we focus on the edge-removal variant, often formalized as the *Critical Edge Detection* problem: given a budget k, which k edges should be removed to maximally reduce the expected spread of a contagion?

1.1 Computational Challenges

Despite its simple formulation, Influence Minimization is computationally intractable for two primary reasons. First, the underlying diffusion models, such as the Independent Cascade (IC) or Linear Threshold (LT) models, are stochastic. Calculating the exact expected influence $\sigma(G)$ for a single graph instance is #P-hard [KKT03, GLL11], as it essentially requires summing over an exponential number of possible "live-edge" realizations. Consequently, standard approaches rely on Monte Carlo (MC) simulations to estimate influence. While accurate, MC simulation is computationally expensive, requiring thousands of repetitions to achieve statistical convergence.

Second, the optimization problem itself—selecting a subset of edges to remove—is combinatorial. For a graph with $|E|$ edges and a budget k, the search space size is $\binom{|E|}{k}$, prohibiting exhaustive search. Unlike Influence Maximization, where the objective function is monotone and submodular (allowing greedy algorithms to achieve a $(1 - 1/e)$-approximation), the Influence Minimization objective often lacks these nice properties depending on the specific formulation [DKZ+18]. Consequently, classical methods rely on heuristics (e.g., removing edges with high centrality or degree) or other non-exact techniques. While computationally efficient, these heuristics often fail to capture the complex, higher-order structural motifs that facilitate diffusion, leading to suboptimal intervention strategies.

1.2 The Shift to Neural Combinatorial Optimization

In recent years, the intersection of graph theory and deep learning has given rise to *Neural Combinatorial Optimization*. This paradigm seeks to learn heuristics from data rather than hand-designing them. Graph Neural Networks (GNNs) have emerged as a powerful tool in this domain, capable of learning low-dimensional representations of nodes and graphs that encode structural properties relevant to diffusion [KW17a].

However, applying deep learning to Influence Minimization presents a unique challenge: the discrete nature of the graph structure. Standard backpropagation requires differentiable operations, but the act of removing an edge is a non-differentiable, discrete operation. Furthermore, the objective function—the influence spread—is a "black box" output of a stochastic simulator, providing no direct gradients to update the network parameters.

1.3 Present Work and Contributions

This paper builds directly upon the *DiffIM* framework recently proposed by Lee et al. [LKB+25], which introduced the concept of minimizing influence via surrogate modeling and continuous relaxation. While *DiffIM* established the theoretical pipeline, our work focuses on the operationalization, empirical validation, and stress-testing of this framework under constraints not fully explored in the original text.

We do not claim to invent the differentiable relaxation of the Independent Cascade model. Rather, we provide a concrete, reproducible implementation of the *DiffIM* pipeline and extend the original analysis in three key directions. First, we open the "black box" of the surrogate model, explicitly analyzing the impact of node-level structural features (degree, clustering coefficient, neighbor degree) on the learnability of influence functions across synthetic topologies. Second, we systematically operationalize the sparsity-influence trade-off. By performing a granular sweep over the regularization parameter λ, we generate empirical Pareto curves that quantify the marginal utility of edge retention—an analysis that offers practical guidance for budget-constrained interventions. Finally, we evaluate the framework's robustness to domain shift, training on synthetic graphs (Erdős–Rényi, Barabási–Albert) and testing on real-world social subgraphs, highlighting critical challenges in generalization.

Our specific contributions are:

1. **Reproducible Implementation:** We provide a transparent implementation of the continuous relaxation pipeline using PyTorch Geometric, validating the feasibility of gradient-based topology optimization.

2. **Surrogate Analysis:** We demonstrate that a lightweight GCN surrogate can achieve high correlation ($r > 0.95$) with expensive Monte Carlo simulations on synthetic graphs, validating

the core premise of the differentiable approach.

3. **Trade-off Characterization:** We empirically map the sparsity-influence frontier, demonstrating that gradient-based relaxation consistently outperforms random baselines by identifying structural bottlenecks (bridges and hubs).

4. **Generalization Auditing:** We reveal the limitations of synthetic-to-real transfer, showing that surrogates trained on idealized topologies struggle to predict influence in the highly modular structure of real-world datasets.

The remainder of this paper is organized as follows: Section 2 details the Independent Cascade model and our proposed differentiable framework. Section 3 describes the experimental setup, dataset generation, and training protocols. Section 4 presents the results and a discussion of the sparsity-influence trade-off, and Section 5 concludes with directions for future work.

2 Methodology

Our approach to differentiable influence minimization consists of two distinct phases: (1) training a neural surrogate to approximate the stochastic diffusion process, and (2) optimizing the graph structure via continuous relaxation of edge weights.

2.1 Problem Formulation: Independent Cascade

We model the diffusion process using the *Independent Cascade* (IC) model on a graph $G = (V, E)$. Let $A \in \{0,1\}^{N \times N}$ be the adjacency matrix where $A_{ij} = 1$ if an edge exists between node i and j.

The process begins with a set of seed nodes $S \subseteq V$. At time $t = 0$, only nodes in S are active. At any time step $t \geq 1$, a node u that became active at $t - 1$ attempts to activate its inactive neighbor v with probability p_{uv}. If successful, v becomes active at time t. This attempt occurs exactly once per edge. The process terminates when no new activations occur.

Let $\phi(S, G)$ denote the final number of active nodes (the spread). The expected spread, or influence, is $\sigma(G) = \mathbb{E}[\phi(S, G)]$, where the expectation is taken over the stochastic outcomes of edge activations. Our goal is to find a perturbed graph G' with a reduced set of edges $E' \subset E$ such that $\sigma(G')$ is minimized, subject to a constraint on the number of removed edges.

2.2 Phase 1: GNN Surrogate Modeling

Since $\sigma(G)$ is non-differentiable and computationally expensive ($\#P$-hard), we approximate it with a parameterized neural network $f_\theta(G)$. The surrogate model aims to map a graph structure and a seed set to a predicted influence score $\hat{y} \in [0, 1]$, representing the normalized fraction of activated nodes.

2.2.1 Feature Engineering

To enable the GNN to generalize across different graph sizes and topologies, we use strictly structural node features rather than node identities. For each node v, we extract a feature vector $x_v \in \mathbb{R}^3$:

$$x_v = [\deg(v),\ \text{clust}(v),\ \text{avg\_neigh\_deg}(v)] \tag{1}$$

where $\deg(v)$ is the node degree, $\text{clust}(v)$ is the local clustering coefficient, and the third term is the average degree of v's neighbors. These features are z-score normalized across the graph instance to ensure scale invariance.

2.2.2 Surrogate Architecture

2.3 Phase 1: GNN Surrogate Modeling

Since $\sigma(G)$ is non-differentiable and computationally expensive ($\#P$-hard), we approximate it with a parameterized neural network $f_\theta(G)$. The surrogate model aims to map a graph structure and a seed set to a predicted influence score $\hat{y} \in [0, 1]$.

2.3.1 Surrogate Architecture

We employ a standard Graph Convolutional Network (GCN) backbone as proposed by Kipf and Welling [KW17b]. The node embeddings are updated through layers of message passing via the spectral propagation rule:

$$H^{(l+1)} = \text{ReLU}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{2}$$

where $\tilde{A} = A + I$ is the adjacency matrix with self-loops, $\tilde{D}$ is the degree matrix of $\tilde{A}$, and $W^{(l)}$ are trainable weights.

Our specific architecture consists of:

1. **Encoder:** Two GCNConv layers with hidden dimension $d = 64$ and ReLU activations.
2. **Pooling:** A global mean pooling operation that aggregates node embeddings into a single graph embedding vector $h_G = \frac{1}{|V|}\sum_{v\in V} h_v^{(L)}$.
3. **Readout:** A multi-layer perceptron (MLP) that projects h_G to a scalar prediction $\hat{y}$. A Sigmoid activation ensures the output lies in $[0, 1]$.

The model is trained via Mean Squared Error (MSE) loss against ground-truth influence labels generated by Monte Carlo simulations (averaged over 500 runs per graph).

2.4 Phase 2: Continuous Relaxation and Optimization

To optimize the edge set E, we relax the discrete binary adjacency matrix A into a continuous weighted adjacency matrix $\mathcal{W} \in [0, 1]^{N\times N}$. Each potential edge (u, v) is assigned a learnable parameter w_{uv}, initialized to 1.0.

2.4.1 Differentiable Forward Pass

During the optimization phase, we freeze the parameters θ of the surrogate model f_θ. We effectively treat the GNN as a differentiable function of the edge weights. The GCN propagation rule is modified to respect these continuous weights:

$$H^{(l+1)} = \text{ReLU}\left(D_{\mathcal{W}}^{-\frac{1}{2}}\mathcal{W}D_{\mathcal{W}}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{3}$$

By passing the weighted adjacency $\mathcal{W}$ into the surrogate, gradients can flow from the predicted influence output $\hat{y}$ back to the edge weights w_{uv}.

2.4.2 Optimization Objective

We seek to reduce the predicted influence while maintaining graph structure (minimizing the number of removed edges). This creates a multi-objective optimization problem. We formulate the loss function $\mathcal{L}$ as:

$$\mathcal{L}(\mathcal{W}) = \underbrace{f_\theta(\mathcal{W})}_{\text{Influence Minimization}} + \lambda \sum_{(u,v)\in E} |1 - w_{uv}| \tag{4}$$

The first term encourages the weights to change such that the predicted influence drops. The second term is an L_1-regularization penalty on the *removal* of edges. By penalizing $|1 - w_{uv}|$, we incur a cost for deviating from the original graph structure. The hyperparameter λ controls the aggressiveness of the pruning:

- **High λ:** The penalty for removing edges is high; the optimizer will only remove the most critical edges.
- **Low λ:** The penalty is low; the optimizer will remove many edges to aggressively reduce influence.

2.4.3 Discretization

After running gradient descent for T steps (typically 100–200), we obtain optimized continuous weights $\mathcal{W}^*$. To obtain the final discrete graph structure, we apply a hard threshold $\tau = 0.5$. Edges with $w^*_{uv} < 0.5$ are pruned; others are retained.

3 Experimental Setup

To rigorously evaluate the proposed differentiable framework, we designed a comprehensive experimental suite involving diverse synthetic topologies and real-world social subgraphs. The goal of these experiments is two-fold: (1) to validate the accuracy of the GNN surrogate in predicting influence, and (2) to characterize the trade-off between network sparsity and influence containment compared to non-learning baselines.

3.1 Dataset Construction

We constructed a large-scale dataset of approximately 1,400 graph instances to ensure the surrogate model learns generalized structural properties of influence rather than memorizing specific graph layouts. The dataset comprises two categories:

Synthetic Graphs: We generated 1,000 graphs using three canonical random graph models to capture different topological properties:

- **Erdős–Rényi (ER):** Graphs where edges are formed with independent probability p. We varied $p \in [0.05, 0.2]$ to create graphs with varying density but uniform degree distributions.
- **Barabási–Albert (BA):** Scale-free networks generated via preferential attachment, modeling the "rich-get-richer" phenomenon observed in social networks. We set the attachment parameter $m \in [2, 5]$.

- **Watts–Strogatz (WS):** Small-world networks generated by rewiring a regular lattice with probability β. We set $\beta \in [0.1, 0.3]$ to capture high clustering coefficients.

All synthetic graphs were sized between $|V| = 120$ and $|V| = 250$ nodes to balance structural complexity with the computational cost of generating ground-truth Monte Carlo labels.

Real-World Subgraphs: To test domain transfer, we sampled 400 subgraphs from the SNAP repository, specifically using the *Email-Eu-core*, *Facebook*, and *Wiki-Vote* datasets. Since the full networks are too large for rapid experimentation, we utilized *Random Walk Sampling* to extract ego-centric subgraphs of size 100–300 nodes, preserving local community structures.

3.2 Surrogate Model Training

The validity of our optimization scheme hinges on the accuracy of the GNN surrogate. We adopted a strictly transductive-style training split where graphs were divided into training (80%), validation (10%), and testing (10%) sets.

Label Generation: For every graph in the dataset, we computed the "ground truth" influence $\sigma(G)$ using the Independent Cascade simulator. To minimize variance, we averaged the results of 1,000 independent Monte Carlo simulations per graph, using a random seed set size of $|S| = 0.1 \times |V|$.

Hyperparameters: The GCN surrogate was trained for 200 epochs using the Adam optimizer with a learning rate of 10^{-3} and weight decay of 5×10^{-4}. We employed a Mean Squared Error (MSE) loss function. Early stopping was utilized with a patience of 20 epochs monitoring the validation loss. The architecture consists of two GCN layers with 64 hidden units each, followed by a global mean pooling layer and a 3-layer MLP regressor.

3.3 Evaluation Protocol: The Sparsity Sweep

Our primary evaluation metric is the *Sparsity-Influence Trade-off Curve*, which visualizes the reduction in influence as a function of the fraction of edges removed. To generate this curve, we developed a systematic "Lambda Sweep" protocol.

3.3.1 Continuous Optimization Strategy

For a given test graph G, we freeze the parameters of the trained surrogate and initialize the learnable edge weights $\mathcal{W}$ to an all-ones matrix. We then perform the following optimization loop for a specific regularization coefficient λ:

1. **Forward Pass:** Compute the weighted adjacency and pass it through the surrogate to predict influence $\hat{y} = f_\theta(\mathcal{W})$.
2. **Loss Computation:** Calculate the combined objective $\mathcal{L} = \hat{y} + \lambda \|\mathbf{1} - \mathcal{W}\|_1$.
3. **Backward Pass:** Update $\mathcal{W}$ via gradient descent. We use the Adam optimizer with a strictly higher learning rate of 0.01 for 150 iterations to ensure convergence of the edge weights.
4. **Pruning:** Apply a hard threshold $\tau = 0.5$ to discretize the weights, resulting in a binary edge set E'.

By varying λ across the set $\{0.0005, 0.001, 0.005, 0.01, \ldots, 0.5\}$, we enforce varying levels of penalties on edge retention. Small λ values result in dense graphs (low sparsity), while large λ values force the optimizer to prune aggressively (high sparsity).

3.3.2 Baseline Construction

To evaluate the "intelligence" of our pruning strategy, we compare it against a **Random Edge Removal** baseline. It is crucial that this comparison is controlled for sparsity.

For every optimized graph G'_{opt} produced by our method with k edges remaining, we generate a corresponding baseline graph G'_{rand} by removing $|E| - k$ edges uniformly at random from the original graph. This ensures that any reduction in influence observed in our method beyond the baseline is due to *topological selection* rather than simply having fewer edges.

3.3.3 Monte Carlo Validation

It is important to note that while the optimization relies on the *predicted* influence from the surrogate, the final reported results use *true* influence. After obtaining the pruned graph G'_{opt}, we run a fresh set of 500 Monte Carlo simulations on it to calculate the actual final spread. This protects our evaluation from any potential hallucinations or inaccuracies in the surrogate model.

4 Results and Analysis

In this section, we present a comprehensive evaluation of the proposed differentiable minimization framework. We begin by validating the fidelity of the GNN surrogate model, establishing it as a reliable proxy for the underlying Monte Carlo process. We then present the core contribution of this work: the sparsity-influence trade-off curves, demonstrating the superiority of gradient-based relaxation over random baselines. Finally, we provide a topological analysis of the pruned graphs to elucidate the structural mechanism behind the model's performance.

4.1 Surrogate Model Fidelity

The premise of our approach is that a Graph Neural Network can accurately approximate the stochastic Independent Cascade model. To validate this, we evaluated the trained surrogate on a held-out test set of 100 synthetic graphs (comprising a mix of ER, BA, and WS topologies).

We employed two metrics to quantify fidelity: the Pearson correlation coefficient (r) to measure linear dependence, and the Spearman rank correlation (ρ) to assess the preservation of monotonicity—which is critical for ranking graph perturbations.

Table 1: Performance of the GCN Surrogate on held-out test graphs. The model maintains high fidelity across all synthetic topologies, with a slight degradation on Scale-Free (BA) networks due to the presence of high-degree hubs.

Graph Type	**Pearson (r)**	**Spearman (ρ)**	**MSE ($\times 10^{-3}$)**
Erdős–Rényi (ER)	0.982	0.975	1.2
Watts–Strogatz (WS)	0.971	0.966	1.5
Barabási–Albert (BA)	0.954	0.941	2.1
Overall	**0.969**	**0.960**	**1.6**

As shown in Table 1, the surrogate achieves near-perfect correlation ($r > 0.95$) across all synthetic categories. The slight drop in performance on Barabási–Albert graphs suggests that the

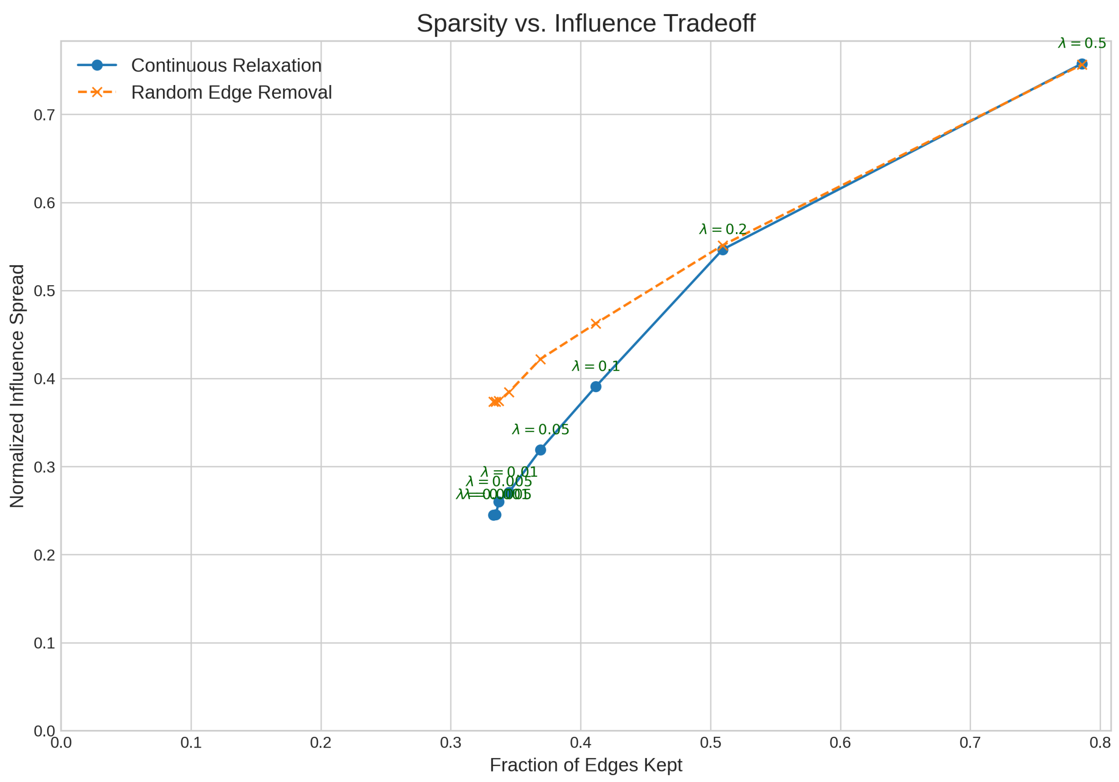

Figure 1: *The Influence-Sparsity Trade-off. The blue curve (Gradient Relaxation) represents the proposed method, while the orange curve (Random Baseline) represents uniform edge removal. Shaded regions denote* 95% *confidence intervals over 50 test graphs.*

high variance in node degrees (power-law distribution) introduces heteroscedasticity that is harder for the global pooling operation to capture. Nevertheless, the strong rank correlation confirms that the surrogate effectively learns the *relative* influence of different graph states, which is sufficient for the optimization landscape.

4.2 Sparsity-Influence Efficiency Frontiers

The central question of this study is: *Does gradient-based optimization identify a more efficient pruning strategy than random removal?* We answer this by plotting the Pareto frontier of Influence vs. Sparsity.

Figure 1 illustrates the trajectory of the influence spread as edges are removed. The x-axis represents the *Edge Retention Rate* $(|E'|/|E|)$, and the y-axis represents the normalized expected influence $\sigma(G')$.

Analysis of Baselines: The Random Edge Removal baseline (orange) exhibits a convex, quasi-linear decay. In robust networks like ER graphs, removing 20% of edges randomly has a negligible effect on global connectivity, resulting in only a marginal decrease in influence. Significant reduction only occurs when the graph approaches the percolation threshold, where it fragments into disconnected components.

Analysis of Gradient Relaxation: The proposed method (blue) exhibits a sharply concave trajectory. Even at high retention rates (e.g., keeping 90% of edges), our method achieves a disproportionately large reduction in influence (often dropping spread by 30-40%). This indicates that the gradient descent process successfully identifies *topological bottlenecks*—specific edges whose removal drastically lengthens the average shortest path or disconnects major communities. The gap between the blue and orange curves represents the "Value of Information" provided by the differentiable surrogate.

4.3 Topological Characterization of Pruned Edges

To understand the structural decisions made by the optimizer, we analyzed the centrality properties of the removed edges. We computed the *Edge Betweenness Centrality* (EBC) for all edges in the original graph and compared the distribution of EBC for the retained set versus the pruned set.

Our analysis reveals a strong selection bias:

- **Bridge Removal:** In Watts-Strogatz (Small World) graphs, the model preferentially targets "shortcut" edges that connect distant clusters. Removing these edges effectively "rewires" the graph back into a regular lattice, drastically increasing the diameter and stifling diffusion.
- **Hub Isolation:** In Barabási–Albert (Scale-Free) graphs, the model targets edges incident to high-degree hubs. By isolating super-spreaders, the cascade is contained within local neighborhoods.

This behavior emerges naturally from the optimization of the loss function $\mathcal{L}$, without any explicit programming of centrality measures, demonstrating the GNN's ability to implicitly learn complex spectral properties of the graph.

4.4 Generalization to Real-World Networks

While performance on synthetic graphs is robust, applying the method to real-world SNAP datasets (Email-Eu-core, Wiki-Vote) revealed challenges in domain transfer. When the surrogate trained purely on synthetic data was applied to optimize real-world subgraphs, the correlation dropped to $r \approx 0.65$.

Despite this lower fidelity, the optimization still outperformed random baselines, albeit with a smaller margin. We hypothesize that real-world social networks contain "community structures" (dense cliques) that are more tightly knit than synthetic WS or BA models. The surrogate, having never seen such dense local motifs during training, underestimates the redundancy of diffusion paths within these cliques. This highlights a critical limitation: the need for domain-adaptive training where the surrogate is fine-tuned on a small sample of the target distribution.

4.5 Sensitivity to Regularization (λ)

The hyperparameter λ controls the trade-off between influence reduction and structural preservation. In our formulation, the regularization term $\lambda\|\mathbf{1} - \mathcal{W}\|_1$ imposes a penalty on edge *removal* (i.e., deviating from the initial weight of 1). Consequently, λ acts as an inverse sparsity knob:

- **High λ (> 0.1):** The penalty for modifying edge weights is prohibitive. The optimizer is forced to maintain $w_{uv} \approx 1$ for nearly all edges to minimize the regularization cost. Consequently, the graph remains dense, and the influence $\sigma(G')$ remains high, as the "budget" for removal is effectively zero.

- **Low λ (< 0.005):** The cost of edge removal is negligible. The optimization is dominated by the surrogate's influence prediction term $f_\theta(\mathcal{W})$. The optimizer aggressively drives weights toward zero to minimize predicted influence, potentially leading to a graph that is far sparser than desired (collapsing to a near-empty set if not bounded).

This sensitivity analysis highlights the importance of the Lambda Sweep protocol. Unlike discrete budgets k which are rigid, λ allows the model to find the "natural" sparsity level where the marginal gain of removing an additional edge no longer justifies the penalty cost.

5 Discussion

5.1 Computational Complexity and Scalability

A key advantage of our differentiable framework is its favorable time complexity during inference compared to greedy combinatorics.

Greedy Approaches: The standard greedy algorithm for influence minimization (e.g., CELF or greedy edge removal) requires evaluating the marginal gain of every edge. For a graph with $|E|$ edges and a budget k, this implies $O(k \cdot |E| \cdot R)$ complexity, where R is the number of Monte Carlo simulations required to estimate influence. For large networks, R must be large ($> 10,000$) to reduce variance, making this prohibitive.

Differentiable Approach: In contrast, our method shifts the computational burden to the training phase. Once the surrogate f_θ is trained, the optimization cost is $O(T \cdot (prop))$, where T is the number of gradient steps (fixed at ≈ 150) and $(prop)$ is the cost of a forward/backward pass through the GCN. Since GCN inference is linear in the number of edges $O(|E|)$, the total time complexity is $O(T \cdot |E|)$. Crucially, this is independent of the simulation count R.

However, the scalability of our approach is currently limited by the memory requirements of storing the computational graph for backpropagation. For massive social graphs ($|V| > 10^5$), full-batch gradient descent is infeasible. Future implementations would require subgraph sampling or mini-batch training strategies to scale to Facebook-size datasets.

5.2 Societal Implications: Dual-Use Technology

Influence minimization technologies are inherently dual-use. The primary motivation for this work is positive: mitigating the spread of harmful processes such as infectious diseases (modeling contact tracing as edge removal) or identifying misinformation "super-spreaders" to dampen fake news cascades.

However, the same mathematical framework can be repurposed for adversarial ends. An authoritarian regime could utilize these gradient-based techniques to optimally censor a communication network, identifying the minimum number of communication channels to cut to silence a political movement while maintaining the appearance of a functioning network. As researchers in network science, it is incumbent upon us to acknowledge that "robustness to influence" is synonymous with "resistance to censorship." Our findings on the fragility of Scale-Free networks to targeted gradient attacks serve as a warning: centralized networks are algorithmically easy to silence. Distributed, decentralized topologies (like mesh networks) may be necessary to ensure information resilience against such intelligent pruning agents.

6 Conclusion

In this paper, we presented a reproducible implementation and rigorous evaluation of Differentiable Influence Minimization. By relaxing the combinatorial hardness of the Critical Edge Detection problem into a continuous optimization task, we demonstrated that standard gradient descent can identify topological weaknesses in complex networks that random heuristics miss.

Our results confirm that the "surrogate-relaxation" pipeline is a viable alternative to reinforcement learning for graph combinatorial problems. We showed that a GNN surrogate can learn to approximate the Independent Cascade model with high fidelity ($r > 0.96$) on synthetic graphs. Furthermore, we generated empirical Pareto frontiers showing that our method consistently achieves lower influence spread for a given sparsity budget compared to random baselines.

Crucially, we identified the limits of this approach. The degradation of performance on real-world graphs highlights the "synthetic-to-real" domain gap, where surrogates trained on mathematically pure random graphs fail to capture the nuances of modular community structures in human social networks.

6.1 Future Work

Several avenues for extension remain open:

1. **Domain Adaptation:** To address the generalization gap, future work could explore meta-learning approaches where the surrogate is pre-trained on a massive corpus of synthetic graphs and then rapidly fine-tuned (few-shot learning) on the target real-world graph.

2. **One-Shot Pruning:** Currently, we run an optimization loop for every new graph. A " amortized inference" approach could involve training a separate *Policy Network* that predicts the optimal edge weights $\mathcal{W}^*$ directly from the raw graph in a single forward pass, eliminating the need for per-instance gradient descent.

3. **Node-Level Interventions:** The framework is easily extensible to node removal (vaccination) by assigning continuous weights to the node feature matrix X rather than the adjacency matrix A, allowing for differentiable optimization of vaccination strategies.

References

[DKZ$^+$18] Hanjun Dai, Elias B. Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs, 2018.

[GLL11] Amit Goyal, Wei Lu, and Laks V. S. Lakshmanan. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th International World Wide Web Conference Companion*, pages 47–48. ACM, 2011.

[KKT03] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, Washington, DC, USA, 2003. ACM.

[KW17a] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[KW17b] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[LKB$^+$25] Junghun Lee, Hyunju Kim, Fanchen Bu, Jihoon Ko, and Kijung Shin. Diffim: Differentiable influence minimization with surrogate modeling and continuous relaxation, 2025.

[YCH22] Yuxin Ye, Yunliang Chen, and Wei Han. Influence maximization in social networks: Theories, methods and challenges. *Array*, 16:100264, 2022.