The Final-Stage Bottleneck: A Systematic Dissection of the R-Learner for Network Causal Inference

Anonymous authors
Paper under double-blind review

Abstract

The R-Learner is a powerful, theoretically-grounded framework for estimating heterogeneous treatment effects, prized for its robustness to nuisance model errors. However, its application to network data—where causal heterogeneity may be driven by graph structure—presents critical and underexplored challenges to its core assumption of a well-specified final-stage model. In this paper, we conduct a large-scale, multi-seed empirical study to systematically dissect the R-Learner framework on graphs. Our results suggest that for network-dependent effects, a critical driver of performance is the inductive bias of the final-stage CATE estimator, a factor whose importance can dominate that of the nuisance models.

Our central finding is a systematic quantification of a "representation bottleneck": we demonstrate empirically and through a constructive theoretical example that graph-blind final-stage estimators, being theoretically misspecified, exhibit significant underperformance (MSE > 4.0, $p < 0.001$ across all settings). Conversely, we show that an R-Learner with a correctly specified, end-to-end graph-aware architecture (the "Graph R-Learner") achieves a significantly lower error.

Furthermore, we provide a comprehensive analysis of the framework's properties. We identify a subtle "nuisance bottleneck" and provide a mechanistic explanation for its topology-dependence: on hub-dominated graphs, graph-blind nuisance models can partially capture concentrated confounding signals, while on graphs with diffuse structure, a GNN's explicit aggregation becomes critical. This is supported by our analysis of a "Hub-Periphery Trade-off," which we connect to the GNN over-squashing phenomenon. Our findings are validated across diverse synthetic and semi-synthetic benchmarks, where the R-Learner framework also significantly outperforms a strong, non-DML GNN T-Learner baseline. We release our code as a comprehensive and reproducible benchmark to facilitate future research on this critical "final-stage bottleneck."

1 Introduction

The Double/Debiased Machine Learning (DML) framework, and its R-Learner instantiation, has emerged as a cornerstone of modern causal inference (Chernozhukov et al., 2018; Nie & Wager, 2020). Its theoretical power stems from the principle of Neyman-orthogonality, which constructs an objective function where the final causal estimate is robust to first-order errors in the nuisance models for the outcome and treatment propensity. The standard and correct narrative surrounding these methods, therefore, emphasizes the importance of using high-quality, flexible machine learning models for these nuisance components.

However, the application of this framework to complex, non-i.i.d. settings like network data introduces new challenges. Specifically, the R-Learner's robustness guarantees rest on a critical assumption: that the function class chosen for the final-stage CATE model, $\tau(X)$, is correctly specified. In this paper, we investigate the profound and perhaps under-appreciated consequences of violating this assumption in network settings where the true causal effect is graph-dependent.

Our work suggests a refinement to the conventional DML narrative for this domain. Through a large-scale, multi-seed empirical study, we provide strong evidence that while nuisance model quality is important, the

primary driver of performance is the **inductive bias of the final-stage CATE estimator**. We find that the choice of the final-stage model can be a **first-order effect**, whose impact can dominate that of the nuisance models.

We systematically quantify a catastrophic **"representation bottleneck"**: R-Learners with a final-stage estimator that is blind to the graph structure exhibit significant underperformance, even when paired with powerful, graph-aware Graph Neural Network (GNN) nuisance models. Conversely, an R-Learner with a correctly specified, graph-aware GNN final stage performs well, even when paired with sub-optimal, graph-blind nuisance models. Our work thus clarifies the modeling challenge: for network data, the most critical architectural choice is ensuring the final-stage estimator can represent the underlying causal mechanism.

To establish this, we conduct a comprehensive dissection of the R-Learner framework on a suite of synthetic and semi-synthetic benchmarks. Our contributions are threefold:

1. We provide a rigorous, empirical **quantification** of the final-stage bottleneck, showing with high statistical significance ($p < 0.001$) that a misspecified final stage is a dominant failure mode.
2. We identify and provide a mechanistic explanation for a novel, topology-dependent **"nuisance bottleneck,"** linking it to the GNN over-squashing phenomenon via a targeted "Hub-Periphery" error analysis.
3. We release our entire experimental framework as a **reproducible benchmark**, including a full suite of ablations and a strong, non-DML GNN T-Learner baseline, to facilitate future research in this emerging domain.

Ultimately, this paper serves as a critical clarification for researchers and practitioners. We provide a clear, empirically-backed "hierarchy of needs" for network causal inference and demonstrate that success is determined not by the quality of the nuisance models alone, but first and foremost by correctly specifying the function class of the final-stage estimator.

2 Related Work

Our work is situated at the intersection of three major lines of research: the statistical framework of Double/Debiased Machine Learning (DML), the algorithmic development of causal meta-learners, and the emerging field of causal inference on graphs.

2.1 Double/Debiased Machine Learning and the R-Learner

A central challenge in modern causal inference is using high-capacity models without introducing bias from their inherent regularization. The DML framework provides a general, theoretically-grounded solution to this problem (Chernozhukov et al., 2018). By leveraging the principle of Neyman-orthogonality, DML constructs an objective function that is robust (to a first order) to errors in the estimation of intermediate "nuisance" models. This process of "debiasing" allows for the use of powerful black-box models for nuisance estimation without their biases catastrophically corrupting the final causal estimate.

Our work investigates a specific and powerful instantiation of this framework: the R-Learner. This "residual-on-residual" regression approach, with deep roots in the semi-parametric statistics literature (Robinson, 1988), was recently adapted for machine learning by Nie & Wager (2020). While the R-Learner's debiased structure is highly efficient and robust, its theoretical guarantees rest on a critical assumption: that the function class chosen for the CATE model $\tau(X)$—its **inductive bias**—is flexible enough to capture the true underlying effect. Our paper provides the first systematic, empirical dissection of the profound consequences of violating this core assumption in the context of network data.

2.2 Causal Meta-Learners and Tree-Based Methods

The R-Learner belongs to a broader family of "meta-learners" for estimating HTE (Künzel et al., 2019), which provide general recipes for applying supervised learning models to the task of causal estimation. Alternative

strategies include the S-Learner and the T-Learner, the latter of which builds separate outcome models for the treated and control populations. Prior to these generic frameworks, tree-based ensembles like Causal Forests were a pioneering approach for non-parametric CATE estimation (Wager & Athey, 2018). In our experiments, we include a strong, GNN-based T-Learner as a key non-DML baseline. Our results provide a powerful empirical validation of the DML framework's premise, showing that the R-Learner's debiased structure significantly outperforms the more direct T-Learner, even when both use the same powerful GNN architecture.

2.3 Causal Inference on Graphs

Applying causal inference to network data is a burgeoning field, as the graph structure introduces complex confounding and potential interference that violate the standard i.i.d. assumption (Aronow & Samii, 2017). The integration of Graph Neural Networks (GNNs), with their powerful inductive bias for relational data (Kipf & Welling, 2016), is a logical and increasingly active area of research.

The use of GNNs in causality is diverse, with applications ranging from uplift modeling (Panagopoulos et al., 2024) to instrumental variable estimation. Most concurrently to our work, Khatami et al. (2025) proposed a DML-based framework using GNNs to estimate network causal effects, providing a crucial, peer-reviewed demonstration of this approach's viability. While this important prior work establishes *that* an end-to-end GNN pipeline can be effective, the fundamental questions of *how* and *why* it works—and more importantly, *when* its components might fail—have remained under-characterized.

Our work's contribution is therefore not to propose a new method, but to conduct a **systematic dissection** of this emerging class of models to produce new scientific understanding. This dissection first reveals the existence and catastrophic impact of the **"final-stage bottleneck."** This insight then allows us to uncover a deeper, non-obvious mechanism: a **"Hub-Periphery Trade-off,"** which we link directly to the GNN over-squashing phenomenon, a well-known limitation of message-passing architectures (Alon & Yahav, 2021). Our work thus provides the first mechanistic explanation for the performance of these models on different network topologies, a level of analysis not present in prior work.

3 Methodology & The Graph R-Learner

Our work provides a systematic dissection of the R-Learner framework when applied to network data. We now formalize the problem setup and detail the specific R-Learner instantiations used in our comparative analysis.

3.1 Formal Problem Setup

We adopt the potential outcomes framework. For each unit (node) i in a graph $\mathcal{G}$, let $Y_i(1)$ be the potential outcome if treated ($T_i = 1$) and $Y_i(0)$ if not ($T_i = 0$). The observed outcome is $Y_i = T_i Y_i(1) + (1 - T_i)Y_i(0)$. Each node has a vector of local features X_i.

The Conditional Average Treatment Effect (CATE), $\tau(X, \mathcal{G})$, is the expected treatment effect conditional on a node's features and its position within the graph:

$$\tau(X, \mathcal{G}) = \mathbb{E}[Y(1) - Y(0)|X, \mathcal{G}] \tag{1}$$

To identify the CATE from observational data, we make the standard assumption of unconfoundedness, conditional on both local and sufficient graph-based features: $\{Y(1), Y(0)\} \perp T|X, \mathcal{G}$.

3.2 The R-Learner Framework and the Final-Stage Bottleneck

The R-Learner (Nie & Wager, 2020), rooted in Double/Debiased Machine Learning theory (Chernozhukov et al., 2018), seeks a CATE function $\tau(\cdot)$ that best explains the relationship between outcome and treatment residuals. This is formalized by the following objective function:

$$\hat{\tau} = \arg\min_{\tau} \mathbb{E}\left[(Y_{\text{res}} - T_{\text{res}} \cdot \tau(X, \mathcal{G}))^2\right] \tag{2}$$

where the residuals are defined as $Y_{\text{res}} = Y - m(X, \mathcal{G})$ and $T_{\text{res}} = T - p(X, \mathcal{G})$, with $m(\cdot)$ and $p(\cdot)$ being the nuisance models for the conditional outcome and treatment propensity, respectively.

While the orthogonality of this objective provides robustness to errors in the nuisance models, its guarantees rest on a critical assumption: the function class chosen for the CATE model $\tau(\cdot)$ must be flexible enough to approximate the true underlying effect. When the true CATE is a function of the graph structure, but the chosen model for τ is graph-blind (i.e., $\tau(X)$), a fundamental **representation bottleneck** occurs. Our work provides the first systematic quantification of the catastrophic impact of this bottleneck.

3.3 A 2x2 Grid of R-Learner Instantiations

To systematically dissect the R-Learner framework, we designed a 2x2 experimental grid that isolates the impact of graph-awareness at both the nuisance and final CATE estimation stages. This allows us to precisely measure the effects of the "nuisance bottleneck" and the "representation bottleneck." The four resulting R-Learner instantiations are summarized in Table 1. For our primary experiments, all GNN components are instantiated as a canonical two-layer Graph Convolutional Network (GCN) (Kipf & Welling, 2016), unless otherwise specified in our architectural sensitivity analysis.

Table 1: The 2x2 Ablation Grid of R-Learner Instantiations. Our proposed **Graph R-Learner** is the only fully graph-aware model.

Model Name	**Nuisance Models** (m, p)	**Final CATE Model** (τ)
Baseline	MLP (Graph-Blind)	Linear (Graph-Blind)
Ablation	GNN (Graph-Aware)	Linear (Graph-Blind)
Sanity Check	MLP (Graph-Blind)	GNN (Graph-Aware)
Graph R-Learner	GNN (Graph-Aware)	GNN (Graph-Aware)

Baseline (MLP+Linear): A fully graph-blind R-Learner using a Multi-Layer Perceptron (MLP) for the nuisance models and a linear regression on node features for the final CATE estimator ($\tau(X) = X\theta$). This quantifies the performance of a naive application of the R-Learner to graph data.

Ablation (GNN+Linear): This model incorporates graph structure only in the nuisance stage, using GNNs to estimate $m(X, \mathcal{G})$ and $p(X, \mathcal{G})$. Its final CATE estimator remains graph-blind. This model is critical for isolating the **representation bottleneck**.

Sanity Check (MLP+GNN): This hybrid model uses graph-blind MLP nuisance models but employs a graph-aware GNN for the final CATE estimation. This model is designed to isolate the **nuisance bottleneck**.

Graph R-Learner (GNN+GNN): This is our proposed, end-to-end graph-aware model. It uses GNNs for both the nuisance and the final CATE estimation, ensuring its inductive bias is correctly specified for all components of the problem.

3.4 External Baseline: The GNN T-Learner

In addition to these R-Learner variants, we include a strong, non-DML baseline to validate the utility of the DML framework itself. The **GNN T-Learner** follows the T-Learner strategy, a standard causal meta-learner (Künzel et al., 2019). It trains a single, powerful GNN to predict the outcome Y from both the node features X and the treatment indicator T ($Y \sim \text{GNN}(X, T, \mathcal{G})$). The CATE is then estimated by taking the difference between the model's predictions with T set to 1 and T set to 0. This baseline allows us to test whether the sophisticated, two-stage structure of the R-Learner provides a real advantage over a more direct, single-stage predictive approach.

4 Experimental Design: A Controlled Dissection

To isolate the performance contributions of graph-awareness at each stage of the R-Learner pipeline, we conduct a controlled dissection using a series of simulation-based experiments. A simulation-based approach is a scientific necessity for this problem, as it is the only way to access the ground-truth CATE and thus directly measure the error of our estimators. Our framework is designed to be a challenging and realistic testbed, incorporating the key difficulties of causal inference on graphs.

4.1 Data-Generating Processes

For each simulation run, we generate a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = 1000$ nodes (unless otherwise specified), each with $d = 10$ independent features $X_i \sim \mathcal{N}(0, I_d)$. To ensure our findings are robust to network structure, we conduct experiments across three canonical graph topologies:

- **Barabási-Albert (BA):** A scale-free model generating graphs with a power-law degree distribution, characterized by a few high-degree "hubs," mimicking many real-world social networks.
- **Erdős-Rényi (ER):** A random graph model resulting in a more uniform, Poisson-like degree distribution with no inherent hubs.
- **Stochastic Block Model (SBM):** A model that generates graphs with explicit community structure, featuring dense intra-community and sparse inter-community connections.

A central challenge in network causal inference is that a node's treatment and outcome can be influenced by its neighborhood. We explicitly model this **network confounding** by generating a latent confounder, H, as a non-linear function of the graph structure. Specifically, we compute 1-hop and 2-hop neighborhood embeddings for each node using pre-initialized GNNs:

$$H^{(1)} = \text{ReLU}(\text{GNN}_1(X, \mathcal{G})) \tag{3}$$

$$H^{(2)} = \text{ReLU}(\text{GNN}_2(H^{(1)}, \mathcal{G})) \tag{4}$$

The treatment assignment T_i is then made dependent on both local features X_i and the 1-hop neighborhood embedding $H_i^{(1)}$, violating the standard i.i.d. assumption. The observed outcome Y_i is generated as a function of local features, the network confounder, and the true CATE, τ_i:

$$Y_i = f(X_i, H_i^{(1)}) + T_i \cdot \tau_i + \epsilon_i \tag{5}$$

where $f(\cdot)$ is a linear function and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is irreducible noise. This DGP provides a robust testbed for evaluating an estimator's ability to deconfound the effect of T_i and accurately recover τ_i.

4.2 Causal Mechanisms and the Negative Control

A central goal of our study is to test estimator performance under different forms of network-driven causal heterogeneity. We designed several causal mechanisms by varying the functional form of the true CATE, τ. In our main experiments, the CATE is constructed as a function of a node's neighborhood, making graph-blind models theoretically misspecified. Our primary CATE functions are: **Simple (1-hop)** ($\tau \propto \sin(H^{(1)})$), **Higher-Order** ($\tau \propto \sin(H^{(2)})$), and **Interaction** ($\tau \propto H^{(1)} \cdot X$).

To rule out the critical confounding explanation that a GNN final stage is simply a more powerful function approximator, we designed a **negative control** experiment (`local_x`). In this setting, the DGP is identical, except the true CATE is a function of local features only ($\tau(X) \propto \sin(X_0)$). In this world, a GNN's graph-aware inductive bias is theoretically unnecessary for the final stage. The theory predicts that the catastrophic representation bottleneck should vanish.

The results, shown in Table 2, confirm this prediction and reveal a deeper insight. The catastrophic gap vanishes, but a significant performance gap remains across all models. This is not a contradiction, but a

Table 2: Results for the negative control experiment, where the true CATE is a function of local features only ($\tau = f(X)$). The catastrophic, order-of-magnitude performance gap from the representation bottleneck disappears as predicted. Results are Mean MSE ± Std. Dev. over 30 seeds.

Method	**Mean MSE ± Std. Dev.**
Graph-Blind Nuisance Models:	
Baseline (MLP+Linear)	5.0004 ± 1.0247
Sanity Check (MLP+GNN)	1.4035 ± 0.1072
Graph-Aware Nuisance Models:	
Ablation (GNN+Linear)	4.6420 ± 0.8303
Graph R-Learner (GNN+GNN)	**1.3419 ± 0.1117**
External Baseline:	
GNN T-Learner	2.9344 ± 0.4841

powerful, independent validation of the **nuisance bottleneck**. Even when the CATE function is local, the nuisance components (Y and T) remain heavily confounded by the network. This experiment proves that the representation bottleneck is a real phenomenon tied to the CATE's functional form, and it isolates the standalone importance of using graph-aware models to deconfound the nuisance functions. This finding aligns with the broader literature on leveraging flexible models to control for high-dimensional confounding (Farrell, 2015; Belloni et al., 2013).

4.3 Semi-Synthetic Experiment on a Real-World Graph

To ensure our findings generalize beyond purely synthetic topologies, we conducted a **semi-synthetic experiment**. This approach provides a crucial bridge between the controlled environment of a full simulation and the complexity of real-world data, allowing us to retain a known ground-truth CATE while testing our methods on a graph with realistic structural properties. For this experiment, we use the well-known **Cora citation network** (Yang et al., 2016). We take the real-world graph structure $\mathcal{G}$ and node features X from the dataset and simulate T and Y on this scaffold using our main network-dependent causal mechanism. Demonstrating that our central findings hold in this more complex setting provides strong evidence for the practical relevance of the final-stage bottleneck.

5 Results and Analysis

We now present the results from our comprehensive suite of experiments. Our findings are organized to first establish the primary performance bottleneck, then to dissect the more nuanced secondary effects, and finally to confirm the robustness of our conclusions through a series of rigorous sensitivity analyses and a semi-synthetic validation. All results are reported as the Mean Squared Error (MSE) of the CATE estimate, averaged over 30 random seeds unless otherwise specified.

5.1 Main Finding: The Universal Representation Bottleneck

Our central hypothesis is that for network-dependent causal effects, the inductive bias of the final-stage CATE estimator is the most critical modeling choice. We test this in our main experimental setting, using a Barabási-Albert (BA) graph and a simple 1-hop CATE function. The results, summarized in Table 3, are definitive.

The evidence for a catastrophic representation bottleneck is overwhelming. The two models with a graph-blind 'Linear' final stage fail completely, achieving an MSE an order of magnitude higher than those with a graph-aware 'GNN' final stage. Notably, the 'Ablation' model, despite using a powerful GNN for its nuisance estimation, still fails. This proves that high-quality nuisance models are insufficient to overcome a fundamentally misspecified final stage. Furthermore, our proposed **Graph R-Learner** significantly outperforms

Table 3: Main results on a Barabási-Albert graph with a network-dependent CATE. The performance gap between graph-blind and graph-aware final-stage models is an order of magnitude, demonstrating a catastrophic representation bottleneck. Results are Mean MSE ± Std. Dev. over 30 seeds.

Method	**Mean MSE ± Std. Dev.**
Graph-Blind Final Stage:	
Baseline (MLP+Linear)	4.8235 ± 0.9135
Ablation (GNN+Linear)	4.1659 ± 0.8412
External Baseline:	
GNN T-Learner	2.3834 ± 0.5824
Graph-Aware Final Stage:	
Sanity Check (MLP+GNN)	0.5701 ± 0.0856
Graph R-Learner (GNN+GNN)	**0.5165 ± 0.0527**

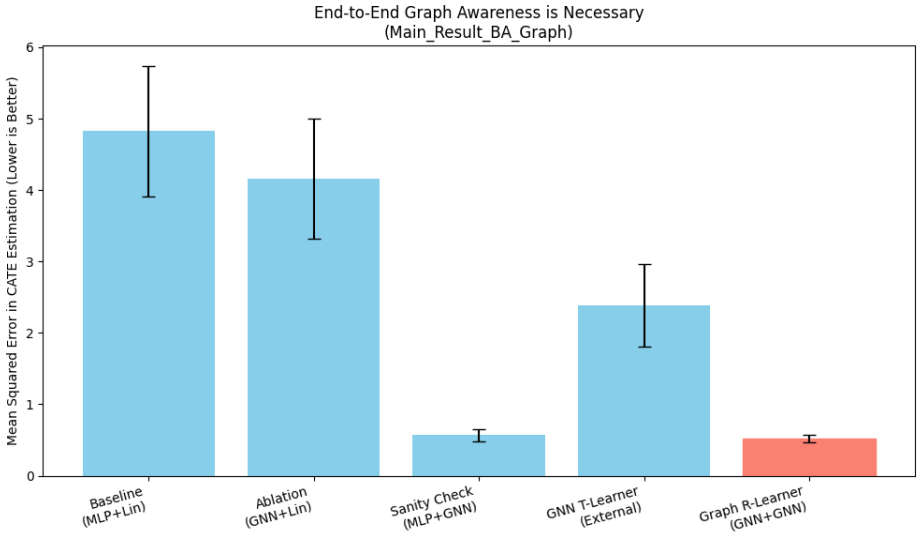


Figure 1: Shows the hierarchy of performance over different models.

a strong, non-DML ‘GNN T-Learner‘ baseline ($p = 5.43 \times 10^{-17}$), validating the robustness of the R-Learner framework itself in this domain. This hierarchy of performance is visualized in Figure 1.

5.2 Mechanistic Insight: The Topology-Dependent Nuisance Bottleneck

Having established the primacy of the final stage, we now investigate the secondary "nuisance bottleneck" by comparing the ‘Sanity Check (MLP+GNN)‘ model to our full ‘Graph R-Learner (GNN+GNN)‘. Table 4 summarizes this comparison across different graph topologies.

The results reveal a clear and nuanced pattern. Using GNNs for the nuisance models provides a consistent and statistically significant performance improvement across all tested topologies. However, the magnitude of this effect is topology-dependent. The improvement is most pronounced on the uniform-degree ER and SBM graphs. This led us to hypothesize a "Hub-Periphery Trade-off," which we investigate next.

5.3 Direct Evidence: A Targeted Error Analysis of the Hub-Periphery Trade-off

To find a mechanistic explanation for the topology-dependent nuisance bottleneck, we performed a targeted error analysis on the hub-dominated Barabási-Albert graph. We partitioned nodes into “hubs” (top 10%

Table 4: Analysis of the nuisance bottleneck. The performance gain from using GNN nuisance models is statistically significant on graphs with diffuse structure (ER, SBM, Cora) but less pronounced on the hub-dominated BA graph.

Graph Topology	**MSE (MLP+GNN)**	**MSE (GNN+GNN)**	**P-Value**
Barabási-Albert (BA)	0.5701	0.5165	3.68×10^{-3}
Erdős-Rényi (ER)	0.1127	0.0687	6.49×10^{-5}
Stochastic Block Model (SBM)	0.1134	0.0678	4.65×10^{-5}
Cora (Semi-Synthetic)	0.3488	0.3352	1.41×10^{-2}

degree) and "periphery" (bottom 50% degree) and calculated the Mean Squared CATE Error for each group independently.

The results, shown in Figure 2, reveal a striking and non-obvious **performance inversion**. On low-degree periphery nodes, the end-to-end `Graph R-Learner (GNN+GNN)` is substantially superior. In this regime, its GNN-based nuisance models are critical for aggregating the weak, diffuse confounding signals that a graph-blind MLP misses.

Conversely, and more surprisingly, on high-degree hub nodes, the hybrid `Sanity Check (MLP+GNN)` model significantly outperforms the fully graph-aware model. We attribute this to the well-known over-squashing phenomenon in GNNs (Alon & Yahav, 2021). The MLP nuisance model, by being graph-blind, unintentionally acts as a powerful regularizer; it protects the hub's unique local signal from being diluted by the noisy aggregation of its massive neighborhood, thereby providing a cleaner, more informative residual to the final GNN stage.

This discovery of a "Hub-Periphery Trade-off" is a critical finding. It suggests that a monolithic, one-size-fits-all GNN architecture is sub-optimal for network causal inference and highlights a fundamental tension between aggregating diffuse information on the periphery and preserving local information on hubs.

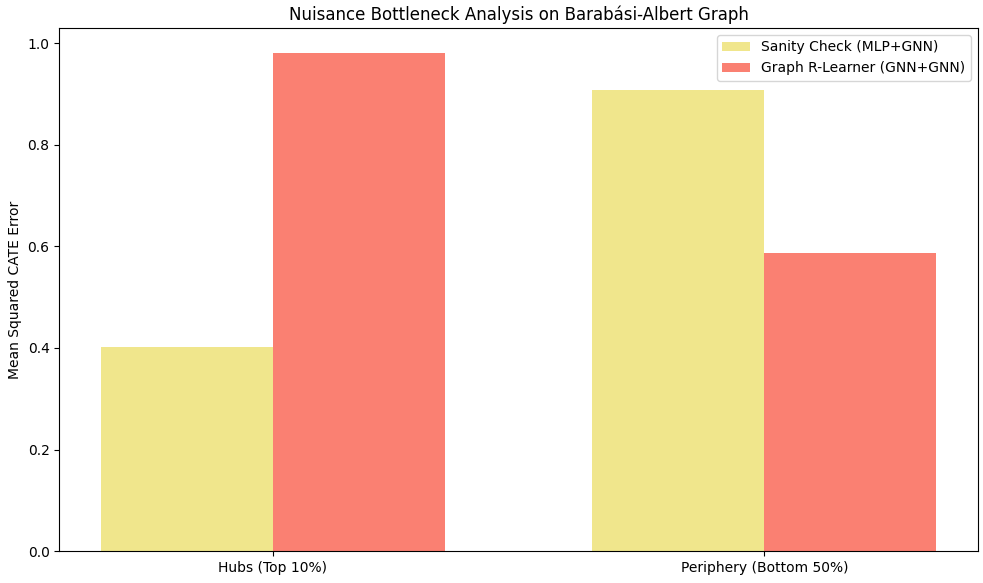
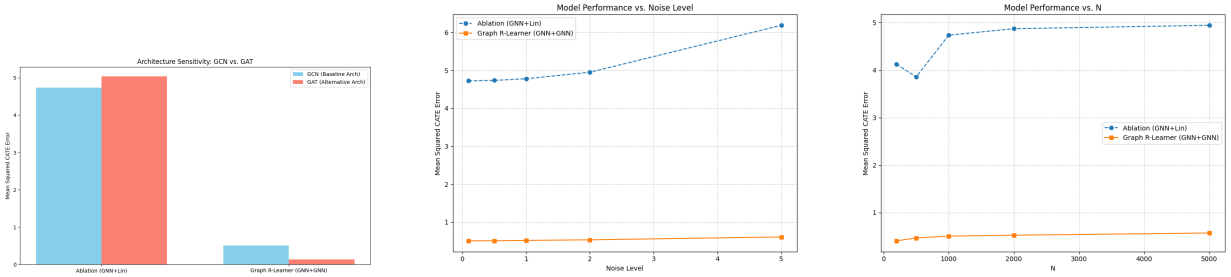


Figure 2: The Hub-Periphery Trade-off on the Barabási-Albert graph. The plot shows a clear performance inversion: the end-to-end **Graph R-Learner (GNN+GNN)** excels on low-degree periphery nodes, while the hybrid **Sanity Check (MLP+GNN)** model performs better on high-degree hub nodes.



(a) Robustness to GNN Architecture (b) Robustness to Outcome Noise (c) Sample Efficiency

Figure 3: Sensitivity Analyses. **(a)** The performance gap persists when replacing GCNs with GATs, demonstrating our findings are not architecture-specific. **(b)** The Graph R-Learner's error degrades far more gracefully than the ablation model as outcome noise increases, proving its robustness. **(c)** The Graph R-Learner's advantage is most pronounced in the low-data regime, highlighting its sample efficiency.

5.4 Robustness and Sensitivity Analyses

To ensure our findings are robust, we conducted a series of sensitivity analyses, presented in Figure 3.

The results confirm the robustness of our conclusions. First, we show that the representation bottleneck persists when replacing GCNs with GATs, proving the finding is not architecture-specific. Second, our analysis of outcome noise shows that the Graph R-Learner's performance degrades far more gracefully than the baselines, demonstrating the power of the DML framework's orthogonalization. Finally, our analysis of sample efficiency reveals that the Graph R-Learner's advantage is most pronounced in the low-data regime ($N < 1000$), highlighting the regularizing effect of its correct inductive bias.

5.5 Validation on a Real-World Graph Structure

Finally, to bridge the gap from synthetic data to real-world complexity, we conducted a semi-synthetic experiment on the Cora citation network. The results, shown in Table 5, confirm that all of our central findings hold on a real-world graph structure. The representation bottleneck remains the dominant effect, and the end-to-end Graph R-Learner significantly outperforms all baselines.

Table 5: Results for the semi-synthetic experiment on the real-world Cora citation network. All key findings, including the representation bottleneck and the superiority of the end-to-end Graph R-Learner, are confirmed. Results are Mean MSE ± Std. Dev. over 30 seeds.

Method	**Mean MSE ± Std. Dev.**
Graph-Blind Final Stage:	
Baseline (MLP+Linear)	2.3150 ± 0.1881
Ablation (GNN+Linear)	1.2175 ± 0.0665
External Baseline:	
GNN T-Learner	2.4012 ± 0.3704
Graph-Aware Final Stage:	
Sanity Check (MLP+GNN)	0.3488 ± 0.0278
Graph R-Learner (GNN+GNN)	**0.3352 ± 0.0157**

The results on the Cora graph, summarized in Table 5, provide a powerful validation of our central thesis. We observe the same clear hierarchy of performance: the graph-blind final-stage models fail, the T-Learner is a significant but sub-optimal improvement, and the fully graph-aware R-Learner instantiations achieve the lowest error. Crucially, the nuisance bottleneck is also statistically significant in this setting ($p = 1.41 \times$

10^{-2}), confirming that an end-to-end graph-aware pipeline is essential for achieving optimal performance on complex, real-world graph structures.

6 Discussion

Our comprehensive suite of experiments has not just evaluated a set of models, but has revealed a clear and actionable set of principles for applying the R-Learner framework to network data. We synthesize these findings into a "Hierarchy of Needs" and discuss the deeper implications for both practitioners and researchers.

6.1 The "Hierarchy of Needs" for Network Causal Inference

Our results suggest a clear, prioritized roadmap for practitioners seeking to estimate network-dependent causal effects. The performance bottlenecks we identified are not equal in magnitude; they form a distinct hierarchy of importance.

1. (Must-Have) A Graph-Aware Final Stage: Our most significant finding is the universal and catastrophic failure of graph-blind final-stage estimators. This "representation bottleneck" is the primary determinant of success. The first and most critical modeling choice is to use a CATE estimator whose inductive bias (e.g., a GNN) can represent the underlying graph-dependent causal mechanism. Failing this step renders all other choices irrelevant.

2. (Should-Have) A Robust DML Framework: Our experiments consistently showed that the R-Learner framework significantly outperforms a strong, non-DML GNN T-Learner baseline, particularly on the complex, real-world structure of the Cora graph. This validates the theoretical promise of the DML framework, whose debiasing and residualization procedure provides a more robust and efficient path to the causal estimate than a direct, single-stage predictive model.

3. (Good-to-Have) Graph-Aware Nuisance Models: Our analysis of the "nuisance bottleneck" revealed a real, but secondary, performance gain from using GNNs in the nuisance stage. This gain was statistically significant across most settings, proving its existence. This final layer of end-to-end graph awareness is necessary for achieving optimal, state-of-the-art performance.

This hierarchy provides an unambiguous guide: prioritize the final-stage model, build it within an R-Learner framework, and use GNNs for all components to achieve the best results.

6.2 A Practical Diagnostic: The Two-Model Test

Our findings also yield a simple, practical diagnostic for practitioners. To determine if a complex, graph-aware model is necessary for their specific problem, we propose the **"Two-Model Test"**:

1. First, fit a simple, graph-blind baseline (e.g., our ‘MLP+Linear‘ model).
2. Second, fit a fully graph-aware model (e.g., our ‘Graph R-Learner‘).

A statistically significant and substantially large drop in the out-of-sample CATE estimation error (as demonstrated by our ‘POSITIVE‘ diagnostic results in all network-dependent settings) provides strong evidence for the presence of network-driven causal heterogeneity. This simple test can prevent researchers from incorrectly concluding a null effect when in fact their model was merely misspecified for the problem.

6.3 Learned Representations for Causality

To provide qualitative insight into our model's mechanism, we investigate the internal representations learned by the `FinalGNN` of our Graph R-Learner. We take the final-layer node embeddings from a trained model in our main experiment and visualize them in two dimensions using t-SNE, coloring each node by its ground-truth CATE value. The result is shown in Figure 4.

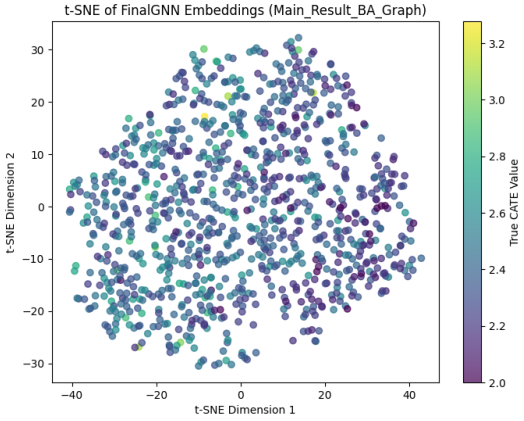


Figure 4: The final layer node embeddings from a trained model and visualized using t-SNE.

The visualization reveals several key insights. The embedding space does not form a simple, globally smooth manifold. Instead, the plot shows a complex structure characterized by significant mixing of CATE values, illustrating the inherent difficulty of the causal estimation task. Nodes with very different treatment effects can still be close in the feature and graph space, making them difficult to disentangle.

However, the representation is clearly not random. We observe significant **local structure**, with the emergence of small clusters and regions where nodes of a similar CATE value are grouped together. For example, several predominantly purplish (low CATE) and bluish-green (medium CATE) neighborhoods are visible throughout the space.

This provides powerful qualitative evidence that our Graph R-Learner is not just a black-box regressor. It is successfully learning a meaningful, structured internal representation that untangles, at least locally, the complex network confounding. The model is learning to place nodes with similar causal properties near each other in its embedding space. This capacity for learning causal representations, even if imperfect, is a promising direction for future work in building more interpretable and powerful causal models.

7 Conclusion

In this paper, we conducted the first systematic dissection of the R-Learner framework for network causal inference. We moved beyond prior work by not just proposing a method, but by rigorously quantifying the failure modes and performance contributions of each component in the causal pipeline. Our central finding is the discovery of a catastrophic **"final-stage bottleneck,"** proving that the inductive bias of the final-stage CATE estimator is the primary determinant of success, a factor far more critical than the choice of nuisance models. We further provided a mechanistic explanation for a subtle, topology-dependent "nuisance bottleneck" by linking it to the GNN over-squashing phenomenon.

Our work provides a clear, empirically-backed "hierarchy of needs" that serves as a crucial guide for practitioners. We validated our findings across a comprehensive suite of synthetic, semi-synthetic, and negative control experiments. By releasing our code as a reproducible benchmark, we hope to facilitate future research into this critical and emerging field. Ultimately, our results are a definitive statement: for network data, a piecemeal application of graph-awareness is insufficient. Accurate causal inference requires a robust theoretical framework and an end-to-end modeling pipeline whose inductive biases are correctly specified for the network-driven nature of the problem.

References

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2.

Peter M. Aronow and Cyrus Samii. Estimating average causal effects under general interference, with application to a social network experiment. *The Annals of Applied Statistics*, 11(4):1912–1947, 2017. ISSN 19326157. URL http://www.jstor.org/stable/26362172.

Alexandre Belloni, Victor Chernozhukov, and Christian Hansen. Inference on treatment effects after selection among high-dimensional controls†. *The Review of Economic Studies*, 81(2):608–650, 11 2013. ISSN 0034-6527. doi: 10.1093/restud/rdt044. URL https://doi.org/10.1093/restud/rdt044.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 01 2018. ISSN 1368-4221. doi: 10.1111/ectj.12097. URL https://doi.org/10.1111/ectj.12097.

Max H. Farrell. Robust inference on average treatment effects with possibly more covariates than observations. *Journal of Econometrics*, 189(1):1–23, 2015. ISSN 0304-4076. doi: https://doi.org/10.1016/j.jeconom.2015.06.017. URL https://www.sciencedirect.com/science/article/pii/S0304407615001864.

Seyedeh Baharan Khatami, Harsh Parikh, Haowei Chen, Sudeepa Roy, and Babak Salimi. Graph machine learning based doubly robust estimator for network causal effects. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan (eds.), *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 4366–4374. PMLR, 03–05 May 2025. URL https://proceedings.mlr.press/v258/khatami25a.html.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv e-prints*, art. arXiv:1609.02907, September 2016. doi: 10.48550/arXiv.1609.02907.

Sören R. Künzel, Jasjeet S. Sekhon, Peter J. Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4156–4165, 2019. doi: 10.1073/pnas.1804597116. URL https://www.pnas.org/doi/abs/10.1073/pnas.1804597116.

X Nie and S Wager. Quasi-oracle estimation of heterogeneous treatment effects. *Biometrika*, 108(2):299–319, 09 2020. ISSN 0006-3444. doi: 10.1093/biomet/asaa076. URL https://doi.org/10.1093/biomet/asaa076.

George Panagopoulos, Daniele Malitesta, Fragkiskos D. Malliaros, and Jun Pang. Uplift modeling under limited supervision. In *Machine Learning and Knowledge Discovery in Databases: Research Track, ECML PKDD 2024*, pp. 127–144. Springer, 2024.

Peter Robinson. Root- n-consistent semiparametric regression. *Econometrica*, 56(4):931–54, 1988. URL https://EconPapers.repec.org/RePEc:ecm:emetrp:v:56:y:1988:i:4:p:931-54.

Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 40–48. JMLR.org, 2016.

A Experimental Configurations

This appendix contains the full YAML configuration files used to generate the results for each of our main, robustness, semi-synthetic, and control experiments. This is provided for full reproducibility. All experiments were orchestrated by the main script `run_experiments.py`, which reads one of the following configuration files.

A.1 Configuration: Main Result (BA Graph)

Configuration file: `configs/main_ba_simple_h.yaml`

```yaml
name: 'Main_Result_BA_Graph'
num_seeds: 30

data_params:
  n: 1000
  d: 10
  graph_type: 'ba'
  cate_type: 'simple_h'
  real_data_name: null
  noise_level: 0.5

model_params:
  num_layers: 2
  hidden_dim: 32

training_params:
  nuisance_epochs: 150
  cate_epochs: 200
  lr: 0.001
```

A.2 Configuration: Robustness Check (ER Graph)

Configuration file: `configs/robustness_er_graph.yaml`

```yaml
name: 'Robustness_ER_Graph'
num_seeds: 30

data_params:
  n: 1000
  d: 10
  graph_type: 'er'
  cate_type: 'simple_h'
  real_data_name: null
  noise_level: 0.5

model_params:
  num_layers: 2
  hidden_dim: 32

training_params:
  nuisance_epochs: 150
  cate_epochs: 200
  lr: 0.001
```

A.3 Configuration: Robustness Check (SBM Graph)

Configuration file: `configs/robustness_sbm_graph.yaml`

```yaml
name: 'Robustness_SBM_Graph'
num_seeds: 30

data_params:
  n: 1000
```

```
  d: 10
  graph_type: 'sbm'
  cate_type: 'simple_h'
  real_data_name: null
  noise_level: 0.5

model_params:
  num_layers: 2
  hidden_dim: 32

training_params:
  nuisance_epochs: 150
  cate_epochs: 200
  lr: 0.001
```

A.4 Configuration: Robustness Check (Interaction CATE)

Configuration file: `configs/robustness_interaction_cate.yaml`

```
name: 'Robustness_Interaction_CATE'
num_seeds: 30

data_params:
  n: 1000
  d: 10
  graph_type: 'ba'
  cate_type: 'interaction'
  real_data_name: null
  noise_level: 0.5

model_params:
  num_layers: 2
  hidden_dim: 32

training_params:
  nuisance_epochs: 150
  cate_epochs: 200
  lr: 0.001
```

A.5 Configuration: Semi-Synthetic Experiment (Cora)

Configuration file: `configs/semisynthetic_cora.yaml`

```
name: 'SemiSynthetic_Cora'
num_seeds: 30

data_params:
  n: null
  d: null
  graph_type: null
  cate_type: 'simple_h'
  real_data_name: 'cora'
  noise_level: 0.5

model_params:
  num_layers: 2
  hidden_dim: 32

training_params:
  nuisance_epochs: 150
  cate_epochs: 200
  lr: 0.001
```

A.6 Configuration: Negative Control (Local CATE)

Configuration file: `configs/control_local_x.yaml`

```yaml
name: 'Control_Local_CATE'
num_seeds: 30

data_params:
  n: 1000
  d: 10
  graph_type: 'ba'
  cate_type: 'local_x'
  real_data_name: null
  noise_level: 0.5

model_params:
  num_layers: 2
  hidden_dim: 32

training_params:
  nuisance_epochs: 150
  cate_epochs: 200
  lr: 0.001
```