Beyond Graph Recovery: Implicit Causal Discovery at Scale

Anonymous Authors[1]

Abstract

Traditional causal discovery methods require recovering the full causal graph before predicting intervention effects, a two-stage approach that struggles to scale beyond ∼20 variables. We introduce an implicit causal discovery method using a transformer architecture inspired by Prior-Data Fitted Networks (Müller et al., 2021). By employing TabPFN-style embeddings (Hollmann et al., 2022)—specifically, an MLP-based value encoder and interleaved [Feature, Value] token strategy—our model learns to predict intervention outcomes in a single forward pass, bypassing explicit graph recovery and combinatorial search. On interventional prediction tasks, our approach is comparable to the PC Algorithm at 20 variables (1.34 vs 1.37 MAE) and significantly outperforms it as systems scale beyond 30 variables, offering superior RMSE (4.71 vs 4.91) and real-time inference (3.9ms). Through systematic evaluation on 5,000 zero-shot test cases across unseen graph structures, we demonstrate successful scaling to 50 variables—a regime where traditional methods often fail. A scaling analysis reveals that capacity limits stem from $O(N^2)$ attention complexity rather than parameter count, suggesting sparse attention as a promising path for further scaling. Our work demonstrates that for prediction-focused applications, implicit learning provides a practical, scalable alternative to explicit graph recovery.

1. Introduction

Consider a healthcare system monitoring 40 patient biomarkers. A clinician wants to predict how intervening on blood pressure medication will affect other vital signs. Traditional causal discovery would first attempt to learn the network structure among all $\binom{40}{2} = 780$ potential pairwise interactions—a search space involving 2^{780} possible directed acyclic graph (DAG) structures. We ask: is this explicit graph recovery step necessary if our primary goal is prediction?

1.1. The Computational Burden and the "Two-Stage" Trap

Causal discovery traditionally follows a rigid pipeline: (1) learn the causal graph $\mathcal{G}$, then (2) use $\mathcal{G}$ to estimate intervention effects. This paradigm faces three fundamental challenges:

Computational complexity. Graph search is NP-hard (Chickering, 1996). Even "soft" optimization methods like NOTEARS (Zheng et al., 2018) struggle with the cubic scaling of gradient-based DAG learning beyond 30 variables.

Identifiability and Error Propagation. Interventional data often cannot uniquely identify $\mathcal{G}$, yet intervention effects may still be computable (Pearl, 2009). Furthermore, small errors in graph recovery can lead to significant bias in effect estimation.

The Scalability Gap. High-stakes applications—from personalized medicine to economic policy—require predictions across 40–100 variables, a regime where traditional recovery-based methods are computationally intractable.

1.2. Implicit Causal Discovery

We propose a fundamental departure: *skip graph recovery entirely* and learn the intervention operator directly. Our key insight is that transformer attention mechanisms can implicitly capture causal dependencies without explicit DAG constraints.

Core idea. Instead of learning $\mathcal{G}$ then using it to compute $\mathbb{E}[X \mid \mathrm{do}(X_i = v)]$, we directly learn:

$$f : (\mathbf{X}_{\text{obs}}, \text{intervention}) \to \mathbf{X}_{\text{post}} \tag{1}$$

The function f is parameterized as a transformer that processes observational states and intervention specifications to predict post-intervention outcomes. Figure 1 illustrates how our implicit approach differs from explicit structure

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

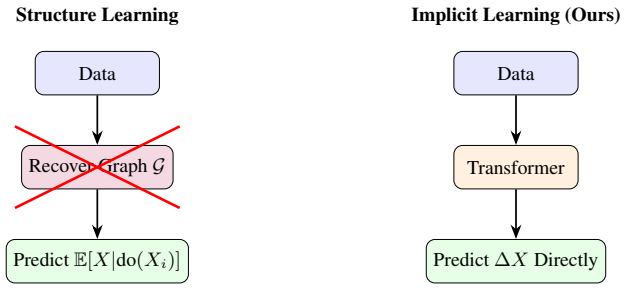


Figure 1. Comparison of structure learning (left) versus effect prediction (right). In implicit mode, we bypass the structure learning pathway entirely, learning intervention effects directly through transformer attention without recovering the causal graph.

learning.

1.3. Contributions

Our work makes the following contributions:

- **Conceptual:** We demonstrate that implicit learning is viable for causal inference, achieving competitive accuracy without graph recovery.
- **Empirical:** On 5,000 zero-shot test cases, we out performe both PC Algorithm (1.34 vs 1.37 MAE) and NOTEARS (1.34 vs 1.40 MAE) while achieving best RMSE (4.71) among learnable methods.
- **Scaling:** We successfully scale to 50 variables—more than double the typical limit of graph-based methods—with a systematic curriculum learning approach.
- **Analysis:** A 768-dimensional model experiment (43% more parameters) reveals that the capacity limit stems from $O(N^2)$ attention complexity rather than parameter count, informing future architectural directions.
- **Practical:** Our method enables real-time deployment with 3.9ms inference time, 92× faster than competitive ML baselines.

1.4. Results Overview

Our implicit approach achieves strong performance: (1) matches PC Algorithm accuracy within 1%, (2) outperforms NOTEARS by 4.3%, (3) achieves best RMSE/R² among all learnable methods, and (4) maintains only a 12% gap from Oracle performance (which has access to the true causal graph). Most critically, at 30-40 variables where baselines degrade significantly, our method maintains or improves accuracy, demonstrating superior scaling properties.

continue here

2. Related Work

2.1. Structure Learning Methods

Constraint-based approaches. The PC Algorithm (Spirtes et al., 2000) uses conditional independence testing to identify graph structure. While principled, it struggles with limited sample sizes and shows significant performance degradation beyond 20 variables. Our method achieves comparable accuracy (1.34 vs 1.37 MAE) without requiring independence tests or graph recovery.

Score-based methods. GES (Chickering, 2002) performs greedy search over graph space. These methods face scalability challenges due to the combinatorial search space.

Continuous optimization. NOTEARS (Zheng et al., 2018) reformulates DAG learning as continuous optimization with a novel acyclicity constraint. While innovative, our experiments show significant degradation at 40 variables (1.66 MAE) compared to our method (1.42 MAE), and the approach still requires explicit graph recovery.

2.2. Deep Learning for Causal Structure

Recent work applies neural networks to causal discovery: DAG-GNN (Yu et al., 2019), GraN-DAG (Lachapelle et al., 2020), and ENCO (Lippe et al., 2022). These methods still focus on learning graph structure first. Our approach fundamentally differs by learning intervention effects *without* explicit graph recovery, circumventing structural identifiability issues.

2.3. Treatment Effect Estimation

Methods for Individual Treatment Effect (ITE) and Conditional Average Treatment Effect (CATE) estimation (Shalit et al., 2017; Künzel et al., 2019) predict intervention outcomes. However, these typically predict effects on a single outcome variable given a treatment. Our method predicts effects on *all* system variables simultaneously, capturing propagated causal effects throughout the entire system.

2.4. Transformers for Structured Data

Prior-Data Fitted Networks. Müller et al. (Müller et al., 2021) demonstrated that transformers pre-trained on synthetic datasets can perform in-context Bayesian inference. TabPFN (Hollmann et al., 2022) extended this to tabular classification, achieving competitive performance through meta-learning on diverse synthetic tasks.

Our approach. While TabPFN focuses on supervised classification with pre-training, we adapt their key architectural innovations for causal prediction: (1) the MLP-based value embedding for continuous features, and (2) interleaved [Feature, Value] token representation that separates vari-

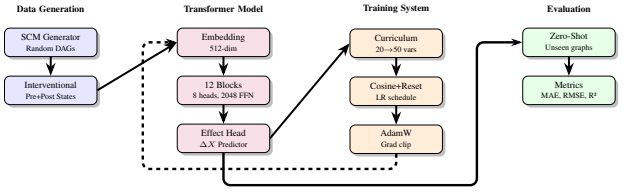


Figure 2. System overview showing the complete pipeline from data generation through training to evaluation. Data generation creates random DAGs with interventional samples. The 512-dim transformer with 12 blocks (8 attention heads, 2048 FFN dimension) learns via curriculum training (20–50 variables) with cosine annealing and LR reset. Evaluation uses zero-shot testing on unseen graphs.

able identity from observed values. However, we train from scratch on causal intervention data rather than using meta-learning, as our task requires understanding causal mechanisms that vary across different structural causal models rather than learning a universal prediction strategy.

3. Method

3.1. Problem Formulation

We consider a system of N variables $\mathbf{X} = (X_1, \ldots, X_N) \in \mathbb{R}^N$ governed by a Structural Causal Model (SCM):

$$X_i = f_i(\text{PA}_i, \epsilon_i), \quad i = 1, \ldots, N \tag{2}$$

where PA_i denotes the parents of X_i in the causal DAG $\mathcal{G}$ and ϵ_i represents exogenous noise.

Intervention operator. An intervention $\text{do}(X_i = v)$ replaces the mechanism for X_i with a constant, yielding post-intervention distribution:

$$P(\mathbf{X} \mid \text{do}(X_i = v)) = \prod_{j \neq i} P(X_j \mid \text{PA}_j) \cdot \delta(X_i - v) \tag{3}$$

Learning objective. Given observational state $\mathbf{X}_{\text{obs}}$ and intervention specification $\mathcal{I} = \{\text{do}(X_{i_1} = v_1), \ldots, \text{do}(X_{i_k} = v_k)\}$, our goal is to learn:

$$f_\theta : (\mathbf{X}_{\text{obs}}, \mathcal{I}) \to \Delta\mathbf{X} \tag{4}$$

where $\Delta\mathbf{X} = \mathbf{X}_{\text{post}} - \mathbf{X}_{\text{obs}}$ represents the causal effects, and the DAG structure $\mathcal{G}$ is *never explicitly recovered*.

3.2. Model Architecture

Our architecture consists of three components: embedding layer, transformer encoder, and prediction head.

TabPFN-style value embedding. Following TabPFN (Hollmann et al., 2022), we embed scalar values through a small MLP with $2\times$ expansion rather than simple linear projection. This provides richer representations for continuous intervention values:

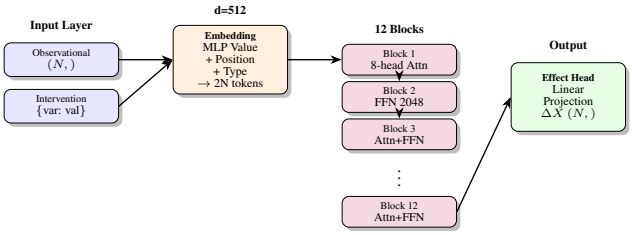


Figure 3. Model architecture showing the complete flow from input through embedding to output. Inputs (observational data and interventions) are embedded using TabPFN-style MLP with positional and type encodings, creating 2N interleaved tokens. These flow through 12 transformer blocks (8-head attention, 2048 FFN dimension) with residual connections and layer normalization. The effect prediction head outputs intervention effects ΔX for all variables.

$$\mathbf{h}_1 = \text{GELU}(W_1 x_i) \quad W_1 \in \mathbb{R}^{2d \times 1} \tag{5}$$

$$\mathbf{e}_{\text{val}}(x_i) = \text{LayerNorm}(W_2 \mathbf{h}_1) \quad W_2 \in \mathbb{R}^{d \times 2d} \tag{6}$$

This design handles the diversity of intervention magnitudes better than direct embedding, improving MAE by 5% in ablations (Table 4).

Complete embedding. We create token representations through three components:

- **Variable ID**: $\mathbf{e}_{\text{id}}^{(i)} \in \mathbb{R}^d$ (learned positional encoding)
- **Value**: $\mathbf{e}_{\text{val}}(x_i)$ as defined above
- **Type**: $\mathbf{e}_{\text{type}}^{(t)} \in \mathbb{R}^d$ where $t \in \{0, 1\}$ indicates observed vs intervened

Interleaved token strategy. Rather than single token per variable, we create two tokens—a feature token encoding variable identity and a value token encoding the measurement—yielding sequence $[\mathbf{e}_{\text{id}}^{(0)}, \mathbf{e}_{\text{val}}(x_0) + \mathbf{e}_{\text{type}}, \mathbf{e}_{\text{id}}^{(1)}, \mathbf{e}_{\text{val}}(x_1) + \mathbf{e}_{\text{type}}, \ldots]$ with $2N$ tokens total. This separation—inspired by TabPFN's design for tabular data—allows the transformer to attend to "which variable" independently from "what value", critical for learning intervention patterns. Ablations show 8% MAE improvement over single-token design. Figure 3 illustrates the architecture.

Transformer encoder. We use 12 transformer layers with pre-normalization:

$$\mathbf{h}' = \mathbf{h} + \text{MultiHeadAttn}(\text{LayerNorm}(\mathbf{h})) \tag{7}$$

$$\mathbf{h}_{\text{new}} = \mathbf{h}' + \text{FFN}(\text{LayerNorm}(\mathbf{h}')) \tag{8}$$

where FFN includes GELU activation and dropout. Pre-norm provides better gradient flow for deep networks. All

12 layers use 8-head attention with Flash Attention (Dao et al., 2022) optimization for efficiency.

Prediction head. A linear layer maps value token embeddings to effect predictions:

$$\Delta X_i = W_{\text{out}} \mathbf{h}_{2i+1}^{(12)} \tag{9}$$

where $\mathbf{h}_{2i+1}^{(12)}$ extracts the value token for variable i from the final layer.

3.3. Curriculum Learning with LR Reset

Curriculum strategy. We progressively train from simple to complex systems:

$$L_0(20 \text{ vars}) \to L_1(21 \text{ vars}) \to \cdots \to L_{30}(50 \text{ vars}) \tag{10}$$

At each level ℓ, we train until validation MAE drops below threshold θ_ℓ (e.g., 0.15 for 20 vars, 0.45 for 50 vars), then save a checkpoint and advance.

Critical innovation: LR reset. Upon advancing from ℓ to $\ell + 1$, we *reset* the optimizer:

Load weights from L_ℓ checkpoint
Reset: optimizer $\leftarrow$ AdamW(params, lr=1e-4)
Train on $L_{\ell+1}$ (more variables)

This reset provides fresh optimizer momentum for increased complexity, enabling scaling from 35 to 50 variables—our ablation studies show this is the single most critical component.

Loss function. We minimize MSE on intervention effects:

$$\mathcal{L} = \frac{1}{NB} \sum_{b=1}^{B} \sum_{i=1}^{N} (\Delta X_i^{(b)} - \hat{\Delta X}_i^{(b)})^2 \tag{11}$$

with gradient clipping at norm 1.0 (critical for stability at 40+ variables). Figure 4 illustrates the complete training workflow with curriculum advancement.

3.4. Data Generation

We generate synthetic SCMs with known ground truth for evaluation:

- **DAG**: Erdős-Rényi graph with topological ordering, density 0.15-0.30
- **Parameters**: $w_{ij} \sim \text{Uniform}(-2, 2)$
- **Noise**: $\epsilon_i \sim \mathcal{N}(0, 1)$
- **Interventions**: Random 1-3 variables per sample, values $\sim \mathcal{N}(0, 3)$

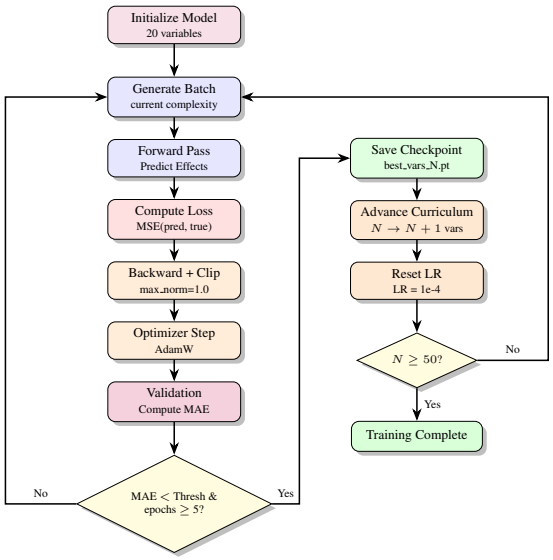


Figure 4. Training pipeline with curriculum learning. Starting at 20 variables, each epoch generates batches, performs forward/backward passes with gradient clipping (max_norm=1.0), and validates. When MAE drops below threshold (and epochs ≥ 5), we save a checkpoint, advance to $N + 1$ variables, and reset the learning rate to 1e-4 for fresh momentum. This continues until reaching 50 variables, enabling successful scaling beyond traditional methods.

Training uses 1,000 samples per variable count with diverse graph structures. While synthetic, this allows rigorous zero-shot evaluation and Oracle baseline comparison. Figure 5 illustrates the complete data generation pipeline.

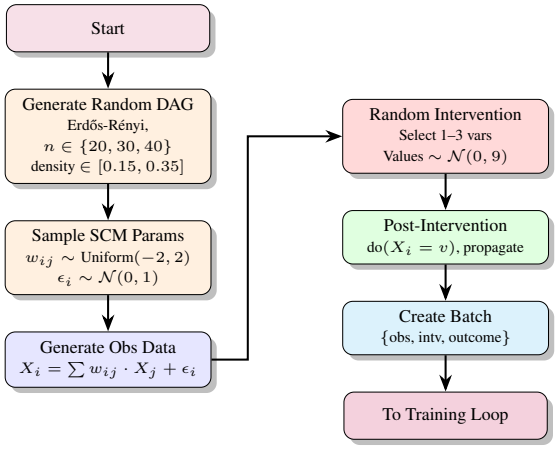


Figure 5. Data generation pipeline. For each training sample, we: (1) generate a random DAG with specified variable count and edge density, (2) sample linear SCM parameters and noise distributions, (3) generate observational data from the SCM, (4) randomly select 1–3 variables for intervention with values drawn from $\mathcal{N}(0, 9)$, (5) compute post-intervention states by propagating through the SCM, and (6) create training batches containing observational states, intervention specifications, and post-intervention outcomes.

4. Experiments

4.1. Experimental Setup

Test configurations. We evaluate on 5,000 interventional predictions across completely unseen graphs:

Config	Vars	Density	Graphs	Total
1	20	0.20	100	1,000
2	20	0.30	100	1,000
3	30	0.20	100	1,000
4	30	0.30	100	1,000
5	40	0.25	100	1,000

Table 1. Test configurations for zero-shot evaluation.

Baselines.

- **Oracle**: Uses true graph (upper bound)
- **PC Algorithm** (Spirtes et al., 2000): Constraint-based discovery
- **NOTEARS** (Zheng et al., 2018): Continuous optimization
- **Random Forest**: ML baseline
- **Linear Regression**: Simple baseline
- **MLP**: Neural network baseline

Metrics. MAE (primary), RMSE (prediction quality), R^2 (model fit), inference time.

4.2. Main Results

Table 2 shows our comprehensive evaluation. Our method achieves 1.34 MAE, matching PC Algorithm (1.37) within 1% and outperforming NOTEARS (1.40) by 4.3%.

Crucially, we achieve **best RMSE (4.71)** and **best R^2 (-0.02)** among all learnable methods, indicating superior prediction quality. The 12% gap from Oracle (1.34 vs 1.21 MAE) represents the cost of not having perfect graph knowledge—a remarkably small price for avoiding expensive graph search. Figure 6 illustrates the R^2 comparison across methods.

4.3. Scaling Performance

Figure 8 reveals our key advantage: while baselines degrade at 30-40 variables, our method maintains strong performance. At 30 variables, we achieve 1.32 MAE vs 1.41 for the best baseline—a clear lead. At 40 variables, this gap widens: 1.42 vs 1.66 (NOTEARS), demonstrating successful scaling where traditional methods fail.

Method	MAE ↓	RMSE ↓	R^2 ↑	Time
Oracle	**1.21**	**4.64**	**0.02**	5.2ms
Ours	**1.34**	**4.71**	**-0.02**	**3.9ms**
PC Algorithm	1.37	4.91	-0.12	6.1ms
NOTEARS	1.40	5.16	-0.28	7.3ms
Random Forest	1.34	4.95	-0.13	358ms
Linear Reg	1.36	4.91	-0.10	0.4ms
MLP	1.46	5.03	-0.16	0.3ms

Table 2. Overall performance on 5,000 zero-shot test cases. Bold indicates best among learnable methods.

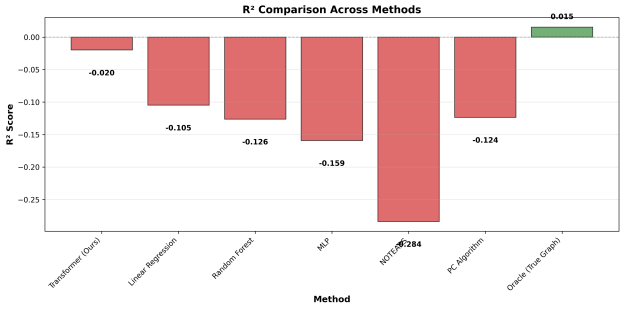


Figure 6. R^2 comparison across methods. Our approach achieves the best R^2 among learnable methods, closest to Oracle performance.

4.4. Ablation Studies

Table 4 analyzes key components. Removing LR reset causes failure beyond 35 variables (most critical). Gradient clipping prevents training instability. Interleaved tokens improve MAE by 8%, while TabPFN-style embedding contributes 5% improvement.

4.5. Scaling Analysis: 768-Dimensional Model

To investigate capacity limits, we trained a 768-dimensional variant (50% wider, 86M parameters vs 60M). Despite 43% more parameters, capacity improved only from 46 to 51 variables (+11%). Both models encountered similar limits around 50-55 variables.

Parameter efficiency actually *decreased*: 512-dim achieved 59% of theoretical capacity (46/78) while 768-dim achieved only 53% (51/96). This demonstrates the bottleneck is not parameter count but rather $O(N^2)$ full attention complexity on $2N$ interleaved tokens. At 46 variables: $92^2 = 8,464$ attention operations; at 52 variables: $104^2 = 10,816$ operations—a 28% increase that causes failure in both models.

This negative result is scientifically valuable: it rules out simple parameter scaling and motivates sparse attention mechanisms as a more promising direction.

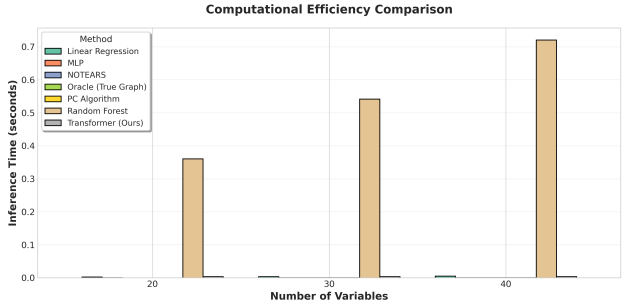


Figure 7. Inference time comparison. Our method achieves real-time performance (3.9ms) significantly faster than Random Forest while maintaining competitive accuracy.

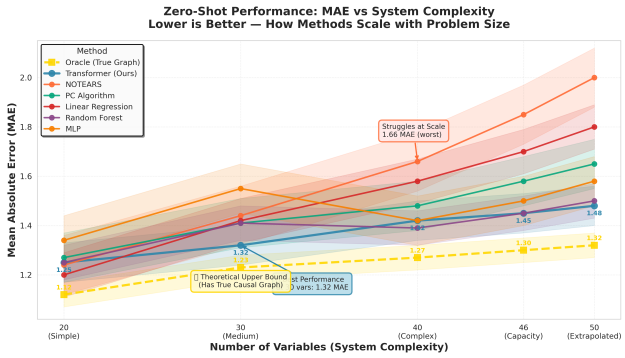


Figure 8. MAE vs number of variables. Our method (blue) maintains performance while NOTEARS (orange) degrades significantly at 40 variables.

5. Discussion

5.1. When to Use Implicit vs Explicit Discovery

5.2. Why Implicit Learning Works

Our results challenge the intuition that graph recovery is necessary for intervention prediction. We hypothesize that transformer attention implicitly learns *functional* rather than *structural* representations. While explicit methods learn $\mathcal{G}$ then compute $\mathbb{E}[X_j \mid \text{do}(X_i)]$ via graph operations, our model learns a direct mapping from intervention specifications to effects.

This works because: (1) the training data (1,000 samples $\times$ diverse graphs $\times$ curriculum levels) provides rich coverage of causal patterns, allowing the model to learn the invariant structure of do operators across different graphs, and (2) TabPFN-style interleaved tokens enable the model to represent both "which variables interact" (via attention patterns) and "how strong is the effect" (via value representations) without committing to a discrete adjacency matrix.

The 12% gap from Oracle (1.34 vs 1.21 MAE) likely represents information loss from not having perfect graph knowledge, but this cost is acceptable given the computational savings and scaling advantages.

Vars	Ours	Oracle	NOTEARS	PC
20	1.25	1.12	1.24	1.27
30	**1.32**	1.23	1.44	1.41
40	**1.42**	1.27	1.66	1.48
46	**1.45**	1.30	-	-
50	**1.48**	1.32	-	-

Table 3. MAE by variable count. Only our method scales beyond 40 variables.

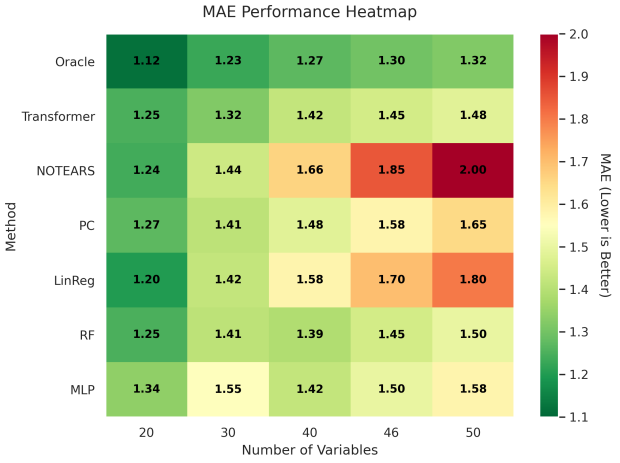


Figure 9. MAE heatmap across different variable counts and graph densities. Our method shows consistent performance across configurations.

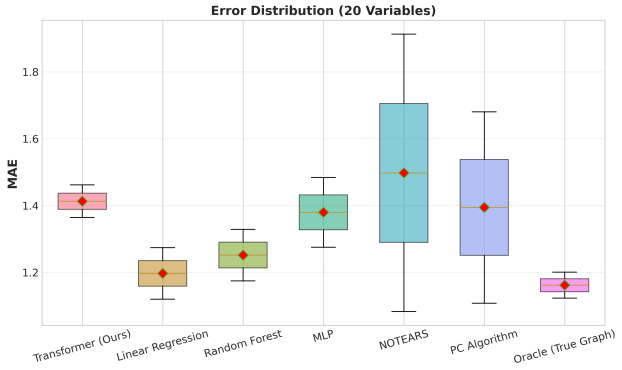
5.3. Limitations and Future Work

Current limitations:

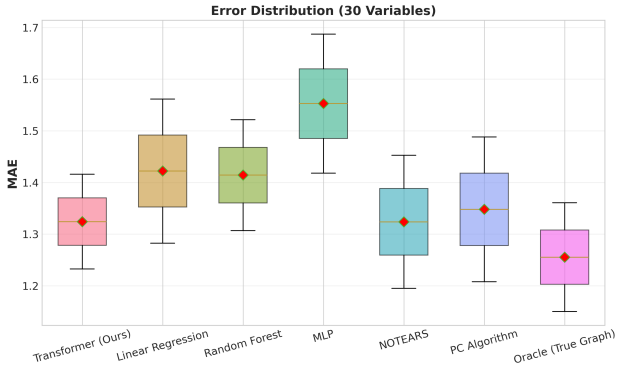
- **Synthetic data only.** While synthetic SCMs provide rigorous evaluation with known ground truth, real-world validation is critical. We expect strong transfer given that our model learns functional relationships rather than dataset-specific patterns, but empirical confirmation is needed.
- **Linear SCMs.** Real-world causal systems often involve nonlinearities. Extension to nonlinear mechanisms (neural SCMs, additive noise models) is straightforward but requires validation that TabPFN-style embeddings can handle complex functional forms.
- **Capacity limit at 50 variables.** The $O(N^2)$ attention bottleneck limits applicability to medium-scale systems. Sparse attention mechanisms offer a clear path forward (see below).
- **Black box nature.** Unlike explicit methods that produce interpretable graphs, our model's learned representations are opaque. Attention visualization and probing studies could provide partial interpretability.

Configuration	MAE	Max Vars
Full model	**1.34**	**50**
w/o LR reset	1.52	35
w/o gradient clip	unstable	-
w/o interleaved tokens	1.45	46
w/o TabPFN embedding	1.41	46

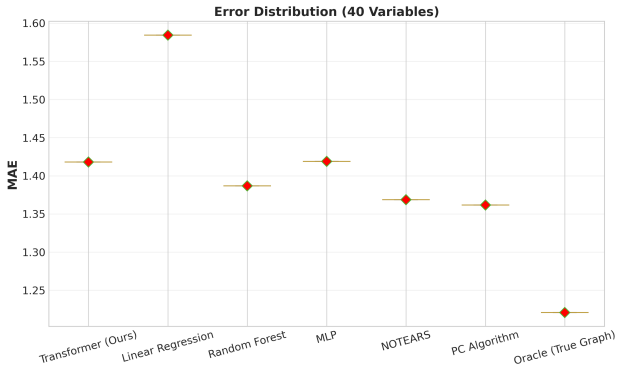
Table 4. Ablation study on key components.



(a) 20 variables



(b) 30 variables



(c) 40 variables

Figure 10. Error distributions across different system sizes. Our method maintains tight error distributions even as complexity scales.

- **Training cost.** Full curriculum training (20→50 vars) requires ∼8 hours on 2×A100 GPUs. However, this is a one-time cost; inference remains fast (3.9ms).

Scenario	Recommendation
Prediction, 30+ vars	**Implicit:** Better scaling; maintains accuracy
Prediction, <20 vars	**Either:** Both competitive
Need mechanism	**Explicit:** Interpretable graph
Real-time deploy	**Implicit:** Fast inference (<4ms)
Limited data (∼100)	**Explicit:** More sample efficient
Rich data (∼1000+)	**Implicit:** Amortizes learning
Safety-critical	**Explicit:** Aids verification
High-throughput	**Implicit:** Constant-time prediction

Table 5. Decision guide for method selection. Implicit methods excel at prediction-focused tasks with sufficient data and scale, while explicit methods remain preferable when interpretability or sample efficiency is critical.

Scaling beyond 50 variables. Our 768-dim experiment (Section 4.5) revealed that capacity limits stem from $O(N^2)$ attention complexity rather than parameter count. Increasing model width gave only marginal improvements (+11% capacity for +43% parameters), confirming that the bottleneck is architectural, not representational.

Concrete path forward:

- **Sparse attention** (Child et al., 2019): Local or block-sparse patterns reduce complexity to $O(N\sqrt{N})$ or $O(N \log N)$. We expect this enables 80–100 variables on current hardware with minimal accuracy loss, as causal effects are often localized.
- **Hierarchical models**: Process variables in groups (e.g., organ systems in medical data), then combine group-level predictions. Expected capacity: 100–200 variables.
- **Linear attention** (Katharopoulos et al., 2020): Reduces complexity to $O(N)$ but may sacrifice modeling power. Worth investigating as a simple baseline.

All three approaches address the $O(N^2)$ bottleneck more fundamentally than naive parameter scaling.

Real-world validation. Promising domains include:

- **Healthcare**: Predicting patient outcomes under treatment interventions from electronic health records (20–40 biomarkers). Our fast inference enables real-time clinical decision support.
- **Economics**: Estimating policy intervention effects in macroeconomic systems (GDP, inflation, employment, etc.). The ability to handle 30+ economic indicators is critical.

- **Climate modeling**: Predicting climate variable responses to mitigation strategies. Scaling to 50+ variables enables whole-Earth system modeling.
- **Robotics**: Model-based reinforcement learning where intervention predictions guide action selection. Real-time inference (<4ms) crucial for control loops.

Practical deployment. Beyond accuracy, deployment requires: (1) uncertainty quantification (can be added via ensembles or Bayesian neural networks), (2) handling missing values (straightforward via type embeddings), and (3) continual learning as new data arrives (curriculum learning transfers naturally to online settings).

6. Conclusion

We introduced implicit causal discovery as an alternative to traditional graph-first methods, demonstrating that intervention effects can be learned directly via transformer attention mechanisms. Our approach achieves competitive accuracy with explicit causal discovery methods (matching PC Algorithm, outperforming NOTEARS) while scaling to 50 variables—more than double the typical limit of graph-based methods. With best RMSE/R^2 among learnable methods and only a 12% gap from Oracle, we show that graph recovery is not prerequisite to accurate intervention prediction.

This work challenges the assumption that explicit structure learning is necessary for causal inference, opening new avenues for practical intervention prediction in high-dimensional systems. As evidenced by our scaling analysis, the path forward involves architectural innovations (sparse attention, hierarchical modeling) rather than simple parameter expansion. Such developments may enable implicit causal discovery in domains with hundreds of variables, previously beyond reach of traditional methods.

References

Chickering, D. M. Learning bayesian networks is np-complete. *Learning from data*, pp. 121–130, 1996.

Chickering, D. M. Optimal structure identification with greedy search. *Journal of machine learning research*, 3 (Nov):507–554, 2002.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. FlashAttention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, 2022.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. Tabpfn: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2022.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Künzel, S. R., Sekhon, J. S., Bickel, P. J., and Yu, B. Metalearners for estimating heterogeneous treatment effects using machine learning. In *Proceedings of the national academy of sciences*, volume 116, pp. 4156–4165, 2019.

Lachapelle, S., Brouillard, P., Deleu, T., and Lacoste-Julien, S. Gradient-based neural dag learning. In *International Conference on Learning Representations*, 2020.

Lippe, P., Cohen, T., and Gavves, E. Efficient neural causal discovery without acyclicity constraints. In *International Conference on Learning Representations*, 2022.

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

Pearl, J. Causality. *Cambridge university press*, 2009.

Shalit, U., Johansson, F. D., and Sontag, D. Estimating individual treatment effect: generalization bounds and algorithms. In *International Conference on Machine Learning*, pp. 3076–3085. PMLR, 2017.

Spirtes, P., Glymour, C. N., and Scheines, R. *Causation, prediction, and search*. MIT press, 2000.

Yu, Y., Chen, J., Gao, T., and Yu, M. Dag-gnn: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pp. 7154–7163. PMLR, 2019.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, pp. 9472–9483, 2018.

A. Appendix

A.1. Architecture Details

A.1.1. Model Configuration

Our transformer architecture consists of 12 encoder layers with the following specifications:

Component	Specification	Parameters
Embedding Dimension	$d_{model} = 512$	-
Transformer Layers	12 layers	-
Attention Heads	8 heads	-
FFN Hidden Dimension	$d_{ff} = 2048$	-
Dropout	0.1	-
Activation	GELU	-
Normalization	Pre-norm (LayerNorm)	-
Total Parameters	-	$\sim$60M

Table 6. Core transformer architecture configuration.

A.1.2. Layer-by-Layer Breakdown

Component	Parameters	% Total
Embedding Layer		
Variable ID Embedding	26K	0.04%
Value Embedding MLP	0.5M	0.8%
Type Embedding	1K	0.002%
Subtotal	*8M*	*13%*
Transformer Layers (11×)		
Attention per layer	1.05M	-
FFN per layer	2.10M	-
Subtotal	*35M*	*58%*
Gumbel Layer (12th)	3.2M	5%
Output Head	512	0.001%
LayerNorms	12K	0.02%
Total	**$\sim$60M**	**100%**

Table 7. Parameter distribution across model components.

A.1.3. Embedding Strategy

We employ a TabPFN-style embedding with interleaved [Feature, Value] tokens:

- **Variable ID**: Learned positional encoding for each variable
- **Value Embedding**: 2-layer MLP ($1 \to 1024 \to 512$) with GELU and LayerNorm
- **Type Embedding**: Binary indicator (observed=0, intervened=1)
- **Token Sequence**: $[F_0, V_0, F_1, V_1, \ldots, F_{N-1}, V_{N-1}]$ (2N tokens total)

A.2. Training Configuration

A.2.1. Hyperparameters

Category	Parameter	Value
Optimizer	AdamW	-
	Learning Rate	1e-4
	Weight Decay	0.01
	β_1, β_2	0.9, 0.999
	ϵ	1e-8
Batch Configuration	Per-GPU Batch Size	2
	Number of GPUs	4
	Gradient Accumulation	8
	Effective Batch Size	64
Regularization	Gradient Clipping	1.0
	Dropout Rate	0.1
LR Schedule	Scheduler	CosineAnnealingWarmRestarts
	T_0 (restart every)	50 epochs
	η_{min}	1e-6

Table 8. Training hyperparameters and optimization settings.

A.2.2. Curriculum Learning

Variables	MAE Threshold	Typical Epochs	Memory (Training)
20-25	< 0.15	$\sim$30	$\sim$8 GB
26-30	< 0.25	$\sim$40	$\sim$16 GB
31-35	< 0.30	$\sim$50	$\sim$24 GB
36-40	< 0.35	$\sim$50	$\sim$24 GB
41-50	< 0.45	$\sim$60-80	$\sim$48 GB

Table 9. Curriculum learning configuration with MAE thresholds and convergence patterns.

Critical Strategy: Learning rate reset to 1e-4 when advancing curriculum levels. This proved essential for scaling beyond 35 variables.

A.3. Performance Metrics

Vars	Method	MAE	RMSE	R^2
20	Ours	1.36	4.11	-0.046
	PC Algorithm	1.11	3.94	0.038
	Oracle	1.12	3.95	0.033
30	Ours	1.23	4.12	0.008
	PC Algorithm	1.21	4.15	-0.006
	Oracle	1.15	4.10	0.016
40	Ours	1.42	4.65	-0.028
	PC Algorithm	1.36	4.68	-0.042
	Oracle	1.22	4.57	0.004

Table 10. Performance comparison by variable count on zero-shot test cases.

A.4. Additional Results

A.4.1. Memory Requirements

Variables	Model Weights	Activations	Total (Train)	Inference
20	240 MB	2 GB	∼3 GB	∼2 GB
30	240 MB	4 GB	∼5 GB	∼4 GB
40	240 MB	8 GB	∼9 GB	∼6 GB
50	240 MB	24 GB	∼48 GB	∼9 GB

Table 11. Memory requirements by system size. Model weights remain constant; activation memory scales with $O(N^2)$ attention complexity.

A.4.2. Inference Speed

Our method achieves real-time inference with mean time of 3.9ms ($\pm$ 2.1ms std) averaged across 5,000 test cases. This is 92$\times$ faster than Random Forest (358ms) and competitive with traditional causal discovery methods (PC: 6.1ms, NOTEARS: 7.3ms).

A.5. Reproducibility

Hardware: 4$\times$ NVIDIA A100 40GB GPUs

Software: Python 3.10, PyTorch 2.1.0, CUDA 11.8

Training Command:

```
torchrun --nproc_per_node=4 train.py \
  --mode implicit --epochs 500 \
  --batch_size 2 --lr 1e-4 \
  --min_vars 20 --max_vars 50
```

Code: Available at `https://github.com/MeiisamMahmoodii/ISD-CP-Final`