Causal Discovery and Inference through Next-Token Prediction

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

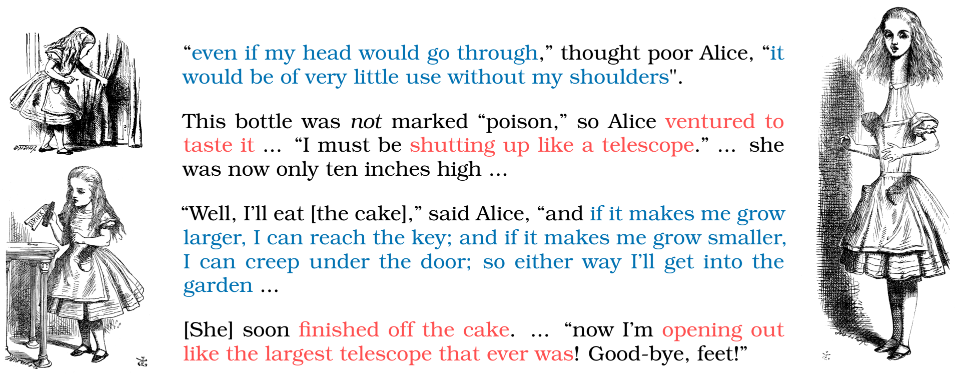
Some have argued that deep neural networks are fundamentally *statistical* systems that fail to capture the causal generative processes that underlie their training data. Here we demonstrate that a GPT-style transformer trained for next-token prediction can simultaneously discover instances of linear Gaussian structural causal models (SCMs) and learn to answer counterfactual queries about them. First, we show that the network generalizes to counterfactual queries about SCMs for which it saw *only* strings describing noisy interventional data. Second, we decode the implicit SCM from the network's residual stream activations and use gradient descent to intervene on that "mental" SCM with predictable effects on the model's output. Our results suggest that statistical prediction may be sufficient to drive the emergence of internal causal models in neural networks trained on passively observed data.

1 Introduction

How can an AI system learn to reason about causes and effects—the mechanisms that remain invariant under local interventions [41, 33]? Pearl [34, 36] has argued that deep neural networks (DNNs) trained using prediction objectives are intrinsically limited in their causal reasoning capacities. His argument rests on Pearl's Causal Hierarchy (PCH) [2], also known as the "Ladder of Causation" [36]. PCH describes three levels of causal capabilities—associational ($\mathcal{L}_1$), interventional ($\mathcal{L}_2$), counterfactual ($\mathcal{L}_3$)—and implies that answers to higher level queries are generally underdetermined by data or information from lower levels. According to Pearl [34], this hierarchy implies that DNNs trained in a "statistical mode" to predict passive observations can only master associations ($\mathcal{L}_1$) and are prevented from reasoning about actions, experiments, and explanations ($\mathcal{L}_2$ and $\mathcal{L}_3$).

If true, Pearl's claim would have important theoretical and practical implications regarding the causal abilities of large language models (LLMs) and other foundation models. Such models are typically pretrained using statistical objectives to predict held-out portions of passive streams of data [8, 38, 37, 6, 5]. For example, LLMs are trained to predict the next token in snippets drawn from a diverse corpus of text. Following Pearl's reasoning [34], at least initial pretrained versions of foundation models should be limited to the level of associations ($\mathcal{L}_1$).

While PCH provides an extremely valuable theoretical framework with wide-reaching implications, we do *not* think that Pearl's claim about DNNs follows from PCH. First, the training data for foundation models contains information about the causal structure of the world. Natural language in particular contains many descriptions of interventions and causal inferences (Fig. 1). Such examples demonstrate that "passive" data is not necessarily equal to "observational" ($\mathcal{L}_1$) data. In fact, in a more recent interview, Pearl [35] acknowledges that text does contain $\mathcal{L}_2$/$\mathcal{L}_3$ information. This means that we cannot rule out the emergence of causal models in LLMs and other foundation models *a priori*, following standard PCH logic.



"even if my head would go through," thought poor Alice, "it would be of very little use without my shoulders".

This bottle was *not* marked "poison," so Alice ventured to taste it ... "I must be shutting up like a telescope." ... she was now only ten inches high ...

"Well, I'll eat [the cake]," said Alice, "and if it makes me grow larger, I can reach the key; and if it makes me grow smaller, I can creep under the door; so either way I'll get into the garden ...

[She] soon finished off the cake. ... "now I'm opening out like the largest telescope that ever was! Good-bye, feet!"

Figure 1: Natural language includes descriptions of interventions and causal inference. Examples and illustrations from *Alice's Adventures in Wonderland* [7]. The drink makes Alice shrink, while the cake causes her to grow; she also reasons counterfactually about fitting through the tiny door. LLMs may discover causal structure—the mechanisms that remain invariant under local interventions [41, 33]—and learn causal inference engines to *predict* the next token in such strings.

However, Pearl [35] and others [49] have still maintained that LLMs cannot possess causal models. The underlying assumption seems to be that the *modeling framework* (deep neural networks) and/or the *learning objective/setup* (prediction of passively observed streams of data) are somehow intrinsically associated with the statistical ($\mathcal{L}_1$) level of the PCH. And in so far as LLMs *can* answer some interventional ($\mathcal{L}_2$) or counterfactual ($\mathcal{L}_3$) queries, they do so by citing information from their training corpus, but they do not possess causal models that would let them generalize [35]. In other words, they are "causal parrots" [49].

Here we propose an alternative hypothesis and provide an existence proof: That the objective to predict the next token, at least in some contexts, may drive LLMs to acquire *real* emergent causal models and causal reasoning capacities at the interventional ($\mathcal{L}_2$) and counterfactual ($\mathcal{L}_3$) levels. We test this hypothesis empirically in a controlled setting. We generate text (in a made-up simple language) describing interventional data and counterfactual inferences from a constrained class of linear Gaussian structural causal models (SCMs). Generated text strings fall into one of two classes (Fig. 3): (1) `DATA` strings describe the behavior of the referenced SCM under interventions; (2) `INFERENCE` strings describe counterfactual inferences, given the referenced SCM. The strings reference the underlying SCMs via arbitrary indices, and the SCM structure is never explicitly provided. Given snippets of these strings, we train a GPT-style transformer model to predict the next token.

To reduce next-token prediction loss in our task, the model could adopt one of two strategies: memorize the training examples, or discover the structure of the underlying SCMs and learn a more general causal inference algorithm. To distinguish these two possibilities, we devise a "generalization challenge" where the trained model has to generalize to counterfactual queries about a separate set of test SCMs (D_{test}) for which the model has *only* been trained with interventional data, and not with any counterfactual inference strings. The only way the model might perform counterfactual inferences about these SCMs is if it had (1) learned how to perform counterfactual inference for our class of SCMs and (2) inferred the structure of the test SCMs from the interventional data. We find that the trained model does indeed generalize to the test SCMs, indicating it has not memorized the answers, but has instead learned a more general counterfactual inference engine.

Crucially, we also probe inside the model and find internal representations of the underlying SCMs that can be decoded from the model's residual activations. We show that we can manipulate these implicit SCMs in the "network's mind" mid-computation using gradient descent with predictable effects on model's output. Finally, we show that decoding accuracy does not necessarily track where in the model the information about the underlying SCMs is *used*.

Importantly, existing critiques of LLMs' causal capabilities rarely define what it means to "possess" a causal model [35, 49]. We address this definitional gap by proposing and demonstrating three capabilities that constitute "possessing" a causal model: (1) generalizing to unseen causal inference query and structure combinations, (2) learning interpretable, decodable representations of the underlying causal structures that (3) can be causally manipulated with predictable effects on model output. Together, our results serve as a carefully crafted and studied existence proof demonstrating how DNNs trained to predict passive streams of data can nonetheless discover and learn to use causal models.

2 Related work

Our work contributes to the broader discussion in the literature about the extent to which LLMs and other foundation models understand the world [4, 47, 27, 48]. Pearl [33, 34] and others [12, 19, 36, 42, 11] have argued that human understanding derives from building and using powerful causal abstractions of the world. Causal capacity of LLMs thus plays an important role in this larger debate about understanding.

Some have been skeptical about the causal abilities of LLMs. Referencing Pearl's hierarchy, Zečević et al. [49] argue that LLMs are "causal parrots" that occasionally correctly answer causal questions only by capturing the "correlation of causal facts". We agree that LLMs may learn to recite often repeated causal claims or exploit correlations between certain words to answer causal queries. However, here we present evidence that LLMs may *also* discover true causal structure from text and learn causal inference engines that generalize to unseen structure and query combinations.

Empirical results on LLM causal capacity have been mixed [18, 50, 17]. Several datasets and benchmarks have been created to make progress on this question [15, 16, 40]. In contrast to evaluating causal abilities of pre-trained LLMs, here we consider a relatively simple causal task (discovery and inference within a constrained set of linear Gaussian SCMs) and train a small transformer from scratch with full control over the training data. We then study not only its behavior, but also *internal representations*. We are inspired by other probing and mechanistic interpretability work [31, 9, 10, 29], particularly on emergent world representations in transformers [21, 22, 30, 23] and knowledge localization/editing [25, 14].

In our setup, we use the structural causal model (SCM) formalism established by Pearl [33] to generate our training data, connecting to recent work on neural-causal models [32, 45, 46, 44].

Finally, our work complements a finding from Lampinen et al. [20] that RL agents can learn "causal strategies" from purely passive offline data, which the agents can then exploit to uncover causal structure using interventions at test time. Interestingly, our results suggest that purely "passive" training on next-token prediction and no interventions (at least *by the model*) may be enough to discover causal structure and learn a causal inference engine.

3 Methods

Our setup involves: (1) generating instances of a constrained set of linear Gaussian structural causal models (SCMs) (section 3.1)); (2) generating text using those SCMs in a simple artificial language that describes interventional data and counterfactual inferences (section 3.2); (3) setting up a generalization challenge for the transformer by providing only interventional data strings for some of the SCMs during training (section 3.3); (4) training a GPT-style transformer to predict the next token in generated text (section 3.4). We then study the emergent behavior of the model and its internal representations.

3.1 Linear Gaussian structural causal models (SCMs)

We consider a constrained class of linear Gaussian structural causal models (SCMs) with 4 variables V_1, V_2, V_3, V_4:

$$U_j \sim \mathcal{N}(0, \sigma^2) \quad \text{with } \sigma = \sqrt{0.1} \approx 0.32 \tag{1}$$

$$V_j := U_j + w_{jj} + \sum_{i<j} w_{ij} V_i \quad \text{where } \forall i, j : w_{ij} \in \{-1, 0, 1\} \tag{2}$$

Background variables $\boldsymbol{U} = \{U_1, U_2, U_3, U_4\}$ (sometimes referred to as "exogenous" or "noise" variables) are sampled independently from a fixed Gaussian distribution. Each endogenous variable

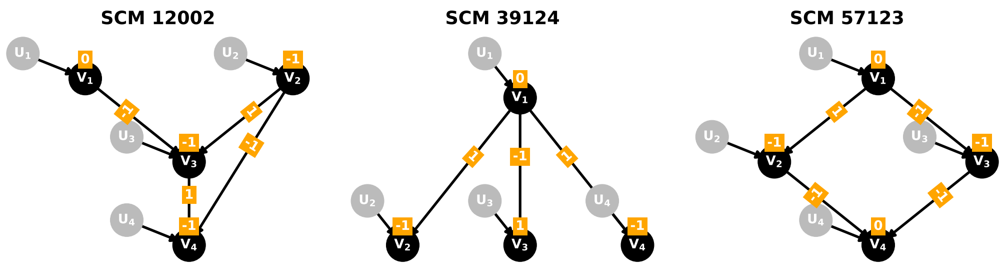


Figure 2: Some examples of the 59,049 unique SCMs. Bias weights w_{ii} are displayed on top of the variables. Non-zero effect weights w_{ij} are displayed on the edges between the variables.

V_j is a linear function of the corresponding background variable U_j, bias term w_{jj}, and a weighted combination of parent values $\sum_{i<j} w_{ij} V_i$, where w_{ij} represents the effect of variable V_i on V_j. Note that each SCM instance within our class is fully defined by a weight vector $\boldsymbol{w}$ with 10 ternary values:

$$\boldsymbol{w} = [w_{11}, w_{12}, w_{13}, w_{14}, w_{22}, w_{23}, w_{24}, w_{33}, w_{34}, w_{44}] \tag{3}$$

We hypothesized that the trained model may represent the SCMs using this (or similar) representation. Foreshadowing the results, it turns out that we can decode and overwrite this vector $\boldsymbol{w}$ in the model's residual stream activations with predictable effects on model's behavior.

We generate all possible weight combinations (3 possible values for 10 weights), resulting in 3^{10} = 59,049 unique SCM instances. Some example SCMs can be seen in Figure 2.

Answers to counterfactual queries within the linear Gaussian SCM class can be computed analytically (details provided in appendix D). Efficient analytical estimates allow us to implement a text-generative model that dynamically samples queries and computes answers for each training batch.

3.2 Generating text (training data)

Natural language includes descriptions of interventions, outcomes under those interventions, and descriptions of various causal inferences (Fig. 1). Our text generation setup using a simple artificial language is intended to emulate this aspect of natural language.

More concretely, we generate two types of strings (Fig. 3). Each string begins with a token that describes its type (`DATA` or `INFERENCE`), followed by an SCM index encoded using 4 letter tokens (e.g. `A R T Q` corresponds to SCM index 12002). Thus SCM structure is never explicitly provided—SCMs are referenced only via these arbitrary indices. We use `OBS` `[V`$_i$`]` `[value]` token sequence to indicate an observation and `DO` `[V`$_i$`]` `[value]` to represent an intervention. Numbers within $[-10, 10]$ are encoded using numerical tokens with one decimal point precision (e.g. `0.3`, `-7.5`). Numbers that fall outside of that range are encoded using `-INF` and `+INF` tokens.

`DATA` strings **provide noisy samples of the underlying interventional distributions**. To construct a `DATA` string, we sample up to two interventions with values drawn uniformly from $\mathcal{U}[-5, 5]$, ensuring that the intervened variable does not repeat in the same string. We then compute the analytical interventional distribution given the referenced SCM and interventions, and take one *sample* from that interventional distribution. We use a random order for the variables and record the sampled values for each variable using `[V`$_i$`]` `[value]` token pairs.

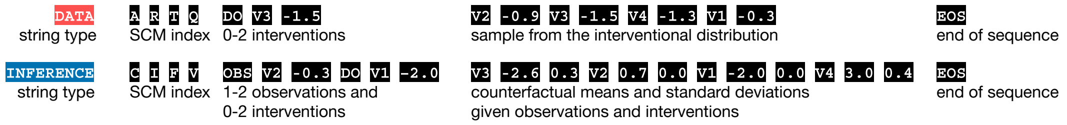

DATA
string type
A R T Q
SCM index
DO V3 -1.5
0-2 interventions
V2 -0.9 V3 -1.5 V4 -1.3 V1 -0.3
sample from the interventional distribution
EOS
end of sequence
INFERENCE
string type
C I F V
SCM index
OBS V2 -0.3 DO V1 -2.0
1-2 observations and
0-2 interventions
V3 -2.6 0.3 V2 0.7 0.0 V1 -2.0 0.0 V4 3.0 0.4
counterfactual means and standard deviations
given observations and interventions
EOS
end of sequence

Figure 3: We generate two types of strings from the SCMs. `DATA` strings provide interventional data about the referenced SCM. `INFERENCE` strings provide examples of counterfactual inference.

SCM set	number of SCMs	INFERENCE strings	DATA strings
D_{train}	58,049	(1) learning a **counterfactual inference engine** ✓	(2) building a **shared representation** ✓
D_{test}	1,000	**generalization challenge!** ✗	(3) **discovering causal structure** from interventional data ✓

strings seen during training? ✓ / ✗

Figure 4: Generalization challenge. To generalize to unseen `INFERENCE` queries for SCMs with `DATA` strings only (D_{test}) the model has to: (1) learn a counterfactual inference engine from descriptions of inferences with the D_{train} SCM set, (2) build a representation for counterfactual inference and interventional data strings that exploits the shared structure, (3) discover the structure of D_{test} SCMs from interventional data.

`INFERENCE` strings **provide examples of counterfactual inference** within our SCM class. To construct an `INFERENCE` string, we sample 1-2 observations (also $\sim \mathcal{U}[-5, 5]$) and 0-2 interventions. We then compute the analytical counterfactual distribution (details in Appendix D). Similarly as with `DATA` strings, we use a random order for the variables and record the counterfactual mean and standard deviation for each variable using a `[V`$_i$`] [mean] [std]` sequence. All strings end with an `EOS` token.

3.3 Generalization challenge

We wanted to distinguish between two possibilities: (1) The model memorizes answers to causal queries. (2) The model learns a more general causal inference engine. For this purpose, we devised a "generalization challenge" by randomly choosing a held-out set of 1,000 SCMs (denoted D_{test}) for which the model only saw `DATA` strings during training (Fig. 4). If the trained model can answer counterfactual queries about this test set, it means that it has (1) learned a more general counterfactual inference engine, (2) built a shared representation for interventional data and counterfactual inference, and (3) discovered the causal structure of SCMs within the D_{test} set from interventional data strings. In other words, it can *compose* the learned counterfactual inference engine from D_{train} strings with the discovered causal structure from interventional data strings in D_{test} .

3.4 Model architecture and training

We use the TransformerLens [28] implementation of a GPT 2-style transformer decoder. TransformerLens is a mechanistic interpretability library that exposes the model's internal activations for reading and editing purposes during the forward pass. We utilize these features of TransformerLens post-training when we decode SCM weights from model's residual stream activations and intervene on the SCM representation mid-computation.

Our transformer model has 12 layers, hidden size 512, 8 attention heads of size 64, MLP size 2048, GELU activation function, and "Pre-LN" type layer normalization. We use AdamW [24] with learning rate 1e-5, [0.9, 0.999] betas, 1e-8 epsilon, and 0.001 weight decay.

We train the model for 300 epochs with batch size 128. In each epoch, we draw 10 `DATA` and 10 `INFERENCE` strings per SCM from our text generative model (3.2), resulting in 1,160,980 strings per epoch. Note that SCMs in the D_{test} set have 0 `INFERENCE` strings in the training data. For the last 10 epochs, we reduce the learning rate to 1e-6 for better convergence.

4 Results

We show the following: (1) The trained model successfully generalizes to counterfactual queries about those SCMs that only had interventional data strings (section 4.1). (2) We can decode the SCM weights from residual stream activations within the model using linear and multi-layer perceptron (MLP) probes (section 4.2). (3) We can use these probes to manipulate the underlying SCM representation within residual activations using gradient descent with predictable effects on model's output (sections 4.3 and 4.4). Our results suggest that the next-token prediction objective drives the model to discover SCMs and learn an algorithm for counterfactual inference that generalizes within our SCM class.

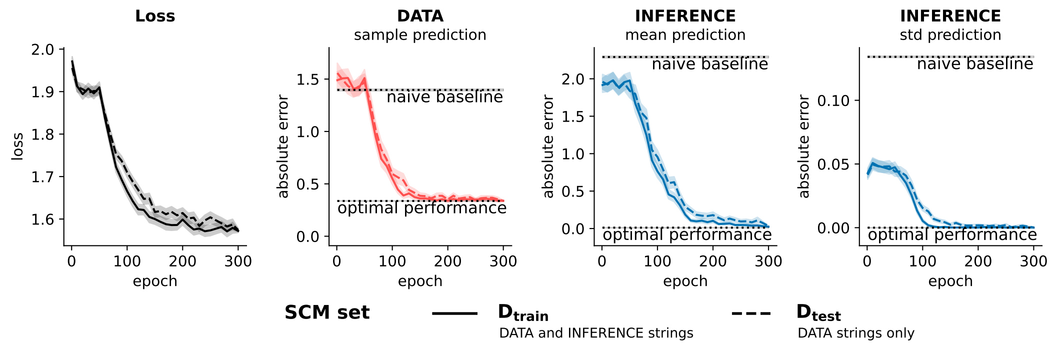


Figure 5: We track model loss and absolute error while predicting interventional `DATA` samples, counterfactual `INFERENCE` means and standard deviations for both D_{train} and D_{test} SCM sets. D_{test} SCM set metrics slightly lag behind, but eventually catch up with performance on the D_{train} set and arrive at near optimal performance. Naive baseline represents predicting the average value. Shaded regions show 95% bootstrapped confidence intervals for the mean across evaluation examples.

4.1 Transformer generalizes to SCMs with interventional data only

Our main behavioral test is the "generalization challenge" (3.3). The core intuition behind the challenge is that SCMs from the D_{test} set are referenced *only* within `DATA` strings during training. If the model can successfully generalize to counterfactual queries about the D_{test} SCMs during test time, it means that: (1) it has learnt a more general counterfactual inference engine (not just memorized the answers to counterfactual queries for the D_{train} set), and (2) it has discovered the causal structure of D_{test} set SCMs from noisy interventional data.

Fig. 5 shows the training trajectory for a few key metrics, including the loss as well as the absolute error for predicting interventional data samples and counterfactual means and standard deviations (see Appendix E for the calculation of absolute error). While the metrics slightly lag behind for the D_{test} SCM set, this is expected since the model is provided with *noisy* interventional samples and only half the number of strings for this set (i.e. 0 `INFERENCE` strings). However, the error for both D_{train} and D_{test} SCMs eventually reaches near-optimal performance.

Fig. 6 shows accuracy of the fully trained model as a function of number of interventions and observations provided in the string. Overall the network performs near optimally on all accuracy metrics. The only more challenging scenario is counterfactual mean prediction with *two observations* and 1-2 interventions. However, we believe that training the model further could reduce this error as the loss was still going down slowly when we terminated training. Note also that average absolute error of ≈ 0.1 is still quite good given that numbers are encoded using one decimal point precision in the text. Most importantly, there is no systematic performance gap between D_{train} and D_{test} SCM sets—the model generalizes successfully to counterfactual queries about D_{test} SCMs. Finally, our

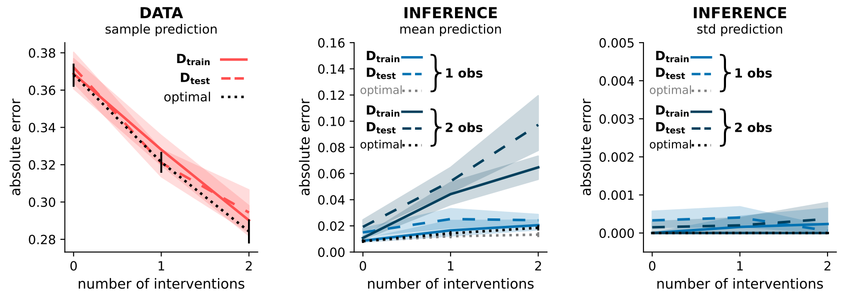


Figure 6: Trained model absolute errors as a function of number of interventions and observations in the query. The model performs near optimally on all prediction tasks (with caveat for `INFERENCE` mean prediction with two observations—see text for details). Shaded regions indicate 95% bootstrapped confidence intervals for the mean across evaluation examples.

results do not depend on the instance (see appendix B) or the fixed topological order assumption (see appendix C). We conclude that the network passes the generalization challenge, i.e. it does not simply memorize strings in the training set, but learns a more general counterfactual inference engine, discovers D_{test} SCMs from strings describing interventional data, and successfully composes the discovered structure with the inference engine.

4.2 SCM weights can be decoded from residual stream activations

Since the weight vector $\boldsymbol{w}$ (discussed in 3.1) fully defines an SCM within our class, we hypothesized that the model may map SCM indices to an internal representation that is going to be closely related to this vector. So we train "probes" [1, 3, 13] that map model's residual activations to elements of this vector. Namely, we train a separate probe for each of the 12 transformer layers to each of the 10 SCM weights, resulting in 120 separate classifiers. We chose to decode from residual activations at the *last SCM index position* as that position had the highest decoding accuracy—consistent with the idea that the model "pulls up" the SCM representation given the full SCM index for downstream computation.

Inspired by [22, 30], we train both linear and multi-layer perceptron (MLP) probes. Linear probes map the 512-dimensional layer l residual stream activations x^l directly to 3-way softmax output for each weight $w_{ij} \in \{-1, 0, 1\}$, while MLP probes include a 256-dimensional hidden layer:

$$p^{\text{linear}}_{w_{ij}}(x^l) = \text{softmax}(W^T x^l), \qquad W \in \mathbb{R}^{512\times 3} \tag{4}$$

$$p^{\text{MLP}}_{w_{ij}}(x^l) = \text{softmax}(W_2^T \text{ReLU}(W_1^T x^l), \qquad W_1 \in \mathbb{R}^{512\times 256}, \quad W_2 \in \mathbb{R}^{256\times 3} \tag{5}$$

We split D_{train} set (Fig. 4) into $D^{\text{probe}}_{\text{train}}$ (57,049 SCMs) and $D^{\text{probe}}_{\text{valid}}$ (1,000 SCMs) sets. We train the probes on $D^{\text{probe}}_{\text{train}}$, obtaining last SCM index position residual activations using `INFERENCE [SCM index]` input strings (see appendix E for details on probe training). We then test probe accuracy on the D_{test} SCM set (Fig. 7). SCM weights can be decoded above chance (33%) using both linear and MLP probes, suggesting that the transformer maps the arbitrary SCM index into a meaningful structured representation of the underlying SCM.

4.3 We can manipulate SCM representation in the residual stream

Probing results should be interpreted carefully, especially for non-linear probes [39]. Although successful decoding establishes the *presence* of information, we do not know whether or how this information is *used* by downstream computation. We hypothesized that probes do in fact recover at least parts of the model's representation of the underlying SCMs. If our hypothesis is correct, we should be able to *control* model's behavior by overwriting the SCM representation in the "network's mind" [10, 22, 26].

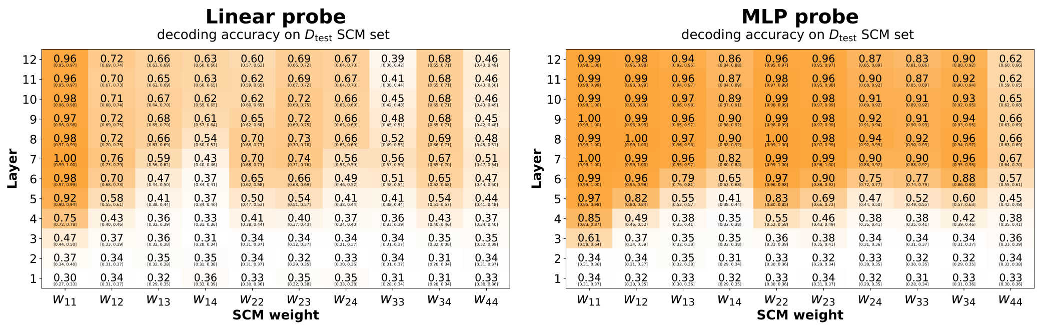


Figure 7: Average SCM weight decoding accuracy. Each cell corresponds to accuracy of the classifier that takes residual stream activations of a single layer and predicts SCM weight $w_{ij} \in \{-1, 0, 1\}$. Smaller values below indicate 95% bootstrapped confidence interval for the mean across the SCM set. Chance performance is 33%. By layer 5-6, most SCM weights can be decoded by linear or MLP probes above chance.

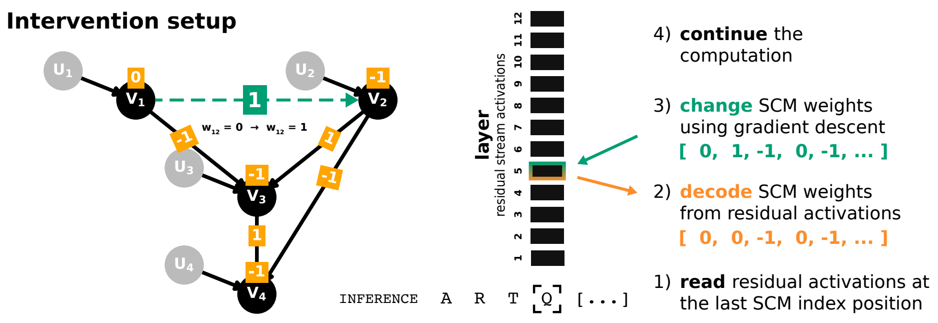

Intervention setup
U₁
U₂
U₃
U₄
V₁
V₂
V₃
V₄
w₁₂ = 0 → w₁₂ = 1
layer
residual stream activations
INFERENCE A R T Q [...]
4) continue the computation
3) change SCM weights using gradient descent
[0, 1, -1, 0, -1, ...]
2) decode SCM weights from residual activations
[0, 0, -1, 0, -1, ...]
1) read residual activations at the last SCM index position

Figure 8: Intervention setup.

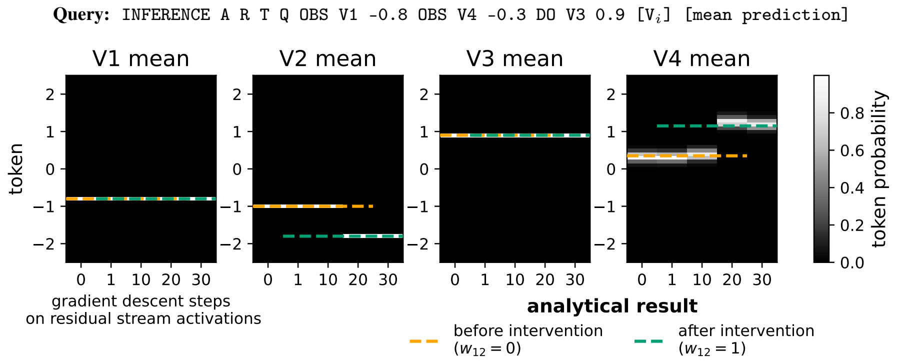


Figure 9: Linear probe intervention example ($w_{12} = 0 \to w_{12} = 1$). Testing whether our intervention on the representation is successful involves: (1) coming up with a causal *query*, (2) computing *analytical* estimates before ($w_{12} = 0$) and after ($w_{12} = 1$) the corresponding intervention in the ground truth SCM, and (3) checking whether the post-intervention model predictions agree with analytical estimates. Model prediction flips for V_2 and V_4, aligning with analytical estimates. The change in V_4 prediction is particularly interesting since weight w_{12} in the underlying SCM has only indirect effects on V_4 (see SCM in Fig 8).

To test our hypothesis, we use the gradient signal from the decoders to intervene on the SCM representation mid-computation (Fig. 8). Fig. 9 provides an intervention example using the linear probe. We see that we can indeed successfully manipulate the representation—the model predictions post-intervention on the residual stream align with analytical estimates for the corresponding change in the ground truth SCM. Further details on our intervention setup and an intervention example using an MLP probe can be found in appendix E. The fact that we can manipulate the SCM weights suggests that the probes do in fact recover a "real" representation of the referenced SCM.

4.4 Decoding accuracy does not imply controllability

We introduce a quantitative metric (the "intervention score") to investigate which SCM weights can be manipulated at which layers. Intuitively, the intervention score represents how well we can *control* model's behavior by changing its residual stream activations at a particular layer. Given an SCM, weight change ($w_{ij} \to w'$), residual stream intervention at layer l ($l \to l'$), and queried variable V_k:

$$\begin{array}{r}\text{intervention}\\ \text{score}\end{array} = \frac{1}{\beta}\Big(\underbrace{p^{l\to l'}_{\text{model}}(\text{token}(E[V_k^{w_{ij}\to w'}]))}_{\substack{\text{probability assigned to the correct token}\\ \text{after residual stream intervention}}} - \underbrace{p_{\text{model}}(\text{token}(E[V_k^{w_{ij}\to w'}]))}_{\substack{\text{probability already assigned to the correct}\\ \text{token before residual stream intervention}}} \Big) \quad (6)$$

Note that this score can be negative when the model assigns some non-zero probability to the correct post-intervention token *before* the intervention, but the intervention on activations ($l \to l'$) lowers that probability. Moreover, the model generally hedges its bets, so we normalize the quantity in the big brackets by the average max probability β model assigns during normal operation (≈ 0.89). An average intervention score of 1 then corresponds to “perfect” interventions—the model gives *correct* answers with the same average *confidence* as during normal operation. Further details on computing intervention scores can be found in Appendix E.

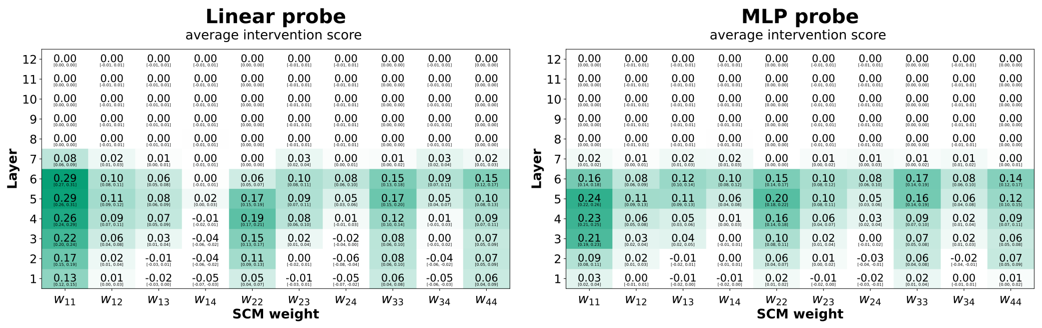


Figure 10: Intervention scores are not coupled to decoding accuracy beyond layer 7 (compare to Fig. 7). Each cell corresponds to the average intervention score for weight w_{ij} and layer l. Values in smaller font indicate 95% bootstrapped confidence interval for the mean across queries, queried variables V_i and weight changes $w_{ij} \to w'$.

The intervention scores in Fig. 10 suggest two things. First, the decoding accuracy is decoupled from the intervention scores beyond layer 7—changing residual stream activations (at the last SCM index position) beyond layer 7 does not have an effect on model output. Although the model does not appear to *use* the information beyond layer 7, it does not destroy it, either. Second, while the linear probe has lower overall decoding accuracy, it achieves *higher* intervention scores for some layer-and-weight pairs (particularly, for w_{11}, see Fig. 16 for a contrast between linear and MLP intervention scores). Overall, the quantitative intervention results underscore the broader point that the presence of information does not imply that the model is *using* that information.

5 Discussion

By demonstrating that transformers can discover causal structure and learn causal inference engines solely through next-token prediction, we challenge fundamental assumptions about the limitations of neural networks trained to predict “passive” streams of data [34, 36, 35, 49]. While the learning setup may be passive, the data is often rich with implicit causal information that a purely predictive (but finite) model may want to discover and exploit. This raises broader questions that apply to all foundation models. For instance, do video generation models (e.g. OpenAI’s Sora [5]) learn simulators of the world—corresponding, at least informally, to $\mathcal{L}_2$ of Pearl’s Causal Hierarchy? Could we find abstract representations of physical objects and agents within their activations that we could intervene upon? If so, could we augment these models with causal reasoning capacities—allowing them to reason counterfactually ($\mathcal{L}_3$)? We are excited to explore these questions in future work.

The primary limitations of our work are the artificial language setup and the constrained SCM class. Our setup, while designed to emulate how natural language implicitly conveys causal information about interventions and inferences, does not capture the full richness of real-world text and causal structures. However, this constrained setting was crucial for providing a clear and unambiguous “existence proof” that statistical prediction can, under the right conditions, give rise to causal models and reasoning. Future work could explore to what extent our results generalize to more naturalistic settings with vague natural language descriptions of the causal scenarios and more complex causal model classes, including non-linear SCMs.

Conclusion. Our work provides a concrete existence proof that statistical prediction objectives can drive the emergence of causal models and causal reasoning. Through careful behavioral tests and mechanistic analysis, we demonstrated that next-token prediction can yield models that discover causal structure, build internal causal representations, and perform counterfactual inference. While our study focuses on a constrained setting, it opens new questions about the causal capabilities that may be emerging in today’s foundation models trained on vast datasets.

References

[1] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[2] Elias Bareinboim, Juan D Correa, Duligur Ibeling, and Thomas Icard. On Pearl's hierarchy and the foundations of causal inference. In *Probabilistic and causal inference: the works of judea pearl*, pages 507–556. 2022.

[3] Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022. Publisher: MIT Press One Broadway, 12th Floor, Cambridge, Massachusetts 02142, USA . . . .

[4] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big?. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

[5] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL `https://openai.com/research/video-generation-models-as-world-simulators`.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and others. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Lewis Carroll. *Alice's Adventures in Wonderland*. Macmillan, London, 1865.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[9] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread*, 2021.

[10] Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems*, 34:9574–9586, 2021.

[11] Alison Gopnik and Henry M Wellman. Reconstructing constructivism: causal models, Bayesian learning mechanisms, and the theory theory. *Psychological bulletin*, 138(6):1085, 2012. Publisher: American Psychological Association.

[12] Alison Gopnik, Clark Glymour, David M Sobel, Laura E Schulz, Tamar Kushnir, and David Danks. A theory of causal learning in children: causal maps and Bayes nets. *Psychological review*, 111(1):3, 2004. Publisher: American Psychological Association.

[13] Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

[14] Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36:17643–17668, 2023.

[15] Zhijing Jin, Yuen Chen, Felix Leeb, Luigi Gresele, Ojasv Kamal, Zhiheng Lyu, Kevin Blin, Fernando Gonzalez Adauto, Max Kleiman-Weiner, Mrinmaya Sachan, and others. Cladder: Assessing causal reasoning in language models. *Advances in Neural Information Processing Systems*, 36:31038–31065, 2023.

[16] Zhijing Jin, Jiarui Liu, Zhiheng Lyu, Spencer Poff, Mrinmaya Sachan, Rada Mihalcea, Mona Diab, and Bernhard Schölkopf. Can large language models infer causation from correlation? *arXiv preprint arXiv:2306.05836*, 2023.

[17] Nitish Joshi, Abulhair Saparov, Yixin Wang, and He He. Llms are prone to fallacies in causal inference. *arXiv preprint arXiv:2406.12158*, 2024.

[18] Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality. *arXiv preprint arXiv:2305.00050*, 2023.

[19] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017. Publisher: Cambridge University Press.

[20] Andrew Lampinen, Stephanie Chan, Ishita Dasgupta, Andrew Nam, and Jane Wang. Passive learning of active causal strategies in agents and language models. *Advances in Neural Information Processing Systems*, 36:1283–1297, 2023.

[21] Belinda Z Li, Maxwell Nye, and Jacob Andreas. Implicit Representations of Meaning in Neural Language Models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827, 2021.

[22] Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *ICLR*, 2023. Publisher: ICLR.

[23] Zichao Li, Yanshuai Cao, and Jackie CK Cheung. Do LLMs build world representations? Probing through the lens of state abstraction. *Advances in Neural Information Processing Systems*, 37:98009–98032, 2024.

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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

[26] Raphaël Millière and Cameron Buckner. Interventionist Methods for Interpreting Deep Neural Networks. In Gualtiero Piccinini, editor, *Neurocognitive Foundations of Mind*. Routledge.

[27] Melanie Mitchell and David C. Krakauer. The debate over understanding in AI's large language models. *Proceedings of the National Academy of Sciences*, 120(13):e2215907120, March 2023. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2215907120. URL `https://pnas.org/doi/10.1073/pnas.2215907120`.

[28] Neel Nanda and Joseph Bloom. TransformerLens, 2022. URL `https://github.com/TransformerLensOrg/TransformerLens`.

[29] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, October 2023. URL `http://arxiv.org/abs/2301.05217`. arXiv:2301.05217 [cs].

[30] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023.

[31] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

[32] Nick Pawlowski, Daniel Coelho de Castro, and Ben Glocker. Deep structural causal models for tractable counterfactual inference. *Advances in neural information processing systems*, 33:857–869, 2020.

[33] Judea Pearl. *Causality*. Cambridge university press, 2009.

[34] Judea Pearl. Theoretical impediments to machine learning with seven sparks from the causal revolution. *arXiv preprint arXiv:1801.04016*, 2018.

[35] Judea Pearl. Judea pearl, AI, and causality: What role do statisticians play? *Interviewed by D. Mackenzie). Amstat News (AI Special Issue)*, 555:6–9, 2023.

[36] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018.

[37] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and others. Improving language understanding by generative pre-training. 2018. Publisher: San Francisco, CA, USA.

[38] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-Shot Text-to-Image Generation, February 2021. URL `http://arxiv.org/abs/2102.12092`. arXiv:2102.12092 [cs].

[39] J. Brendan Ritchie, David Michael Kaplan, and Colin Klein. Decoding the Brain: Neural Representation and the Limits of Multivariate Pattern Analysis in Cognitive Neuroscience. *The British Journal for the Philosophy of Science*, 70(2):581–607, June 2019. ISSN 0007-0882, 1464-3537. doi: 10.1093/bjps/axx023. URL `https://www.journals.uchicago.edu/doi/10.1093/bjps/axx023`.

[40] Angelika Romanou, Syrielle Montariol, Debjit Paul, Leo Laugier, Karl Aberer, and Antoine Bosselut. CRAB: Assessing the Strength of Causal Relationships Between Real-world Events, 2023. URL `https://arxiv.org/abs/2311.04284`. _eprint: 2311.04284.

[41] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5): 612–634, 2021. Publisher: IEEE.

[42] Steven Sloman and Steven A Sloman. *Causal models: How people think about the world and its alternatives*. Oxford University Press, 2009.

[43] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008.

[44] Kevin Xia and Elias Bareinboim. Neural Causal Abstractions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18):20585–20595, March 2024. doi: 10.1609/aaai.v38i18.30044. URL `https://ojs.aaai.org/index.php/AAAI/article/view/30044`.

[45] Kevin Xia, Kai-Zhan Lee, Yoshua Bengio, and Elias Bareinboim. The causal-neural connection: Expressiveness, learnability, and inference. *Advances in Neural Information Processing Systems*, 34:10823–10836, 2021.

[46] Kevin Xia, Yushu Pan, and Elias Bareinboim. Neural causal models for counterfactual identification and estimation. *arXiv preprint arXiv:2210.00035*, 2022.

[47] Blaise Agüera y Arcas. Do large language models understand us? *Daedalus*, 151(2):183–197, 2022. Publisher: MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ....

[48] Ilker Yildirim and LA Paul. From task structures to world models: what do LLMs know? *Trends in Cognitive Sciences*, 2024. Publisher: Elsevier.

[49] Matej Zečević, Moritz Willig, Devendra Singh Dhami, and Kristian Kersting. Causal Parrots: Large Language Models May Talk Causality But Are Not Causal, August 2023. URL `http://arxiv.org/abs/2308.13067`. arXiv:2308.13067 [cs].

[50] Cheng Zhang, Stefan Bauer, Paul Bennett, Jiangfeng Gao, Wenbo Gong, Agrin Hilmkil, Joel Jennings, Chao Ma, Tom Minka, Nick Pawlowski, and others. Understanding causality with large language models: Feasibility and opportunities. *arXiv preprint arXiv:2304.05524*, 2023.

A Code

`https://anonymous.4open.science/r/causal-discovery-and-inference-through-next-token-prediction`.

B Additional Instances

As a robustness check, we trained three additional model instances from different random seeds (Fig. 11). All instances successfully generalize to counterfactual queries about D_{test} SCMs, reaching near-optimal performance despite somewhat idiosyncratic training trajectories. ‘

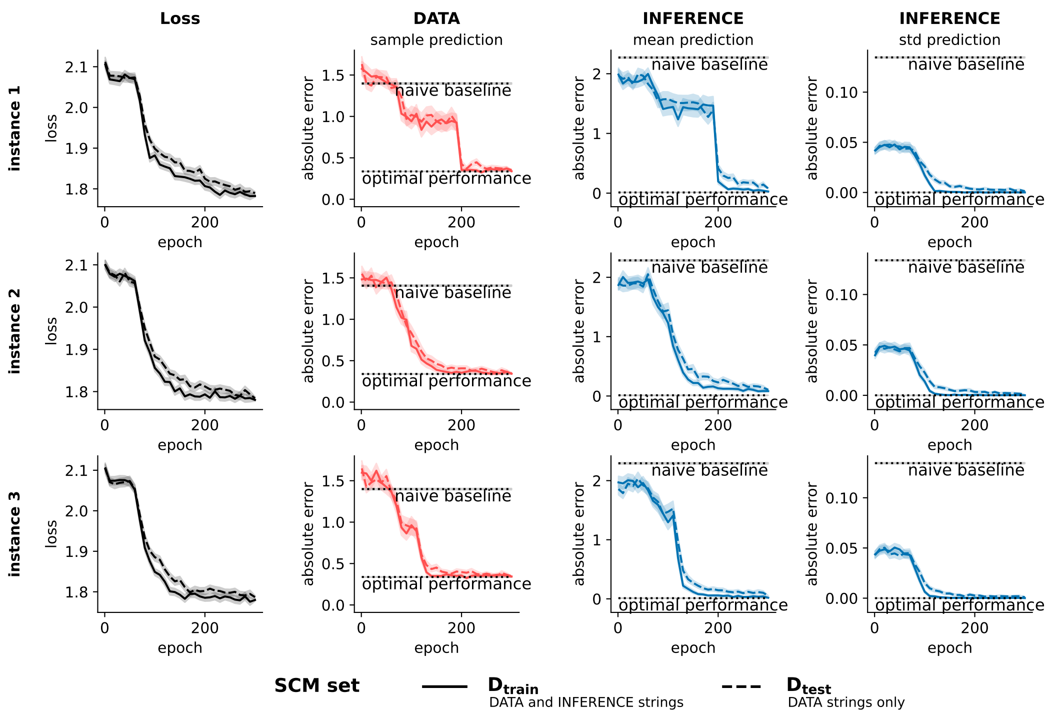


Figure 11: Three additional model instances (different random seeds) all pass the generalization challenge, reaching near-optimal counterfactual inference performance on D_{test} SCMs. Same metrics as in Fig. 5.

C Causal Discovery Without a Fixed Topological Order

In our main results, the topological order of the variables was fixed across all SCM instances ($V_1 \prec V_2 \prec V_3 \prec V_4$). For instance, `V1` always referred to the root node. So the model could learn to exploit this order, substantially simplifying the causal discovery problem.

To make sure that our results do not depend on the fixed variable order, we trained a model instance where the variable names are randomly permuted in each SCM, i.e. `V1` can refer to the root node, or it can be the last variable in the SCM.

Figures 12 and 13 demonstrate that the model generalizes to counterfactual queries about the D_{test} SCMs even in this more challenging (and more typical) causal discovery scenario where the topological order cannot be assumed. Interestingly, consistent with our intuition that this scenario should be more difficult, the model takes much longer to converge and reach optimal performance (over 1000 epochs compared to 300 for the main results).

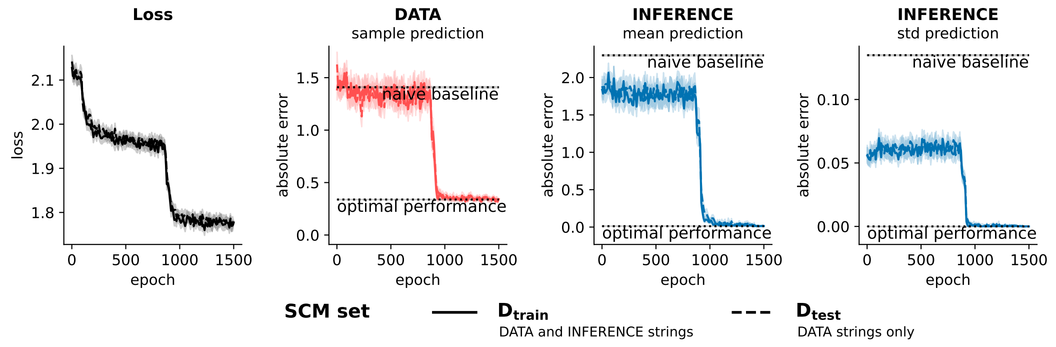


Figure 12: Setting without a fixed topological order of the variables. Interestingly, it takes much longer for the model to converge in this setting (over 1000 epochs vs. 300 epochs), but our results do not depend on the fixed topological order. Same metrics as in Fig. 5.

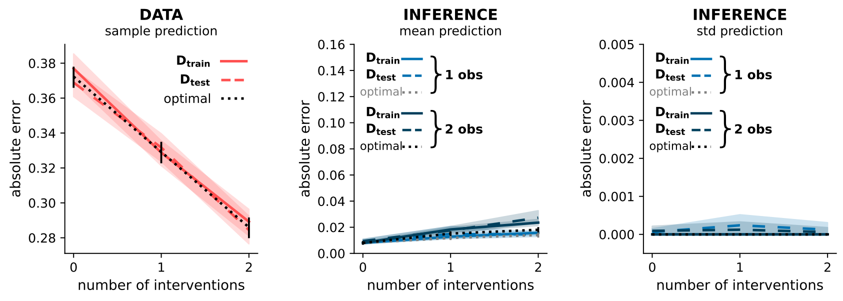


Figure 13: Last epoch performance without a fixed topological order of the variables. Same metrics as in Fig. 6.

D Methods Appendix

Analytic counterfactual inference in linear Gaussian SCMs

We implement analytic counterfactual inference for linear Gaussian SCMs, computing the quantity $P(\boldsymbol{V}_{\boldsymbol{x}_{\text{int}}}|\boldsymbol{Y} = \boldsymbol{y}_{\text{obs}})$, where $\boldsymbol{Y} = \boldsymbol{y}_{\text{obs}}$ are observed values in the factual world, and $\boldsymbol{x}_{\text{int}}$ represents the intervention $do(\boldsymbol{X} = \boldsymbol{x}_{\text{int}})$ in the counterfactual world. Counterfactual inference involves three steps [33]:

1. **Abduction** – finding the posterior over the background variables given observed values $p(\boldsymbol{U}|\boldsymbol{X} = \boldsymbol{x}_{\text{obs}})$.
2. **Action** – modifying the SCM according to the intervention; in our case, this corresponds to setting a subset of the endogenous variables to constant values in the structural equations $\boldsymbol{X} := \boldsymbol{x}_{\text{int}}$ and setting incoming weights to zero for intervened variables $\mathbf{w} \to \mathbf{w}_{\boldsymbol{x}_{\text{int}}}$.
3. **Prediction** – given the posterior over background variables $\boldsymbol{U}$ and the modified SCM, computing the counterfactual distribution over the endogenous variables $\boldsymbol{V}$.

To analytically compute answers in steps (1) and (3) for a Gaussian linear SCM, it is useful to define the "total effects" matrix T. Let T_{ji} denote the total effect of the background variable U_i on the endogenous variable V_j, where $i, j \in \{1, \ldots, m\}$ and m is the number of variables (in our case, $m = 4$). The entries of the total effects matrix $\boldsymbol{T}$ are defined recursively as follows:

1. **Base Cases:** For $i = 1, \ldots, m$:
   - $T_{ii} = 1$ (Direct effect $U_i \to V_i$).
   - $T_{ji} = 0$ for all $j < i$ (No effect from U_i on preceding variables).

2. **Recursive Step:** For all $j > i$:

$$T_{ji} = \sum_k T_{ki} \times w_{kj}$$

Note that $\boldsymbol{T}$ is a lower triangular matrix with ones on its diagonal. We can then use it to define the structural equations in matrix form:

$$\boldsymbol{V} = \boldsymbol{T}\boldsymbol{U} \tag{7}$$

Let $\boldsymbol{x}_{\text{obs}} \in \mathbb{R}^n$ be the vector of n observed values. Let $\mathcal{O} = \{o_1, o_2, \dots, o_n\}$ be the ordered set of indices such that the r-th element of $\boldsymbol{x}_{\text{obs}}$, denoted $(\boldsymbol{x}_{\text{obs}})_r$, corresponds to the observed value of variable V_{o_r}. That is, $V_{o_r} = (\boldsymbol{x}_{\text{obs}})_r$ for $r = 1, \dots, n$.

We construct the observation matrix $\boldsymbol{H} \in \mathbb{R}^{n \times m}$ by selecting the rows of the total effects matrix $\boldsymbol{T}$ corresponding to the indices in $\mathcal{O}$, maintaining the order defined by $\mathcal{O}$.

The posterior over the background variables given the observations $p(\boldsymbol{U}|\boldsymbol{x}_{\text{obs}}) = \mathcal{N}(\boldsymbol{U}; \boldsymbol{\mu}_{\text{post}}, \boldsymbol{\Sigma}_{\text{post}})$ can be computed from the prior mean $\boldsymbol{\mu}_{\text{prior}}$ and covariance $\boldsymbol{\Sigma}_{\text{prior}}$ using standard Gaussian conditioning:

$$\boldsymbol{K} = \boldsymbol{\Sigma}_{\text{prior}} \boldsymbol{H}^T (\boldsymbol{H} \boldsymbol{\Sigma}_{\text{prior}} \boldsymbol{H}^T)^{-1} \tag{8}$$

$$\boldsymbol{\mu}_{\text{post}} = \boldsymbol{\mu}_{\text{prior}} + \boldsymbol{K}(\boldsymbol{x}_{\text{obs}} - \boldsymbol{H}\boldsymbol{\mu}_{\text{prior}}) \tag{9}$$

$$\boldsymbol{\Sigma}_{\text{post}} = \boldsymbol{\Sigma}_{\text{prior}} - \boldsymbol{K}\boldsymbol{H}\boldsymbol{\Sigma}_{\text{prior}} \tag{10}$$

where $\boldsymbol{K}$ is the Kalman gain matrix.

Let the counterfactual mean $\boldsymbol{\mu}_{\text{counter}}$ and covariance matrix $\boldsymbol{\Sigma}_{\text{counter}}$ parameterize the counterfactual distribution $P(\boldsymbol{V}_{\boldsymbol{x}_{\text{int}}}|\boldsymbol{Y} = \boldsymbol{y}_{\text{obs}}) = \mathcal{N}(\boldsymbol{V}_{\boldsymbol{x}_{\text{int}}}; \boldsymbol{\mu}_{\text{counter}}, \boldsymbol{\Sigma}_{\text{counter}})$.

To compute the counterfactual mean $\boldsymbol{\mu}_{\text{counter}}$, we change the structural equations to incorporate interventions as assignments to constants (intervened values), and compute the variable values in topological order.

To compute the covariance matrix $\boldsymbol{\Sigma}_{\text{counter}}$, we first compute a modified posterior covariance matrix over the background terms $\tilde{\boldsymbol{\Sigma}}_{\text{post}}$ where we set all the rows and columns of the intervened variable to 0. We also compute the total effects matrix for intervened weights $\boldsymbol{T}_{\boldsymbol{x}_{\text{int}}}$. Then $\boldsymbol{\Sigma}_{\text{counter}} = \tilde{\boldsymbol{T}}_{\boldsymbol{x}_{\text{int}}} \tilde{\boldsymbol{\Sigma}}_{\text{post}} \tilde{\boldsymbol{T}}^T_{\boldsymbol{x}_{\text{int}}}$.

For `DATA` strings, we take one sample from the counterfactual distribution. For `INFERENCE` strings, we provide counterfactual means and standard deviations (square root of $\boldsymbol{\Sigma}_{\text{counter}}$ diagonal values).

E Results Appendix

4.1 Appendix: Details on absolute error calculation

To calculate absolute error when predicting `DATA` samples, we prompt the model with:

```
DATA [SCM index] [interventions] [query variable] → [sample prediction]
```

and convert the highest probability next token to a numerical value. We then calculate mean absolute error (MAE) as $\text{MAE}_{data} = \frac{1}{n}\sum_i^n |x_i - \hat{x}_i|$, where x_i is sampled value as presented in the text, $\hat{x}_i$ is model predicted value and n is the number of examples queries we are averaging over.

To calculate prediction accuracy for `INFERENCE` mean and standard deviation, we prompt the model with:

```
INFERENCE [SCM index] [observations & interventions] [query variable]
              → [mean prediction] [std prediction]
```

and auto-regressively predict two tokens. We interpret the first as the counterfactual mean prediction and the second as the standard deviation prediction. We convert both of these tokens to numerical values and use the same equations for mean absolute error (but now the ground truth tokens represent analytically derived mean and standard deviation).

In Fig. 5 we present model evaluations at every 10 epochs. For each evaluated epoch and metric we sample 1,000 D_{train} and 1,000 D_{test} queries (where query is a combination of an SCM index, set of interventions, and a set of observations for `INFERENCE` strings). For the last epoch (zoomed in in Fig. 6), we sample 10,000 D_{train} and 10,000 D_{test} queries to get better estimates. For each query, we run the model with four different queried variables. We exclude *intervened* queried variables from the analysis since the answer is trivial (i.e. copy the intervened value from the query or predict zero for standard deviation) and the model makes no errors. Excluding intervened queried variables leaves ≈ 2.9 strings per query, resulting in around 5,800 for each epoch (last one has about 58,000).

4.2 Appendix: Probe training and probe accuracy on $D^{\text{probe}}_{\text{valid}}$ SCM set

We probe the residual stream *pre-activations* in the transformer model. So layer 1 activations correspond to token embeddings with positional embeddings, layer 2 activations correspond to the output of layer 1, etc.

We train both linear and MLP probes using AdamW optimizer [24] with batch size of 128, learning rate 0.001, eps 1e-08, betas [0.9, 0.99] and weight decay 0.0. Due to different convergence rates, linear probes were trained for 20 epochs, and MLP probes were trained for 40 epochs. We did not observe any signs of overfitting for either of the probes based on the validation set performance.

We trained on $D^{\text{probe}}_{\text{train}}$ and tested on D_{test} SCM sets (see Fig. 4.2). In Fig. 14 we provide probe decoding accuracy on the $D^{\text{probe}}_{\text{valid}}$ set. This is an easier generalization for the probes than the case in the main text since validation set examples come from the same D_{train} set used to train the probes.

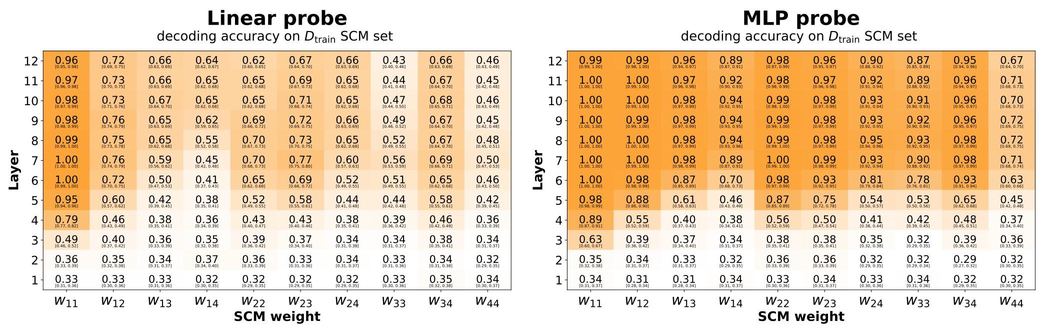


Figure 14: SCM weight decoding validation accuracy on $D^{\text{probe}}_{\text{valid}}$ set. Probe accuracy is slightly higher than for D_{test} set presented in the main text. This is expected because the probe does not need to generalize as much—both $D^{\text{probe}}_{\text{train}}$ and $D^{\text{probe}}_{\text{valid}}$ come from the same D_{train} SCM set. Smaller values below indicate 95% bootstrapped confidence interval for the mean across the SCM set.

4.3 Appendix: Intervention algorithm and MLP intervention example

For all intervention experiments we only consider the `INFERENCE` counterfactual mean prediction task since that is the most challenging for the model (see Fig. 6).

Our intervention setup (Algorithm 1) is directly inspired by Li et al. [22].

For linear intervention example (Fig. 9), we intervened on layer 3 with learning rate $\alpha = 0.08$. For MLP example (Fig. 15), we intervened on layer 5 with learning rate $\alpha = 0.08$.

4.4 Appendix: Intervention score details

For intervention score calculation, we only consider mean prediction in `INFERENCE` strings. Note also that we only consider cases where post-intervention analytic mean is different from the pre-intervention analytic mean ($E[V_k^{w_{ij}\to w'}] \neq E[V_k^{w_{ij}}]$), i.e. we only consider variables for which the ground truth intervention "does something".

We set normalization factor $\beta = 0.89$ in eq. 6 by computing the average max probability the model assigns when predicting the mean.

Algorithm 1 Gradient-Based Residual Stream Intervention

```
Require: Model $M$, probe $P$, weight index $(i,j)$, target weight $w'$, layer $l$, learning rate $\alpha$, steps $k$
 1: $\mathbf{a} \leftarrow \mathbf{h}_\ell[:, 4, :].clone()$        ▷ Extract activations at the last SCM index position
 2: Adam($\mathbf{a}, lr = \alpha$)        ▷ Initialize optimizer
 3: $\mathbf{y}_{\text{prev}} \leftarrow P(\mathbf{a}, \ell)$        ▷ Store initial probe predictions
 4: for step $i = 1$ to $k$ do
 5:     $\mathbf{y} \leftarrow P(\mathbf{a}, \ell)$        ▷ Current probe predictions
 6:     $\mathcal{L}_{\text{target}} \leftarrow \text{CrossEntropy}(\mathbf{y}_{ij}, w')$        ▷ Push target weight probe prediction to $w'$
 7:     $\mathcal{L}_{\text{others}} \leftarrow \text{KL}(\mathbf{y}_{\setminus ij}, \mathbf{y}_{\text{prev}\setminus ij})$        ▷ Push other probes to maintain initial predictions
 8:     $\mathcal{L} \leftarrow \mathcal{L}_{\text{target}} + \mathcal{L}_{\text{others}}$
 9:     Update $\mathbf{a}$ using gradient descent on $\mathcal{L}$
10: end for
11: $\mathbf{h}_\ell[:, 4, :] \leftarrow \mathbf{a}$
12: return Model output logits
```

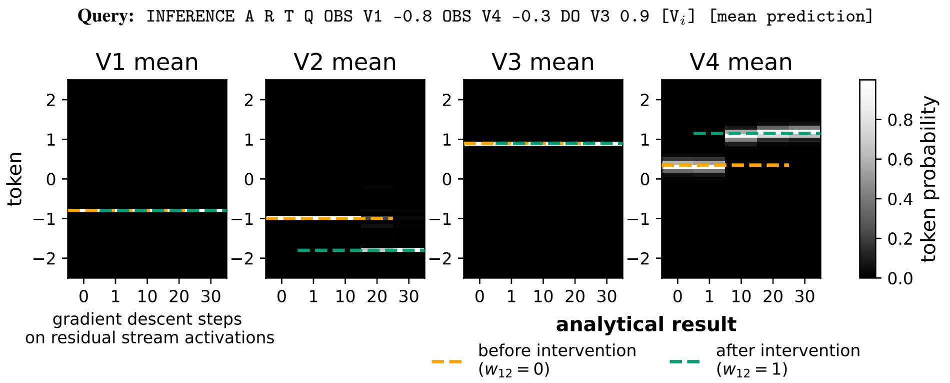


Figure 15: MLP probe intervention example ($w_{12} = 0 \rightarrow w_{12} = 1$) for the same query as in the main text (Fig. 9).

We first generate a set of 200 queries that are fixed across all layers and SCM weights. We then evaluate intervention scores for each queries and for each layer-SCM weight combination by changing the weight to all possible values $w_{ij} \rightarrow w' \in \{-1, 0, 1\}$ with number of steps $k = 30$ and learning rate $\alpha = 0.08$. We found these values to results in generally good interventions qualitatively and we found quantitatively that intervention scores were quite robust to, say, changing number of steps to $k = 20$. However, in future work we should consider a much more extensive intervention hyperparameter sweep. For instance, it may be that certain layer-SCM weight combinations require a completely different number of steps k or learning rate α.

F Miscellaneous

Visualizing trained model embeddings

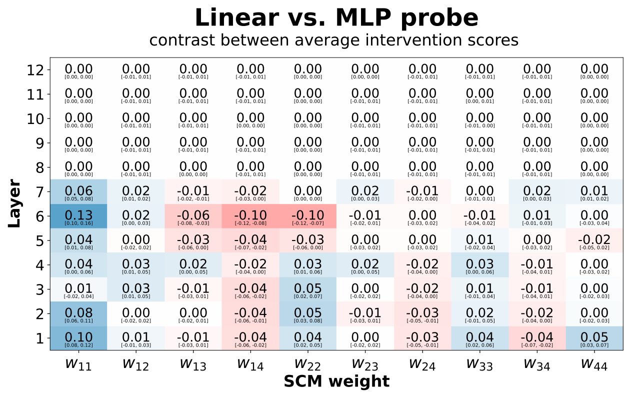


Figure 16: Intervention score contrast between linear (blue) and MLP probes (red) (intervention score for MLP probe subtracted from the score for linear probe). While linear probe has lower decoding accuracy (Fig. 7 and 14), MLP probe does not strictly dominate it in terms of intervention scores. Smaller values below indicate 95% bootstrapped confidence interval across queries, queried variables V_i and weight changes $w_{ij} \to w'$.

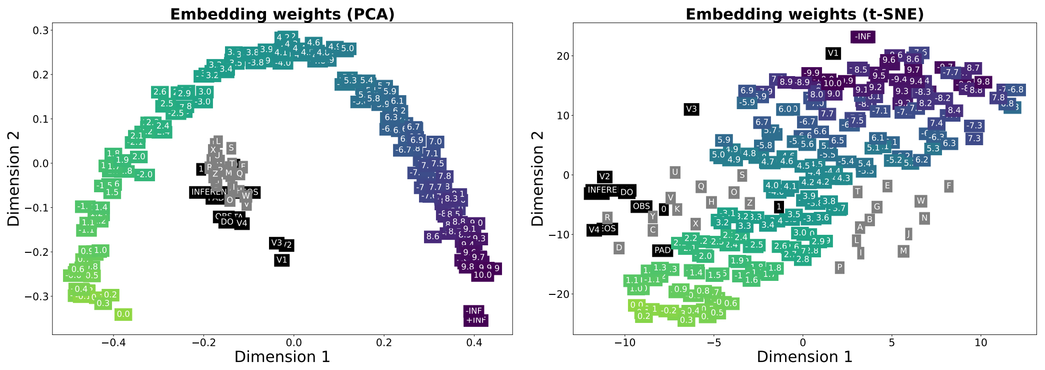


Figure 17: We visualize the trained transformer token embedding weights using PCA and t-SNE [43]. The model seems to learn the number line (numerical tokens color coded based on their absolute value). The model keeps negative and positive versions of the same absolute value nearby, possibly because the prediction task requires sign flipping (when parent value is negative and weight is -1).