

840 A Posterior predictive distribution

841 *Proof of posterior predictive distribution.* Let $\Theta \subset \mathbb{R}$, $B \subseteq \mathbb{R}$ and $\beta \in B$. Define the observational
842 data $\mathbf{x} = (x_1, y_1, \dots, x_n, y_n)$. Then, with query $:= (x_1, y_1, \dots, x_n, y_n, z, x^{\text{CF}})$,

$$\begin{aligned} & p_{Y_Z^{\text{CF}}}(y_z^{\text{CF}}|\text{query}) \\ &= \int_{\Theta} p_{Y_Z^{\text{CF}}}(y_z^{\text{CF}}|\text{query}, \theta) \pi(\theta|\text{query}) d\theta \\ &= \int_{\Theta} p_{Y_Z^{\text{CF}}}(y_z^{\text{CF}}|\mathbf{x}, \theta, x_z^{\text{CF}}, z) \pi(\theta|\mathbf{x}, x_z^{\text{CF}}, z) d\theta \\ &\stackrel{\text{II}}{=} \int_{\Theta} p_{Y_Z^{\text{CF}}}(y_z^{\text{CF}}|\mathbf{x}, \theta, x_z^{\text{CF}}, z) \pi(\theta|\mathbf{x}) d\theta \end{aligned} \quad (7)$$

$$\begin{aligned} &= \int_{\Theta} \int_B p_{Y_Z^{\text{CF}}}(\beta(x_z^{\text{CF}} - x_z) + y_z|\mathbf{x}, \theta, x_z^{\text{CF}}, z, \beta) p_{\beta}(\beta|\mathbf{x}, \theta, x_z^{\text{CF}}, z) \pi(\theta|\mathbf{x}) d\beta d\theta \\ &\stackrel{\text{dF}}{=} \int_{\Theta} \int_B p_{Y_Z^{\text{CF}}}(\beta(x_z^{\text{CF}} - x_z) + y_z|x_i, y_i, \theta, x_z^{\text{CF}}, z, \beta) p_{\beta}(\beta|\mathbf{x}, \theta) \pi(\theta|\mathbf{x}) d\beta d\theta \quad (8) \\ &= \int_{\Theta} \int_B \delta(Y_Z^{\text{CF}} = \beta x_z^{\text{CF}} + y_z - \beta x_z) p_{\beta}(\beta|\mathbf{x}, \theta) \pi(\theta|\mathbf{x}) d\beta d\theta, \end{aligned}$$

843 where (7) follows by independence, as $\theta \perp\!\!\!\perp (X^{\text{CF}}, Z)$, and (8) by de Finetti [de Finetti, 1931, Klenke,
844 2008], as the in-context examples $((x_1, y_1), \dots, (x_n, y_n))$ are exchangeable. \square

845 *Mean and variance in linear additive framework.* Both follow immediately by the tower law (III) of
846 iterated expectations,

$$\begin{aligned} \mathbb{E}[Y^{\text{CF}}] &= \mathbb{E}[\beta X^{\text{CF}} + U_Y] \\ &\stackrel{\text{III}}{=} \mathbb{E}[\mathbb{E}[\beta|\theta]] \mathbb{E}[X^{\text{CF}}] + \mathbb{E}[\mathbb{E}[U_Y|\theta]] \\ &= \mathbb{E}[\theta] = 0 \\ \text{Var}(Y^{\text{CF}}) &= \mathbb{E}[(Y^{\text{CF}})^2] = \mathbb{E}[(\beta X^{\text{CF}} + U_Y)^2] \\ &= \mathbb{E}[(\beta X^{\text{CF}})^2 + 2\beta X^{\text{CF}} U_Y + U_Y^2] \\ &\stackrel{\text{II}}{=} \mathbb{E}[\beta^2] \mathbb{E}[(X^{\text{CF}})^2] + 2\mathbb{E}[\beta U_Y] \mathbb{E}[X^{\text{CF}}] + \mathbb{E}[U_Y^2] \\ &\stackrel{\text{III}}{=} \mathbb{E}[\mathbb{E}[\beta^2|\theta]] \text{Var}(X^{\text{CF}}) + \mathbb{E}[\mathbb{E}[U_Y^2|\theta]] \\ &= \mathbb{E}[1 + \theta^2] (\text{Var}(X^{\text{CF}}) + 1) \\ &= (1 + \text{Var}(\theta))(12 + 1) = 13^2 = 169. \end{aligned}$$

847 The proof for the variance under the multiplicative extension, $\text{Var}(\beta X^{\text{CF}} U_Y)$, is analogous. \square

848 B Transformation lemma

849 *Proof of Lemma I* As T is invertible in U given $f(X)$,

$$Y^{\text{CF}} = T(f(X^{\text{CF}}), U) = T(f(X^{\text{CF}}), T^{-1}(f(X), Y)).$$

850 We require injectivity in u to uniquely determine u from $(f(X), Y)$. Else, $Y^{\text{CF}} = T(f(X^{\text{CF}}), u)$ is
851 ambiguous and counterfactuals are ill-defined. Thus, the counterfactual completion writes

$$Y^{\text{CF}} = h(f(X), f(X^{\text{CF}}), Y)$$

852 for $h : \mathcal{F} \times \mathcal{F} \times \mathcal{U} \longrightarrow \mathcal{Y}$. \square

C Training details

We construct synthetic datasets with fixed noise. Conditional on a uniformly distributed latent parameter $\theta_i \in \Theta$, we draw $n_i \sim \mathcal{U}(\{1, \dots, 50\})$ observational data points with noise $\mathbf{U}_X^{i,j} | \theta_i, \mathbf{U}_Y^{i,j} | \theta_i \in \mathbb{R}^E, j \in \{1, \dots, n_i\}$ and weights $\beta_i | \theta_i \in \mathbb{R}^E, i \in \{1, \dots, N \cdot B\}$ to enforce *non-i.i.d.* data. We choose $E = 5$. Next, we sample $N \cdot B$ data points with

$$\begin{aligned} \theta &\sim \mathcal{U}([-6, 6]^E) & \beta | \theta &\sim \mathcal{N}(\theta, \mathbf{I}_E) \\ \mathbf{U} | \theta &\sim \mathcal{N}(\theta, \mathbf{I}_E) & \mathbf{X}^{\text{CF}} &\sim \mathcal{U}([-6, 6]^E) \end{aligned} \quad (9)$$

for $N = 200000$ training steps of batch size $B = 64$. We set

$$\mathbf{Y} = \beta \odot \mathbf{U}_X + \mathbf{U}_Y$$

with \odot denoting element-wise products to have input and output embedding dimension agree without zero-padding. Including the indicator token $Z_b = z_b \cdot \mathbf{1}_E, z_b \in \{1, \dots, n_b\}$, the corpus of queries then consists of N batches of the form

$$\{(\mathbf{x}_{b;1}, \mathbf{y}_{b;1}, \dots, \mathbf{x}_{b;n_b}, \mathbf{y}_{b;n_b}, \mathbf{z}_b, \mathbf{x}_b^{\text{CF}})\}_{b \in \{1, \dots, B\}}$$

on which the model has to predict $\mathbf{y}_{b;z_b}^{\text{CF}}, b \in \{1, \dots, B\} =: [B]$. Our target Y^{CF} is zero-mean with $\text{Var}(Y^{\text{CF}}) = 169$ and $\log(\text{Var}(Y^{\text{CF}})) = 5.1299$.

We train on minimizing the per-batch mean squared error (MSE) between the counterfactual prediction $\widehat{\mathbf{y}}_{[B]}^{\text{CF}}$ and the ground truth $\mathbf{y}_{[B]}^{\text{CF}}$,

$$\text{MSE}(\widehat{\mathbf{y}}_{[B]}^{\text{CF}}, \mathbf{y}_{[B]}^{\text{CF}}) = \frac{1}{B \cdot E} \sum_{b=1}^B \left\| \widehat{\mathbf{y}}_{b;z_b}^{\text{CF}} - \mathbf{y}_{b;z_b}^{\text{CF}} \right\|_2^2$$

and evaluate on an unseen test set following (9). We use the notational forms $\text{MSE}(\widehat{y}, y)$ and $\text{MSE}(\widehat{y} - y)$ interchangeably.

Our code is based on the repository by Garg et al. [2022]. We therefore adopt the Adam optimizer [Kingma and Ba, 2015] and a learning rate of 10^{-4} for all function classes and models. We implement the experiments in pytorch [Paszke et al., 2019] and use one NVIDIA GeForce RTX 3090 GPU for training. All conducted experiments require between 10 minutes and 3 hours of training depending on model setup and task complexity.

C.1 A note on embedding dimension E

Following the approach of Garg et al. [2022], we train models using various embedding dimensions and analyze their performance. For $E = 20$, we observe in-context performance that is comparable to the case with $E = 5$. However, training convergence is noticeably slower. In particular, for $E = 50$, the training loss has *not yet* fully converged after $N = 200000$ steps. Similar trends are observed for $E = 75$ and $E = 100$. For the model trained with $E = 250$, the training loss plateaus at a level consistent with the theoretical variance.

Given the extensive number of experiments conducted, we opt for a small embedding dimension of $E = 5$. This choice is especially important for experiments involving data diversity and cyclic causal structures, which are trained with fewer steps. In these cases, larger embedding dimensions would entail higher computational costs. Since our study relies on synthetic data, we prioritize efficiency and reproducibility by choosing a smaller E .

D Model details

The Transformer architecture is described in subsection 2.2. Here we lay out the details on the three recurrent models used and investigate the relevance of model depth in Transformers more closely.

888 D.1 Model details on RNN architectures

889 The Elman RNN [Elman, 1990] is a foundational recurrent neural network architecture. It consists of
 890 an input layer, a hidden layer with recurrent connections, and an output layer. The hidden state at
 891 time step t , denoted by h_t , is computed as

$$\begin{aligned} h_t &= \tanh(W_{ih}x_t + W_{hh}h_{t-1} + b_h) \\ y_t &= W_{ho}h_t + b_o, \end{aligned}$$

892 where x_t is the input at time t , and \tanh denotes the *tangens hyperbolicus*, a non-linear activation
 893 function. The output is given by y_t and W , b represent learnable weight matrices and biases.

894 The Long Short-Term Memory (LSTM) network [Hochreiter and Schmidhuber, 1997] addresses
 895 the vanishing gradient problem by incorporating a memory cell and three gates: input (i_t), forget
 896 (f_t), and output (o_t). These gates regulate the flow of information and enable the network to retain
 897 long-term dependencies. The LSTM cell is defined by the following equations,

$$\begin{aligned} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ g_t &= \tanh(W_g x_t + U_g h_{t-1} + b_g) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t). \end{aligned}$$

898 In this formulation, c_t is the cell state and h_t the hidden output state. Activation vectors of the three
 899 gates are given by i_t, f_t, o_t and g_t is the cell input activation vector. By σ , we denote the sigmoid
 900 function and element-wise multiplication by \odot . U represents another weight matrix.

901 The Gated Recurrent Unit (GRU) [Cho et al., 2014] introduces gating mechanisms to better control
 902 the flow of information. It simplifies the LSTM architecture by combining the forget and input gates
 903 into a single update gate, z_t . The GRU computes its hidden state using the following equations,

$$\begin{aligned} r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\ z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ n_t &= \tanh(W_n x_t + U_n (r_t \odot h_{t-1}) + b_n) \\ h_t &= (1 - z_t) \odot n_t + z_t \odot h_{t-1}. \end{aligned}$$

904 Here, r_t is the reset gate, z_t the update gate, n_t the new gate, the candidate update vector. Note that
 905 the dependence of the hidden state on the update gate varies across documentations. We follow the
 906 definition in the pytorch package.

907 We perform a hyperparameter sweep on hidden size D and the number of layers L over the grid
 908 $D \in \{64, 128, 256\}$, $L \in \{2, 3\}$. We select the configuration with hidden size 256 and 3 layers for
 909 each of the three architectures. Figure 6 sheds more light on the performance results illustrated by
 910 Figure 2. For long contexts of at least 17 in-context examples, the Elman RNN achieves significantly
 911 higher in-context MSE than the three other architectures. For the LSTM, GRU and STANDARD
 912 Transformer, we observe no significant performance difference across models.

913 D.2 Varying depth for additional values

914 Analogous to Figure 3b, we note that increasing model depth leads to declining in-context MSE. This
 915 holds for contexts both shorter and longer than the 35-example version considered above. Figure 7
 916 illustrates the in-context MSE for varying depth, evaluated at 1 and 50 in-context examples, respec-
 917 tively. We note that the loss of the 1-layer, 8-head Transformer exhibits no significant differences
 918 between 1 in-context example and 50. We interpret this as indication that the model is unable to
 919 infer information on the latent θ from the context. Reinforcing our findings that the 4-layer, 2-head
 920 and 8-layer, 1-head Transformers achieve the lowest in-context MSE, this pattern serves as evidence
 921 that in-context inference occurs across layers. A contrasting notion may be that the Transformer
 922 maps different junks of information to different subspaces of the embedding space. Acting on each
 923 subspace individually, each attention head would then focus on a different task before the MLP would
 924 combine the information and output the final prediction. Instead, the model appears to benefit from

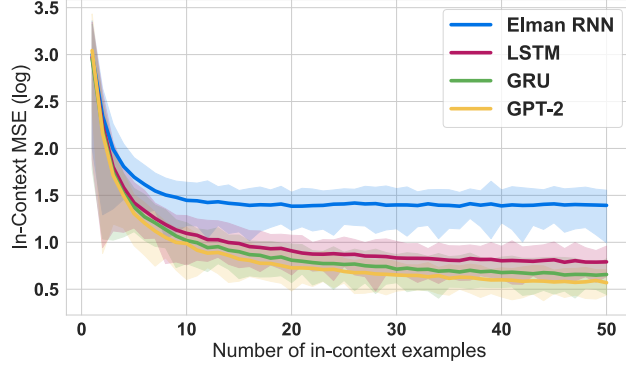


Figure 6: **Model comparisons for in-context counterfactual reasoning.** In-context counterfactual prediction accuracy measured via log-transformed mean-squared error loss averaged over 100 sequences versus the number of in-context examples observed in a prompt. We compare the following model architectures: GPT-2 (STANDARD), LSTM, GRU, and Elman RNN. All models are capable of in-context counterfactual reasoning. GPT-2 (STANDARD) achieves lowest error and fastest convergence rate for a small number of in-context examples. For more than 17 in-context examples, the Elman RNN has significantly [Efron, 1979] higher in-context MSE than the three other architectures.

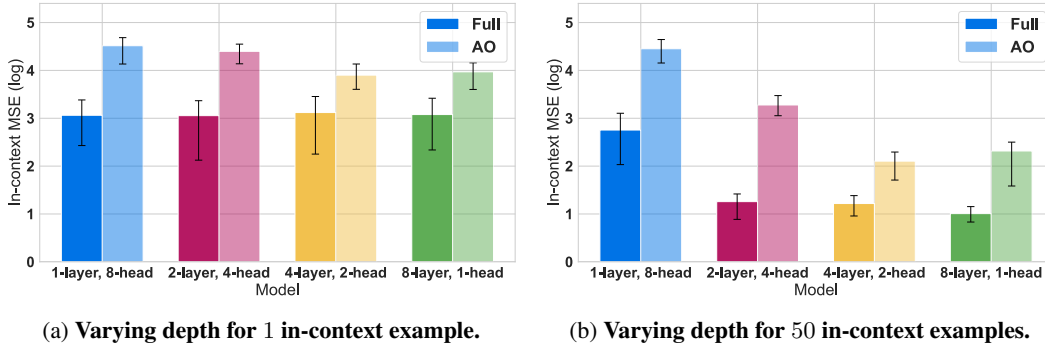


Figure 7: **Varying depth more closely.** Building on top of the results displayed in Figure 3b, we compare **Full** and **AO** Transformers at a constant number of 8 attention heads. We observe a decrease in in-context loss as model depth increases across shorter and longer contexts of 1 and 50 in-context examples, respectively. We evaluate on 100 sequences.

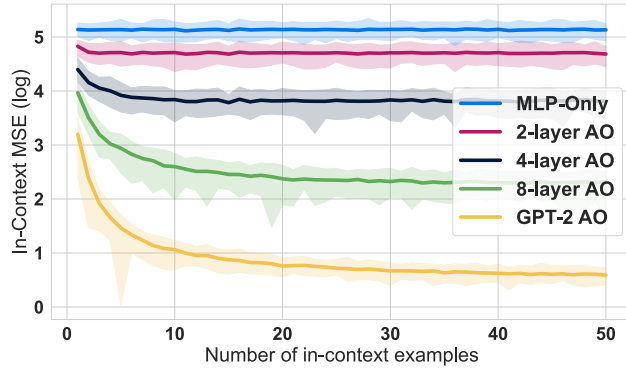


Figure 8: **Switching Attention.** In-context MSE stagnates with increasing number of in-context examples observed for **MLP-only** model. We compare to **AO** Transformers of 2, 4, 8 layers and the STANDARD GPT-2 setup. For all Transformers, in-context MSE decreases as more in-context examples are observed. Error bars are basic bootstrap confidence intervals [Efron, 1979].

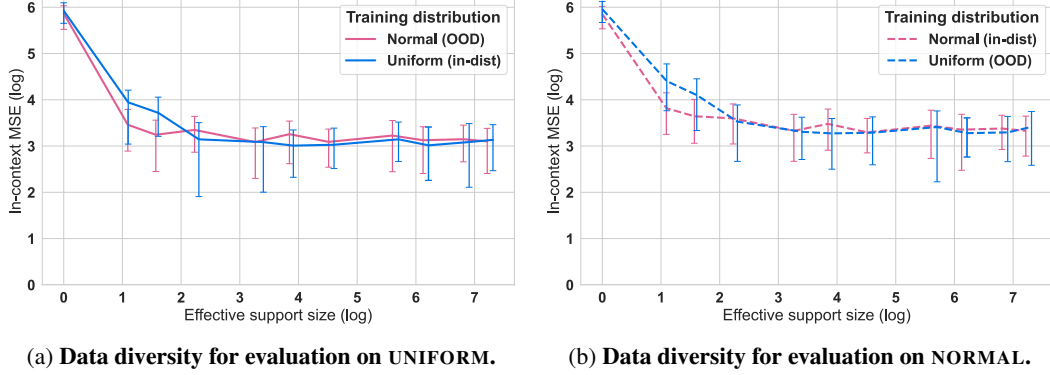


Figure 9: **Data diversity for pre-training on 1 in-context example.** We measure in-context MSE (log-scaled) averaged over 100 prompts at one in-context example against log-scaled effective support size. Each point represents one fully pre-trained model on either the UNIFORM sampled or NORMAL sampled θ . We train the models for 50000 training steps. All models in are evaluated on both datasets. Error bars are the 95% basic bootstrap confidence intervals.

the interplay between layers, which supports the claim that attention heads across layers communicate through the residual stream [Elhage et al., 2021].

Figure 8 underscores the results illustrated by Figure 3a by indicating significant differences between the MLP-only setting and each of the four considered AO Transformer variants.

E Data diversity and robustness

E.1 Data diversity adjustments and out-of-distribution

In 4.3, we discuss data diversity and generalizability to OOD. The UNIFORM setup largely follows the overall setup described in 9) with the following adjustment: instead of $N \cdot B \cdot E$, we sample $d \ll N$ realizations of the latent and compute the effective support size [Grendar, 2006]. Note that we control the absolute number of one-dimensional realizations as our setup effectively analyzes E independent examples at every turn. Thus, we draw d realizations and allocate across iterations and embedding dimensions. The parameter d corresponds to the diversity argument in our code.

The NORMAL setup follows the above structure. Compared to 9), we sample the latent from $\theta \sim \mathcal{N}(\mathbf{0}, \sqrt{12} \cdot \mathbf{I}_E)$ such that the theoretical variance is equivalent across setups. Everything else is analogous. Note that the effective support size is equivalent to the support size under the UNIFORM setup. This can be seen by interpreting Ess as the logarithm of the Shannon entropy. Indeed, $p(\vartheta)$ is uniformly distributed over discrete Θ_0 if ϑ denotes a realization of the latent θ ,

$$H_{\text{UNIFORM}}(\theta) = - \sum_{\vartheta \in \Theta_0} \log p(\vartheta).$$

Therefore, we allocate the $d = |\Theta_0|$ realizations uniformly across iterations and embedding dimensions. Under the NORMAL setup, $p(\vartheta)$ depends on the distance of ϑ to 0. Here, we sample d Gaussian realizations. We then allocate them proportional to $p(\vartheta)$ across iterations and embedding dimensions. In analogy to Figure 4, we plot the in-context MSE against Ess for pre-training on the NORMAL setting and evaluating both in-distribution and on UNIFORM. Figure 9 illustrates the in-context MSE relative to effective support size on contexts of one example. Here we train all models for 50000 training steps. Although the loss is at a higher level than for 35 in-context examples, we observe that the MSE declines with increasing effective support size. In terms of sufficient data diversity, the findings are consistent with above, as we require pre-training effective support size of around 10.

E.2 Non-linear and non-additive extensions

On top of the linear regression setting, we extend in-context counterfactual reasoning to non-linear, non-additive functions which are invertible in u . The general dataset setup remains the same with

adjustments made only to the computation of y, y^{CF} . We evaluate the the 8-layer, 1-head **Full** and **AO** Transformers. [Table 1](#) reports the in-context MSE on 100 queries with 35 in-context examples each. Depending on the exact configuration of architecture and extension, the in-context MSE is at least one tenth of the empirical variance of the test set.

	$f(x, u)$	Empirical variance	MSE of AO Transformer	MSE of Full Transformer
Tanh	$\tanh(\tau(\beta x + u))$	0.3144	0.0186	0.0058
Sigmoid (σ)	$\frac{1}{1 + \exp(-\tau(\beta x + u))}$	0.0355	0.0016	0.0005
Multiplicative	$\frac{1}{\sqrt{\text{Var}(\beta X^{\text{CF}} U_Y)}} \beta x u$	0.7888	0.0671	0.0437

Table 1: **Robustness to different function classes for regression tasks.** We report in-context MSE averaged over 100 sequences for the 8-layer **AO** and **Full** Transformers under invertible nonlinear activation functions and multiplicative noise. We compute the empirical variance of target Y^{CF} on the test set. We observe for both architectures that the MSE is 10 to 70 times lower than the empirical variance, suggesting both can in-context learn more complex function classes beyond linear regression and complete the counterfactual query. All functions are applied element-wise in one dimension. \tanh and σ are scaled by $\tau = \frac{1}{13}$ to counteract congestion around $\{-1, 1\}$ and $\{0, 1\}$, respectively. To guarantee theoretical variance of 1, we divide the multiplicative term by $\sqrt{\text{Var}(\beta X^{\text{CF}} U_Y)} = \sqrt{3410.4}$.

Note that under multiplicative noise, counterfactual reasoning reduces to a pure copying task as

$$Y^{\text{CF}} = \frac{1}{\sqrt{\text{Var}(Y^{\text{CF}})}} \beta X^{\text{CF}} U_Y = \frac{1}{\sqrt{\text{Var}(Y^{\text{CF}})}} \beta X^{\text{CF}} \frac{Y \sqrt{\text{Var}(Y^{\text{CF}})}}{\beta X} = \frac{X^{\text{CF}}}{X} Y,$$

where estimation of β is not required for counterfactual completion.

F Model details for cyclic sequential dynamical systems

F.1 Training details

In accordance with our exchangeable setting, we construct the dataset

$$\begin{aligned} \theta &\sim \mathcal{U}([1, 2]^E) & \alpha, \beta, \gamma, \delta | \theta &\sim \text{Exp}(\theta) \\ \mathbf{U} | \theta &\sim \mathcal{N}(\theta, \mathbf{I}_E) & \mathbf{X}^{\text{CF}} &\sim \mathcal{U}([1, 2]^E) \end{aligned} \quad (10)$$

where we sample $N_i \sim \text{Pois}(\lambda)$ distinct time steps for $i \in \{1, \dots, N \cdot B\}$ and $\lambda = 200$. We truncate the sequence on $[0, 0.1]$ to overcome exploding concentrations and obtain the expected number of distinct time steps $\mathbb{E}[N_i] = 0.1 \cdot 200 = 20$. We again consider E independent sequences at once by stacking independent one-dimensional SDEs. Within one example, all SDEs are evaluated at the same distinct time steps. At any i , we then provide the observational sequence as

$$(\mathbf{x}_0, \mathbf{y}_0, \mathbf{x}_{t_1}, \mathbf{y}_{t_1}, \dots, \mathbf{x}_{t_{N_i}}, \mathbf{y}_{t_{N_i}}, \mathbf{z}, \mathbf{x}_0^{\text{CF}}, \mathbf{y}_0^{\text{CF}})$$

for \mathbf{z} a counterfactual delimiter token, identical across i . Given the query, we ask the model to predict the *full* completion of the counterfactual sequence, $\text{comp}_i := (\mathbf{x}_{t_1}^{\text{CF}}, \mathbf{y}_{t_1}^{\text{CF}}, \dots, \mathbf{x}_{t_{N_i}}^{\text{CF}}, \mathbf{y}_{t_{N_i}}^{\text{CF}})$, autoregressively. We again evaluate the model on the per-batch MSE of the predicted completion, $\widehat{\text{comp}}_{[B]}$, and the underlying ground truth sequence, $\text{comp}_{[B]}$,

$$\text{MSE}(\widehat{\text{comp}}_{[B]}, \text{comp}_{[B]}) = \frac{1}{B \cdot E} \sum_{b=1}^B \|\widehat{\text{comp}}_b - \text{comp}_b\|_2^2$$

and evaluate on an unseen test set following [\(10\)](#).

We train for $N = 10000$ training steps at batch size of $B = 64$ and numerically approximate the solutions to the SDEs in (3) using the Euler–Maruyama [Maruyama, 1955] scheme. The counterfactual sequence is generated analogously but by recycling the Brownian motion used for the observational sequence. The manifestation of this can be seen in Figure 5b as *observational* and *counterfactual* y exhibit corresponding dynamics at each realized time step $t_n \leq t_{N_i}$. We inherit the torchsde package from Li et al. [2020] and follow Charleux et al. [2018] for the implementation of the Lotka–Volterra equations. All other specifications are similar to the regression setup described in Appendix C

F.2 Lotka–Volterra model

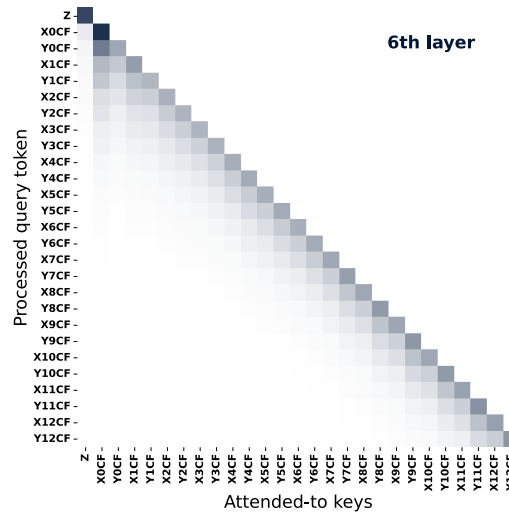
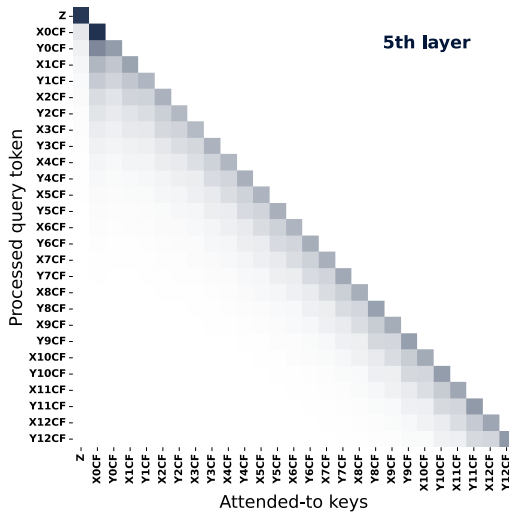
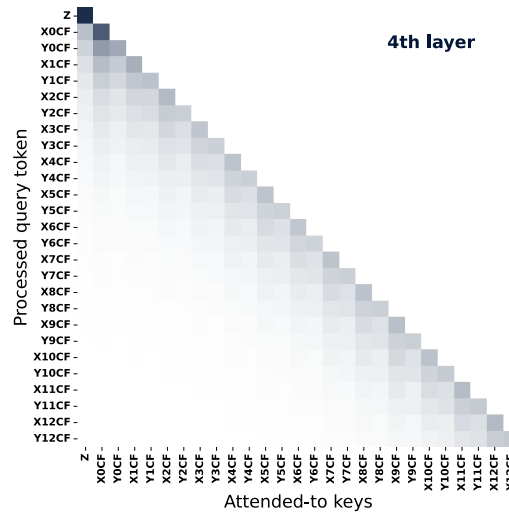
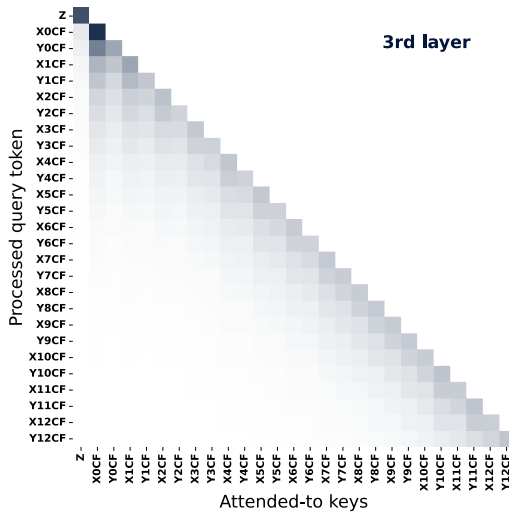
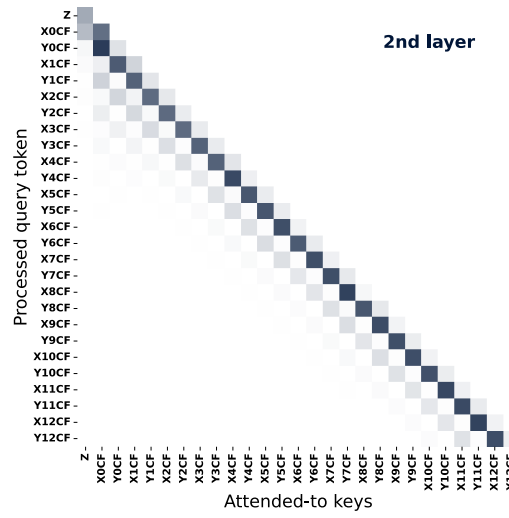
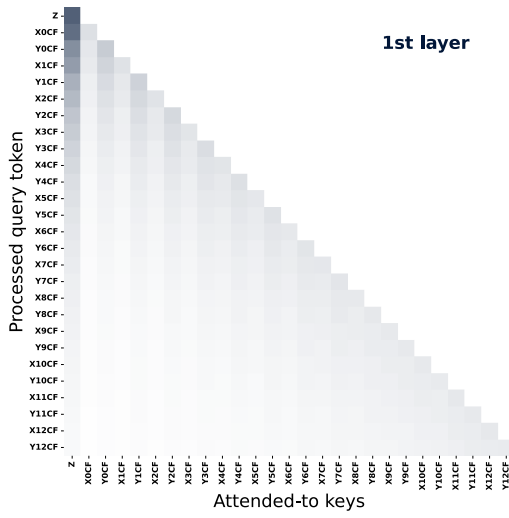
The Lotka–Volterra framework models the temporal evolution of the concentrations of two interacting species: predators and prey. In the absence of predators, the prey concentration grows continuously over time. Conversely, the predator concentration declines unless sustained by consuming prey. Predation enables predator growth, which simultaneously reduces prey concentration. We model the four individual dynamics by real, nonnegative parameters. The dynamics of the prey concentration (5) is parameterized by the intrinsic growth rate α and the rate at which the prey concentration decreases¹ with predator concentration, β . For predator concentration (6), we capture the species-inherent decreasing concentration by γ and the rate at which predators feed off prey by δ . In the context of biological species, *concentration* can be thought of as the population density.

F.3 Attention behavior

Figure 11 shows that the 8-layer, 1-head AO Transformer performs counterfactual reasoning by repeatedly moving information forward. Trained on variable sequence length, the model exhibits this pattern for different context lengths. Across a diverse set of test sequences, we observe that the tokens of the counterfactual sequence attend to the position of the delimiter token \mathbf{z} at the first layer. Note that we only plot attention values between tokens of the counterfactual sequence. The Transformer at the second layer implements an induction head [Olsson et al., 2022] that copies forward the information one step. Then, we find five attention heads which implement decaying attention over prior positions. Maximum weight is here put on the current token, and progressively lower weights on all previous tokens relative to token distance. This design facilitates forward information propagation across tokens. The final attention head aggregates signals from the current and preceding token, integrating both token-specific information and temporally local context shared within the token pair at time t_n .

In accordance with the hypothesis that induction heads drive in-context learning [Elhage et al., 2021, Olsson et al., 2022, Akyürek et al., 2024], we notice this pattern for larger models, with multiple heads per layer, and including the MLP. For instance, the Full STANDARD GPT-2 Transformer contains three heads across the first two layers which mostly attend to the delimiter token \mathbf{z} . At the third layer, the model implements an induction head. Taken together, this provides more evidence that counterfactual reasoning emerges, at least partly, in the self-attention layer of the Transformer.

¹In Equation 5 we write $g(X_t, Y_t) := \alpha X_t + \beta X_t Y_t$, but specify $\alpha, \beta \geq 0$. This is of course a typo as prey concentration *decreases* with increasing predator concentration. We replace with $g(X_t, Y_t) := \alpha X_t - \beta X_t Y_t$. Our code is not affected by this error.



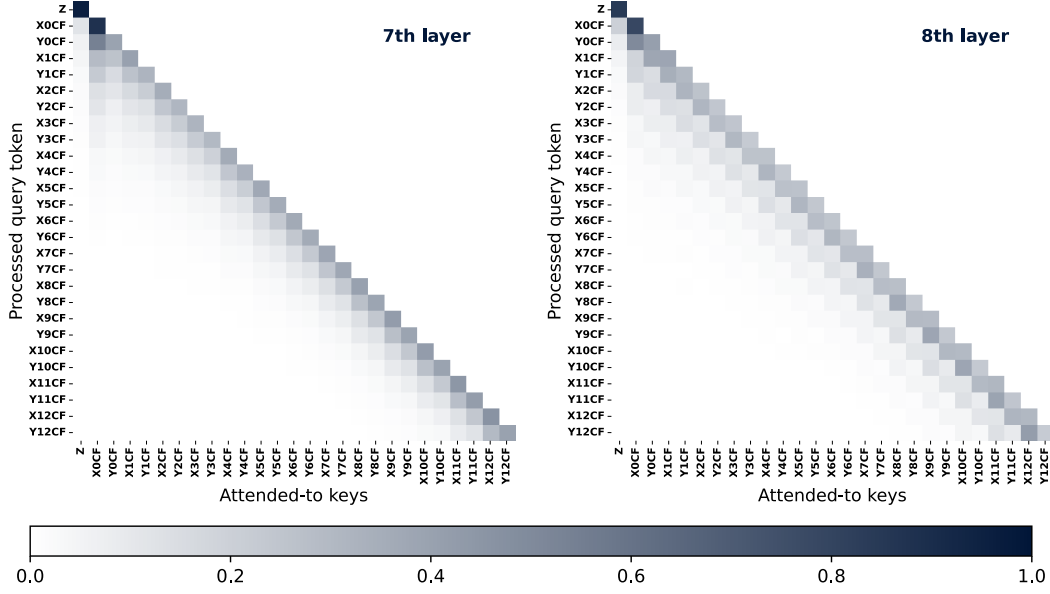


Figure 11: **Cyclic causal relationship.** The 8-layer AO Transformer, trained on counterfactual reasoning in Lotka-Volterra SDEs, includes two dedicated copying heads. At layer 2, the induction head shifts residual stream information from the previous token forward one position. The subsequent five attention heads implement decaying attention over prior positions, with maximum weight on the current token and progressively lower weights on all previous tokens relative to token distance.

G Connection between synthetic setup and natural language

G.1 Discrete regression setup

Due to the synthetic nature of our setup, the connection to natural language is not immediately apparent. To illustrate how unobserved noise u can be interpreted linguistically, we provide a three-shot example. Each pair (x_j, y_j) can be thought of as a pair of sentences, where the second (y_j) disambiguates the first (x_j) for $j \in \{1, \dots, n\}$. In this setup, x_j is semantically ambiguous, and y_j provides a clarifying continuation. After presenting the model with n such in-context examples, we prompt it to resolve the ambiguity in a new instance, indexed by z . Figure 12 visualizes this process in natural language. The examples marked \diamond , \triangle , and \square each consist of an ambiguous sentence x_j , followed by one or more possible concretizations y_j , with the realized continuation shown in **bold**. We then present a new ambiguous sentence $x_z^{\text{CF}} = \text{He was stuck in a tough spot}$, accompanied by the index token \blacksquare . Based on the previously seen \square example, the model infers that this sentence should be interpreted in a startup context, and not related to softball/baseball or the viscoelastic polymer called *pitch*.

G.2 Continuous SDE setup

The cyclic causal dependencies modeled by the SDEs in (3) align with the sequential structure of natural language. Each query can be interpreted as a factual narrative of length t_N . Given a hypothetical initial condition (x_0, y_0) , we task the model with counterfactually completing the story while keeping the noise fixed. This noise may represent semantic variability as well as narrative-internal uncertainty. By doing so, the model can generate plausible counterfactual continuations conditioned on the observed prompt. This approach resonates with the work of Qin et al. (2019), who explore how language models can reason about alternative story outcomes through counterfactual perturbations to character actions or events.

◇ They went to the bank.	{ There, they withdrew some money before the shops closed. They sat down and watched the ducks swim in the river. After last night's storm, they had to check on the erosion.
△ He broke the bat.	{ The wooden handle snapped after he hit the ball too hard. His little sister was very sad about the loss of her favorite stuffed toy.
□ She made a pitch.	{ The ball curved slightly as it left her hand and hit the strike zone. Her voice rose as she delivered the startup idea to the investors. She spread the sticky substance across the seams of the roof tiles.
■ He was stuck in a tough spot.	{ With the bases loaded and two outs, he had no room for error. The product wasn't gaining traction, and investors were starting to lose interest. The thick black goo clung to his boots as he tried to move across the roof.

Figure 12: **Discrete ambiguity resolution via in-context examples.** Each row shows a pair (x_j, y_j) , where x_j is an ambiguous sentence and y_j its concretization. The correct interpretation of y_j is shown in **bold**, while alternative continuations illustrate other plausible meanings of x_j . The final row (■) presents a new ambiguous sentence x_z^{CF} . Based on the prior example □, the model is expected to resolve the ambiguity in a startup context, rejecting interpretations tied to baseball or material substances.