Amortized Active Causal Induction with Deep Reinforcement Learning

Anonymous Authors[1]

Abstract

We present Causal Amortized Active Structure Learning (CAASL), an active intervention design policy that can select interventions that are adaptive, real-time and that does not require access to the likelihood. This policy, an amortized network based on the transformer, is trained with reinforcement learning on a simulator of the design environment, and a reward function that measures how close the true causal graph is to a causal graph posterior inferred from the gathered data. On synthetic data and a single-cell gene expression simulator, we demonstrate empirically that the data acquired through our policy results in a better estimate of the underlying causal graph than alternative strategies. Our design policy successfully achieves amortized intervention design on the distribution of the training environment while also generalizing well to distribution shifts in test-time design environments. Further, our policy also demonstrates excellent zero-shot generalization to design environments with dimensionality higher than that during training, and to intervention types that it has not been trained on.

1. Introduction

Infer, design and experiment is a three step loop in the empirical scientific discovery paradigm. Causal induction (a.k.a. causal structure learning), the problem of finding causal relationships present in data, also falls under this paradigm when experiments in the form of interventions are permissible (Spirtes et al., 2001; Heinze-Deml et al., 2018). Causal structure learning has gained increasing importance in empirical sciences, for example in single-cell biology, where perturbation experiments like gene knockouts can be carried out with high-precision (Tejada-Lapuerta et al., 2023). Such interventions are not only more informative to infer the underlying causal graph than just observational data, but in certain cases essential to go beyond the Markov equivalence class (Peters et al., 2017), making the problem of design of interventions both relevant and important. For the problem of structure learning with interventions, however, inference and design both involve significant challenges. For instance, inference of the causal graph from data usually involves search over the space of graphs with a likelihood (usually weighted by a prior) or score function (Annadani et al., 2023; Brouillard et al., 2020; Hauser and Bühlmann, 2012), which is slow and not robust to violations of data generation assumptions (Montagna et al., 2024). The design of informative interventions, on the other hand, utilizes the inferred causal graph from existing data to select promising designs and rank them according to a scoring criterion. This scoring criterion is usually based on an approximation of mutual information between the unknown causal graph and the interventional data (Tigas et al., 2023; 2022), which also involves the (interventional) data likelihood. In problems related to empirical sciences where causal discovery is essential, like inferring a gene regulatory network with gene knockouts or knockdowns, the likelihood of the data is typically intractable. While progress has been made in terms of likelihood-free inference of causal graphs (Lorch et al., 2022; Ke et al., 2022), existing intervention design algorithms have been largely restricted to likelihood-based strategies.

With a focus on addressing practical intervention design challenges that arise in empirical sciences like inferring the gene regulatory network, in this work, we propose an intervention design method called CAASL that significantly differs from existing approaches. Instead of following the infer, design and experiment loop, we amortize the intervention design procedure by training a single design network policy, based on the transformer (Vaswani et al., 2017), which encodes key design space symmetries. During test-time, our trained policy directly predicts the next intervention to perform by just a forward-pass of the data collected so far, without the need to undergo slow and expensive inference of the causal graph corresponding to that data. We train the transformer policy with Soft Actor-Critic (SAC) (Haarnoja et al., 2018) to maximize cumulative rewards over a fixed number of design iterations (budget), thereby making the policy adaptive. The choice of a good reward function is

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

Preliminary work. Under review by the SPIGM workshop at ICML 2024. Do not distribute.

essential for informative designs. We discuss various reward function choices, primarily based on an estimate of the true causal graph obtained from a likelihood-free amortized causal discovery approach. Both our policy and the reward function only require access to a simulator of the design environment. Further, we present connections of our approach to amortized sequential Bayesian experimental design (Foster et al., 2021). We demonstrate that the reward function is related to an approximation of expected information gain based on the amortized posterior distribution over causal graphs. As such, CAASL is an intervention design method for performing sample efficient causal structure learning, but is not a new causal structure learning method in itself.

On synthetic data and the single-cell gene expression simulator SERGIO (Dibaeinia and Sinha, 2020), we empirically study various aspects of our trained policy—the amortization performance on training distribution of the design environment as well as on design environments with distribution shifts from the training environment. We find that our policy obtains better causal structure learning performance for a given budget than alternate intervention strategies. Overall, we observe excellent generalization capability of the transformer for intervention design, similar to what has been demonstrated in other domains (Brown et al., 2020; Kaplan et al., 2020; Zhai et al., 2022). The robustness of the amortized policy opens up the possibility for lab-in-the-loop intervention design for single-cell data, wherein a single network can propose informative interventions across different cell lines and experimental conditions.

2. Background and Related Work

Structural Causal Models. Let $\boldsymbol{y} = \{y_1, \ldots, y_d\}$ be the random variables of interest associated with the vertices of a graph G. Let $A \in \{0,1\}^{d \times d}$ be the adjacency matrix corresponding to G. A Structural Causal Model (SCM) (Peters et al., 2017) is a framework for causality which consists of a set of equations in which each variable y_i is a deterministic function of its direct causes $y_{\text{pa}_G(i)}$ as well as an exogenous noise variable ϵ_i with a distribution P_{ϵ_i}

$$y_i \coloneqq f_i(\boldsymbol{y}_{\text{pa}_G(i)}, \epsilon_i; \theta_i). \tag{1}$$

The functions f_i, with parameters θ_i, are mechanisms that relate how the direct causes affect the variable y_i. The structural assignments are typically assumed to be acyclic, with G being a directed acyclic graph whose edges indicate direct causes. In addition, an SCM defines the likelihood of any data sample $\boldsymbol{y}$ under this model, denoted as $p(\boldsymbol{y} \mid \{A, \theta\})$. Further, we assume that the SCM is causally sufficient, i.e. all the variables are measurable (but can be missing at random), and the noise variables are mutually independent.

Interventions. The SCM framework admits reasoning about effects of interventions on any variable in $\boldsymbol{y}$. Most notable types of intervention include a perfect (do) intervention, and a shift intervention (Rothenhäusler et al., 2015). A perfect intervention on any variable y_i corresponds to changing the structural equation of that variable to the desired value, $y_i \coloneqq v_i$. It is denoted by the do-operator (Pearl, 2009) as $\text{do}(y_i = v_i)$. In a shift intervention, the conditional mean of the interventional variable $\mathbb{E}[y_i \mid \boldsymbol{y}_{\text{pa}_G(i)}]$ is shifted by v_i. The likelihood of any data under an intervention I is denoted as $p(\boldsymbol{y} \mid \{A, \theta\}, I)$. For perfect and shift interventions, I can be parameterized as a $d \times 2$ dimensional matrix, where the first column corresponds to one-hot encoding of whether a particular variable is intervened or not, and the second column corresponds to the value (or the shift) of the intervention corresponding to each potential intervention target.

Causal Structure Learning. The problem of causal structure learning corresponds to estimating A (and other parameters of the SCM θ) given samples from p_{data} (Heinze-Deml et al., 2018). In general, there could be multiple models (and hence graphs) that can be consistent with a given joint distribution over $\boldsymbol{y}$, which necessitates causal structure learning with interventional data (Peters et al., 2017). There are various approaches, either based on independence tests (Dai et al., 2024; Spirtes et al., 2001), or graph search by maximizing a score function (likelihood of the data with certain assumptions on the SCM) (Brouillard et al., 2020; Rolland et al., 2022; Hauser and Bühlmann, 2012). Reinforcement learning has also been used for search over graphs with a score function (Zhu et al., 2019), however it differs entirely from our approach wherein we use RL for intervention design. Alternately, based on the tractable likelihood, there are also causal structure learning methods that estimate the posterior distribution $q(A \mid \mathcal{D})$ of graphs (Annadani et al., 2023; Deleu et al., 2024) for a dataset $\mathcal{D}$ that is sampled from p_{data}.

Likelihood-Free Amortized Causal Structure Learning (AVICI) (Lorch et al., 2022). More recently, instead of inferring causal graphs over specific datasets, amortized posterior inference of causal graphs has also been studied (Lorch et al., 2022; Ke et al., 2022). In particular, the amortized posterior from Lorch et al. (2022), called AVICI, makes use of a transformer to directly predict the posterior $q(A \mid \mathcal{D})$ by just a forward-pass of any dataset. The amortized posterior, parameterized as a product of independent Bernoulli random variables over the presence of edges in the causal graph, is trained from a simulator without having access to the likelihood of the data. Since the simulator provides the ground truth value of A, the amortized posterior can be trained by maximum likelihood with a combination of observational and interventional data to maximize the probability of the true edges. The AVICI model can amortize over datasets with different dimensionalities d, while gener-

alizing well to new datasets that have not been seen during training. Since it is computationally cheap to obtain the posterior distribution with AVICI, we use it for computing the reward for our intervention design policy.

Active Intervention Design. Active intervention design is the problem of designing interventional experiments to obtain data that enables causal structure learning in a sample efficient manner (under a fixed budget). While adaptive strategies have been explored (Choo and Shiragur, 2023; Greenewald et al., 2019), these approaches still require intermediate inference of the SCM, and are also not amortized. Intervention design based on Bayesian optimal experimental design (Lindley, 1956; Chaloner and Verdinelli, 1995) has also been considered, although only with additive noise models, which enable likelihood evaluation (Tigas et al., 2022; 2023; Agrawal et al., 2019; Zhang et al., 2024; Sussex et al., 2021). Reinforcement learning has also been used in intervention design (Sauter et al., 2023; Lampinen et al., 2024), however, they have been limited to non-amortized and small scale settings. In contrast to earlier work, we demonstrate the applicability of our method to single-cell simulated gene expression data, wherein the mechanisms are defined by differential equations and also include technical noise.

3. Amortized Intervention Design

We first present our active intervention design strategy with reinforcement learning, the corresponding amortized network and its training. In Section 4.1, we then present connections of our reward to sequential Bayesian experimental design.

Setting. Given a budget T, intervention design is the problem of finding a sequence of informative interventions with a policy $I_1, \dots, I_T \sim \pi$ that results in an estimate of the causal graph that is close to A. For any intervention I, a causal model defines a generative model of the data with likelihood $p(\boldsymbol{y} \mid \{A, \theta\}, I)$ and prior $p(A, \theta)$. We indicate initial (observational) data, if available, as $\boldsymbol{y}_0 = \{\boldsymbol{y}_0^{(i)}\}_{i=1}^{n_0}$ and the corresponding interventions with I_0, where $I_0 = \{\emptyset\}^{n_0}$ if the initial data is fully observational. Let $h_t \in \mathbb{R}^{(n_0+t)\times d\times 2}$ denote the interventional history $(\boldsymbol{y}_0, I_0), \dots, (\boldsymbol{y}_t, I_t)$, obtained by concatenation of $\boldsymbol{y}$ and first column of I that correspond to interventional targets. We do not explicitly encode intervention values in history, since for a do intervention, the intervention values are already present in $\boldsymbol{y}$[1]. Existing intervention design strategies like (Tigas et al., 2023) approximate a posterior on $\{A, \theta\}$ at each step t, approximate expected information gain (EIG) (Lindley, 1956; Rainforth et al., 2024) and greedily maximize it to compute I_{t+1}. Details of this greedy approach are given in Appendix A.1.

[1]We train our policy only on do interventions.

3.1. Intervention Design with Reinforcement Learning

In this work, we instead treat intervention design as a Reinforcement Learning (RL) problem and train a single policy network π_ϕ with parameters ϕ to obtain a sequence of adaptive interventions $I_1, \dots, I_T$ for any underlying causal graph with adjacency matrix A. In order to do so, we first describe the RL environment under which the interventions are performed.

Intervention Design Environment. Similar to Blau et al. (2022), we define an interventional design environment as a Hidden-Parameter Markov Decision Process (HiP-MDP) (Doshi-Velez and Konidaris, 2016). The HiP-MDP we use, $\mathcal{M}(\{A, \theta\})$, has hidden parameters $\{A, \theta\}$ and can be fully described by the tuple $(\mathcal{S}, \mathcal{A}, \rho, \beta, \mathcal{T}, R, \gamma, p_\beta)$. The state-space $\mathcal{S}$ consists of the histories $s_t = h_t$, the initial state $\rho = h_0 = (\boldsymbol{y}_0, I_0)$ corresponds to initial data, the action-space $\mathcal{A}$ corresponds to interventions $a_t = I_t$ and β describes the space of all causal models (graphs and parameters) with prior $p_\beta = p(A, \theta)$. The hidden parameters are sampled for each episode at the beginning from the prior. γ is the discount factor. In a HiP-MDP, the transition function $\mathcal{T}$ and reward R depend on the hidden parameters. The transition function $\mathcal{T}(h_t \mid h_{t-1}, I_t, \{A, \theta\})$ is Markovian, and it involves two operations: (1) sampling interventional data $\boldsymbol{y}_t \sim p(\boldsymbol{y} \mid \{A, \theta\}, I_t)$, and (2) updating the history state $h_t = \text{Concat}[h_{t-1}, (\boldsymbol{y}_t, I_t)]$. For a reward function $R(h_t, I_t, h_{t-1}, \{A, \theta\})$ that we define below, intervention design corresponds to finding the parameters ϕ of the amortized policy that maximizes the expected cumulative reward of all interventions:

$$\max_{\phi} \mathop{\mathbb{E}}_{\pi_\phi, \rho, \mathcal{T}, p(A,\theta)} \left[\sum_{t=1}^{T} \gamma^{t-1} R\left(h_t, I_t, h_{t-1}, \{A, \theta\}\right) \right] \quad \text{with} \quad I_t \sim \pi_\phi(h_{t-1}) \tag{2}$$

Reward Function. For the purpose of amortized intervention design, a good reward function should be cheap to evaluate while leading to informative interventions. In this work, we propose to utilize the estimate of the causal graph from an amortized causal graph posterior $q(\hat{A} \mid h_t)$. In particular, we use the pretrained AVICI model (Lorch et al., 2022). AVICI is a transformer based neural network trained with (interventional) data from a simulator to directly predict the probability of presence or absence of any edge in the causal graph by just a forward pass of the data, without requiring access to the likelihood. For any history h_{t-1}, we define the reward for performing intervention I_t and reaching state h_t as the improvement in the number

of correct entries in the predicted adjacency matrix of the AVICI model:

$$R(h_t, I_t, h_{t-1}, \{A, \theta\}) = \mathop{\mathbb{E}}_{q(\hat{A}|h_t)} \left[\sum_{i,j} \mathbb{I}\left[\hat{A}_{i,j} = A_{i,j}\right] \right] - R(h_{t-1}, I_{t-1}, h_{t-2}, \{A, \theta\}) \tag{3}$$

where $\mathbb{I}[\cdot]$ is the indicator function and $R(h_0, I_0, \{A, \theta\}) = \mathbb{E}_{q(\hat{A}|h_0)} \left[\sum_{i,j} \mathbb{I}\left[\hat{A}_{i,j} = A_{i,j}\right]\right]$. We note that our choice of reward function revolves around obtaining a good estimate for the causal graph, A; we do not (directly) reward learning about θ.

The above RL problem for intervention design is intuitive: reward the intervention in proportion to the improvement it brings in terms of number of correct entries of the adjacency matrix from the amortized posterior. Also, for any t, the cumulative reward, eq. (2), for $\gamma = 1$ of all interventions including I_0, which includes an additional term $R(h_0, I_0, \{A, \theta\})$, is simply the number of correct entries of the adjacency matrix predicted by the amortized posterior model for h_t. This reward telescoping was inspired by Blau et al. (2022). We also show in Section 4.1 that this reward function is also related to an approximation of multi-step EIG, the quantity of interest in sequential Bayesian experimental design (Foster et al., 2021).

3.2. Policy

Architecture. In order for the policy to achieve amortization and generalize to new environments not seen during training, it should encode key design space symmetries. In particular, for the problem of intervention design, the interventions should be permutation equivariant to ordering of the variables and permutation invariant to the ordering of the history. This can be ensured by a transformer architecture (Vaswani et al., 2017) wherein self-attention is applied alternately—once over the variable axis and next over the samples axis (Kossen et al., 2021). More precisely, we input $h_t \in \mathbb{R}^{(n_0+t)\times d\times 2}$ and apply self-attention[2] over first the $n_0 + t$ axis and next over the d axis. This ensures that the history representation is permutation equivariant over both the axes (Lee et al., 2019). After multiple layers of alternating self-attention, we apply max pooling over the samples (dim. $n_0 + t$) axis, which gives an encoding of size l of the history $B_t \in \mathbb{R}^{d\times l}$ that respects the desired symmetries. The same symmetries apply for amortized causal structure learning, hence the reward model AVICI also leverages the alternate attention architecture. The history embedding B_t is then passed through a multi-layer perceptron, whose outputs parameterize the logits of the Gaussian-Tanh distribution (Haarnoja et al., 2018), from which the interventions are sampled. In our setting, we model both the intervention targets and intervention values, hence I_t is $d \times 2$ dimensional. Gaussian-Tanh samples range from -1 to 1. We use $I_t[:, 0]$ to encode the interventional targets by discretizing the values to a binary mask (0 and 1) by thresholding at 0, where 1 indicates intervention on the variable y_i. If an intervention on y_i is active, the value to intervene with is given by $I_t[i, 1]$.

[2]Every self-attention is multi-headed, followed by a position-wise feedforward layer and layer normalization, as in a standard transformer block.

Training. Training the policy involves addressing two main challenges: computing the reward in eq. (3) since $\{A, \theta\}$ would be unknown for real datasets, and optimizing this reward, which is discrete. In order to address the first challenge, we simulate interventional data $\boldsymbol{y}_t \sim p(\boldsymbol{y} \mid \{A', \theta'\}, I_t)$ for a sample $\{A', \theta'\} \sim p(A, \theta)$ from the prior using a *simulator*. Such simulators exist for single cell gene regulatory networks (e.g. (Dibaeinia and Sinha, 2020; Cannoodt et al., 2021; Schaffter et al., 2011)) and are becoming increasingly widespread in other domains (Gamella et al., 2024; Ahmed et al., 2020). The reward model AVICI is pretrained on datasets from the prior $p(\{A, \theta\})$ using the same simulator. During training of the policy, we only use the pretrained reward model for inference and do not update its parameters. To address the second challenge, we train our policy using Soft-Actor Critic (SAC) (Haarnoja et al., 2018), an off-policy reinforcement learning algorithm that does not require rewards to be differentiable. We use the REDQ version of SAC to improve sample efficiency (Chen et al., 2021). REDQ trains multiple Q-function networks to optimize the reward. For each Q-function network, we use a transformer based history state encoder with architecture similar to that in the policy, but the weights are not shared. This is beneficial because the same equivariance–invariance properties that hold for the policy should also hold for the Q-function.

Inference. Deploying the policy in a real (i.e. not simulated) environment amounts to a rollout of the policy through interaction with the real environment. This requires just a forward pass of policy network for each time step t. Note that we do not need intermediate Bayesian inference or other estimation of the causal graph on the collected data.

4. Choice of Reward Function

4.1. Connection to Sequential Bayesian Experimental Design

As discussed in Appendix A, the problem we tackle has a connection to likelihood-free sequential Bayesian experimental design (Foster et al., 2021; Ivanova et al., 2021). With the aim of gathering data to learn about the causal

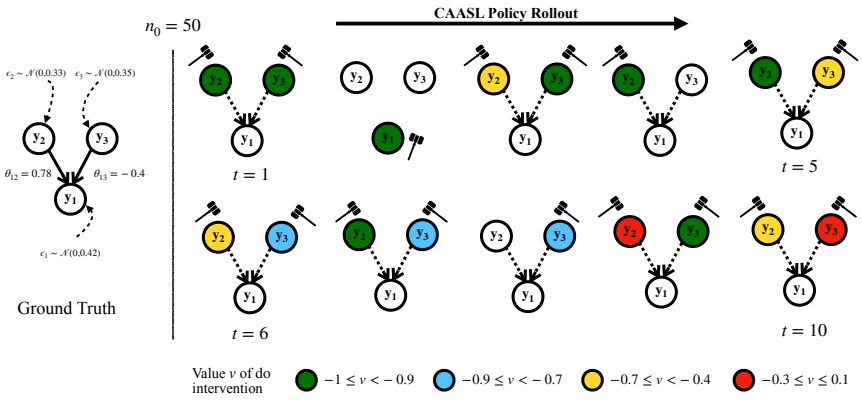

Figure 1. Visualization of the rollout of the trained CAASL policy on a randomly sampled environment with $n_0 = 50$ initial observational samples. Colored circles indicate nodes with a do intervention. The policy selects interventions that mostly correspond to the variables with a child in the ground truth graph. At $t = 2$, the policy selects the only child y_1, which breaks all direct causal effects. This gives lesser information about the overall causal model. After this, y_1 is never chosen. Initially, the policy is exploratory wrt targets and exploitative wrt values. This trend is reversed as the episode progresses. The policy is trained on environments with $d = 2$, therefore it has not seen any graphs with $d = 3$ before.

graph A, the multi-step expected information gain (EIG) can be written

$$\text{EIG}(A; \pi_\phi) = \mathop{\mathbb{E}}_{\pi_\phi, \rho, \mathcal{T}, p(\{A,\theta\})} [\log p(A \mid h_t)] + \text{const.} \tag{4}$$

Since the posterior $p(A \mid h_t)$ is intractable, we could replace it by an approximate posterior $q(A \mid h_t)$. This gives rise to the Barber-Agakov (BA) bound (Barber and Agakov, 2004; Foster et al., 2020), which was recently explored in a sequential context by Blau et al. (2023). This tells us that we have an EIG lower bound by using q in place of p:

$$\text{EIG}(A; \pi_\phi) \geq \mathop{\mathbb{E}}_{\pi_\phi, \rho, \mathcal{T}, p(\{A,\theta\})} [\log q(A \mid h_t)] + \text{const.} \tag{5}$$

We can interpret eq. (5) in simple terms—taking $\log q(A \mid h_t)$ as a reward function is equivalent to optimizing a lower bound on the EIG. Although eq. (5) implies that we only receive a reward on the final state h_t, it is possible to rewrite this using telescoping rewards (Blau et al., 2022) exactly as we do in eq. (3). The BA bound therefore represents the closest point of comparison between the method we outline in Section 3 and sequential Bayesian experimental design. As with the BA bound, we make use of an amortized approximate posterior distribution $q(A \mid h_t)$ that works backwards from data h_t to predict the graph that might have generated it. Unlike the BA bound, however, we use the adjacency matrix accuracy to compare the true A to samples $\hat{A}$ from the amortized posterior, rather than computing the log-likelihood of the true graph under that amortized posterior, $\log q(A \mid h_t)$. We found that this worked better in practice. Nevertheless, we see a close relationship between the approach we take and the methods of sequential Bayesian experimental design.

4.2. Other Possible Reward Functions

Any target metric for causal structure learning like structural hamming distance computed on the amortized posterior could be used as a reward function. Depending on the application, domain specific causal graph objectives could also be considered. While a large number of possibilities exist, we use expected number of correct entries of adjacency matrix, eq. (3), as the reward for training CAASL. As opposed to Structural Hamming Distance (SHD) and Area Under Precision Recall Curve (AUPRC), eq. (3) is straightforward to compute and parallelize.

5. Experiments

We train CAASL policy on two challenging environment domains: 1. Synthetic design environment with a causal model defined by linear mechanisms and additive noise, and 2. SERGIO (Dibaeinia and Sinha, 2020), a single-cell simulator corresponding to any gene regulatory network. For each domain, we train a single CAASL policy on a distribution of design environments with $d = 10$. A distribution of intervention design environments is defined by the choice of prior over causal models $p(A, \theta)$, which includes priors over graphs A (e.g. Erdős–Rényi (Erdős et al., 1960)), mechanism parameters θ and noise. We define an Out-of-Distribution (OOD) environment as any environment with the choice of prior (either over graphs, mechanisms parameters or noise) that is different from training. In addition to these distribution shifts, we also consider OOD environments wherein the priors remain the same, but either the dimensionality of the data d increases (i.e. $d > 10$), or the performed intervention type changes. Precise choice

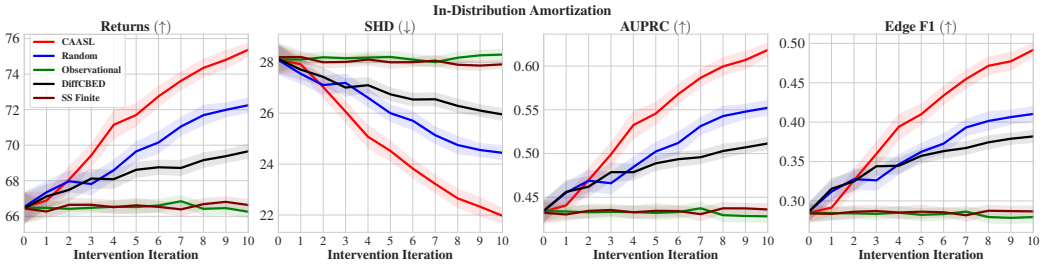

Figure 2. Amortization results of various intervention strategies on 100 random test environments. CAASL significantly outperforms other intervention strategies. Shaded area represents 95% CI.

of training and OOD testing distributions are given in Appendix C. All evaluation experiments are conducted *in silico*, on environments with causal model parameters that CAASL has never seen during training, regardless of whether the environment is in-distribution or OOD. In addition, all evaluation is done by just forward passing the history through the policy.

Baselines. We compare our approach with two amortized strategies: Random and Observational. Random corresponds to obtaining data from random interventions, while Observational corresponds to collecting more observational data. For the synthetic design environment domain, we also compare with DiffCBED (Tigas et al., 2023) and SS Finite (Sussex et al., 2021). These intervention strategies use likelihood of the data to perform designs. So in certain OOD synthetic design environments and the single-cell simulator SERGIO where the likelihood is not available, we omit these baselines. DiffCBED and SS Finite rely on an approximate causal graph posterior to design interventions. As suggested in (Tigas et al., 2023), we use bootstrapped GIES (Hauser and Bühlmann, 2012; Friedman et al., 2013) as the approximate posterior distribution for these baselines. For an evaluation task on 100 random design environments with a budget of 10, DiffCBED and SS Finite methods require approximate posterior inference of the causal graph 1000 times.

Metrics. All evaluation is done on 100 random test environments. As CAASL is an intervention design method, we measure the cumulative rewards with $\gamma = 1$ (returns) obtained from the graph predicted by the amortized posterior. However, for the sake of completeness, we also measure structure learning related metrics like the Structural Hamming Distance (SHD), the Area under Precision Recall Curve (AUPRC) and F1 score (Edge F1) between the graph predicted by the amortized posterior and the true graph (Lorch et al., 2022; Annadani et al., 2023). Precise definition of these metrics is provided in Appendix G.2. We find that in most cases all the metrics are correlated. Therefore, unless otherwise mentioned, we only report the returns and relegate the other metrics to Appendix G.2.

5.1. Synthetic Design Environment

Training Distribution of the Design Environment. We train CAASL on synthetically generated data, wherein $p(A, \theta)$ consists of linear SCMs with additive homoskedastic Gaussian noise. The dimensionality during training is $d = 10$. The prior over causal graphs is Erdős–Rényi (Erdős et al., 1960), with 3 edges per node in expectation. The prior over linear coefficients is chosen such that the marginal variance of each variable is close to 1. This is done to ensure that structure learning algorithms are not sensitive to the scale of the data (Reisach et al., 2021). During training, an intervention exclusively corresponds to a do intervention. Further, we set $n_0 = 50$ and the budget T is fixed to 10.

Training Details. We train CAASL with 4 layers of alternating attention for the transformer, followed by a max pooling operation over the history, to give an embedding with size $l = 32$. SAC related hyperparameters are tuned based on performance on held-out design environments. Details of the architecture, hyperparameter tuning and optimizer is given in Appendix D. For the reward model, we use AVICI that is pretrained on random linear additive noise datasets.

Amortization Performance. We test on novel environments with hidden parameters sampled from the training prior $p(A, \theta)$. Results are provided in fig. 2. We find that our policy significantly outperforms the random baseline in terms of returns as well as more common structure learning metrics like the SHD, AUPRC and Edge F1. For instance, our method achieves returns close to 76 with just 10 interventional samples, while the random baseline achieves close to 72. Other intervention strategies like DiffCBED (Tigas et al., 2023) and SS Finite (Sussex et al., 2021) perform worse, while still making use of the likelihood and performing intermediate inference of causal structure.

Zero-Shot OOD Generalization. We also test the trained CAASL policy on environments when the prior changes.

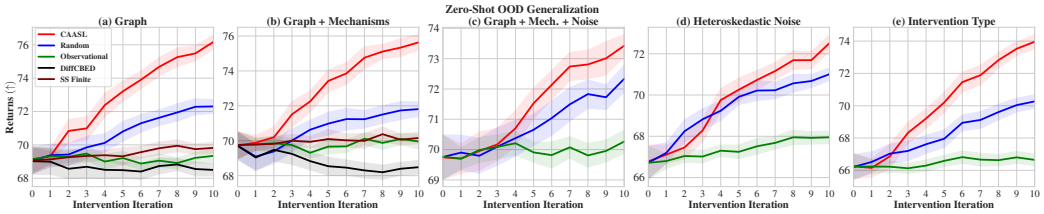

Figure 3. Zero-shot OOD returns of CAASL on 100 random environments with distribution shift coming from (a) graphs, (b) graphs and mechanisms, (c) graphs, mechanisms and noise, (d) noise changing from homoskedastic to heteroskedastic, and finally (e) intervention changing from do to a shift intervention. CAASL outperforms other intervention strategies. Shaded area represents 95% CI.

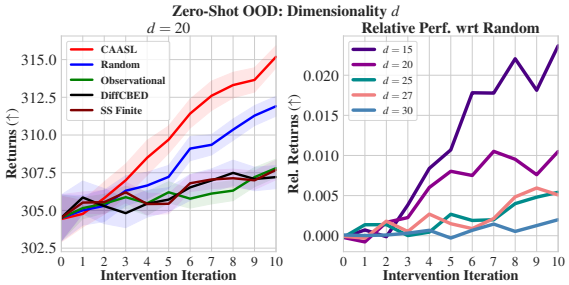

Figure 4. Zero-Shot OOD generalization results when dimensionality d changes for synthetic environment. For training, $d = 10$. Left: Zero-Shot test returns with $d = 20$. Right: Relative mean zero-shot returns of CAASL wrt random for different d. Results on 100 random environments. Shaded area represents 95% CI.

All results correspond to zero-shot performance, obtained by just a forward forward pass of the trained policy. Figure 3 presents the returns of CAASL alongside other applicable baselines. We consider shifts which become increasingly different from training: (1) the graph prior changes from Erdős–Rényi (Erdős et al., 1960) to Scale-Free (Barabási and Albert, 1999) (fig. 3 (a)), (2) apart from the graph, prior over mechanisms also change (fig. 3 (b)), (3) apart from graph and mechanisms, the noise distribution changes from Gaussian to Gumbel (fig. 3 (c)). We find that our policy achieves better performance than random strategy by a significant margin. Further, our method also outperforms DiffCBED and SS Finite which explicitly optimize for designs corresponding to these environments. In addition to these OOD settings, we also consider OOD environments in which the prior remains the same, but the noise is heteroskedastic instead of homoskedastic (fig. 3 (d)). Although the random strategy is very competitive, CAASL performs better. Finally, we consider OOD environments wherein the intervention design suggested by the policy during testing is used for performing a shift intervention instead of a do (fig. 3 (d)). CAASL performs better than baselines even in this setting.

Slightly different to the above OOD environments, we also consider OOD environments in which the dimensionality of the data changes during testing, but the prior remains the same. fig. 4 presents the results, with further details in fig. 6. CAASL obtains better returns on average than random at all points of acquisition. The relative performance of CAASL decreases as d increases (up to $d = 30$) from training, although it still performs better than random.

5.2. Single-Cell Gene Regulatory Network Environment

In this setting, we train a CAASL policy based on the single-cell gene expression simulator SERGIO (Dibaeinia and Sinha, 2020). Given a causal graph that corresponds to interaction between different genes in terms of their transcription regulation, SERGIO simulates expressions of genes that correspond to steady state of differential equations that govern the interaction between the genes. Each variable entry indicates the count of mRNA that is produced corresponding to that gene, similar to the output of modern single-cell RNA sequencing (scRNA-seq) technological platforms (Macosko et al., 2015). In addition, SERGIO can be extended to support interventions. Interventions in this setting correspond to either gene knockouts, wherein the transcription rate of the intervened gene is actively set to 0, or gene knockdown, wherein the intervened gene's transcription rate is actively halved. Since there is no value selection in this setting, the dimensionality of the policy is d instead of $d \times 2$. SERGIO also simulates technical noise such that the statistics of the data match that obtained from real scRNA-seq platforms. Some of the technical noise includes dropouts (missingness of the data), library size effects and random outlier effects. Most notably, atleast 70% of the data is missing in most single cell platforms. Therefore, in this domain, not only is the likelihood intractable, but also there is high amount of missing data. We do not impute the missing data, but just encode it with 0.

Training Distribution of the Design Environment. For training, we set $d = 10$, with $n_0 = 50$ observational (wild-type) data with budget $T = 10$. The statistics of the data corresponds to 10X Chromium platform (Dibaeinia and Sinha, 2020) wherein around 74% of the data is dropped out. The prior over causal graphs is set to Erdős–Rényi (Erdős et al., 1960) with 3 edges per node on average. An inter-

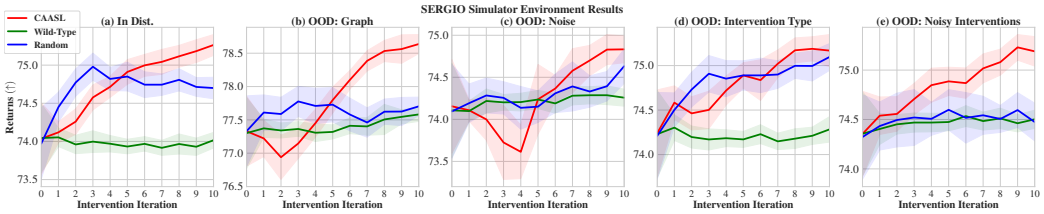

Figure 5. Results on SERGIO environment with 100 random environments. (a) corresponds to in-distribution performance, (b)-(e) correspond to zero-shot OOD performance with distribution shift coming from either (b) graphs, (c) technical noise, (d) intervention changing to a gene-knockdown (e) Noisy interventions, which include off-target effects. Shaded area represents 95% CI.

vention exclusively corresponds to a gene knockout. We provide details of the simulator in appendix B.2 and the training prior parameters in appendix C.2.

Training Details. We train CAASL with 3 layers of alternate attention, followed by a max pooling operation, giving an embedding of size $l = 32$. Just like in the synthetic linear domain, SAC related hyperparameters are tuned based on performance on held-out design environments. Details are given in appendix D. Once trained, we perform a forward pass of the history through the policy to obtain intervention designs for all test environments. For the reward model, we use AVICI that is pretrained on this simulator with post-noise data statistics matching that of 10X chromium platform.

Amortization Performance. The in-distribution amortization performance is presented in fig. 5(a). After 5 acquisitions, CAASL obtains better returns than random.

Zero-Shot OOD Generalization. We test the CAASL policy when the environment is subject to various test-time distribution shifts. Robustness to distribution shifts is important in real world-settings, where experimental conditions can change. We consider 4 different OOD environments: (1) the prior over graphs changes from Erdős–Rényi to Scale-free (fig. 5(b)), (2) The perturbation platform changes to Drop-Seq (Macosko et al., 2015), wherein among other noise parameters, the amount of missing data increases from 74 to 85% (fig. 5(c)), (3) The intervention type changes from knockout to knockdown (fig. 5(d)) and, (4) Noisy knockout interventions, where there is a 10% chance that either the intended gene does not get knocked out, or an off-target gene is knocked out (fig. 5(e)). We find that CAASL shows excellent robustness to these distribution shifts, and obtains better returns than baselines. When the intervention type changes, the random baseline is still competitive. An interesting observation is that the for the OOD graph and the OOD noise setting, the model shows exploratory behavior in the beginning where the returns decrease, but later becomes better than random. Robustness to various distribution-shifts demonstrates the generality of the policy.

Limitation. For the zero-shot OOD generalization when the dimensionality of the data increases, we noticed that the performance of CAASL is on par with random, but is not necessarily better (fig. 7). We hypothesize that since almost 74% of data is missing, the incorporated design space symmetries might not be as relevant, which might limit the extent of zero-shot generalization.

6. Conclusion

We have presented an amortized and adaptive intervention design strategy CAASL, that does not require intermediate inference of the causal graph. CAASL is based on a policy parameterized by the transformer which is permutation equivariant to ordering of the variables and permutation invariant to ordering of the collected data. Through various experiments, including on a simulator which respects the data statistics of real gene-expression readouts, we find that our method shows excellent amortized intervention design and zero-shot generalization to significant distribution shifts. The achieved performance motivates intervention design in more complex settings - high-throughput experiments with large batch sizes and utilization of existing real offline data for designing interventions.

References

Raj Agrawal, Chandler Squires, Karren Yang, Karthikeyan Shanmugam, and Caroline Uhler. Abcd-strategy: Budgeted experimental design for targeted causal structure discovery. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3400–3409. PMLR, 2019.

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. *arXiv preprint arXiv:2010.04296*, 2020.

Yashas Annadani, Nick Pawlowski, Joel Jennings, Stefan Bauer, Cheng Zhang, and Wenbo Gong. Bayesdag:

Gradient-based posterior sampling for causal discovery. *arXiv preprint arXiv:2307.13917*, 2023.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

David Barber and Felix Agakov. The im algorithm: a variational approach to information maximization. *Advances in neural information processing systems*, 16(320):201, 2004.

Tom Blau, Edwin V Bonilla, Iadine Chades, and Amir Dezfouli. Optimizing sequential experimental design with deep reinforcement learning. In *International conference on machine learning*, pages 2107–2128. PMLR, 2022.

Tom Blau, Edwin Bonilla, Iadine Chades, and Amir Dezfouli. Cross-entropy estimators for sequential experiment design with reinforcement learning. *arXiv preprint arXiv:2305.18435*, 2023.

Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data. *Advances in Neural Information Processing Systems*, 33: 21865–21877, 2020.

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

Robrecht Cannoodt, Wouter Saelens, Louise Deconinck, and Yvan Saeys. Spearheading future omics analyses using dyngen, a multi-modal simulator of single cells. *Nature Communications*, 12(1):3942, 2021.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical science*, pages 273–304, 1995.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.

Davin Choo and Kirankumar Shiragur. Adaptivity complexity for causal graph discovery. In *Uncertainty in Artificial Intelligence*, pages 391–402. PMLR, 2023.

Haoyue Dai, Ignavier Ng, Gongxu Luo, Peter Spirtes, Petar Stojanov, and Kun Zhang. Gene regulatory network inference in the presence of dropouts: a causal view. *arXiv preprint arXiv:2403.15500*, 2024.

Tristan Deleu, Mizu Nishikawa-Toomey, Jithendaraa Subramanian, Nikolay Malkin, Laurent Charlin, and Yoshua Bengio. Joint bayesian inference of graphical structure and parameters with a single generative flow network. *Advances in Neural Information Processing Systems*, 36, 2024.

Payam Dibaeinia and Saurabh Sinha. Sergio: a single-cell expression simulator guided by gene regulatory networks. *Cell systems*, 11(3):252–271, 2020.

Finale Doshi-Velez and George Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *IJCAI: proceedings of the conference*, volume 2016, page 1432. NIH Public Access, 2016.

Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60, 1960.

Adam Foster, Martin Jankowiak, Matthew O'Meara, Yee Whye Teh, and Tom Rainforth. A unified stochastic gradient approach to designing bayesian-optimal experiments. In *International Conference on Artificial Intelligence and Statistics*, pages 2959–2969. PMLR, 2020.

Adam Foster, Desi R Ivanova, Ilyas Malik, and Tom Rainforth. Deep adaptive design: Amortizing sequential bayesian experimental design. In *International conference on machine learning*, pages 3384–3395. PMLR, 2021.

Nir Friedman, Moises Goldszmidt, and Abraham Wyner. Data analysis with bayesian networks: A bootstrap approach. *arXiv preprint arXiv:1301.6695*, 2013.

Juan L Gamella, Jonas Peters, and Peter Bühlmann. The causal chambers: Real physical systems as a testbed for ai methodology. *arXiv preprint arXiv:2404.11341*, 2024.

Kristjan Greenewald, Dmitriy Katz, Karthikeyan Shanmugam, Sara Magliacane, Murat Kocaoglu, Enric Boix Adsera, and Guy Bresler. Sample efficient active learning of causal trees. *Advances in Neural Information Processing Systems*, 32, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

Alain Hauser and Peter Bühlmann. Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *The Journal of Machine Learning Research*, 13(1):2409–2464, 2012.

Christina Heinze-Deml, Marloes H Maathuis, and Nicolai Meinshausen. Causal structure learning. *Annual Review of Statistics and Its Application*, 5:371–391, 2018.

Xun Huan and Youssef M Marzouk. Sequential bayesian optimal experimental design via approximate dynamic programming. *arXiv preprint arXiv:1604.08320*, 2016.

Desi R Ivanova, Adam Foster, Steven Kleinegesse, Michael U Gutmann, and Thomas Rainforth. Implicit deep adaptive design: Policy-based experimental design without likelihoods. *Advances in Neural Information Processing Systems*, 34:25785–25798, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Nan Rosemary Ke, Silvia Chiappa, Jane Wang, Anirudh Goyal, Jorg Bornschein, Melanie Rey, Theophane Weber, Matthew Botvinic, Michael Mozer, and Danilo Jimenez Rezende. Learning to induce causal structure. *arXiv preprint arXiv:2204.04875*, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jannik Kossen, Neil Band, Clare Lyle, Aidan N Gomez, Thomas Rainforth, and Yarin Gal. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. *Advances in Neural Information Processing Systems*, 34:28742–28756, 2021.

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, 2024.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

Dennis V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

Lars Lorch, Scott Sussex, Jonas Rothfuss, Andreas Krause, and Bernhard Schölkopf. Amortized inference for causal structure learning. *Advances in Neural Information Processing Systems*, 35:13104–13118, 2022.

Evan Z Macosko, Anindita Basu, Rahul Satija, James Nemesh, Karthik Shekhar, Melissa Goldman, Itay Tirosh, Allison R Bialas, Nolan Kamitaki, Emily M Martersteck, et al. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 161(5): 1202–1214, 2015.

Francesco Montagna, Atalanti Mastakouri, Elias Eulig, Nicoletta Noceti, Lorenzo Rosasco, Dominik Janzing, Bryon Aragam, and Francesco Locatello. Assumption violations in causal discovery and the robustness of score matching. *Advances in Neural Information Processing Systems*, 36, 2024.

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

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Tom Rainforth, Adam Foster, Desi R Ivanova, and Freddie Bickford Smith. Modern bayesian experimental design. *Statistical Science*, 39(1):100–114, 2024.

Alexander Reisach, Christof Seiler, and Sebastian Weichwald. Beware of the simulated dag! causal discovery benchmarks may be easy to game. *Advances in Neural Information Processing Systems*, 34:27772–27784, 2021.

Paul Rolland, Volkan Cevher, Matthäus Kleindessner, Chris Russell, Dominik Janzing, Bernhard Schölkopf, and Francesco Locatello. Score matching enables causal discovery of nonlinear additive noise models. In *International Conference on Machine Learning*, pages 18741–18753. PMLR, 2022.

Dominik Rothenhäusler, Christina Heinze, Jonas Peters, and Nicolai Meinshausen. Backshift: Learning causal cyclic graphs from unknown shift interventions. *advances in neural information processing systems*, 28, 2015.

Andreas WM Sauter, Erman Acar, and Vincent François-Lavet. A meta-reinforcement learning algorithm for causal discovery. In *Conference on Causal Learning and Reasoning*, pages 602–619. PMLR, 2023.

Thomas Schaffter, Daniel Marbach, and Dario Floreano. Genenetweaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*, 27(16):2263–2270, 2011.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2001.

Scott Sussex, Caroline Uhler, and Andreas Krause. Near-optimal multi-perturbation experimental design for causal structure learning. *Advances in Neural Information Processing Systems*, 34:777–788, 2021.

Alejandro Tejada-Lapuerta, Paul Bertin, Stefan Bauer, Hananeh Aliee, Yoshua Bengio, and Fabian J Theis.

Causal machine learning for single-cell genomics. *arXiv preprint arXiv:2310.14935*, 2023.

Panagiotis Tigas, Yashas Annadani, Andrew Jesson, Bernhard Schölkopf, Yarin Gal, and Stefan Bauer. Interventions, where and how? experimental design for causal models at scale. *Advances in neural information processing systems*, 35:24130–24143, 2022.

Panagiotis Tigas, Yashas Annadani, Desi R Ivanova, Andrew Jesson, Yarin Gal, Adam Foster, and Stefan Bauer. Differentiable multi-target causal bayesian experimental design. In *International Conference on Machine Learning*, pages 34263–34279. PMLR, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113, 2022.

Zeyu Zhang, Chaozhuo Li, Xu Chen, and Xing Xie. Bayesian active causal discovery with multi-fidelity experiments. *Advances in Neural Information Processing Systems*, 36, 2024.

Shengyu Zhu, Ignavier Ng, and Zhitang Chen. Causal discovery with reinforcement learning. *arXiv preprint arXiv:1906.04477*, 2019.

A. Connections to Bayesian Experimental Design using Expected Information Gain

A.1. Greedy Approaches

We consider the model with unknown parameters $\{A, \theta\}$, prior $p(\{A, \theta\})$ and likelihood of the data $p(\boldsymbol{y} \mid \{A, \theta\}, I)$ under an intervention I. The Expected Information Gain (EIG) is given by:

$$\mathrm{EIG}(I) = \mathop{\mathbb{E}}_{p(\{A,\theta\})p(\boldsymbol{y}|\{A,\theta\},I)} \left[\log p(\boldsymbol{y} \mid \{A, \theta\}, I) - \log p(\boldsymbol{y} \mid I)\right]. \tag{6}$$

In the standard greedy approach to Bayesian experimental design (Rainforth et al., 2024), given a history h_{t-1}, we replace the prior $p(\{A, \theta\})$ with the posterior conditional on existing data $p(\{A, \theta\} \mid h_{t-1})$ and then estimate the one-step EIG

$$\mathrm{EIG}(I) = \mathop{\mathbb{E}}_{p(\{A,\theta\}|h_{t-1})p(\boldsymbol{y}|\{A,\theta\},I)} \left[\log p(\boldsymbol{y} \mid \{A, \theta\}, I) - \log p(\boldsymbol{y} \mid h_{t-1}, I)\right]. \tag{7}$$

where $p(\boldsymbol{y} \mid h_{t-1}, I) = \int_{A,\theta} p(\{A, \theta\} \mid h_{t-1}) p(\boldsymbol{y} \mid \{A, \theta\}, I)$. The EIG is estimated for each candidate design I, and the one with the largest EIG is selected. This gives rise to the policy π_{greedy}, which was applied to causal graph discovery by e.g. Tigas et al. (2023).

A.2. Non-greedy Approaches

Non-greedy approaches to experimental design using EIG were also explored (Huan and Marzouk, 2016; Foster et al., 2021). Using the parameters $\{A, \theta\}$ of our model and the notation of Foster et al. (2021), the EIG of a sequence of t experiments generated using policy π_ϕ about $\{A, \theta\}$ is given by

$$\mathrm{EIG}(\{A, \theta\}; \pi_\phi) = \mathop{\mathbb{E}}_{p(\{A,\theta\})p(h_t|\{A,\theta\},\pi_\phi)} \left[\log p(h_t \mid \{A, \theta\}, \pi_\phi) - \log p(h_t \mid \pi_\phi)\right] \tag{8}$$

$$\text{where } p(h_t \mid \{A, \theta\}, \pi_\phi) = \prod_{\tau=1}^{t} p(I_\tau \mid \pi_\phi(h_{\tau-1})) p(\boldsymbol{y}_\tau \mid \{A, \theta\}, I_\tau) \tag{9}$$

and $p(h_t \mid \pi_\phi)$ is the marginal of this quantity over $p(\{A, \theta\})$. Equation (8) cannot be computed exactly, so likelihood-based (Foster et al., 2021) and likelihood-free (Ivanova et al., 2021) approximations have both been explored. The likelihood-based sPCE lower bound on EIG (Foster et al., 2021) was also used as a reward function to train an RL policy (Blau et al., 2022).

The problem we consider in this paper is decidedly likelihood-free for two reasons: (1) some simulators do not have explicit likelihoods, (2) even where a likelihood is available, it is generally conditional on both A and θ. We made the choice to focus on experimental design to learn A (ignoring information gain about θ). In this case, the relevant EIG is

$$\mathrm{EIG}(A; \pi_\phi) = \mathop{\mathbb{E}}_{p(\{A,\theta\})p(h_t|\{A,\theta\},\pi_\phi)} \left[\log p(h_t \mid A, \pi_\phi) - \log p(h_t \mid \pi_\phi)\right]. \tag{10}$$

We would have to perform a costly marginalization over θ to obtain the relevant likelihood, $p(h_t \mid A, \pi_\phi)$.

Equation (10) can be rearranged using Bayes Theorem to read

$$\mathrm{EIG}(A; \pi_\phi) = \mathop{\mathbb{E}}_{p(\{A,\theta\})p(h_t|\{A,\theta\},\pi_\phi)} \left[\log p(A \mid h_t) - \log p(A)\right] \tag{11}$$

$$= \mathop{\mathbb{E}}_{p(\{A,\theta\})p(h_t|\{A,\theta\},\pi_\phi)} \left[\log p(A \mid h_t)\right] + \text{const}, \tag{12}$$

where we make the observation that $\mathbb{E}[-\log p(A)]$ is a constant with respect to the design policy π_ϕ. The form of EIG eq. (12) is the jumping off point for the BA bound, in which we replace $p(A \mid h_t)$ with an approximate posterior.

B. Details of Design Environments

B.1. Synthetic Design Environment

We consider linear additive noise models. For homoskedastic noise, they can be written as

$$y_i := \theta_i^T \boldsymbol{y}_{\mathbf{pa}_G(i)} + \epsilon_i \tag{13}$$

where $\theta_i \sim p(\theta)$ and $\epsilon_i \sim p_{\text{noise}}$ which can be either a Gaussian or a gumbel distribution.

For heteroskedastic noise, the above equation can be written as:

$$y_i := \theta_i^T \boldsymbol{y}_{\mathbf{pa}_G(i)} + \sigma_i(\boldsymbol{y}_{\mathbf{pa}_G(i)}) \cdot \epsilon_i \tag{14}$$

where $\sigma_i(\cdot)$ is scaling factor that is obtained by a squash operation $\sigma_i(\boldsymbol{x}) = \log(1 + \exp(g_i(\boldsymbol{x})))$ on any nonlinear function g_i. Similar to (Lorch et al., 2022), we implement g_i with 100 random Fourier feature features (Rahimi and Recht, 2007). Random Fourier feature functions require a kernel, for which we use a Squared-Exponential Kernel with length $ls = 10$ and output scale $os = 2$.

B.2. Single-Cell Gene Regulatory Network Environment

SERGIO (Dibaeinia and Sinha, 2020) is a single-cell simulator of gene expression for any user provided gene-regulatory network that resembles the data obtained with modern single-cell RNA sequencing (scRNA-seq) technologies technologies like Drop-Seq (Macosko et al., 2015) and 10X Chromium (Dibaeinia and Sinha, 2020).

We provide a brief overview of simulation procedure of SERGIO. Our simulation is based on the original simulator provided by Dibaeinia and Sinha (2020) and hence further details can be found in the paper. This simulator was extended by Lorch et al. (2022) to support knockouts and knockdowns. We further vectorize the simulator to produce datasets for multiple regulatory networks parallely. The simulator of Dibaeinia and Sinha (2020) is publicly available under GPL-3.0 license.

B.2.1. Simulation of Gene Expressions

We first describe the data that is generated without interventions, also called as wild-type measurements. Later, we then describe interventional simulations.

For an observational dataset of size $n \times d$, the data produced from the simulator corresponds to the count of mRNA corresponding to each gene of n different single cells. In particular, given a regulatory network A, steady-state of the regulatory differential equation is simulated for n single cells that is regulated according to A. SERGIO allows for biological variations within this pool of n single cells, such as varying basal rates of master regulator genes. A master regulator gene is a gene with no upstream genes in A. The transcription rate of a master regulator gene is usually a constant, called the basal rate. Usually, cell of the same *type* have the same basal rates. For simulation, we consider c cell types and n_c single cells of each type such that $n = c \cdot n_c$. In this work, we fix $c = 5$. If n is less than 5, we sample n single-cells at random after simulating 5 single-cells corresponding to each cell type. The expression of all the downstream genes is effected nonlinearly by the mRNA production and decay of their respective regulatory genes. This expression is simulated according to a Langevin equation. Finally, the clean data is the continuous-valued mRNA concentration that is measured at random-time points of the steady-state Langevin simulation.

The clean data is then subject to technical measurement noise. The series of simulated noise is as follows:

1. With probability $p_{\text{outlier}} \in [0, 1]$, a gene is converted to an outlier gene that has unusually high expression across different cells. This is done by multiplying the current expression with values from a log-normal with mean μ_{outlier} and scale σ_{outlier}.
2. Based on the single-cell pool considered, different cells have different count distribution data. This is called as library-size effect, which is modeled as a log-normal distribution with mean μ_{lib} and scale σ_{lib}.
3. The dropouts are simulated with parameters dropout percentile $\delta \in [0, 100]$ and the temperature of the logistic function $\eta \in \mathbb{R}_+$

The actual values of the noise parameters differs across different scRNA-seq technologies. The final scRNA-seq resembling simulated data is obtained by sampling from a poisson distribution that is parameterized by the post-noise mRNA concentration levels.

During a gene knockout, the upstream genes do not effect the knocked out gene. The activity of the gene is set to 0 and is propagated downstream as before. In gene knockdowns, the upstream genes still work the same way as before, however, the expression of the knocked down gene is multiplied by 0.5 for every time step of the steady state simulation. The reduced gene expression of the knocked down gene is propagated to downstream genes as before.

	10X **Chromium**	**Drop-Seq**
p_{outlier}	0.01	0.01
$\mu_{\text{outlier}}, \sigma_{\text{outlier}}$	3.0, 1.0	3.0, 1.0
$\mu_{\text{lib}}, \sigma_{\text{lib}}$	6.0, 0.3	4.4, 0.8
δ, η	74, 8	85, 8

Table 1. Technical noise parameters for 10X Chromium and Drop-Seq Single-Cell RNA sequencing platforms that is used for experiments in this work.

B.2.2. Simulation Parameters

For generating the clean data, we use the following parameters across all settings, which is similar to what is used for training the reward model (Lorch et al., 2022):

- Number of cell types $c = 5$.
- Basal rates $b \sim \text{Uniform}(1, 3)$.
- Rate of decay of each gene $\lambda = 0.8$.
- Langevin equation related parameters: Hill function coefficient $\gamma = 1$, system noise scale $\epsilon_s = 1.0$, interaction strength $k \sim \text{Uniform}(1, 5)$ and the sign of the interaction which indicates a promotive or repressive regulation $\text{sgn}(k) \sim \text{Bernoulli}(p_k)$ with $p_k \sim \text{Beta}(0.5, 0.5)$.

For technical noise, we consider two different platforms: 10X Chromium and Drop-Seq. The noise parameters used are suggested in (Dibaeinia and Sinha, 2020). These parameters are presented in table 1.

C. Training and OOD Distributions of Design Environments

C.1. Synthetic Design Environment

For training distribution, we make the following choices:

- Prior over graphs $p(A) = p_{\text{ER}}(k_{in} = 3)$ is an Erdős–Rényi (Erdős et al., 1960) with 3 edges per node in expectation.
- Prior over parameters $p(\theta) = \mathcal{N}(0, \sigma_\theta^2)$ where σ_θ^2 is chosen such that marginal variance of each variable is 1 (Reisach et al., 2021).
- Noise $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ where $\sigma_\epsilon \sim \text{InvGamma}(10, 1)$.

For an OOD distribution, all the priors except for the parameter that undergoes distribution shift remain the same as during training. We define the following OOD environments and their corresponding distribution shifts:

a **Graphs:** $p(A) = p_{\text{SF}}(k_{in} = 3)$ is a Scale-Free (Barabási and Albert, 1999) with 3 edges per node in expectation.

b **Graphs+Mechanisms:** Prior over graphs is $p(A) = p_{\text{SF}}(k_{in} = 3)$ and prior over parameters $p(\theta) = \mathcal{N}(0.1, \sigma_\theta^2)$ with σ^2 chosen as during training.

c **Graphs+Mech.+Noise:** Prior over graphs is $p(A) = p_{\text{SF}}(k_{in} = 3)$ and prior over parameters $p(\theta) = \mathcal{N}(0.1, \sigma_\theta^2)$ with σ^2 chosen as during training. Noise $\epsilon \sim \text{Gumbel}(0, \sigma_\epsilon)$ where $\sigma_\epsilon \sim \text{InvGamma}(10, 1)$.

d **Heteroskedastic Noise:** The causal model changes from equation 13 to equation 14.

e **Intervention Type:** The performed intervention in the environment changes from a do to a shift intervention.

f **Dimensionality** d**:** The dimensionality of the environment increases from training distribution ($d < 10$).

C.2. Single-Cell Gene Regulatory Network Environment

For training distribution, we make the following choices:

- Prior over graphs $p(A) = p_{\text{ER}}(k_{in} = 3)$ is an Erdős–Rényi (Erdős et al., 1960) with 3 edges per node in expectation.
- Prior over mechanisms are as given in appendix B.2.2.
- For technical noise, we consider the 10X Chromium platform whose parameters are given in table 1.

For an OOD distribution, all the priors except for the parameter that undergoes distribution shift remain the same as during training. We define the following OOD environments and their corresponding distribution shifts:

a **Graphs:** $p(A) = p_{\text{SF}}(k_{in} = 3)$ is a Scale-Free (Barabási and Albert, 1999) with 3 edges per node in expectation.

b **(Technical) Noise:** The single-cell RNA sequencing platform changes from 10X Chromium to Drop-Seq, thereby changing the noise levels table 1.

c **Intervention Type:** The performed intervention in the environment changes from a gene knockout to a gene knockdown.

d **Noisy Interventions:** With a 10% probability, the gene suggested by the policy for knockout is either does not happen, or there is an off-target that is knocked-out. We achieve this by flipping the one hot encoding of the intervention target labels with 10% probability. But the history is only appended with the intervention sampled from the policy. Therefore, the policy has no knowledge of the noisy intervention.

D. Training Details

D.1. Architecture Details

We use the alternating attention based transformer for both the policy and the Q-function approximation. We maintain the same architecture for the transformer for both the policy and the Q-function, which we describe below.

For the transformer, we use a standard transformer block (Vaswani et al., 2017) with 8 heads of self-attention. As our transformer has alternating attention, each layer has two such self-attention operations. Each self-attention is followed by a feedforward layer, whose dimension is set to $4 * l$ where l is the size of the state representation. We choose $l = 32$. After L layers of alternate attention, we perform max pooling over ordering of the data to obtain the state representation. The state representation is passed through a two hidden layer MLP with 128 hidden dimensions each and ReLU nonlinearity.

D.2. Hyperparameter Tuning

REDQ (Chen et al., 2021) algorithm based on SAC (Haarnoja et al., 2018) trains M different Q-function networks and updates the gradients of each of them G times before updating the policy. We treat both these quantities as hyperaprameters. All the parameters are updated with the Adam optimizer (Kingma and Ba, 2014) and the learning rate is tuned. We list all the hyperparameters and the corresponding grid search in table 2.

E. Computational Resources

We train all models on 3 40GB NVIDIA A100 GPU accelerators. We provide a wall time of 3 days, which results in a total computational budget of 216 GPU hours for each model. We also tune hyperparameters as outlined in table 2 for both environments, resulting in a total usage of 70,000 hours. For testing, we just rollout the policy on a CPU, which can be completed in seconds.

F. Licenses

For the single-cell gene simulator, we make use of the publicly available repository which is released under GPL-3.0 License. For the reward model AVICI, we make use of the publicly released code and trained models. These are released under MIT License. For baselines, we use the DiffCBED open source repository, which is released under MIT License.

Table 2. Hyperparameters used for training in CAASL.

		Hyperameter Search	Synthetic Environment	SERGIO Environment
Transformer parameters (History state encoder)	No. attention layers (for policy, Q-Function)		4	3
	No. attention heads (for policy, Q-Function)		8	8
	l		32	32
	Dropout (Policy)		0.1	0.1
	Pooling (Policy)		Max pool over samples	Max pool over samples
	Pooling (Q Function)		Max pool over samples, sum pool over variables	Max pool over samples, sum pool over variables
Decoder parameters	Hidden sizes (for policy and Q)		(128, 128)	(128, 128)
	Non-linearity		ReLU	ReLU
REDQ/ SAC training parameters	M	$\{2, 3, 5\}$	5	2
	G	$\{1, 3, 5\}$	1	1
	γ	$\{0.9, 0.95\}$	0.9	0.95
	Buffer Size	$\{10e6, 10e7\}$	$10e7$	$10e6$
	Policy LR	$\{0.01, 0.001\}$	0.001	0.01
	Q-Function LR	$\{3e-5, 3e-6\}$	$3e-5$	$3e-6$
	τ		0.01	0.01

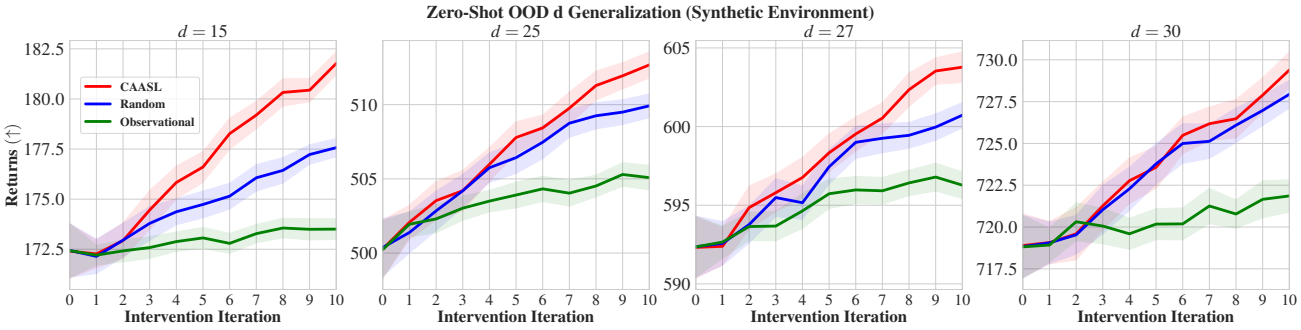

Figure 6. Results of zero-shot OOD generalization when dimensionality of the data increases in the synthetic environment. Results are performed on 100 random test environments. Shaded area represents 95% CI.

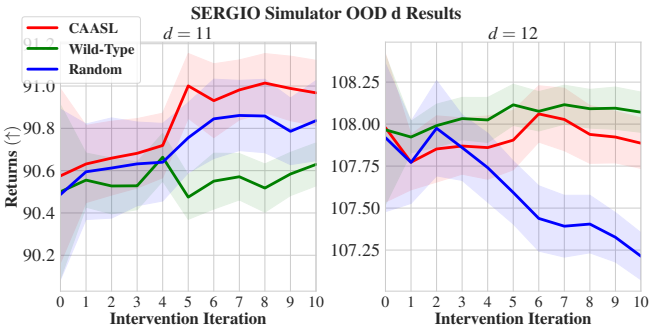

Figure 7. Results of zero-shot OOD generalization when dimensionality of the data increases in the SERGIO environment. We notice that the random baseline is very competitive. We hypothesize that the symmetries encoded in the policy, which are crucial for generalization, might not be so relevant in this setting due to high amount of missing data. Results are performed on 100 random test environments. Shaded area represents 95% CI.

G. Full Results

G.1. Results on Zero-Shot OOD Generalization to Higher Dimensions

The results for Zero-shot OOD generalization to problems of higher dimensions is available in figs. 6 and 7.

G.2. Results on all Metrics

Herein we include all the results that correspond to other metrics omitted in the main text. In particular, apart from returns, we measure Structural Hamming Distance (SHD), Area under Precision Recall Curve (AUPRC), and Edge F1 (Edge F1) score. These additional metrics are defined as follows:

- **SHD**: Structural Hamming Distance (SHD) measures the hamming distance between graphs. In particular, it is a measure of number of edges that are to be added, removed or reversed to get the ground truth from the estimated graph. Since we have a posterior distribution $q(\hat{A} \mid h_t)$ over graphs, we measure the *expected* SHD:

$$\text{SHD} \coloneqq \mathop{\mathbb{E}}_{\hat{A} \sim q(\hat{A} \mid h_t)} [\text{SHD}(\hat{A}, A^{GT})] \approx \frac{1}{100} \sum_{i=1}^{100} [\text{SHD}(\hat{A}^{(i)}, A^{GT})] \quad \text{, with} \quad \hat{A}^{(i)} \sim q(\hat{A} \mid h_t)$$

  where A^{GT} is the ground-truth causal graph.

- **Edge F1**: It is F1 score of each edge being present or absent in comparison to the true edge set, averaged over all edges.
- **AUPRC**: It is the area under the precision recall curve obtained by thresholding the edge probabilities of the amortized graph posterior $q(\hat{A} \mid h_t)$.

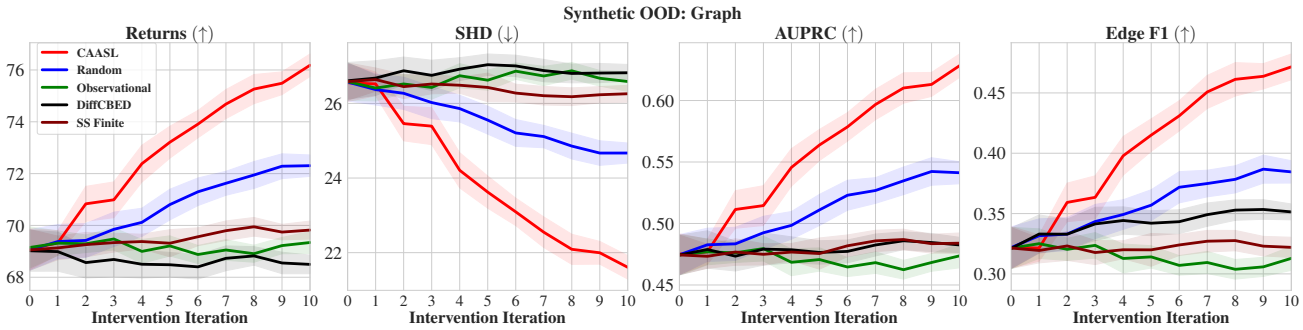

Figure 8. Results of zero-shot OOD graph setting with various intervention strategies on 100 random test environments. Shaded area represents 95% CI.

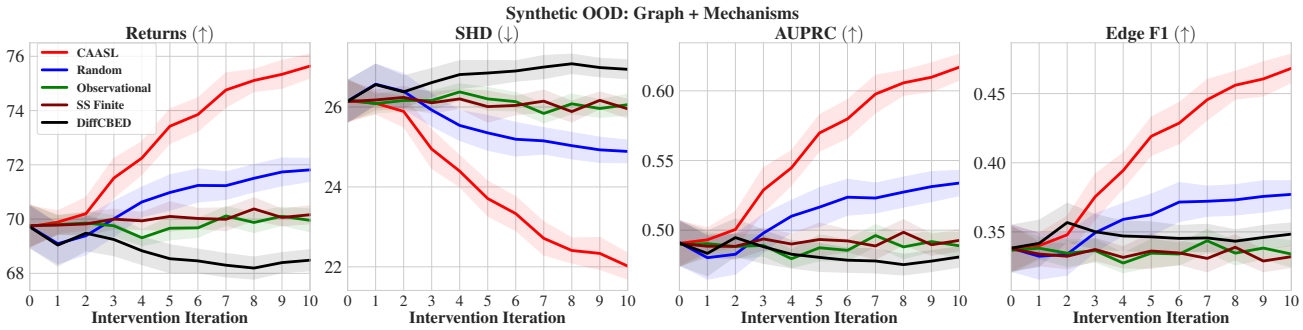

Figure 9. Results of zero-shot OOD graph and mechanisms setting with various intervention strategies on 100 random synthetic test environments. Shaded area represents 95% CI.

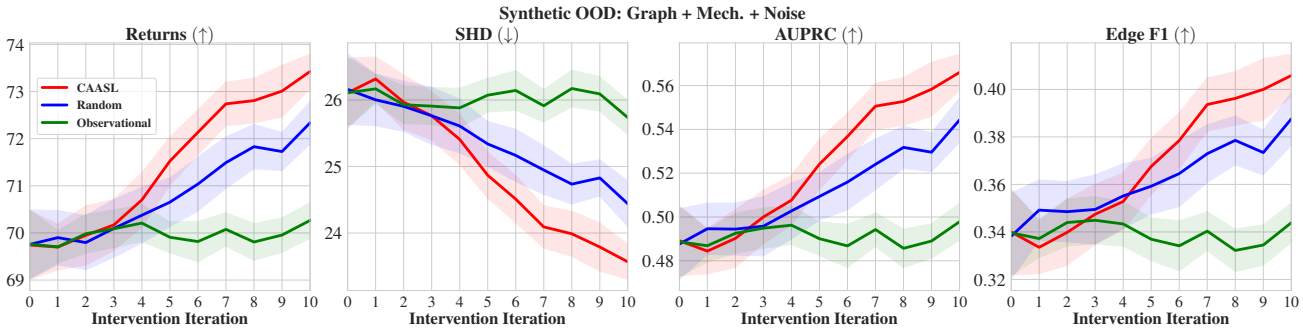

Figure 10. Results of zero-shot OOD graph, mechanisms and noise setting with various intervention strategies on 100 random synthetic test environments. Shaded area represents 95% CI.

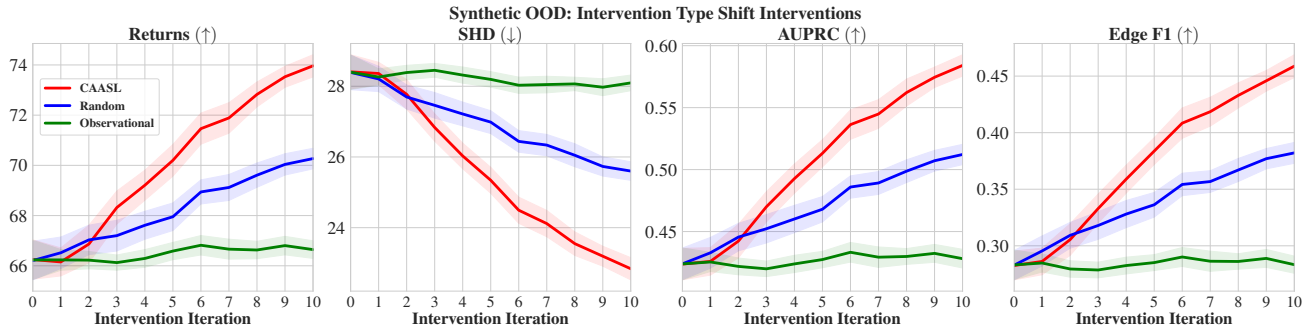

Figure 11. Results of zero-shot OOD intervention type setting with various intervention strategies on 100 random synthetic test environments. Shaded area represents 95% CI.

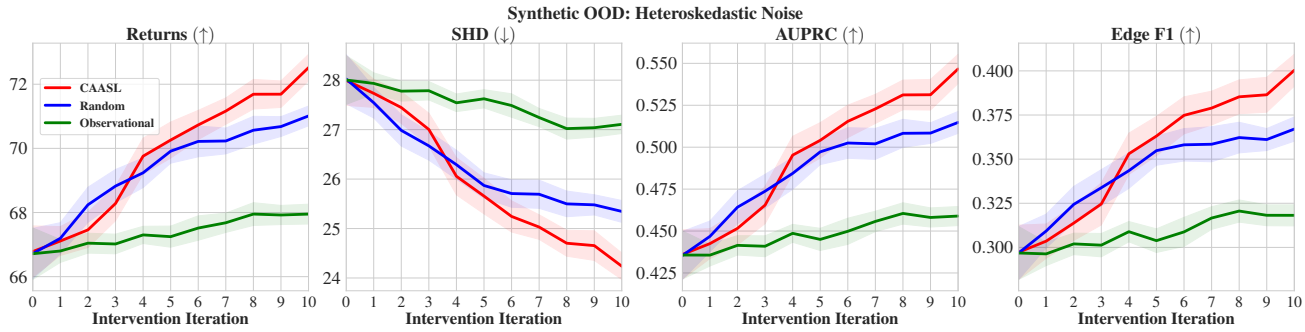

Figure 12. Results of zero-shot OOD heteroskedastic noise setting with various intervention strategies on 100 random synthetic test environments. Shaded area represents 95% CI.

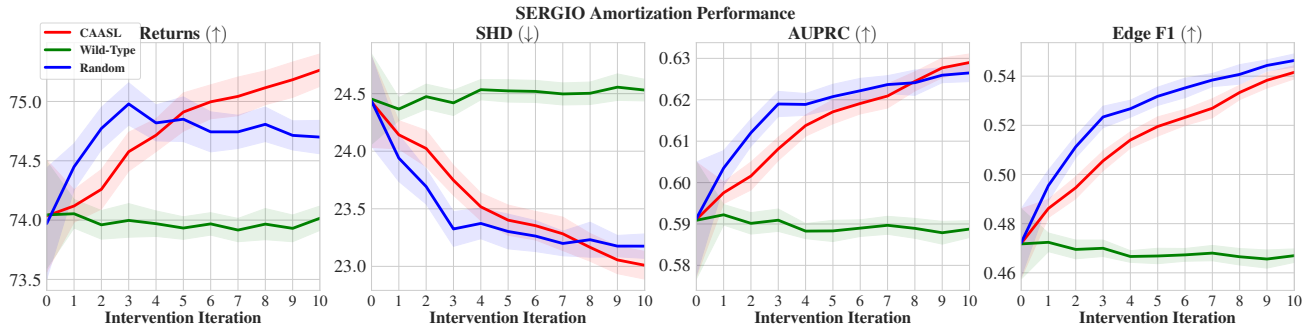

Figure 13. Results of amortization with various intervention strategies on 100 random SERGIO test environments. Shaded area represents 95% CI.

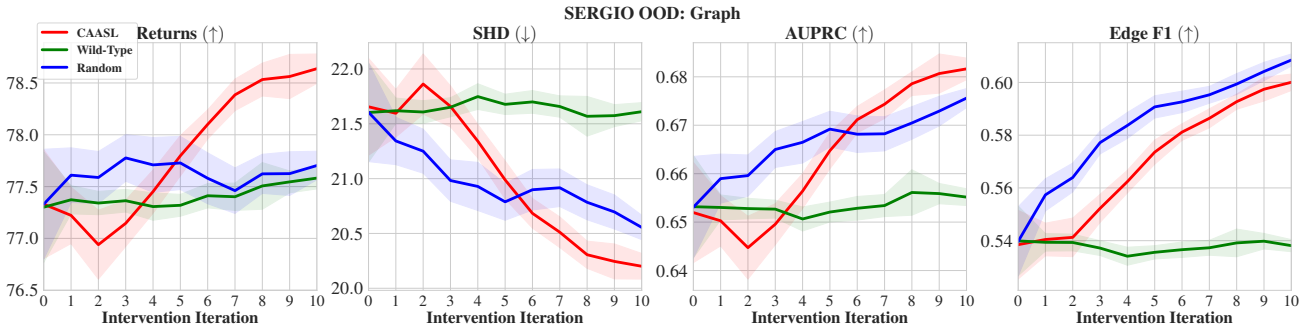

Figure 14. Results of zero-shot OOD graph setting with various intervention strategies on 100 random SERGIO test environments. Shaded area represents 95% CI.

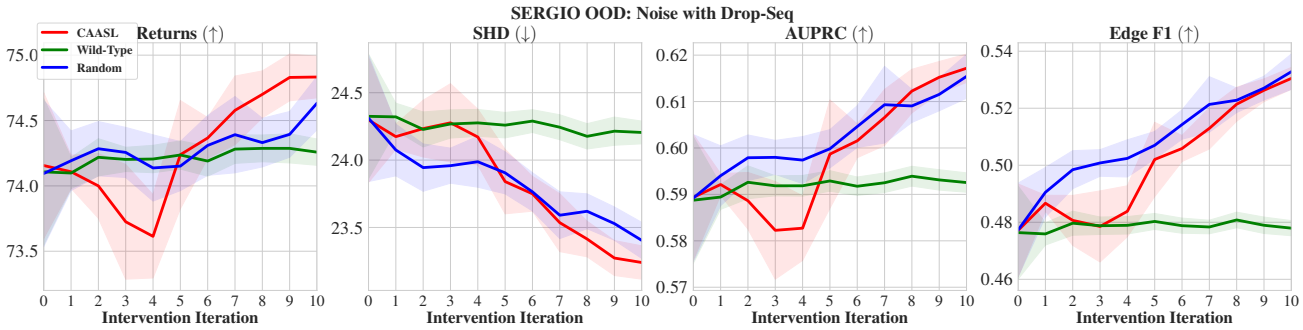

Figure 15. Results of zero-shot OOD scRNA-seq platform and their noise setting with various intervention strategies on 100 random SERGIO test environments. Shaded area represents 95% CI.

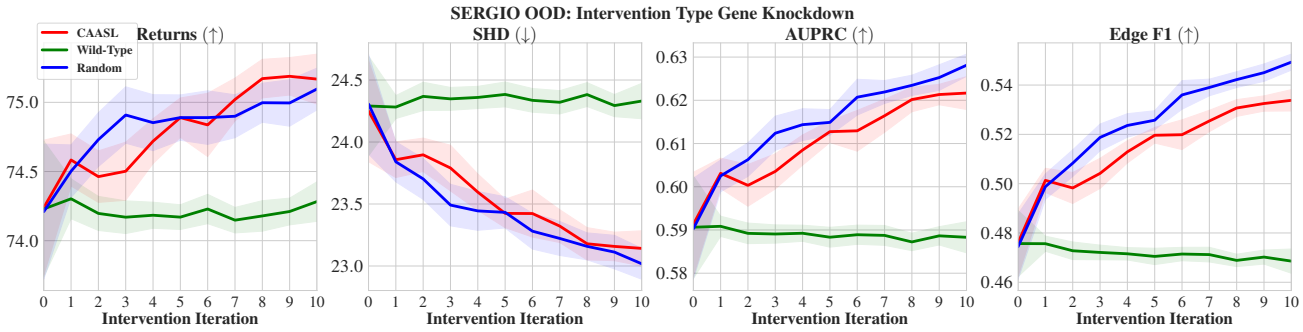

Figure 16. Results of zero-shot OOD intervention type changing to gene knockdown with various intervention strategies on 100 random SERGIO test environments. Shaded area represents 95% CI.

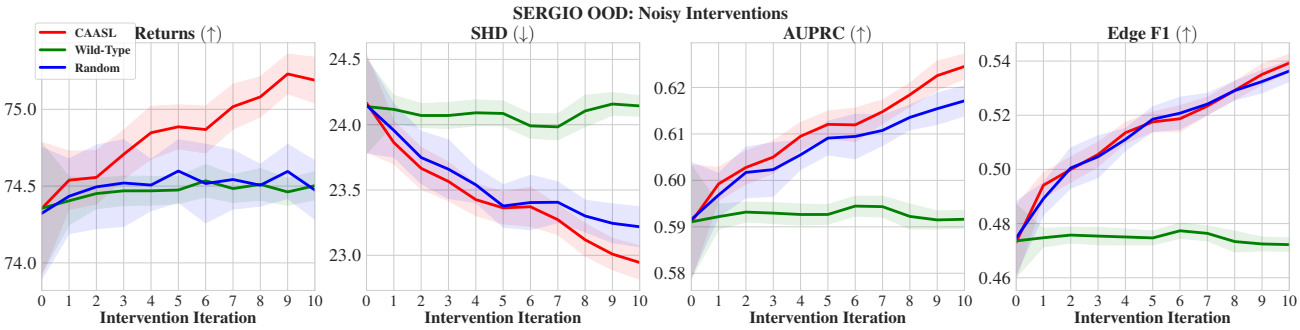

Figure 17. Results of zero-shot OOD noisy gene knockouts with various intervention strategies on 100 random SERGIO test environments. Shaded area represents 95% CI.