Causal Influence Aware Counterfactual Data Augmentation

Núria Armengol Urpí
ETH Zürich & Max Planck Institute for Intelligent Systems
Tübingen, Germany
nuriaa@ethz.ch

Georg Martius
University of Tübingen & Max Planck Institute for Intelligent Systems
Tübingen, Germany
georg.martius@uni-tuebingen.de

Abstract

Pre-recorded data and human demonstrations are practical resources for teaching robots complex behaviors. However, the combinatorial nature of real-world scenarios requires a huge amount of data to prevent neural network policies from picking up on spurious and non-causal factors. We propose CAIAC, a data augmentation method that creates synthetic samples from a fixed dataset without the need to perform new environment interactions. Motivated by the fact that an agent may only modify the environment through its actions, we swap causally *action*-unaffected parts of the state-space from different observed trajectories. In several environment benchmarks, we observe an increase in generalization capabilities and sample efficiency.

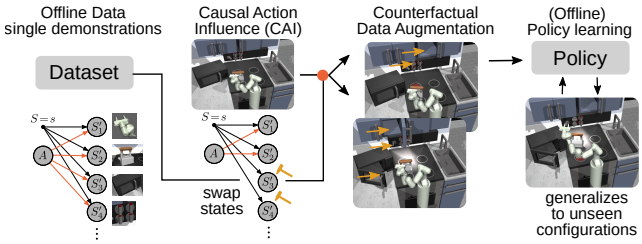


Figure 1: Overview of the proposed approach. Interactions between the agent and entities in the world are sparse. We use causal action influence (CAI), a local causal measure, to determine action-independent entities and create counterfactual data by swapping states of these entities from other observations in the dataset. Offline learning with these augmentations leads to better generalization.

1 Introduction

Teaching robots via collected datasets shows promise towards capable robotic assistants[2, 5, 6], but faces challenges when applied to realistic scenarios due to combinatorial complexity spurring from the existence of many entities [3]. Demonstrations can only capture a fraction of all possible configurations, hindering robust generalization to unfamiliar situations. While learning algorithms are prone to picking up on spurious correlations in the data, humans excel at inferring task-relevant parts of the environment, possibly due to relying on a causal representation of the world [17].

NeurIPS 2023 Workshop on Robot Learning: Pretraining, Fine-Tuning, and Generalization with Large Scale Models, New Orleans, United States of America

Our approach, rooted in a local causal inference perspective, introduces counterfactual data augmentations without the need for additional environment interactions, nor relying on counterfactual model rollouts. Instead, we leverage recorded data by substituting locally causal independent factors with those of different observed trajectories.

We exploit the assumption that an agent can only affect its environment through actions. Hence, reduce the problem of estimating the full causal structure to only measuring the influence of actions over objects. This quantity can then be explicitly estimated through the local Causal Action Influence (CAI) measure [22], thus removing the need for heuristics for causal discovery [19]. The proposed method generates training data that would otherwise be far out-of-distribution, as illustrated in Fig. 1.

Our framework works as an independent module and can be used with any learning algorithm. We demonstrate this through empirical results in high-dimensional offline goal-conditioned tasks. We show that our method, which we refer to as ***Ca**usal **I**nfluence **A**ware **C**ounterfactual Data Augmentation* (CAIAC) leads to enhanced generalization and improved performance when learning from a modest amount of demonstrations.

2 Causal graphical models

We consider a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, consisting of state space, action space, transition kernel, reward function and discount factor, respectively. We assume a known and fixed state-space factorization $\mathcal{S} = \mathcal{S}_1 \times ... \times \mathcal{S}_N$ for N entities, where each factor $\mathcal{S}_i$ corresponds to the state of an entity.

We model the state evolution of the underlying MDP from time t to $t+1$ using a causal graphical model (CGM) [18, 16] over the set of random variables $\mathcal{V} = \{S_1, ..., S_N, A, S'_1, ..., S'_N\}$. with a directed acyclic graph (DAG) $\mathcal{G}$ and a conditional distribution $P(V_j \mid \mathrm{Pa}_{\mathcal{G}}(V_j))$ for each node $V_j \in \mathcal{V}$. Due to the structure of the MDP and our knowledge about time, we assume, as in [22, 19], a causal graph without connections among nodes at the same time step and without edges from the future to the past, as depicted in Fig. 2(a). If the graph $\mathcal{G}$ is structurally minimal, we can think of its edges as representing global causal dependencies. Each node is independent of its non-descendants given its parents, so that $S'_j \perp\!\!\!\perp V_j \mid \mathrm{Pa}(S'_j)$ for all nodes $V_j \notin \mathrm{Pa}(S'_j)$ [18, Def 6.21]. The probability distribution of S'_j is hence fully specified by its parents $P(S'_j|S, A) = P(S'_j \mid \mathrm{Pa}(S'_j))$.

3 Method

Our method enables offline learning algorithms to learn a good policy in states that are not necessarily within the support of the data distribution. This is achieved by augmenting real data with counterfactual modifications to *causally action-unaffected* entities. We use a local causal graph formulation and rely on an independence assumption to explicitly compute causal influence through the Causal Action Influence (CAI) [22] measure.

3.1 Local causal graphs

In most non-trivial environments the global causal graph is fully connected between timesteps: an edge $S_i/A \to S'_j$ is present as long as there is a single timestep for which S_i/A affects S'_j (Fig. 2(a)). In most environments, however, given a concrete timestep, there is limited interaction between entities. For example, given the state configuration in Fig. 2(b), the robot can only influence the kettle and its own end-effector, but none of the other entities.

With this in mind, we focus on the causal structure implied by a specific state configuration $S = s$, i.e. the *local causal model* in s, as proposed in [22, 19]. In the *state-conditioned local causal graph* $\mathcal{G}_{S=s}$, the absence of an edge (V, S'_j) for $V \in \{S_1, \ldots, S_N, A\}$ is implied by $S'_j \perp\!\!\!\perp V|S = s$, i.e., entity j's next state is independent of V. An example is given in Fig. 2(b).

To synthesize counterfactual experience, we are left with inferring the local factorization, i.e. with the hard problem of discovering the conditional causal structure[18]. Therefore, we make the key assumption that interactions between entities only rarely occur and are thus negligible. While the correctness of generated counterfactuals will rely on this assumption to hold, we argue that this is realistic in several robotics tasks of interest, including the ones we empirically evaluate. For example

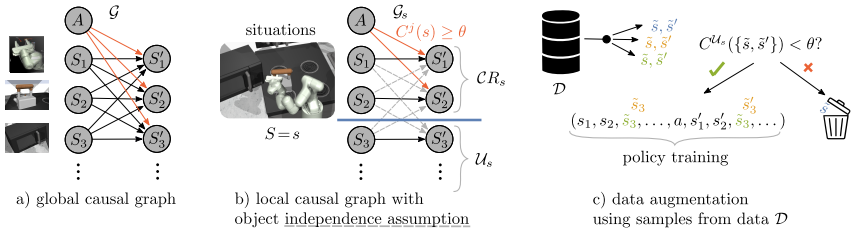


Figure 2: Illustration of counterfactual data augmentation. The global causal graph does not allow for factorization (a). Our local causal graph (b) is pruned by causal action influence. Object-object interactions are assumed to be rare/not existing (gray dashed). We swap elements not under control (in set $\mathcal{U}$) by samples from the data, creating alternative perceptions that yield the same outcome (c).

in the kitchen environment depicted in Fig. 1, the entities can hardly influence each other. More formally, and in a graphical sense, we assume that there is no arrow $S_i \to S'_j, i \neq j$ as visualized by the gray dashed lines in Fig. 2(b). We note that only two groups of arrows remain in the causal graph: $S_j \to S'_j$, which we assume to always be present, and $A \to S'_j$.

Crucially, this practical assumption allows us to reduce the hard problem of local causal discovery to the more approachable problem of local action influence detection, that is, to predict whether given a specific state configuration, the agent can influence an entity through its actions. As a result, instead of resorting to heuristics [19], we can use an *explicit* measure of influence, namely the recently proposed method CAI [22], which we introduce below.

3.2 Causal Action Influence detection

To predict the existence of the edge $A \to S'_j$ in the local causal graph $\mathcal{G}_{S=s}$, [22] use conditional mutual information (CMI) [7] as a measure of dependence. Therefore, in each state $S = s$ we use the point-wise CMI as a state-dependent quantity that measures causal action influence (CAI), given by $C^j(s) := I(S'_j; A \mid S = s) = \mathbb{E}_{a\sim\pi}\big[D_{KL}\big(P_{S'_j|s,a} \,\big|\big|\, P_{S'_j|s}\big)\big]$. See Appendix A.7 for more details.

3.3 Inferring local factorization

For each state s in our data set $\mathcal{D}$, we compute the uncontrollable set, as the set of entities in s for which the agent has no causal action influence, expressed as $\mathcal{U}_s = \{s_j \mid C^j(s) \leq \theta, j \in [1, N]\}$ where θ is a fixed threshold. The set $\mathcal{U}_s$ contains all entities j for which the arrow $A \to S'_j$ in the local causal graph $\mathcal{G}_s$ does not exist. The remaining entities are contained in the set of controllable entities $\mathcal{CR}_s = \{s_1, \ldots, s_N\} \setminus \mathcal{U}_s$. An illustration is given in Fig. 2(b). With our assumptions and the sets $\mathcal{U}_s$ and $\mathcal{CR}_s$ we find that the local causal graph $\mathcal{G}_s$ is divided into the *disconnected* subgraphs $\mathcal{G}_s^{\mathcal{CR}}$, that contains the entities in $\mathcal{CR}$ and A, and into $|\mathcal{U}_s|$ *disconnected* subgraphs $\mathcal{G}_{s_i}^{\mathcal{U}}, i \in [1, |\mathcal{U}_s|]$, each of which contains an entity in $\mathcal{U}_s$ with only self-links, see Fig. 2(b).

3.4 Computing counterfactuals

Given the partitioning of the graph described above, we can think of each subgraph as an independent causal mechanism that can be reasoned about separately. Hence, we can create counterfactuals in the following way: given two transitions (s, a, s') and $(\hat{s}, \hat{a}, \hat{s}') \in \mathcal{D}$, which have at least one uncontrollable subgraph structure in common (i.e. $\mathcal{U}_s \cap \mathcal{U}_{\hat{s}} \neq \emptyset$), we generate a counterfactual transition $(\tilde{s}, \tilde{a}, \tilde{s}')$ by swapping the entity transitions (s_i, s'_i) with $(\hat{s}_i, \hat{s}'_i)$ and $i \in \mathcal{U}_s \cap \mathcal{U}_{\hat{s}}$.

Importantly, the causal structure needs to stay the same after the intervention. The counterfactual $\tilde{s}$ should then be discarded if it alters the set of controllable entities: i.e. $\mathcal{CR}_s \neq \mathcal{CR}_{\tilde{s}}$. This operation is only possible when causal influence can be correctly measured in the counterfactual. As CAI, like previous heuristics, relies on a learned transition model, the counterfactual is an out-of-distribution sample, and the output of the model will likely be inaccurate. In practice, we avoid this additional check and accept creating a small fraction of potentially unfeasible situations. The pseudocode of our method CAIAC is given in Algorithm 1.

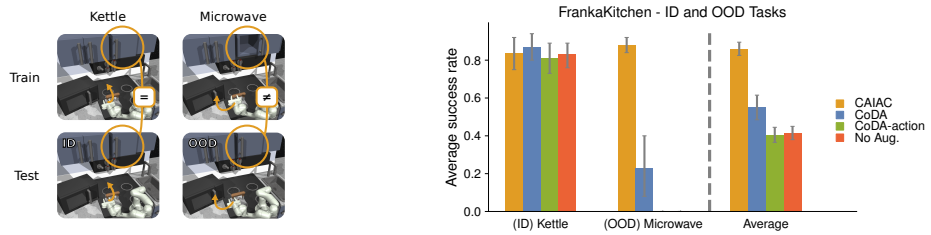


Figure 3: Motivating Franka-Kitchen example. The experimental setup (left) and success rates for in-distribution and out-of-distribution tasks (right). Metrics are averaged over 10 seeds and with 95% simple bootstrap confidence intervals.

4 Experiments

We evaluate CAIAC in two goal-conditioned settings: offline RL (see A.3.1) and offline self-supervised skill learning. We are interested in evaluating whether CAIAC 1) leads to better generalization to unseen configurations, 2) enlarges the support of the joint distribution over the state space in low data regimes. **Baselines** We compare CAIAC with CoDA [19], a counterfactual data augmentation method, which uses the attention weights of a transformer model to estimate the local causal structure, and CoDA-ACTION, an ablated version of CoDA, which only accounts for the influences of the action and thus is a 'heuristic'-sibling of our method. We also include a baseline without data augmentation (NO-AUGM). Additional experiments are in Appendix A.2.1, A.3.1, A.4.

4.1 Goal conditioned offline self-supervised skill learning

Our first experiment is on the Franka-Kitchen environment [11] and is designed to verify claim 1). As a downstream learning algorithm we use LMP [14], an offline goal-conditioned self-supervised learning algorithm. See A.5.1 and A.6.1 for details on the LMP and the Franka-Kitchen environment. We create a reduced dataset from the original D4RL dataset [9], that contains only demonstrations for the microwave task (MW) and the kettle (K) task. During demonstrations for the (MW) task, we initialize the cabinet to be always open, whereas for the (K) task, it remains closed. At inference time, we initialize the environment with its default initial configuration (crucially, the cabinet is closed), and we evaluate both tasks ((K) and (MW)), as shown in Fig. 3(left). While the (K) task was demonstrated for the current configuration (in-distribution, ID), the agent is evaluated on an out-of-distribtion (OOD) configuration for the (MW) task.

As shown in Fig. 3, as expected, all methods are able to solve the (K) task, since it is in-distribution (ID). However, we observe fundamentally different results for the OOD (MW) task. The performance of CAIAC is not affected, because it detects that the sliding cabinet is never under control of the agent, and thus can create relevant counterfactuals. However, the performance of CoDA and CoDA-ACTION is drastically impaired in the OOD setting. Despite the simplicity of the setting, the input dimensionality of the problem is high, and the transformer attention weights are not able to recover the correct causal graph. Hence, they create dynamically-unfeasible counterfactuals which affect performance. Finally, as expected, NO AUGM. fails to solve the OOD (MW) task.

5 Discussion

We proposed CAIAC as a method for counterfactual data augmentation without the need for additional environment interaction nor counterfactual model rollouts, which can be used with any learning algorithm. By adding an inductive bias on the causal structure of the graph, we circumvented the problem of full causal discovery and reduced it to the computation of an explicit measure of the agent's causal action influence over objects. Empirically, we show that CAIAC leads to enhanced performance and generalization to unseen configurations, suggesting that further advances in addressing causal discovery problems can be substantially beneficial for robot learning.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[2] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. 2022.

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[4] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[7] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[8] J.-L. Durrieu, J.-Ph. Thiran, and F. Kelly. Lower and upper bounds for approximation of the kullback-leibler divergence between gaussian mixture models. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4833–4836, 2012. doi: 10.1109/ICASSP.2012.6289001.

[9] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[10] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

[11] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning. *Conference on Robot Learning (CoRL)*, 2019.

[12] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

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

[14] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. *Conference on Robot Learning (CoRL)*, 2019. URL https://arxiv.org/abs/1903.01973.

[15] Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34:24379–24391, 2021.

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

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

[18] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2017. ISBN 978-0-262-03731-0. URL https://mitpress.mit.edu/books/elements-causal-inference.

[19] Silviu Pitis, Elliot Creager, and Animesh Garg. Counterfactual data augmentation using locally factored dynamics. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[20] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[21] Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task agnostic offline reinforcement learning. 2022.

[22] Maximilian Seitzer, Bernhard Schölkopf, and Georg Martius. Causal influence detection for improving efficiency in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS 2021)*, December 2021. URL https://arxiv.org/abs/2106.03443.

[23] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Acknowledgments and Disclosure of Funding

We thank Marco Bagatella, Marin Vlastelica, Cansu Sancaktar and Max Seitzer for their help throughout the project. Núria is supported by the Max Planck ETH Center for Learning Systems. Georg Martius is a member of the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645. We acknowledge the support from the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039B).

A Appendix

A.1 Pseudocode

Algorithm 1 CAIAC

input Dataset $\mathcal{D}$
  Compute uncontrollable set $\mathcal{U}_s,\ \forall s \in \mathcal{D}$ (Eq. 3.3).
  while `Training` **do**
    Sample $(s, a, s') \sim \mathcal{D}$
    $(\tilde{s}, \tilde{s}') \leftarrow (s, s')$
    for $s_i \in \mathcal{U}_s$ **do**
      Sample $(\hat{s}, \hat{a}, \hat{s}') \sim \mathcal{D}$
      if $\hat{s}_i \in \mathcal{U}_{\hat{s}}$ **then**
        $(\tilde{s}_i, \tilde{s}'_i) \leftarrow (\hat{s}_i, \hat{s}'_i)$
      end if
    end for
    Yield training samples (s, a, s') and $(\tilde{s}, a, \tilde{s}')$
  end while

A.2 Additional experiments: Goal Conditioned self-supervised skill learning

A.2.1 Franka-Kitchen: all tasks

Having evaluated CAIAC in a controlled setting, we now scale up the problem to the entire Franka-Kitchen D4RL dataset. While in the standard benchmark the agent is required to execute a single fixed sequence of tasks, we train a goal-conditioned agent and evaluate on the full range of tasks, which include the microwave, the kettle, the slider, the hinge cabinet, the light switch and the bottom left burner tasks [15]. One task is sampled for each evaluation episode. While alleviating the need for long-horizon planning, this results in a challenging setting, as only a subset of tasks is shown directly from the initial configuration. However, the largest challenge in our evaluation protocol lies in the creation of unobserved state configurations at inference time. While the provided demonstrations always start from the same configuration (e.g., the microwave is always initialized as closed), at inference time, we initialize all non-target entities (with $p = 0.5$) to a random state, hence exposing the agent to OOD states. We expect that agents trained with CAIAC will show improved performance to unseen environment configurations, as those can be synthesized through counterfactual data augmentation. The results, shown in Table 1, are consistent with the challenging nature of this benchmark, as the evaluated tasks involve OOD settings in terms of states and actions. Nevertheless, we find that CAIAC is significantly better than baselines in 4/7 tasks, while being on par with the best method in the remaining 3. As observed in the simplified setting, methods relying on heuristic-based causal discovery (CoDA and CoDA-action) suffer from misestimation of causal influence, and thus from the creation of dynamically-unfeasible training samples. Without any data augmentation, the learning algorithm cannot perform the OOD tasks.

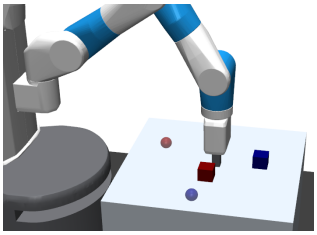
A.3 Additional experiments: Goal-conditioned Offline RL

A.3.1 Fetch-Push with 2 cubes

With this experiment, our aim is to verify claim 2) , i.e., that CAIAC can enlarge the support of the joint distribution in low data regimes. We evaluate CAIAC in Fetch-Push, where a robotic arm

Table 1: Average success rates for Franka-Kitchen tasks with OOD initial configurations, computed over 10 seeds and 10 episodes per task with 90% simple bootstrap confidence intervals.

Algorithm	**CAIAC**	CoDA	CoDA-action	No-Augmentation
Kettle	**0.41 ± 0.06**	0.18 ± 0.05	0.16 ± 0.1	0.06 ± 0.04
Microwave	**0.30 ± 0.06**	0.07 ± 0.05	0.0 ± 0.03	0.01 ± 0.03
Bottom-burner	**0.10 ± 0.07**	0.01 ± 0.01	0.0 ± 0.0	0.0 ± 0.0
Slide cabinet	0.04 ± 0.01	**0.10 ± 0.05**	0.02 ± 0.02	0.06 ± 0.03
Light switch	**0.03 ± 0.03**	0.0 ± 0.0	0.0 ± 0.0	**0.00 ± 0.04**
Hinge cabinet	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0



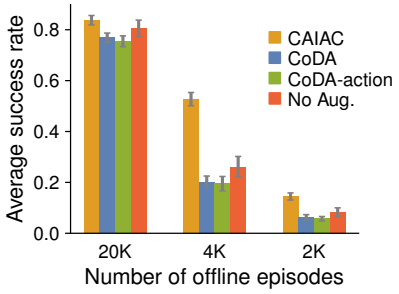


Figure 4: Success rates in different offline data regimes for Fetch-Push with 2 cubes. Metrics are averaged over 30 seeds and 50 episodes with 95% simple bootstrap confidence intervals.

has to slide two blocks to target locations. For this experiment we collect 20k trajectories using an expert policy (30%) and random policy (70%) and train an agent offline using TD3 [10] in different data regimes: namely 100% of data, 20%, and 10%. We use HER [1] to relabel goals with `future` strategy on real data and with `random` strategy on the counterfactual data. To disentangle the impact of the relabel strategy from the impact on the counterfactual data generation, we also relabel the same percentage of goals with `random` strategy for the NO AUGM. baseline. More details are given in Appendix A.6.2 and A.5.2. This choice of learning algorithms and dataset is taken to also validate claim 3), and confirm that CAIAC can be applied to fundamentally different methods (i.e., hierarchical behavioral cloning and flat reinforcement learning, trained on near-expert and mostly random data). We compare success rates between baseline and CAIAC among different data regimes in Fig. 4(right).

For high data regimes, all methods perform similarly, given that there is enough coverage of the state space in the original dataset. Additionally, given enough data and the low dimensionality of the setting, the transformer model is able to discover the causal graph and creates realistic counterfactuals. When moving to medium data regimes we observe the occurrence of a significant performance gap. Given sufficient support for the marginal distribution over the state of each entity, CAIAC can substantially increase the support of the joint distribution state space and therefore achieves higher performance. In the smallest data regime, we see that CAIAC is still significantly better than the baselines, but the performance gap is reduced, showing that when the support of the marginal distributions shrinks it is not possible to retrieve samples to cover the entire joint state space. On the other hand, for medium and low data regimes CODA and CODA-ACTION do not outperform the NO AUGM. baseline. This is because, with a lack of data, the transformer model is not able to disentangle the local causal graph, resulting in a densely connected estimated graph. Thus, very little augmentation can be applied. We note that, while previous work [19] has shown good performance of CODA in this environment (although in an online setting), it resorted to a handcrafted heuristic to detect influence using domain knowledge, namely "objects are disentangled if more than 10cm apart".

A.4 Ablation: Ratio of observed-to-counterfactual data

In this section, we study the effect of the ratio of observed-to-counterfactual data generated with CAIAC, by evaluating downstream performance on the Franka-Kitchen motivating example, as presented in 4.1. Empirical results for this ablation are shown in Fig. 5. As expected, we observe that

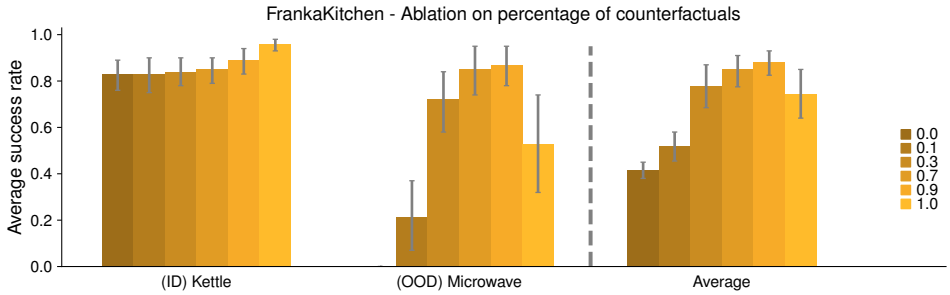


Figure 5: Performance of CAIAC on motivating Franka-Kitchen example when controlling the percentage of counterfactual samples in each batch. Metrics are averaged over 10 seeds and 10 episodes per task, with 95% simple bootstrap confidence intervals.

the ratio of counterfactuals does not have any significant impact on the success rate on the `(k)` task. This is because the task is evaluated in distribution, and hence the downstream learning algorithm does not require observing counterfactual experience (but still doesn't suffer from it). For the OOD `(mw)` task we see that increasing the number of counterfactuals up to a 0.9 ratio has a positive effect in performance, leading the agent to generalize better to the OOD distribution. However, when the ratio is increased up to 1., performance degrades, showing that the agent also needs to observe real data and can suffer from the induced selection bias, as also observed in [19]. With a ratio 0.0, we recover the performance of the No Augm. baseline.

A.5 Implementation of downstream learning algorithms

In this section, we report implementation details concerning the learning algorithms. For a fine-grained description of all hyperparameters, we refer to our codebase at `https://sites.google.com/view/caiac`.

A.5.1 Goal-conditioned offline self-supervised skill learning

For the goal-conditioned self-supervised learning experiments we used LMP [14], a goal-conditioned self-supervised method. Formally, LMP is a sequence-to-sequence VAE [23, 4] autoencoding random experiences extracted from the dataset through a latent space. In our case, we use experiences of fixed window length κ. It consists of a stochastic sequence encoder, or learned posterior, which maps a sequence τ to a distribution in latent plan space $q(z|\tau)$, a stochastic encoder or learned goal-conditioned prior $p(z|s, g)$ and a decoder or plan and goal conditioned policy: $\pi(a|z, s, g)$. The main difference with the original implementation is that the latent goal representation is only added to the prior, but not the decoder. Additionally, we also implemented KL balancing in the loss term between the learned prior and the posterior: we minimize the KL-loss faster with respect to the prior than the posterior. Given that the KL-loss is bidirectional, in the beginning of training, we want to avoid regularizing the plans generated by the posterior towards a poorly trained prior. Hence, we use different learning rates, $\alpha = 0.8$ for the prior and $1 - \alpha$ for the posterior, similar to Hafner et al. [12]. These two modifications were also suggested in [21]. Additionally, our decoder was open-loop (instead of close loop): given a sampled latent plan z it decodes the whole trajectory of length `skill length`$= N$, i.e. our decoder is $\pi(\hat{a}|z)$, where $\hat{a} = a_t, ..., a_{t+N}$ is the sequence of decoded actions, instead of $\pi(a|z, s, g)$. This modification was needed due to the *skewness* of the dataset. Since the demonstrations were provided from an expert agent, given most of the states, the distribution over actions is unimodal: when the robot is close to the microwave, the only sequence of actions in the dataset is the one that opens the microwave. Hence, a close-loop decoder would learn to ignore the latent plan, and only rely on the state. To solve this issue, we make the decoder open-loop.

Given the inherent temporal abstraction of the algorithm, we generate counterfactuals of fixed length $\kappa > 1$ by computing the uncontrollable set $\mathcal{U}_{(s_t, s_{t+1}, ..., s_{t+\kappa})}$ for the sampled window instead of a single time slice, where

$$\mathcal{U}_{(s_t, s_{t+1}, ..., s_{t+\kappa})} = \bigcap_{\tau=t}^{t+\kappa-1} \mathcal{U}_{s_\tau}. \tag{1}$$

A.5.2 Goal-conditioned Offline RL

For the goal-conditioned offline RL setting we implement the TD3 algorithm [10] with HER [1]. Unless specified the hyperparameters used were the ones from the original implementation. We use HER [1] to relabel the goals for real data, with a `future` relabeling strategy with $p = 0.5$, where the step where sampled from a geometric distribution with $p_{geom} = 0.2$. For the counterfactual data we relabel the goals with $p = 0.5$ randomly sampling from the achieved goals in the buffer of counterfactual samples. In the experiments we realized that the relabeling strategy had an impact on the performance of the downstream agent. To disentangle the impact of the relabel strategy from the impact on the counterfactual data generation and do not benefit any of methods, we also relabeled the same percentage of goals (i.e. $p = 0.25$) with `random` strategy for the No Augm. baseline. We train each method for 1.2M gradient steps, although all methods reach convergence after 600k gradient steps. The percentage of counterfactuals in each batch is set to 0.5.

A.6 Experimental details

A.6.1 Franka-Kitchen

We use the kitchen environment from the D4RL benchmark [9] which was originally published by Gupta et al. [11], which consists of a series of teleoperated sequences in which a 7-DoF robot arm manipulates different parts of the environment (e.g., it opens microwave, switches on the stove). Crucially, all demonstrations are limited to a few manipulation sequences (for example, first opening the microwave, turning on a burner, and finally the light). Thus, the support of the joint distribution over entities in the environment is reduced to only a few combinations. To illustrate this using a concrete example, the light is never on if the microwave hasn't been opened yet. The D4RL dataset contains different dataset versions: `kitchen-complete, kitchen-partial, kitchen-mixed`, which contain 3690, 136950 and 136950 samples respectively, making up to approximately 14 demonstrations for `kitchen-complete` and 400 demonstrations for each `kitchen-partial` and `kitchen-mixed`. The simulation starts with all of the joint position actuators of the Franka robot set to zero. The doors of the microwave and cabinets are closed, the burners turned off, and the light switch also off. The kettle will be placed in the bottom left burner. The observation are 51-dimensional, containing the joint positions of the robot (9 dim), the positions of the all the kitchen items (21 dim) and the goal positions of all the items (21 dim). The length of the episode is 280 steps, but the episode will finish earlier if the task is completed. The task is only considered solved when all the objects are within a norm threshold of 0.3 with respect to the goal configuration. While in the standard benchmark the agent is required to execute a single fixed sequence of tasks, we train a goal-conditioned agent, and evaluate on one task per each evaluation episode. For the Franka-kitchen motivating example (see 4.1) we query for either the kettle or the microwave task, in the Franka-Kitchen: All tasks (see A.2.1) we query for the full range of tasks, which include the microwave, the kettle, the slider, the hinge cabinet, the light switch and the bottom left burner tasks. While alleviating the need for long-horizon planning, this results in a challenging setting, as only a subset of tasks is shown directly from the initial configuration.

Franka-Kitchen: motivating experiment For the first experiment (see Subsection 4.1), we modify the dataset version `kitchen-mixed` to only contain ~ 50 demonstrations of length ~ 40 timesteps for each `(mw)` and `(k)` task. During demonstrations for the `(mw)` task, we initialize the cabinet to be always open, whereas for demonstrations for the `(k)` task, it remains closed. The rest of the objects are set to the default initial configuration. The goal configuration for all the objects was set to their initial configuration (as defined above), except for the microwave or the kettle, which were set to the default goal configuration when querying for the `(mw)` and `(k)` tasks respectively.

Franka-Kitchen: all tasks For the second experiment (see Subsection A.2.1) we merge the 3 provided datasets `kitchen-complete, kitchen-partial, kitchen-mixed`. For this experiment, each object (except the one related to the task at hand to ensure non-trivial completion), was randomly initialized with $p = 0.5$, otherwise it was initialized to the default initial configuration (as defined above).

A.6.2 Fetch-Push with 2 cubes

Expert data for the experiment in A.3.1 includes 6000 episodes collected by an agent trained online using TD3 and HER up to approximately 95% success rate. We additionally collect 14000 episodes

with a random agent, which make up for the random dataset. This sums up to a total of 20000 episodes (each of length 100 timesteps), with 30% expert data and 70% random data. Initial positions and goal positions of the cubes are sampled randomly on the table, whereas the robot is initialized in the center of the table with some additional initial random noise . The rewards are sparse, giving a reward of -1 for all timesteps, except a reward of 0 when the position of each of the 2 blocks are within a $2-$norm threshold of 0.05. The observation space is 34-dimensional, containing the position and velocity of the end effector (6dim), of the gripper (4dim) and the object pose, linear and rotational velocities of the objects (12dim each). In contrast to the original `Fetch-Push-v1` [20] environment and similarly to Pitis et al. [19] we do include parts of the state space accounting for relative position or velocities of the object with respect to the gripper, which would entangle the two. The goal is 6-dimensional encoding the position for each of the objects. The action space is 4-dimensional encoding for the end-effector position and griper state. At test time we count the episode as successful upon reaching the goal configuration (i.e., observing a non-negative reward).

A.7 Details on Influence Detection Evaluation

To detect causal action influence we use CAI, as described in 3.2.

$$C^j(s) := I(S'_j; A \mid S = s) = \mathbb{E}_{a\sim\pi}\big[D_{KL}\big(P_{S'_j|s,a} \;\big|\big|\; P_{S'_j|s}\big)\big]. \tag{2}$$

The transition model $P_{S'_j|s,a}$ is modeled as a Gaussian neural network (predicting mean and variance) that is fitted to the training data $\mathcal{D}$ using negative log likelihood. The marginal distribution $P_{S'_j|s}$ is computed in practice using M empirical action samples with the full model: $P_{S'_j|s} \approx \frac{1}{M}\sum_{m=1}^{M} P_{S'_j|s,a^{(m)}},\ a^{(m)} \sim \pi$. We estimate the KL using an approximation for Gaussian mixtures from Durrieu et al. [8]. We refer the reader to Seitzer et al. [22] for more details.

For it we need to learn the transition model $P_{S'_j|s,a}$.

World model training For the Franka-Kitchen experiments all models were trained to predict the full state of the environment. For increased performance in the Fetch-Push task, all models were trained to predict the next position of the end effector of the agent gripper and of the objects (3 dimensions each). For CAIAC the transition model $P_{S'_j|s,a}$ is modeled as a Gaussian neural network (predicting mean and variance) that is fitted to the training data $\mathcal{D}$ using negative log likelihood. We used a simple multi-layer perceptron (MLP) with two separate output layers for mean and variance. To constrain the variance to positive range, the variance output of the MLP is processed by a softplus function (given by $\log(1 + exp(x))$), and a small positive constant of 10^{-8} was added to prevent instabilities near zero. We also clip the variance to a maximum value of 200. For weight initialization, orthogonal initialization is used. For the Franka-kitchen we use larger MLPS, with 3 layers for the simplified and 4 layers for the full experiment, each with 256 units and a learning rate of $8e^{-4}$.

For `CODA` and `CODA-action` we use a self-implementation of the transformer model. We use a model with 3 layers and 4 attention heads for the Fetch-Push task and 5 layers and 4 heads for all the Franka-Kitchen tasks, with an embedding space and output space of 128 dimensions each. We also used a learning rate of $8e^{-4}$.

All models were trained for 100k gradient steps, and tested to reach low MSE error for the predictions in the validation set (train-validation split of 0.9-0.1). We trained all models using the Adam optimizer [13] , with default hyperparameters.

In general, the models were trained using the same data as for the downstream task for all experiments. However, for the Franka-Kitchen task, we add some additional collected data on the environment when acting with random actions. The reason is that, in order to compute CAI, we query the model on randomly sampled actions from the action space. Due to the expert nature of the kitchen dataset comes from an informed agent, the original dataset might lack random samples and hence we would query the model OOD when computing CAI. For both experiments in the Franka-Kitchen simplified experiment we added 1x the original dataset of random data. Further experiments on the impact of the amount of random data could be beneficial. This was not needed for the Fetch-Push task since the dataset already contains random action. We note that this additional data is also provided for training all transformer-based baselines.

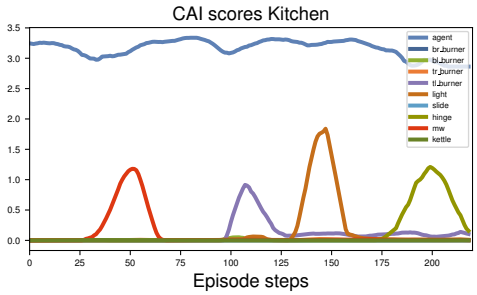


Figure 6: Computed CAI scores per each entity on one of the demonstrations for the Franka-Kitchen dataset. We can observe how the influence of the agent's actions over objects changes over time. First influencing the microwave (mw), then the top-left bottom burner (tl_burner), then the light switch (light) and finally the hinge cabinet (hinge). We selected a threshold of $\theta = 0.3$.

CAI scores In practice, we compute the cai scores using the estimator:

$$C^j(s) = \frac{1}{K}\sum_{i=1}^{K}[D_{KL}\big(p(s'_j \mid s, a^{(i)}||\frac{1}{K}\sum_{k=1}^{K} p(s'_j \mid s, a^{(k)})\big)] \tag{3}$$

with $a \sim \pi$, where $\pi(A) := \mathcal{U}(\mathcal{A})$ (i.e. a uniform distribution over the action space) and with K = 64 actions. We refer the reader to [22] for more details. Given the scores we need a threshold θ to get a classification of control (see Equation 3.3). In Fig. 6 we show the computed CAI scores over a demonstration in the Franka-kitchen. Given the scores we need a threshold θ to get a classification of control (see Equation 3.3). For Fetch-Push we set $\theta = 0.05$ and for Franka-Kitchen $\theta = 0.3$.

Transformer scores To compute causal influence, we use the attention weights of a Transformer, where the score is computed as follows. Letting A_i denote the attention matrix of the i'th of N layers, the total attention matrix is computed as $\prod_{i=1}^{N} A_i$. For `CoDA` the score is computed by checking the corresponding row i and column j for the check $s_i \rightarrow s'_j$, whereas for `CoDA-action` we restrict ourselves to the row corresponding to the input position of the action component, and the output position of the object component. Our implementation follows Pitis et al. [19], to which we refer for more details. Given the scores we need a threshold θ to get a classification of control. For Fetch-Push we set $\theta = 0.2$ for `CODA` and `CODA-action` and for Franka kitchen we set $\theta = 0.3$ for all methods.