Causal Decision Transformers

Parand Alizadeh Alamdari*
Department of Computer Science
University of Toronto
parand@cs.toronto.edu

Kateryna Nekhomiazh*
Department of Computer Science
University of Toronto
kateryna.nekhomiazh@mail.utoronto.ca

Bohdan Naida*
Department of Computer Science
University of Toronto
bohdan.naida@mail.utoronto.ca

Abstract

There is an emerging body of work in using transformers Vaswani et al. [2017] in machine learning tasks resulting in modeling high-dimensional distributions. Recent work has demonstrated that offline reinforcement learning (RL) can be formulated as a sequence modeling problem Janner et al. [2021] and can benefit from these approaches. The offline RL problem can be converted to a supervised learning problem Levine et al. [2020], and by using decision transformers (DTs) Chen et al. [2021], we can output optimal actions by leveraging a causally masked Transformer. However, offline RL solutions, including transformers, need a large amount of data, and by leveraging guidance specific to the problem context, the training process can be accelerated. In this project, we investigate the effect of adding causal information about the dynamics of the environment as guidance to the decision transformers. We evaluate our approach "Causal Decision Transformer" in a custom environment inspired by Minecraft, and compare it with the original decision transformer. The results show that adding the causal structure significantly impacts the policy that the agent learns, causing it to achieve a higher return with less amount of training.

1 Introduction

Reinforcement learning (RL) is a robust mathematical framework for learning-based control. RL agents are trained to optimize a specified reward signal (based on an objective) through actively interacting with the environment. The agent learns how to act optimally in the environment in order to achieve the highest cumulative reward by trial and error, which means taking actions and observing the changes in the state of the environment and the reward signal. Deep RL approaches such as Mnih et al. [2015] and Schulman et al. [2017] have become very popular due to their ability to approximate high-dimensional functions and allow the agents to learn the complex structure of the environments. However, an essential factor in the success of Deep RL approaches is the availability of collecting many online interactions (dataset) with the environment. This is not the case in most real-world problems, such as robotics or healthcare. In many settings, online data collection can be impractical, expensive (e.g. robotics), and dangerous (e.g. healthcare and autonomous driving). To address these concerns, a large body of work in offline reinforcement learning algorithms is proposed to utilize

*Equal contribution. Parand created the grid-world environment, added the causal embedding to the decision transformer architecture, ran the experiments, and created the plots. Kateryna worked on the initial idea of applying the method to the Alchemy environment and created the offline dataset of the Alchemy, and ran the experiments on it. Bohdan cooperated with the creation of the offline dataset for Alchemy, worked on adding causal relationships as a different form of embeddings to the decision transformer architecture, and ran experiments. Everyone participated in preparing and giving the presentation and writing the report.

previously collected data with no further online interactions, and data collection Janner et al. [2021], Levine et al. [2020]. Although the RL agent is trained using the static previously-collected dataset, the goal is still achieving the highest cumulative reward in the environment.

On the other hand, the Transformer architecture has become a standard for natural language processing tasks Radford et al. [2018], and it is shown that they can achieve state-of-the-art performance in computer vision tasks as well Dosovitskiy et al. [2020]. Recent work has demonstrated that offline RL can also be formulated as a sequence modeling problem and can benefit from transformer-based approaches such as Decision Transformers (DTs) Chen et al. [2021]. Despite the simplicity of the architecture of DTs, their performance matches or exceeds the performance of state-of-the-art offline RL approaches.

A significant disadvantage of offline RL approaches is that they are highly time and resource-consuming. This is due to the amount of data needed for training the agents. As the environment gets more complex, the amount of data required for the agents to learn the dynamics of the environment also increases. Furthermore, the rewards are usually sparse, and without online interactions, a large amount of data is needed to capture the complexity of the reward function and environment.

This project addresses this issue by adding causal guidance to offline RL approaches. Currently, existing RL exploration methods typically do not focus on causal exploration: they do not form and test causal hypotheses or plan active interventions to obtain causal data. Instead, existing methods primarily focus on expanding the set of experiences of the agent, for instance, by visiting novel or surprising areas of the state space Pathak et al. [2017]. However, we hypothesize that if the agent is given a high-level knowledge of the environment, such as the underlying causal relations of the variables of the environment, the agent can learn the dynamics of the environment and the optimal policy faster and with less amount of data. Our intuition is that we can save computational time by giving the agent "hints" about how to do the tasks or essential things to pay attention to.

We formalize the offline RL setting in the context of a causal problem, a well-known formalization in RL as in Bareinboim et al. [2015]. Among all the offline RL techniques, we specifically focus on decision transformers Chen et al. [2021]. A decision transformer is a modification of an autoregressive language model (that predicts the next agent's actions instead of words). Thus, we attempt to augment the decision transformer's causal self-attention mask with a causal structure that can be either hand-crafted, driven from natural language, or discovered from the same offline dataset prior to the training of the RL agent. Finally, we evaluate our method in a grid-world environment where the agent needs to collect different objects, and a randomly generated causal graph determines the order of collecting the objects. The results show that adding the causal structure to the decision transformers significantly impacts the number of iterations needed for training, and the agent learns a policy that is approximately optimal and achieves a higher return.

2 Related work

GPT Most NLP models before GPT required a large amount of labeled data, which complicates its usage in domains with a lack of annotated data. Unsupervised representation learning can significantly improve the model's performance in cases with available model supervision. GPT was the first NLP transformer model, trained in a semi-supervised manner. The authors proposed a two-stage training procedure. The first stage is an unsupervised pre-training on a large corpus of text. The second stage is a supervised fine-tuning that adapts the model to the downstream task. The GPT is an auto-regressive decode-only model trained using the left-to-right language objective. It is unidirectional and utilizes information only from the left side. The model has shown robust transfer performance across a variety of tasks and created a family of various GPT models such as GPT-2 Radford et al. [2019], GPT-3 Brown et al. [2020], GPT-Neo Black et al. [2021], GPT-J Wang [2021], and others.

Decision Transformer Chen et al. [2021] is a framework that allows bringing up RL tasks as a sequence modeling problem. This approach allows using language models, such as GPT, in the RL domain. DT generates actions based on future desired returns and represents trajectories as a sequence of states, actions, and rewards. Given the data of the last K timesteps, the DT model predicts the future action token via autoregressive modeling.

Causal Structure Identifiability The existence of a reliable causal structure discovery method would significantly improve the performance of models, optimize learning and decrease training time. Thus

this topic is being actively researched by AI community Zhu et al. [2019] Sun et al. [2007], Yu et al. [2019]. However, for most problems, the task of discovering a trustworthy causal structure remains unsolved.

Causal Reinforcement Learning. The concept of combining RL and a causal structure has several benefits. It improves the explainability and helps to understand the agent's behavior Madumal et al. [2020]. For meta-reinforcement learning tasks, it enhances the agent's ability to adapt to new settings Dasgupta et al. [2019], as it accelerates the exploration of a structure of the environment. Moreover, learning can be efficiently guided by knowing when and what the agent can influence with its actions. Such situation-dependent causal influence detection can help improve reinforcement learning agents Seitzer et al. [2021].

Guidance in Reinforcement Learning. The topic of giving guidance to reinforcement learning agents has gained particular popularity among the machine learning communityDriessens and Džeroski [2004]. Guidance types vary from human guidance Zhang et al. [2019], which includes evaluative feedback given by a human trainer, as policy shaping Griffith et al. [2013], reward shaping Isbell et al. [2001], intervention Saunders et al. [2018], and policy-dependent feedback MacGlashan et al. [2017]. The guidance allows the agent to explore the action space in a more optimal way Kosoy et al. [2022], for example, allowing to avoid researching the field of actions that will not affect the agent's achievement goal. Other types of "directions" for an agent include logic guidance Hasanbeig et al. [2019] and adaptive guidance Gaudet and Linares [2019] Gaudet et al. [2020].

Alchemy Wang et al. [2021] is a meta-reinforcement learning benchmark by DeepMind that presents tasks sampled from a task distribution with a deep underlying structure. The environment consists of 'trials', which fit into 'episodes'. Within each trial, the goal is to use a set of potions to transform each in a collection of stones into more valuable forms, collecting points when the stones are dropped into a central cauldron. The value of each stone depends on its features, but this relationship changes from episode to episode, as well as the potions' transformative effects. This 'chemistry' is fixed across trials within an episode but resampled with a new episode. The challenge within each episode is thus to transform the stones and get the maximum points.

3 Preliminaries

3.1 Markov Decision Process (MDP) with Causal Structure

We define an *MDP with Causal Structure*, as a tuple $M = \langle S, A, T, R, \gamma, C, \rangle$, where the first five terms are the standard items in an MDP, i.e. S is a finite set of states, A is a finite set of actions, $T(s_{t+1}|s_t, a_t)$ gives the probability of transitioning to state s_{t+1} when taking action a_t in state s_t, $r : S \times A \times S \to \mathbb{R}$ is the reward function, and γ is the discount factor. We can add the causal structure (e.g. a causal graph) to the MDP where $\mathcal{C}$ is a high-level abstraction of causal relations of the variables in the environment. In this project, we define C as a set of nodes (causal variables) in the causal graph. In each step, the nodes in c_t are aligned with the current state of the MDP s_t and contain high-level information about the dynamics of the environment.

3.2 Decision Transformers

We can define a trajectory as a sequence of states, actions, and rewards:

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ..., s_T, a_T, r_T)$$

Where T is the episode length. The return-to-go of a trajectory at timestep t is defined as $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$, the sum of future rewards from that timestep. The goal is to learn a policy that maximizes the expected return $\mathbb{E}[\sum_{t=1}^{T} r_t]$. The trajectories that are fed to the decision transformer as the input are as follows:

$$\tau = (\hat{R}_0, s_0, a_0, \hat{R}_1, s_1, a_1, ..., \hat{R}_T, s_T, a_T)$$

The last K timesteps are passed to the Decision Transformer for a total of 3K tokens (one for each modality). To obtain an embedded token, a linear layer is learned for each modality that maps the raw input to the size of the embedded, followed by layer normalization. Then, the embedding is learned for each time step and added to each symbol (one time step corresponds to three tokens). The tokens are then processed by a GPT model, which predicts future action tokens via autoregressive modeling.

In offline reinforcement learning approaches such as decision transformers, we only have access to an offline dataset consisting of trajectory rollouts of arbitrary policies, which is gathered previously. The abilities of exploration and online interactions with the environment are removed, and the problem setting is generally harder to solve.

4 Method

4.1 Adding causal information to trajectories

We investigate our hypothesis and evaluate the effects of adding the causal structure as guidance to decision transformers. We feed the causal structure to the decision transformers as part of the input. Consider the trajectories fed to the model as described in section 3. We add the causal information to the trajectories and change them to the following:

$$\tau = (\hat{R}_0, s_0, c_0, a_0, \hat{R}_1, s_1, c_1, a_1, ..., \hat{R}_T, c_T, s_T, a_T)$$

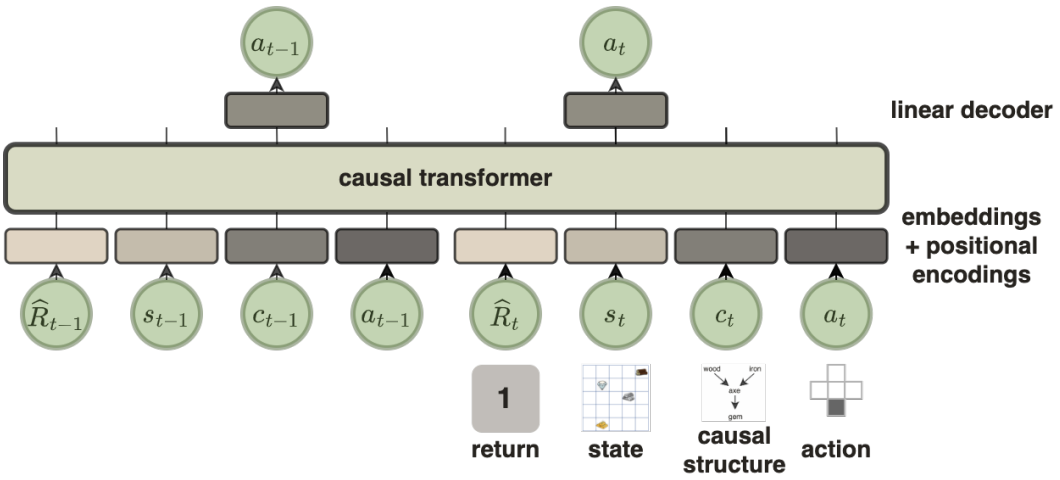


Figure 1: Causal Decision Transformer's architecture

For implementation purposes, we concatenate the causal information to the state representation (instead of adding it separately) and change the state representation as $s'_t = \langle s_t, c_t \rangle$, so our trajectory is as follows:

$$\tau = (\hat{R}_0, s'_0, a_0, \hat{R}_1, s'_1, a_1, ..., \hat{R}_T, s'_T, a_T)$$

The architecture of the causal decision transformer is shown in Figure 1. In test time, the initial return-to-go value $\hat{R}_0$ is the desired return value that we expect the agent achieves.

4.2 Extracting causal information

Suppose we are given a causal directed acyclic graph (DAG) $\mathcal{G}$ that describes the dynamics of the environment. Some samples of this causal graph are shown in Figure 3. In this project, we assume that the variables in the causal graph can only take the values of 0 and 1. Each variable X in the graph, in addition to the known variables of $\mathcal{G}$, has an unknown confounder U_X, which is not shown in the causal DAGs. Furthermore, we assume that these conditions exist among the variables:

$$\begin{cases} P(X=1) = 0 & \text{if } \{\exists Y = 0 | Y \text{is a parent of } X\} \\ P(X=1) = P(U_X = 1) & \text{if } \{\forall Y \text{parent of } X | Y = 1\} \end{cases} \tag{1}$$

So, intuitively in each state of MDP s_t, an observed value of a variable X is 1 if all the observed values of parents of X are 1. We assume all variables are 0 at the beginning (initial state of the MDP s_0) unless we get new observations of the variables in later timesteps.

In each timestep t, c_t in the MDP is a set of nodes from $\mathcal{G}$ where $\forall X \in c_t$, we observed the value 1 for all of the parents of X, but the value of X has not been observed yet. If we observe the value 1 for X in timestep t, then $X \notin c_{t+1}$. For example, in DAG D of Figure 3, c_t can be equal to these sets: $\{$wood, iron$\}$, $\{$wood, axe$\}$, $\{$gold, iron$\}$, $\{$gem$\}$, $\{\}$. In $\{$wood, axe$\}$ case, the agent collected the iron previously, and now wood and axe can be collected if the agent moves to their locations (the location of the agent acts as the unknown confounder).

5 Experiments

5.1 Selection of an RL environment with causal abstraction

One of the significant challenges that we encountered in this project was finding a proper RL environment to implement and evaluate our idea. Confounding factors should highly impact the dynamics of the environment. In addition, these factors and the abstraction of the causal graph should be available to use or at least feasible to extract. In contrast to other domains such as medical problems, where experts can determine the confounding factors and the causal graph from the dataset, it is extremely difficult to extract the causal graph from well-known tasks and environments that decision transformers are initially evaluated on, such as MuJoCo environments Todorov et al. [2012]. Thus, for this course project, we decided to evaluate our idea of an environment with a pre-defined causal abstraction of dynamics.

Our first choice was the Alchemy environment Wang et al. [2021] due to its interesting and non-trivial task with causal confounders. The goal of alchemy is transforming the given stones using the available potions in the environment to reach their highest value. The potions and the chemistry rules of the environment (e.g. the effect of different potions on different stones) are randomly generated in each episode. Initially, we collected an offline dataset of Alchemy with 1000 episodes and tried to train our model on it. However, the results were unsatisfactory. Later, we realized that because of the complexity of the task and the number of possible scenarios, the authors of this benchmark initially used a billion episodes for the preliminary training. As we could not gather a significantly large dataset and train for millions of episodes, we decided to create a simple custom gridworld environment inspired by Minecraft™ to evaluate our idea.

5.2 Environment

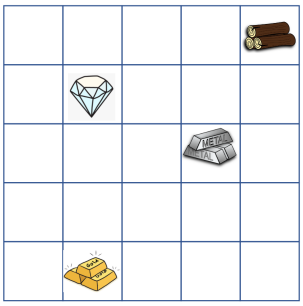
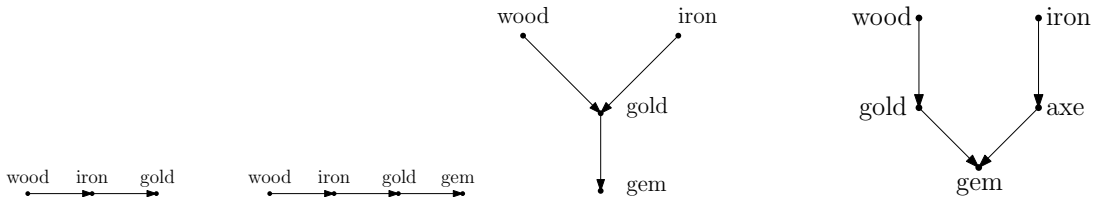


Figure 2: Craft-World Environment

The environment is depicted in Figure 2. The RL agent enters the environment through the upper-left corner and aims to collect all of the objects. The agent can perform five different actions: moving right, left, up, down and collecting the objects if the agent is in the same location of the object and the object can be collected. The objects can not be collected unless the agent has previously collected the necessary tools. For example, if an agent wants to collect an axe, iron and wood should have been collected by the agent before. Similarly, the axe should be collected before if the agent aims to collect gold. The dependencies between the objects can be described as a causal graph. The agent gets a +1 reward by collecting an object; otherwise, it gets 0 rewards. So, the maximum cumulative reward in the environment equals the number of objects.

In the following subsection, we provide a causal graph for each task in Figure3 which describes the dependencies between objects (the objects that need to be collected before each object). In each task,



(a) DAG A (b) DAG B (c) DAG C (d) DAG D

Figure 3: Causal DAG of different tasks in experiments

the structure of the dependency graph and the location of objects in the map are fixed, but the nodes in the graph are randomly generated. For example, a possible scenario is the agent needs to collect wood before collecting gold (wood -> gold). Another likely scenario could be that the agent needs to collect gold before wood (gold -> wood). For example in Scenario A of Figure 3, the randomly generated causal graph is one of the following six cases: (wood -> iron -> gold), (wood -> gold -> iron), (gold -> iron -> wood), (gold -> wood -> iron), (iron -> wood-> gold), (iron -> gold -> wood). So, by not knowing the dependencies, the agent must check the locations of all objects until it can collect one.

One of the challenging parts of the experiments was training the decision transformers on our discrete grid-world environment. Unlike simple classic RL methods such as DQN Van Hasselt et al. [2016] that are trained easily on the grid-world environments with the dense representation of the domain (such as passing the locations of agent and the objects), decision transformers behave differently. We found out that decision transformers work best with the one-hot encoding of the whole environment map and do not train well on an input with short and dense representation.

5.3 Dataset

We evaluate our method in four different scenarios (A, B, C, D). The causal graphs corresponding to each scenario are shown in Figure 3. Scenarios B, C, and D are happening in the environment shown in Figure 2. In Scenario D, there is an extra object of the axe in the bottom-right corner. Scenario A is executed in a smaller environment (3×3) with only three objects wood, iron, and gold. For each scenario, a dataset consists of 12000 trajectories generated using a random agent.

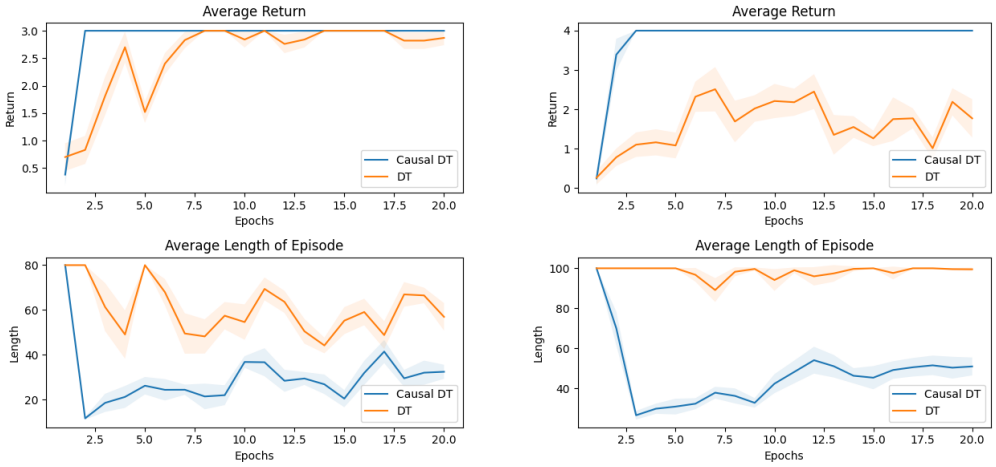
5.4 Baselines

We compare our method "Causal Decision Transformer" (described in section 4) with the original decision transformer method with no modifications. Both of them are trained on each dataset for 20 epochs (approximately 2.5 GPU-hours each). Each model is evaluated on 100 episodes after each epoch. The code for training both models is available at https://github.com/praal/decision-transformer.

5.5 Results

The results are shown in Figure 4 and Figure 5. For each scenario, the "Average Return" figure compares the cumulative reward that the agent gets in an episode with each model, and "Average Length of Episode" is the average amount of time (number of timesteps) that it takes the agent to achieve the maximum cumulative reward (number of objects in the causal DAG). As it is shown in different scenarios, adding the causal embedding to the model significantly impacts the performance of the agent (achieves higher return). In addition, the agent trained with causal decision transformer, actually uses the available causal information and as a result collects all the objects in significantly less amount of time compared to the agent trained with original decision transformer (which can not collect all the objects in some scenarios).

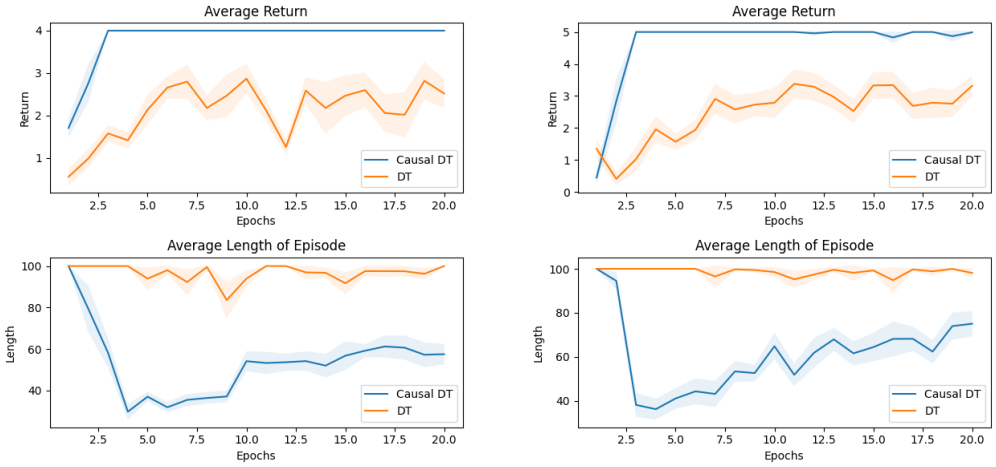
Using the random dataset, and the causal information, our agent learns to perform an almost optimal policy (with some noises) to collect the right object in each turn (and go to next object after). In contrast, without any additional information about the causal DAG, the DT agent has to randomly check all the possible locations of objects to finally collect one (and repeat the process for the next object).



(a) Scenario A

(b) Scenario B

Figure 4: Comparison of our approach "Causal DT" with DT in scenarios A and B.



(a) Scenario C

(b) Scenario D

Figure 5: Comparison of our approach "Causal DT" with DT in scenarios C and D.

Furthermore, the causal DT agent learns a policy that collects all the objects in the first three epochs (low amount of training time) in all scenarios. However, the DT agent can not learn a policy that collects all the objects in every scenario; even in the smaller environment (Scenario A), it takes almost 10 epochs to learn that policy.

6 Limitations and Future Work

In this project, we proposed a framework that by adding information about the causal relations of the environment dynamics, we can train an agent with significantly better policy, and in less amount of time. However, to test the performance of the proposed architecture we used a relatively simple environment. The grid-world environment we implemented is fairly static (only the causal DAG is randomly changing), and the tasks are straightforward. In addition, we limited the possible causal graphs by the assumptions in section 4.2. The next step in evaluating our approach is to test it in different and more complex environments (such as Alchemy Wang et al. [2021]) and relax the assumptions about the causal DAG. However, we hypothesize that the results will be almost the same

in more complex environments, and we can significantly reduce computational expenses, and reach the same or better results in a smaller number of iterations.

A serious challenge that we might face in using our method, especially in a high-dimensional environment with complex logic and structure is extracting the causal graph and dependencies between the variables. Not all RL environments benefit from a causal abstraction in their dynamics. Another challenging task is to identify such problems and environments. In those cases, a solution could be hand-specifying the causal graph (as we did in this project) using the knowledge of domain experts. Another interesting solution that we believe is worth investigating in the future is that we can use the offline dataset that is gathered for training the RL agent and use causal discovery methods to learn the causal graph. This can broaden the area of problems where our approach can be applied, as now it is limited to the environments with predefined "logic" or the ones where the experts can extract the causal information.

Another concern that should be addressed in the future is the representation of the causal DAG in the model. As we limited the graphs in section 4.2, we were able to represent the graph by a subset of nodes that the parents were observed before. However, this might not work well in general causal graphs. We propose a direction toward addressing this issue as representing the causal DAG using Graph Neural Network approaches Zhou et al. [2020].

Furthermore, we can incorporate a causal structure in the architecture of decision transformers and augment the causal self-attention mask with an explicit causal structure. Especially, we can design it in a way that the causal DAG directly influences the return-to-go values, and shape them according to the causal information about the dynamics of the environment (e.g. increasing it by following the causal dynamic or decreasing it by doing the opposite).

7 Conclusion

To summarize, in this project, we augmented Decision Transformer with causal guidance, seeking to optimize the learned policy and accelerate the training process. On a custom grid-world environment, we compared the performance of our model with classical Decision Transformer architecture. We showed that an extra set of information in the form of a causal structure can significantly enhance the learned policy and efficiency of the training process. Our model outperformed the regular Decision Transformer on the offline RL tasks we defined in this environment.

We hope that our work is a small step toward exploring the causal effects of solving RL tasks.

A Appendix

The code for training "Causal Decision Transformer" and "Decision Transformer" is available at https://github.com/praal/decision-transformer.

References

Elias Bareinboim, Andrew Forney, and Judea Pearl. Bandits with unobserved confounders: A causal approach. *Advances in Neural Information Processing Systems*, 28:1342–1350, 2015.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL `https://doi.org/10.5281/zenodo.5297715`. If you use this software, please cite it using these metadata.

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.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Ishita Dasgupta, Jane X. Wang, Silvia Chiappa, Jovana Mitrovic, Pedro A. Ortega, David Raposo, Edward Hughes, Peter W. Battaglia, Matthew M. Botvinick, and Zeb Kurth-Nelson. Causal

reasoning from meta-reinforcement learning. *CoRR*, abs/1901.08162, 2019. URL http://arxiv.org/abs/1901.08162.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Kurt Driessens and Sašo Džeroski. Integrating guidance into relational reinforcement learning. *Machine Learning*, 57:271–304, 12 2004. doi: 10.1023/B:MACH.0000039779.47329.3a.

Brian Gaudet and Richard Linares. Adaptive guidance with reinforcement meta-learning. *CoRR*, abs/1901.04473, 2019. URL http://arxiv.org/abs/1901.04473.

Brian Gaudet, Richard Linares, and Roberto Furfaro. Adaptive guidance and integrated navigation with reinforcement meta-learning. *Acta Astronautica*, 169:180–190, 2020. ISSN 0094-5765. doi: https://doi.org/10.1016/j.actaastro.2020.01.007. URL https://www.sciencedirect.com/science/article/pii/S0094576520300072.

Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/e034fb6b66aacc1d48f445ddfb08da98-Paper.pdf.

Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Certified reinforcement learning with logic guidance. *CoRR*, abs/1902.00778, 2019. URL http://arxiv.org/abs/1902.00778.

Charles Lee Isbell, Jr., Christian R. Shelton, Michael Kearns, Satinder Singh, and Peter Stone. A social reinforcement learning agent. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 377–384, Montreal, Canada, 2001. ACM Press.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Eliza Kosoy, Adrian Liu, Jasmine L Collins, David Chan, Jessica B Hamrick, Nan Rosemary Ke, Sandy Huang, Bryanna Kaufmann, John Canny, and Alison Gopnik. Learning causal overhypotheses through exploration in children and computational models. In *First Conference on Causal Learning and Reasoning*, 2022. URL https://openreview.net/forum?id=6GLEuG0d8i.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, Guan Wang, David L. Roberts, Matthew E. Taylor, and Michael L. Littman. Interactive learning from policy-dependent human feedback. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2285–2294. JMLR.org, 2017.

Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. Explainable reinforcement learning through a causal lens. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03): 2493–2500, Apr. 2020. doi: 10.1609/aaai.v34i03.5631. URL https://ojs.aaai.org/index.php/AAAI/article/view/5631.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

William Saunders, Girish Sastry, Andreas Stuhlmüller, and Owain Evans. Trial without error: Towards safe reinforcement learning via human intervention. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 2067–2069, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Maximilian Seitzer, Bernhard Schölkopf, and Georg Martius. Causal influence detection for improving efficiency in reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 22905–22918. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/c1722a7941d61aad6e651a35b65a9c3e-Paper.pdf.

Xiaohai Sun, Dominik Janzing, Bernhard Schölkopf, and Kenji Fukumizu. A kernel-based causal learning algorithm. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 855–862, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273604. URL https://doi.org/10.1145/1273496.1273604.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

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.

Ben Wang. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Jane Wang, Michael King, Nicolas Porcel, Zeb Kurth-Nelson, Tina Zhu, Charlie Deck, Peter Choy, Mary Cassin, Malcolm Reynolds, Francis Song, Gavin Buttimore, David Reichert, Neil Rabinowitz, Loic Matthey, Demis Hassabis, Alex Lerchner, and Matthew Botvinick. Alchemy: A structured task distribution for meta-reinforcement learning. *arXiv preprint arXiv:2102.02926*, 2021. URL https://arxiv.org/abs/2102.02926.

Yue Yu, Jie Chen, Tian Gao, and Mo Yu. DAG-GNN: DAG structure learning with graph neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7154–7163. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/yu19a.html.

Ruohan Zhang, Faraz Torabi, Lin Guan, Dana H. Ballard, and Peter Stone. Leveraging human guidance for deep reinforcement learning tasks. *CoRR*, abs/1909.09906, 2019. URL http://arxiv.org/abs/1909.09906.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

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