Did I do that? Blame as a means to identify controlled effects in reinforcement learning

Oriol Corcoll *
Institute of Computer Science
University of Tartu
oriol.corcoll.andreu@ut.ee

Youssef Mohamed
Institute of Computer Science
University of Tartu
youssef.mohamed@ut.ee

Raul Vicente
Institute of Computer Science
University of Tartu
raul.vicente.zafra@ut.ee

Abstract

Identifying controllable aspects of the environment has proven to be an extraordinary intrinsic motivator to reinforcement learning agents. Despite repeatedly achieving State-of-the-Art results, this approach has only been studied as a proxy to a reward-based task and has not yet been evaluated on its own. We show that solutions relying on action-prediction fail to model critical controlled events. Humans, on the other hand, assign blame to their actions to decide what they controlled. This work proposes Controlled Effect Network (CEN), an unsupervised method based on counterfactual measures of blame to identify effects on the environment controlled by the agent. CEN is evaluated in a wide range of environments showing that it can accurately identify controlled effects. Moreover, we demonstrate CEN's capabilities as intrinsic motivator by integrating it in the state-of-the-art exploration method, achieving substantially better performance than action-prediction models.

1 Introduction

The recent success of reinforcement learning (RL) methods in difficult environments such as Hide & Seek (Baker et al., 2019), StarCraft II (Vinyals et al., 2019), or Dota2 (OpenAI et al., 2019) has shown the potential of RL to learn complex behavior. Unfortunately, these methods also show RL's inefficiency to learn (Espeholt et al., 2018; Kapturowski et al., 2019; Gulcehre et al., 2020), requiring a vast amount of interactions with the environment before meaningful learning occurs. Consequently, environments with sparse rewards are known to be extremely difficult making imperative a good exploration strategy. A popular approach to exploration is to introduce behavioral biases in the form of intrinsic motivators (Chentanez et al., 2005; Mohamed and Rezende, 2015). This technique aims to facilitate the learning of task-agnostic behavior by producing dense rewards, driving the agent to discover novel states and by doing so increase the chance of discovering the environment's reward.

Numerous motivators have been developed taking inspiration from humans, e.g. curiosity or control (Bellemare et al., 2012b; Pathak et al., 2017; Burda et al., 2018; Choi et al., 2019; Badia et al., 2020b). Recent work (Choi et al., 2019; Song et al., 2019; Badia et al., 2020a,b) has achieved State-of-the-Art on the Atari benchmark (Bellemare et al., 2012a) by rewarding agents for the discovery of novel ways to control their environment. A common design principle among these methods is the use of an inverse model to predict the chosen action from two consecutive observations. The hope is that the latent representation learned by these models encloses aspects of the environment controlled by the agent. We hypothesize that these methods are ill-suited to model anything other than the agent, limiting their applicability. Fig. 1 (left) shows an scenario where an agent moves a box by moving left; an inverse model would only need to represent the agent to predict the action but not the box.

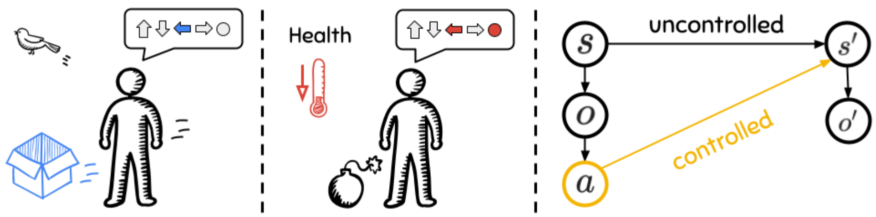


Figure 1: Left) Action-prediction models (e.g. inverse models) are encouraged to model the movement of the agent but not more complex effects like the box. Instead, blame compares an imagined normative world against reality to attribute the movement of the agent and the box to the action. Middle) Using a do-nothing action as normative world is not enough since do-nothing has an effect. Right) Causal graph of a typical RL setting; controlled effects depend on the agent's action.

A more causal approach is to compare counterfactual worlds (Pearl, 2009), i.e., an effect is controllable if the effect would have been different had the agent taken another action. A caveat of this approach is that things become trivially controllable. Following the previous example, a box becomes controllable even when the agent performs the action "do-nothing" since there is an action, "move-left", that would move the box. Contrarily, it is believed that humans identify controlled effects by assigning a degree of blame to their actions. In particular, humans compare what happened to a normative world (Halpern, 2016; Morris et al., 2018; Langenhoff et al., 2019; Grinfeld et al., 2020). If what happened is normal, humans would not assign blame to their actions, e.g. when performing "do-nothing", the box's effect would not be controlled since normally the box would not move. However, it would be considered controlled when performing "move-left" since its normative state is to not move.

This work proposes Controlled Effects Network (CEN), a completely unsupervised approach to identify controlled aspects of the environment based on the human notion of blame. In contrast to models based on action prediction, CEN is composed of a two-branch forward model that creates a normal and controlled view of the world. Our experiments show that CEN can disentangle effects precisely, outperforming state-of-the-art approaches to detect controlled effects. Additionally, we evaluate CEN as an intrinsic motivator by replacing the action-prediction model in Never Give Up's (Badia et al., 2020b) episodic reward with CEN, leading to a substantial gain in performance.

2 Identifying controlled effects using blame

Our goal is to identify changes in the environment that were controlled by the agent. This section introduces Individual Causal Effect (ICE), a fundamental measure in causal literature, and frames it in the context of RL. We show how this measure can be used to identify controllable effects but argue that these are not suitable for RL. In contrast, the human perception of causality is associated with the concept of blame (Gerstenberg and Lagnado, 2014). For example, if lightning hits a forest tree and starts a fire, humans would point to the lightning as the cause of the fire, not the oxygen or wood since they are normally present in the forest. Consequently, we expand the idea of blame to identify controlled effects by using measures of normality and counterfactuals.

2.1 Controllable effects

What does it mean to cause something? Pearl et al. (2016) provide an intuitive definition of cause-effect relations: "A variable X is a cause of a variable Y if Y, in any way, relies on X for its value". This kind of formulation tries to answer questions like "does smoking causes cancer?". Actual causality, proposed in Halpern (2016), studies causal relations between individual events of X and Y. It aims to answer questions like, "did smoking for 30 years caused David's cancer?". In the following, we introduce the concept of causal effect to then define controllable effect in the RL context.

The individual causal effect (ICE) of an event $X = x$ on a variable Y_i can be measured by comparing counterfactual worlds

$$ICE^x_{Y_i} \equiv Y^x_i \neq Y^{\tilde{x}}_i \ , \tag{1}$$

where Y^x_i reads as "what would the value of an individual Y_i be if X is forced to be x". Similarly, $Y^{\tilde{x}}_i$ describes the value of Y_i when X is forced to not be x. Note that the sub-index i refers to an individual, and hence in the following, we use Y^x_i and Y^x interchangeably.

The *fundamental problem of causal inference* states that we can only observe one of these counterfactual worlds and the other needs to be imagined. Intuitively, Eq. 1 compares the world where the event x happened to an alternative world where event x had not happened. Consequently, we say that x has a causal effect on Y if there is an $\tilde{x} \in X$

that satisfies Eq. 1. In the context of RL, X and Y take the form of actions, states and observations. Fig. 1 (right) illustrates the causal relations present in a typical RL setting, where a state s has an effect on both the next state s' and the produced observation o which, in turn, has an effect on the agent's choice of action $a \in \mathcal{A}$. Similarly, an action has an effect on the next state. Since states are typically not accessible by the agent, we do not use states as variables; nevertheless the same principle can be applied if these are accessible. We define the perceived effect e_p^a as the difference between consecutive observations when taking action a, i.e. $e_p^a \equiv o' - o$. As in Eq. 1, we say that a perceived effect was controllable by the agent's action when

$$\exists \tilde{a} \in \mathcal{A} \colon e_p^a \neq e_p^{\tilde{a}} \, . \tag{2}$$

Since we want to know what elements of the perceived effect are controllable, the inequality is an element-wise operation. It is important to notice that Eq. 2 has far-reaching consequences, for example, an agent next to a box would have a causal effect on it even when not moved since there is a counterfactual world where that box would have moved. Using Eq. 2 as reward, the agent would be rewarded for almost every action at every state! Note that taking $\tilde{a}$ as a special "do-nothing" action would not work since even doing nothing does something, e.g., Fig. 1 (middle) shows a scenario where doing nothing has an effect on the agent's health. Taking do-nothing as $\tilde{a}$ would not attribute the effect to the agent. Instead, we would want a more human-like definition of what is controlled where an agent controls a box if moved or its life if a bomb could have been easily avoided.

2.2 Blame

It has been shown that human notion of causality is affected by what is normal (Kahneman and Miller, 1986; Cushman et al., 2008; Knobe and Fraser, 2008; Hitchcock and Knobe, 2009). Here, we resort to concepts of normality from actual causality to find if the agent's action is to blame for what happened. Halpern and Hitchcock (2014) propose to compare what actually happened with what normally would happen. Following this idea we use a normative world in replacement to $Y^{\tilde{x}}$

$$ICE_Y^x = Y^x - \beta_Y \, , \tag{3}$$

where β_Y is the value Y would normally take. Such a value is of course contingent to the notion of normality used, which is to us to define. Note that since we are interested in the magnitude and direction of the effect, Eq. 3 uses the difference rather than the less specific inequality used in Eq. 1.

Advantage function as Blame: A typical use of this formulation is to compute the causal effect of an action on the return G as

$$\begin{aligned} ICE_G^a(s) &= G^a(s) - \beta_G(s) \\ &= Q(s,a) - V(s) \\ &= A(s,a). \end{aligned} \tag{4}$$

$G^a(s)$ is the return the agent would get if action a were to be taken at state s and is typically estimated using a state-action value function $Q(s,a)$. The choice of normality for $\beta_G(s)$ is to estimate the expected return with the state-value function $V(s)$, giving us the advantage function $A(s,a)$.

As described in Sutton and Barto (2018), Generalized Value Functions aim to integrate general knowledge of the world; leaving return as special case. Following the same idea, we can reformulate Eq. 3 to compute the controlled effect of an action as

$$\begin{aligned} ICE_{e_p}^a &= e_p^a - \beta_{e_p} \\ &= e_p^a - \mathop{\mathbb{E}}_{\forall \tilde{a} \in A} \left[e_p^{\tilde{a}} \right]. \end{aligned} \tag{5}$$

To simplify notation, the following sections use controlled effect as $e_c^a = ICE_{e_p}^a$ and normal effect $e_n = \beta_{e_p}$. Intuitively, Eq. 5 builds a normal world by observing every alternative e_p produced by each action creating an average perceived effect. Consider the example in Fig. 1 (middle), moving left or doing nothing would make the agent's health decrease. Eq. 5 would indicate that no changes to the health bar are normal; thus, the loss of health when moving left or staying would be attributed to the agent. On the other hand, moving right would only attribute the change in the agent's position as controlled. Note that the explosion would never be credited to the agent.

Special care needs to be taken when constructing the normal world β_{e_p} for continuous action spaces. Computing counterfactuals on an infinite number of possibilities cannot be done and some approximation needs to be implemented. Although our experiments use discrete actions, the proposed method in the following section is equipped to handle continuous action spaces since it does not compute counterfactuals for each possible action but approximates the normal world directly. It is also important to notice that the controlled effects Eq. 5 can identify in a partially observable setting ($o \neq s$), are constrained to those observed by the agent. Nevertheless, humans cannot perceive every change in state but can identify relevant controlled effects for their survival and joy.

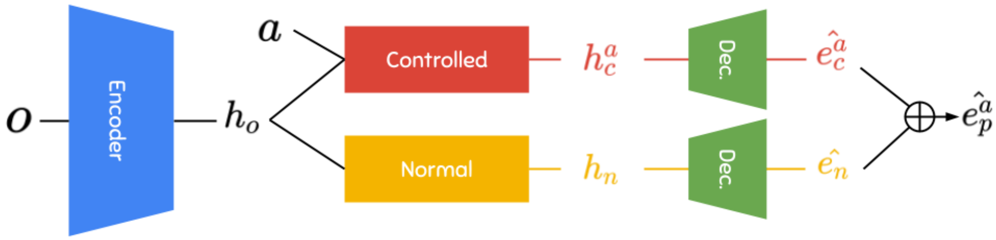


Figure 2: CEN divides the latent space of a forward model into controlled and normal branches. Each branch disentangles controlled and normal effects and decodes each into pixel space independently. Note that decoder weights are shared.

3 Unsupervised learning of controlled effects

In practice, we do not have access to every world and cannot compute Eq. 5 directly. We propose an unsupervised method that disentangles controlled and normal effects only using perceived effects as a self-supervised training signal.

Here, we introduce **Controlled Effects Network (CEN)**, depicted in Fig. 2. CEN is based on a forward model, where observation and action are used to predict the outcome of performing such action on the environment. In contrast to conventional forward models, CEN divides its latent space into controlled and normal representations; similarly to Dueling Networks (Wang et al., 2015). These two representations approximate the controlled and normal effects in latent space. A decoder converts these latent representations into pixel space allowing to estimate $e_c^a + e_n = e_p^a$ as in Eq. 5.

The **controlled branch** has privileged access to the action; having only this branch would make CEN a regular forward model, i.e., the controlled branch alone can predict the perceived effect resulting from the action. Then, why do we need the **normal branch**? The role of the normal branch is to force the controlled branch to predict only what is not predictable from the observation alone and hence, modeling what is controlled by the agent. In a way, the normal branch acts as a distillation mechanism where only what can be controlled will be represented by the controlled branch. To promote the controlled branch to model only controlled effects, we use the following loss

$$\mathcal{L} = \text{MSE}\left(\hat{e_c^a} + \hat{e_n}, e_p^a\right) + \alpha \, \text{MSE}\left(\hat{e_n}, e_p^a\right), \tag{6}$$

where the first part is the reconstruction loss in which the predicted target $\hat{e_p^a} = \hat{e_c^a} + \hat{e_n}$ is compared to the perceived effects provided by the environment. The second part of the loss enforces the network to use the normal branch as much as possible to model the world. Since this branch cannot predict everything without the action the model will converge to the expected effect due to the MSE loss. Additionally, a hyperparameter α regulates how much the normal branch should model the environment. In practice, we found that this hyperparameter creates an agreement between branches on uncertain futures which seemed to be critical in environments with stochastic entities.

Let us look again at Fig. 1 (middle) and assume the agent picks the do-nothing action. The normal branch is encouraged to model the bomb since it does not depend on the action. Furthermore, it should not predict any change in health since what is normal is for health to not change. Thus, the controlled branch must model the health changes.

4 Experiments

This section evaluates CEN[2] on three main questions: 1) Can CEN identify controlled effects at pixel-level? i.e. can it produce an accurate segmentation mask? 2) Some applications may not require pixel-level precision; we asses CEN on predicting controlled effects at attribute-level from both pixel masks and latent representations. 3) Can RL agents benefit from using CEN as intrinsic motivator? We use different baselines to compare CEN against. We use Attentive Dynamics Model (ADM) (Choi et al., 2019) in 1) and 2). For 2) and 3) we rely on Never Give Up (NGU) (Badia et al., 2020b), the current state-of-the-art exploration method.

Environments: we use multiple environments (Fig. 3) to answer the above questions, each showcasing a different aspect of what can be controlled. These environments are based on Griddly (Bamford et al., 2020) and Atari ALE (Bellemare et al., 2012a), both using the Gym interface (Brockman et al., 2016). More details of these environments are given in each experiment and appendix.

[2]Networks, training and evaluation have been implemented using PyTorch (Paszke et al., 2019), NumPy (Harris et al., 2020) and PFRL (Fujita et al., 2021); our experiments are managed using W&B (Biewald, 2020).

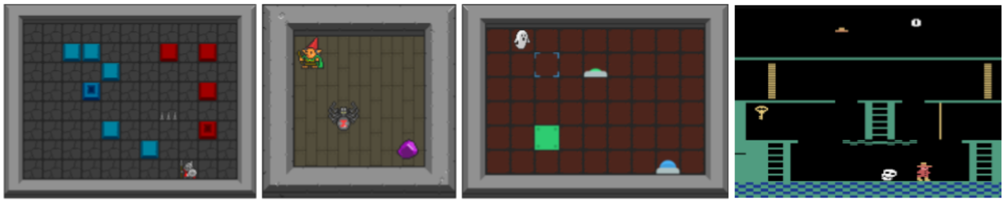


Figure 3: Suite of environments used for the experiments. From left/right top/bottom: Clusters, Spiders, Lights, and Montezuma's Revenge (MZR).

Implementation: CEN is implemented as an encoder-decoder architecture with 2D convolutional layers and ReLU activation functions; the normal and controlled branches are implemented with linear layers. Throughout the experiments we use the same neural networks and hyperparameters unless specified otherwise. Our implementation of ADM uses the same architecture and hyperparameters proposed in Choi et al. (2019). See appendix for more details on the architecture and hyperparameters.

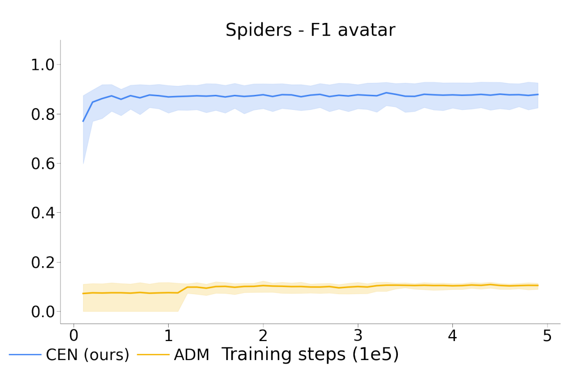
4.1 Controlled effects at pixel-level

This set of experiments explores CEN's ability to identify pixels corresponding to controlled entities. Although Eq. 5 computes the magnitude and direction of the effects, we create a binary mask by setting a threshold for the predicted controlled effects (see exact details in the appendix). We report pixel F1 scores between ground truth and predicted binary masks. The network is trained to minimize Eq. 6 using the ADAM optimizer (Kingma and Ba, 2015) and 300K samples of the form (o, a, e_p^a) collected using a random policy. The evaluation of the model is done using an environment with different seed to the one used for training for 5K steps. To generate the ground truth masks for Griddly environments, we use the internal environment state and transform object coordinates onto pixel coordinates. In experiments using the Atari environment the ground truth is collected by computing Eq. 5 using the ALE's special calls *cloneSystemState* and *restoreSystemState*. ADM produces an attention mask for pixels controlled by the agent therefore it is used as baseline in these experiments.

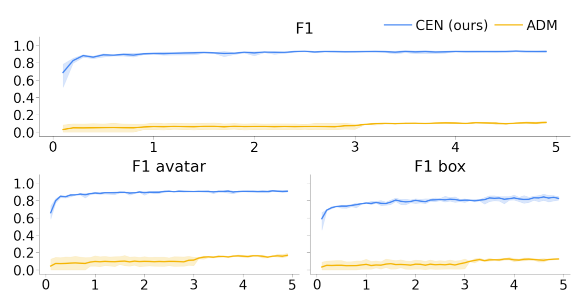
4.1.1 Controlled vs uncontrolled effects

Here we use the Spiders environment to evaluate CEN's ability to disentangle controlled from uncontrolled effects. This environment has two main entities, the agent and a spider. The controlled masks must only focus on the agent and ignore the spider.

Fig. 4a shows the pixel F1 score for our model and the baseline. CEN is able to correctly disentangle controlled effects and can produce accurate masks. Although our implementation of ADM can predict the agent's action with 88% accuracy, it is not capable of modeling the agent's controlled pixels. We conjecture that this is due to ADM's sparse softmax mechanism; nonetheless this behavior persisted when increasing its entropy weight which should produce more dense masks.

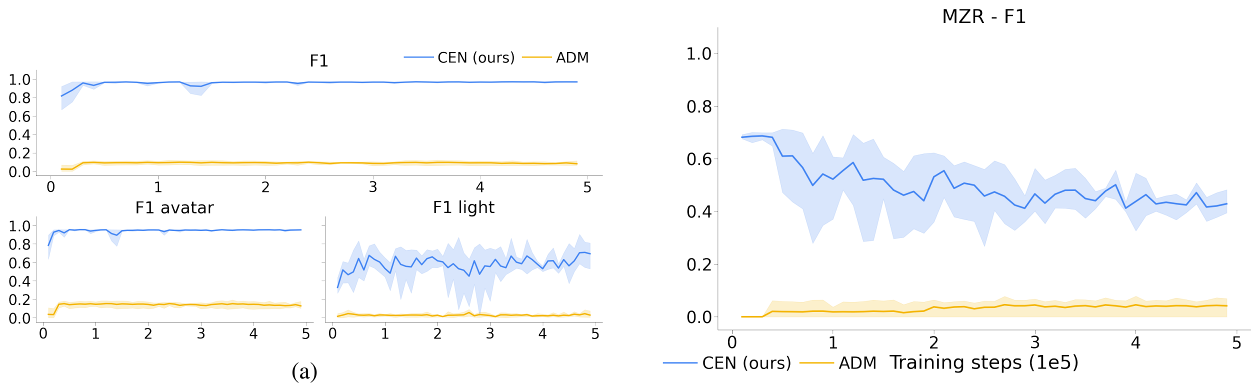


(a)



(b)

Figure 4: a) CEN can correctly disentangle the agent from the randomly moving objects. b) Clusters environment where CEN is able to model not just the agent but also the movement of boxes.



(a)

(b)

Figure 5: a) F1 score on the Lights environment. CEN can model faraway effects. b) Montezuma's Revenge F1 pixel score. CEN outperforms the baseline even in an environment with more complex dynamics and visual features.

4.1.2 Nearby controlled effects

Models based on action prediction are expected to work well on aspects related to the agent. For example, if an agent moves a box due to moving right; the box's movement is also controlled. It is unclear why these models would pay attention to the box since just knowing where the agent is, suffices to predict the chosen action. CEN's controlled branch, on the other hand, is motivated to model the box's effect since the normal branch would predict that the box stays where it is. We call "nearby" controlled effect to an effect that happens adjacent the agent, like the box's movement. To evaluate CEN on this kind of effects we use the Clusters environment where an agent needs to move colored boxes to their corresponding fixed colored blocks. Fig. 4b shows that CEN can precisely model effects on the agent and boxes. We breakdown individual effects to account for the class imbalance between the agent's movement and the boxes. CEN seems to make more mistakes with boxes than the agent but nonetheless, it can consistently model both.

4.1.3 Faraway controlled effects

In contrast to the previous experiment, we want to evaluate if CEN can model distant effects, i.e. effects that are reasonably far away from the agent's location. In this case, we use the Lights environment. Here the environment presents two buttons of different color that, when pressed, turn on their corresponding light. Lights are relatively far away from their corresponding buttons thus making it difficult to model them. As show in Fig. 5a, CEN is able to model this kind of controlled effects. We believe the decrease in F1 score is due to the complex shape of lights and buttons.

4.1.4 Controlled effects in Montezuma's Revenge

This last pixel-level experiment evaluates CEN on Montezuma's Revenge environment. Although agents in Atari environments have limited control over the environment, the relatively complex dynamics of Montezuma's Revenge makes it a challenging test-bed. Results shown in Fig. 8 indicate that CEN can also model controlled effects but that the increase in complexity affects its ability to predict accurate masks. Similarly to the Lights environment the shape of some elements is more complex which we conjecture is the cause of the model decrease in performance.

4.2 Controlled effects at attribute-level

In some cases requiring pixel-level precision can be excessive. The following experiments analyze how different representations can predict effects on attributes from the environment's state, e.g. changes on the agent's (x, y) location or if a light turned on. To this end, we use a probing technique similar to the one described in Anand et al. (2019). This approach trains a classifier per each attribute of interest in the environment's state using frozen versions of trained networks to produce the classifiers inputs. More specifically we use two different sources for the probing classifiers, pixel-level masks (as in the previous section) or the model's latent representation.

For the first case, we produce a binary mask, as in the previous experiments, to occlude perceived effects and use these to train each probing classifier. Classifiers have to predict if there was a positive, negative or none effect. Note that the classifiers need to predict any effect, not just controlled. Thus, the classifier should only be able to predict accurately controlled attributes such as the agent's position but should fail at predicting uncontrollable effects like the location of

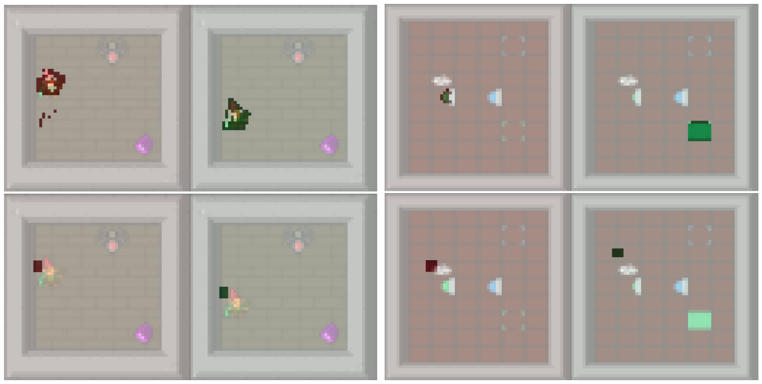


Figure 6: Controlled masks for Spiders and Lights. CEN is on the top row and ADM on the bottom. Columns correspond to the current and next observation with the mask for each method on top.

the spider or skull. We use a random policy to collect a dataset of 35K samples of the form $(m * e_p, y)$ where m is the mask produced by the model and y is the ground truth class. We ensure that each class is relatively balanced, allowing a 20% class imbalance. Each dataset is split into a typical 70/20/10, we report F1 score of each attribute on the test set.

4.2.1 Latent probing from pixels

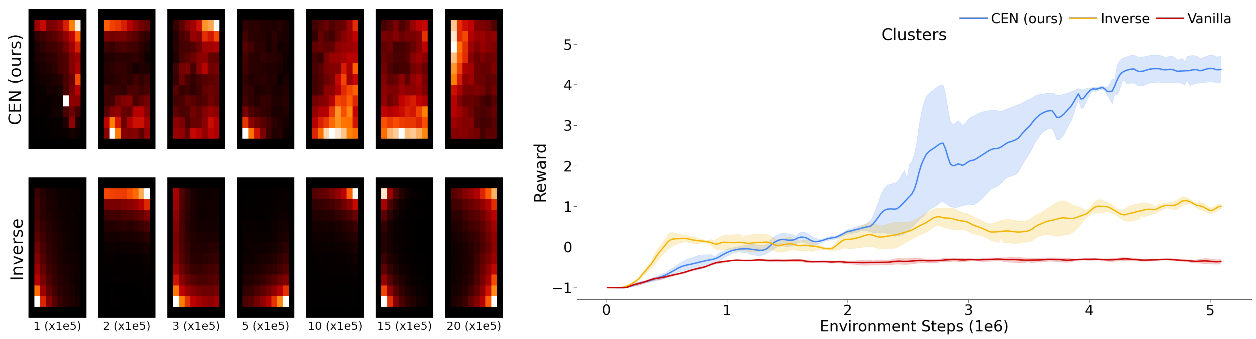
The results in Table 1(first block) indicate that CEN significantly outperforms the baseline when predicting controlled effects for state's attributes, and thus modeling controlled effects accurately. Furthermore, for both Spiders and Montezuma's Revenge environments the model cannot predict the uncontrolled effects, as expected. Even though ADM's action prediction accuracy was high ($\sim 88\%$) on every environment, it is not able to consistently predict controlled effects at the attribute-level.

4.2.2 Latent probing from latent representations

In this case, we train classifiers using a latent representation instead of pixels. We use the latent representation from CEN's controlled branch (h_c). It is unclear how to create a latent representation in ADM's architecture, thus, we use another extremely popular action-prediction model. In this experiment, we evaluate the inverse model as in Badia et al. (2020b). The features of current and next observations are concatenated to create a latent representation of what is controlled. As before, we train a linear probe to predict controlled attributes from the latent space of these models. The probe predicts changes on the x, y and direction (if applies) for agent, spider, skull and boxes; and on/off for lights and buttons.

		F1 Pixels		F1 Latent	
Environment	Attribute	CEN (ours)	ADM	CEN (ours)	Inverse
Spiders	Agent	**1.0±0.00**	0.47±0.23	**0.97±0.01**	0.67±0.05
	Spider ↓	0.35±0.03	**0.25±0.03**	**0.41±0.01**	0.44±0.02
Clusters	Agent	**0.76±0.41**	0.28±0.08	**0.97±0.01**	0.56±0.09
	Box	**0.78±0.37**	0.32±0.19	**0.95±0.02**	0.77±0.00
Lights	Agent	**0.97±0.01**	0.33±0.15	**1.0±0.01**	0.84±0.08
	Button	**0.93±0.05**	0.33±0.01	**0.99±0.0**	**0.99±0.0**
	Light	**0.93±0.04**	0.41±0.14	**1.0±0.0**	0.99±0.0
MZR	Agent	**0.66±0.08**	0.42±0.23	**0.91±0.02**	0.88±0.02
	Skull ↓	**0.19±0.03**	0.20±0.08	**0.61±0.03**	0.61±0.04

Table 1: F1 score for the state attributes when predicted from pixel or latent space. CEN outperforms ADM and the inverse model in modeling the different types of effects. Note that the agent does not control the locations of the spider and skull; the worse it can be predicted, the better.



(a) (b)

Figure 7: a) State visitation maps at different points of training of the Empty environment. CEN values different locations equally, and consequently, the agent learns to explore more uniformly. The inverse model encourages going to walls where predicting the action is hard leading high reward but poor exploration. b) CEN promotes the movement of boxes and consequently faster learning

Table 1 (second block) presents the results of this experiment, showing that CEN improves on the baseline's performance. Although the inverse model is closer to CEN's performance than ADM, it still has difficulties predicting the agent and box changes in location. We hypothesize that when probing from latent space instead of masked effects, correlations between attributes can be more easily exploited. For example, in the lights environment the representation of pushing a button may be enough to predict the button and light turning on. In the case of pixels, predicting the light turning on from the pixels representing the agent becomes a harder task.

4.3 CEN as intrinsic motivator

We have shown that CEN can learn controlled effects in an unsupervised manner. Here we showcase the use of this ability as an intrinsic motivator of a reinforcement learning agent. We consider two tasks, an empty environment without any extrinsic reward where the agent can only control itself and Clusters. The RL agent is implemented using PPO (vanilla). Additionally, PPO is augmented with the exploration bonus proposed in Never Give Up (NGU + Inverse) (Badia et al., 2020b). NGU is composed of two modules for computing episodic and life-long rewards. For simplicity the following experiments only use the episodic module which in NGU consists of a count-based method using an episodic memory to approximate the number of times an agent visited each state, and an inverse model to identify controlled states. We replace the inverse model with CEN (NGU + CEN) and use the latent representation of the controlled branch to compute NGU's episodic reward.

Empty environment: the goal of an empty environment is to showcase how each NGU variant rewards each controlled event. In this environment, only the agent's movement is controlled; thus, rewarding for controlling the agent location should promote a uniform exploration of the environment since once a location is visited, it should not be rewarded as much as visiting a new location. We hypothesize that an inverse model will create similar representations for the same action disregarding the location where it was taken. This should impair exploration since the reward will be similar regardless of where the action is performed. Fig. 7a shows how an inverse model does not. Furthermore, since it is hard to predict the action near a wall or corner, these states have higher value, and consequently, the agent tends to hog them. Contrarily, CEN promotes different locations equally, and the agent learns to explore them more uniformly.

Clusters environment: A more challenging environment is Clusters, where the agent needs to move colored boxes to their respective colored blocks. This environment provides a reward at the end of the episode corresponding to the total number of boxes correctly placed. Results provided in Fig. 7b. Due to the sparsity of the reward PPO does not learn a correct behavior in the given time. Similarly, NGU + Inverse learns to place one box but fails to learn a general behavior to solve the task. Conversely, NGU + CEN quickly learns to move boxes leading to a high extrinsic reward.

5 Related work

Intrinsic motivators: A popular way of introducing behavioral biases in RL agents is the use of intrinsic motivators (Singh et al., 2005; Mohamed and Rezende, 2015). These motivators can promote different types of exploration, from observational surprise (Burda et al., 2018) to control seeking agents (Pathak et al., 2017; Choi et al., 2019). Methods in the latter category have shown extremely good results achieving State-of-the-Art in important benchmarks. Choi et al. (2019) proposed Attentive Dynamics Model (ADM), an attention based method to discover controlled elements in the environment and used it in combination to a count-based exploration to reward the agent for discovering these elements. This method showed State-of-the-Art in Montezuma's Revenge. This work was extended in Song et al. (2019) to incorporate multi-step controlled effects as part of the intrinsic reward, again achieving human-level performance just with intrinsic rewards. Badia et al. (2020b) combined control and observational surprise motivators to create a so called "reward soup" where multiple rewards were combined into one. Their method uses an episodic memory in combination to an inverse model to promote the discovery of controlled effects in a single episode and Random Network Distillation (Burda et al., 2018) to promote long term progress through the game. This method, again, achieves State-of-the-Art in Atari's hard exploration environments. One thing in common among these methods is the use of an inverse model to model controlled aspects of the environment. Our work, instead, uses a two-branch forward model showing a better coverage of these effects. These methods show the importance of identifying what an agent can control, an avenue that deserves to be explored in depth.

Causality in deep reinforcement learning: Causality is central to humans; we think in terms of cause-effect. A similar method was proposed in Chattopadhyay et al. (2019), where they use causal attribution methods to analyzed the effect of inputs on a neural network's outputs via causal attribution. Recent work has introduced causality into deep reinforcement learning (Foerster et al., 2018; Buesing et al., 2018; Jaques et al., 2018; Dasgupta et al., 2019; Goyal et al., 2019; Nair et al., 2019; Madumal et al., 2020) showing that this is a promising avenue for the training of agents. Corcoll and Vicente (2020) proposed an attribution method to learn temporal abstractions for object-centric hierarchical RL. Bellemare et al. (2012b) compute controllable aspects of the environment by generating a mask with all possible controllable areas of an image and uses it as part of the policy's input. In this work, we identify the controlled effects of individual actions using causal concepts of normality and blame.

6 Conclusions

The identification of controlled effects is a central task in RL. This work proposes a fully unsupervised approach to this problem named Controlled Effect Network (CEN). CEN creates a normative world using counterfactuals and compares what actually happened with this normative world to attribute changes on the environment to the agent. The presented experiments show that, despite being unsupervised, this method precisely identifies controlled effects. Furthermore, we show that popular methods based on action prediction do not model controlled effects besides the agent. CEN is showcased as intrinsic motivator for RL agents where results suggest that a more targeted exploration leads to substantially better policies.

Understanding the world in terms of what can be controlled is a promising avenue. In this direction, more sophisticated modeling of normality, e.g., social norms or time-based norms, can provide a measure of relevance or importance, useful to prioritize the learning of effects. Additionally, we assumed that agents only control effects happening immediately after performing an action. A valuable extension to this work is to identify the consequences of an action multiple steps ahead.

Acknowledgments

The authors would like to thank Jaan Aru and Meelis Kull for insightful comments on the manuscript. This work was supported by the University of Tartu ASTRA Project PER ASPERA, financed by the European Regional Development Fund and the University of Tartu HPC.

References

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised State Representation Learning in Atari. 2019.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark, 2020a.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies, 2020b.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula, 2019.

Chris Bamford, Shengyi Huang, and Simon Lucas. Griddly: A platform for ai research in games, 2020.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, Vol. 47:253–279, 2012a. doi: 10.1613/jair.3912. URL http://arxiv.org/abs/1207.4708. cite arxiv:1207.4708.

Marc G. Bellemare, J. Veness, and Michael Bowling. Investigating contingency awareness using atari 2600 games. In *AAAI*, 2012b.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL http://arxiv.org/abs/1606.01540. cite arxiv:1606.01540.

Lars Buesing, Theophane Weber, Yori Zwols, Sebastien Racaniere, Arthur Guez, Jean-Baptiste Lespiau, and Nicolas Heess. Woulda, coulda, shoulda: Counterfactually-guided policy search, 2018.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov Openai. Exploration by Random Network Distillation. 2018.

Aditya Chattopadhyay, Piyushi Manupriya, Anirban Sarkar, and Vineeth N Balasubramanian. Neural network attributions: A causal perspective. 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 981–990. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/chattopadhyay19a.html.

Nuttapong Chentanez, Andrew Barto, and Satinder Singh. Intrinsically motivated reinforcement learning. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 1281–1288. MIT Press, 2005. URL https://proceedings.neurips.cc/paper/2004/file/4be5a36cbaca8ab9d2066debfe4e65c1-Paper.pdf.

Jongwook Choi, Yijie Guo, Marcin Moczulski, Junhyuk Oh, Neal Wu, Mohammad Norouzi, and Honglak Lee. Contingency-Aware Exploration in Reinforcement Learning. *ICLR*, pages 1–20, 2019.

Oriol Corcoll and Raul Vicente. Disentangling causal effects for hierarchical reinforcement learning, 2020.

Fiery Cushman, Joshua Knobe, and Walter Sinnott-Armstrong. Moral appraisals affect doing/allowing judgments. *Cognition*, 108(1):281–289, July 2008. doi: 10.1016/j.cognition.2008.02.005. URL https://doi.org/10.1016/j.cognition.2008.02.005.

Ishita Dasgupta, Jane Wang, Silvia Chiappa, Jovana Mitrovic, Pedro Ortega, David Raposo, Edward Hughes, Peter Battaglia, Matthew Botvinick, and Zeb Kurth-Nelson. Causal reasoning from meta-reinforcement learning, 2019.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. URL https://ojs.aaai.org/index.php/AAAI/article/view/11794.

Yasuhiro Fujita, Prabhat Nagarajan, Toshiki Kataoka, and Takahiro Ishikawa. Chainerrl: A deep reinforcement learning library. *Journal of Machine Learning Research*, 22(77):1–14, 2021. URL http://jmlr.org/papers/v22/20-376.html.

T. Gerstenberg and D. A. Lagnado. Attributing responsibility: Actual and counterfactual worlds. In Joshua Knobe, Tania Lombrozo, and Shaun Nichols, editors, *Oxford Studies in Experimental Philosophy*, volume 1, pages 91–130. Oxford University Press, 2014.

Anirudh Goyal, Shagun Sodhani, Jonathan Binas, Xue Bin Peng, Sergey Levine, and Yoshua Bengio. Reinforcement Learning with Competitive Ensembles of Information-Constrained Primitives. pages 1–21, 2019.

Guy Grinfeld, David Lagnado, Tobias Gerstenberg, James F. Woodward, and Marius Usher. Causal responsibility and robust causation. *Frontiers in Psychology*, 11:1069, 2020. ISSN 1664-1078. doi: 10.3389/fpsyg.2020.01069. URL https://www.frontiersin.org/article/10.3389/fpsyg.2020.01069.

Caglar Gulcehre, Tom Le Paine, Bobak Shahriari, Misha Denil, Matt Hoffman, Hubert Soyer, Richard Tanburn, Steven Kapturowski, Neil Rabinowitz, Duncan Williams, Gabriel Barth-Maron, Ziyu Wang, Nando de Freitas, and Worlds Team. Making efficient use of demonstrations to solve hard exploration problems. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SygKyeHKDH.

Joseph Y. Halpern. *Actual Causality*. The MIT Press, 2016. ISBN 0262035022.

Joseph Y. Halpern and Christopher Hitchcock. Graded Causation and Defaults. *The British Journal for the Philosophy of Science*, 66(2):413–457, 04 2014. ISSN 0007-0882. doi: 10.1093/bjps/axt050.

Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Christopher Hitchcock and Joshua Knobe. Cause and norm. *Journal of Philosophy*, 106(11):587–612, 2009. doi: 10.5840/jphil20091061128.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A. Ortega, DJ Strouse, Joel Z. Leibo, and Nando de Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning, 2018.

D. Kahneman and D. T. Miller. Norm theory - comparing reality to its alternatives. *Psychol Rev Psychol Rev*, 93(2): 136–153, 1986.

Steven Kapturowski, Georg Ostrovski, Will Dabney, John Quan, and Remi Munos. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1lyTjAqYX.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Joshua Knobe and Benjamin Fraser. *Causal judgment and moral judgment: Two experiments*, pages 441 – 447. The MIT Press, United States of America, 2008. ISBN 9780262195690.

Antonia F Langenhoff, Alex Wiegmann, Joseph Y Halpern, Joshua Tenenbaum, and Tobias Gerstenberg. Predicting responsibility judgments from dispositional inferences and causal attributions, Sep 2019. URL psyarxiv.com/63zvw.

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.

Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning, 2015.

Adam Morris, Jonathan S Phillips, Thomas Icard, Joshua Knobe, Tobias Gerstenberg, and Fiery A Cushman. Causal judgments approximate the effectiveness of future interventions, Apr 2018. URL psyarxiv.com/nq53z.

Suraj Nair, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Causal induction from visual observations for goal directed tasks, 2019.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction, 2017.

J Pearl, M Glymour, and N P Jewell. *Causal Inference in Statistics: A Primer*. Wiley, 2016. ISBN 9781119186847.

Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009. ISBN 052189560X.

Satinder Singh, Andrew G Barto, and Nuttapong Chentanez. Intrinsically Motivated Reinforcement Learning. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2005. ISSN 19430604. doi: 10.1109/TAMD.2010.2051031.

Yuhang Song, Jianyi Wang, Thomas Lukasiewicz, Zhenghua Xu, Shangtong Zhang, Andrzej Wojcicki, and Mai Xu. Mega-reward: Achieving human-level play without extrinsic rewards, 2019.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning - An Introduction*. 2018. ISBN 9780262039246.

Oriol Vinyals, Igor Babuschkin, Wojciech Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John Agapiou, Max Jaderberg, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575, 11 2019. doi: 10.1038/s41586-019-1724-z.

Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015.

Appendices

A Environments

A.1 Griddly

Griddly (Bamford et al., 2020) is a highly optimized grid-world based suite of environments. Environments used in this work based on Griddly generate 64×64 pixel observations, although the size of the grid-world may vary. Griddly supports multiple rendering formats, this work uses the 2D rendering of sprites.

Spiders: is a 6x6 arena where a Gnome (the agent) has to grab a Gem without being killed by a Spider. The agent dies if it collides with the spider. In this environment, the agent can move *left*, *right*, *up*, *down* or *stay*. The spider takes an action randomly from the following: rotate left, rotate right or move forward. This environment's controlled entities are: Agent.

Clusters: is a 13x10 arena where a Knight has to move boxes of the same color to their corresponding colored-block without touching the spikes. There are two different colors, blue and red. The agent is rewarded with +1 whenever a box is pushed towards a similar colored block. The agent dies if it collides with spikes or if a box is destroyed by spikes. The agent can move *left*, *right*, *up*, *down* or *stay*. This environment's controlled entities are: Agent and Boxes.
For RL experiments, we made the environment more sparse by removing all intermediate reward and only rewarded the agent after all the boxes of the same color are pushed to blocks. Since the agent can not get any reward whenever a box is stuck to a wall. We removed boxes that touches the wall and punished the agent with -0.01. We then scaled the reward of solving a color with the number of boxes pushed to the block. This modifications encourage the agent to solve the environment by pushing the maximum number of boxes into blocks while preventing it from getting deprived from reward by accidentally pushing boxes to walls.

Lights: is a 11x8 arena where a Ghost (the agent) has to turn all the lights on by pressing each button. Buttons and lights are colored either blue or green. Pressing a button of one color turns the light of the same color on. The agent can move *left*, *right*, *up*, *down* or *stay*. This environment's controlled entities are: Agent, Buttons and Lights.

Empty: this environment is a copy of the clusters environment where all the boxes, blocks, spikes and rewards were removed.

Ground truth: Griddly provides access to each entity's state (x, y, light is on/off, etc); a binary mask is produced for each controlled entity with any of its attributes changed when transitioning between two time steps. x and y coordinates are projected into pixel space and a bounding box is generated using the size of that entity. Note that the coordinates may be different between steps thus the generated mask may enclose multiple locations. The resulting masks for each entity are combined into a single mask m by taking the maximum value among them. Since we want to know what pixels were actually controlled, the final ground truth mask is produced as: $m \cdot e_a^p$.

A.2 Atari Montezuma's Revenge

The ALE (Bellemare et al., 2012a) provides access to Atari 2600 games to learning methods like RL. As it has been a popular choice by methods using inverse models, in this work we use the game Montezuma's Revenge to evaluate CEN. This environment provides uncontrolled as well as controlled effects with more complex entities. The environment typically generates observations of 210x160 pixels which we downscale to 64x64 pixels. Additionally the action space is of size 10.

Ground truth: in contrast to Griddly, we can actually compute counterfactual worlds by saving and loading the state of the game (RAM) multiple times when taking different actions. For this, we directly compute Eq. 5 using ALE's special calls *cloneSystemState* and *restoreSystemState*. More precisely, we compute every possible perceived effect reachable from the current state and build a normal effect using the mode over all possible effects. Then, we compare the perceived effect for the agent's chosen action against the normal effect. The ground truth mask will have 1s where these two effects are different.

B Training

B.1 Architecture

Encoder: is composed of two 2D convolutional layers with 4x4 kernels, stride 2 and padding of 1. Additionally, we have 2 residual blocks each with two 2D convolutional layers with stride 1 and padding of 1. The first layer has a kernel of 3x3 and the second layer of 1x1. ReLU is used as activation function throughout the network; BatchNorm is used between each layer; and 64 channels on every convolutional layer. We project the resulting maps into a flatten vector of size 32 using a linear layer with ReLU activation function.

Decoder: this module is composed of six 2D transposed convolutional layers all having 4x4 kernels, stride 2 and padding of 1. Each layer uses ReLU as activation function but the output layer which uses Tanh activation. Every layer uses 64 channels with the exception of the last layer which outputs a 1 channel prediction of the perceived effects. Parameters are shared among the controlled and normal branch decoders.

Controlled and normal modules: both modules are composed of three linear layers with 32 hidden units, each with a ReLU as activation function. The input to the controlled branch are the encoded observation and an embedding of size 8 of the chosen action.

PPO Agent: uses an encoder consisting of 3 convolutional layers with (channels, padding, strides) equal to (32, 8, 4), (64, 4, 2), (64, 3, 1) respectively. The encoder is followed by two linear layers of sizes 512 and number of actions respectively, to transform the feature map to the environment's number of actions.

B.2 Mask generation

CEN's controlled masks are generated using the encoder, controlled branch and decoder. The predicted controlled effect is binarized using a threshold as $m_{\text{CEN}} = (-T < \hat{e^c})|(\hat{e^c} > T)$. In the case of ADM, its attention mask is thresholded in the same way and resized to the size of the effect.

B.3 Hyperparameters

Name	Value	Sweep
hidden size	32	[16, 32, 64, 128]
latent size	128	[16, 32, 64, 128, 256]
channels	64	[16, 32, 64, 128]
learning rate	0.0001	[0.0001, 0.0005, 0.001, 0.005]
α	0.01	[0.001, 0.01, 0.1, 1, 5, 10, 20, 30, 50]
T	0.01	-

Table 2: CEN hyperparameter sweeps and final values used.

Name	Value	Sweep
entropy	0.05	[0.01, 0.05, 0.1, 0.5, 1, 5]
hidden size	64	[16, 32, 64, 128]
attention size	128	[32, 64, 128, 256]
learning rate	0.0001	[0.0001, 0.0005, 0.001, 0.005]
T	0.01	-

Table 3: ADM hyperparameter sweeps and final values used.

Name	Value	Sweep
batch size	512	[64, 128, 512, 1024]
latent size	32	[8, 16, 32]
CEN encoder output size	128	[16, 32, 64, 128]
learning rate	0.0005	[0.00005, 0.0001, 0.0002, 0.0005, 0.001]
IR Beta	0.001	[0.0001, 0.001, 0.002, 0.005, 0.01, 0.1]
Rollout size	2048	[1024, 2048, 4096]
discount	0.95	-
epochs	10	-

Table 4: PPO hyperparameter sweeps and final values used.