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

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

Abstract

Modeling controllable aspects of the environment enable better prioritization of interventions and has become a popular exploration strategy in reinforcement learning methods. 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 important events. Humans, on the other hand, assign blame to their actions to decide what they controlled. Here we propose Controlled Effect Network (CEN), an unsupervised method based on counterfactual measures of blame. CEN is evaluated in a wide range of environments showing that it can identify controlled effects better than popular models based on action prediction.

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. The inefficiency of these algorithms makes it necessary to provide a dense reward. Environments with sparse rewards are known to be extremely difficult to solve and a good exploration strategy is imperative.

A popular approach to exploration is to introduce behavioral biases in the form of intrinsic motivators (Chentanez et al., 2005). This technique aims to produce dense rewards that facilitate the learning of task-agnostic behavior. Typical motivators drive the agent to discover novel states and by doing so increase the chance of reaching the environment's extrinsic reward. Numerous motivators have been developed by 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 of controlling their environment. A common design principle among these methods is the use of an inverse model to predict the chosen action. 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. For example, if an agent moves to the right and pushes a box by doing so, the model only needs to represent the agent's change of location to predict its action. Thus, the model is free to ignore the box's movement which the agent controlled. Surprisingly the ability to identify controlled effects has only been evaluated as a proxy to a reward-based task but not in isolation. In this work, we evaluate action prediction models on the problem of controlled effect identification and show that, indeed, they do not model controlled effects other than those upon the agent.

A causal approach is to compare counterfactual worlds, i.e. an effect is controllable if the agent would have had another effect by taking a different action (Pearl, 2009). 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-right", that moves the box. 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 imagined 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-right" since its normative state is to not move.

To improve upon this issue we propose Controlled Effects Network (CEN), an unsupervised method 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 disentangles what is normal and what is controlled to predict the next step effects. Our experiments show that CEN can disentangle effects correctly and that it can model both distant and agent-centric controlled effects.

2 Identifying controlled effects using blame

Our goal is to identify what was controlled by the agent. Take Fig. 1 as an example of a skull and an agent about to collide, leading to losing a life. In both scenarios losing a life is controllable by the agent, i.e. there is a world (moving right) where the agent would have not lost a life. In contrast, humans perception of causality is associated to blame (Gerstenberg and Lagnado, 2014), e.g. in the first scenario we would say that the skull caused the agent to lose a life, whereas in the second one the agent would be to blame since it moved onto the skull and lost a life.

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"*. In the previous example, skulls have a causal effect on the agent's life since the agent's life can decrease due to an skull's attack. Actual causality, proposed in Halpern (2016), studies causal relations between individual events of X and Y. It aims to answer questions like, did the skull moving right cause the agent to lose a life? In the following, we introduce the concept of (actual) causal effect to then define controllable effect in the context of RL.

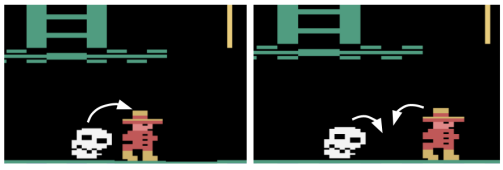

Figure 1. Is losing a life caused by the skull, the agent or both? Left: the skull moves onto the agent. Right: the agent and skull move towards each other. In both cases the agent loses a life.

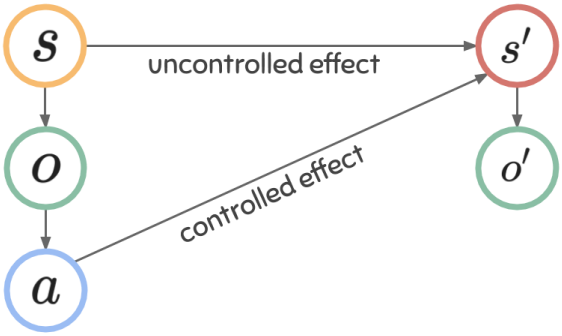

Figure 2. Causal diagram of a typical RL setting. Controlled effects depend on the agent's choice of action.

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

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

where Y_i^x reads as "what would the value of an individual Y_i be if X is forced to be x". Similarly, $Y_i^{\tilde{x}}$ describes the value of Y_i when X is forced to not be x. Note that since we are using the actual causality framework the sub-index i refers to an individual event. In the following, we use Y_i^x and Y^x interchangeably.

The world $Y^{\tilde{x}}$ does not exist and 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 realm of RL, X and Y take the form of actions, states and observations. Figure 2 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 $a \in \mathcal{A} \subseteq \mathbb{N}$. 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 the definition of controllable effect has far-reaching consequences: a motionless agent has a causal effect on itself since it could move right; an agent next to a button that can be pressed has a causal effect on

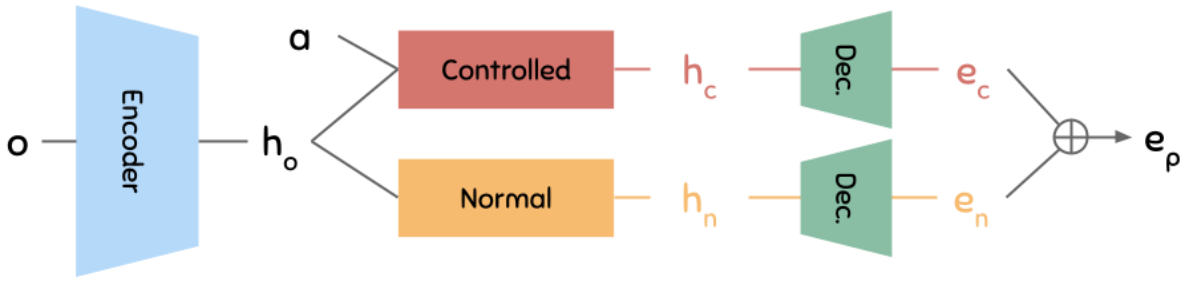

Figure 3. CEN is a forward model that predicts effects from a given observation and action. It disentangles controlled and normal effects in latent space to then decode them into pixel space. Note that the weights of the decoders are shared.

that button, even if it does not press it; an agent shot by a gun has a causal effect on its life because it could have dodged it. If Eq. 2 is used 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. if an agent's oxygen level decreases unless still, doing nothing has an effect on the agent's oxygen level. Instead, we would want a more human-like definition of what is controlled where an agent controls a button if pressed; a box if moved or its life if the bullet is dodged.

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). For example heat, fuel and oxygen are necessary causes for a forest fire but people typically say that heat (a match or lightning) and not oxygen caused the fire. Thus, heat is to blame for the forest fire since oxygen and fuel are normally present in a forest. 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 operator used in Eq. 1.

In RL, 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 Richard S. and Barto (2018), Generalized Value Functions (GVF) 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 as

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

Intuitively, Eq. 5 builds a normal world by observing every alternative perceived effect reached by each action and choosing the most frequent of them. To simplify notation, the following sections use controlled effect as $e_c^a = ICE_{e_p}^a$ and normal effect $e_n = \beta_{e_p}$. Consider the example in Fig. 1, scenario on the right. Moving left would cause the agent to die but staying or moving right would keep him alive. If the agent chooses to move left, Eq. 5 indicates that staying put would be normal and that losing a life is controlled by the agent. Contrarily, the scenario on the left would take the loss of a life as normal.

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 directly but approximates the final normal world. It is also important to notice that the controlled effects identifiable by Eq. 5. in a partially observable setting ($o \neq s$), where the full state is not observed, 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.

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 using perceived effects as training signal.

Here, we introduce **Controlled Effects Network (CEN)**, depicted in Fig. 3. CEN is a forward model where observations are encoded into a latent representation and used to predict the perceived effect resulting from taking a specific action. Contrarily to conventional forward models CEN is composed of two branches, where each branch creates a specific latent representation for controlled and normal effects; in a similar fashion to Dueling Networks (Wang et al., 2015).

The controlled branch has access to the action (in contrast to the normal branch), which allows to accurately predict effects controlled by the agent, something that the normal branch cannot do. Notice that the controlled branch alone can predict the perceived effect resulting from an action, i.e. the normal branch would not be needed to just act as a forward model.

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 using 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.

In order to promote the controlled branch to model only controlled effects, we use the following loss

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

where the first part of the loss promotes the modeling of perceived effects, just as in a conventional forward model in which the predicted target $\hat{e_p^a} = \hat{e_c^a} + \hat{e_n}$ is compared to the ground truth (e_p^a). The second part of the loss enforces the network to use the normal branch as much as possible to model the world. Additionally, a hyperparameter α regulates how much the normal branch should model the dynamics of the environment.

Let's take again the example in Fig. 1 (right). The normal branch is forced to model the skull's movement since it does not depend on the agent's action. Furthermore, if α is large enough the normal branch is forced to also model the agent being still. Since the controlled branch can only add to what the normal branch predicts, it needs to model the movement of the agent to minimize its loss.

4 Experiments

In this section we evaluate[1] CEN's ability to model controlled effects and compare it to the action-prediction model presented in Choi et al. (2019), Attentive Dynamics Model (ADM). The experiments are divided in two main categories, pixel and attribute prediction. Pixel prediction experiments aim to compare these two methods as in a binary-class segmentation task where each method produces a mask that covers as much ground truth as possible. In some applications such as for example making use of intrinsic motivators, knowing the position/value of controlled objects may be enough, and therefore, learning to produce a spatial mask is unnecessarily precise. Thus, the second set of experiments aims to measure how well these methods can model controlled effects of individual attributes in the environment's state instead of at pixel-level.

To achieve this, we use multiple environments 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); see an example in Fig. 4. We give more details of these environments in each experiment and appendix.

CEN is implemented with 2D convolutional layers and ReLU activation function for the encoder and decoders; the normal and controlled branches are implemented with linear layers and ReLU activation function. 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, hyperparameters, and training.

4.1 Controlled effects at pixel-level

Next we present the results for the pixel-level experiments. The goal is to evaluate CEN's predictions of controlled effects at pixel-level. We report pixel F1 scores between ground truth and prediction binary masks. Although Eq. 5 computes the magnitude and direction of the effects, we thresholded the predicted controlled effect. We provide more details in the appendix. 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 with a random policy. The evaluation of the model is done using

[1] This work's networks, training and evaluation have been implemented using PyTorch (Paszke et al., 2019) and NumPy (Harris et al., 2020); our experiments are managed using Weights & Biases (Biewald, 2020).

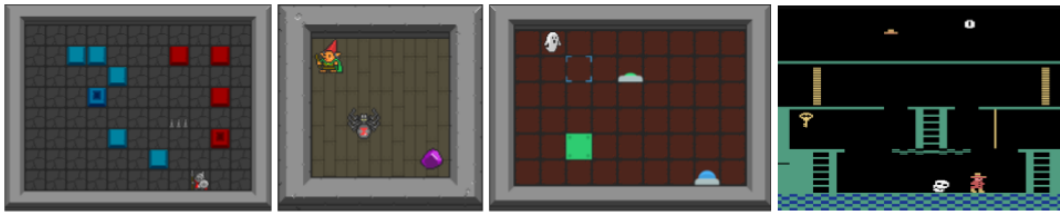

Figure 4. Suite of environments used in our experiments. From left to right: Clusters, Spiders, Lights and Montezuma's Revenge (MZR).

an evaluation environment with different seed to the one used for training during 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 Atari environments the ground truth was collected by computing Eq. 5 using the ALE's special calls *cloneSystemState* and *restoreSystemState*.

4.1.1 Controlled vs uncontrolled effects

In this experiment we evaluate CEN's ability to disentangle controlled from uncontrolled effects. Here we use the Spiders environment which has two main entities, the agent and an uncontrolled angry spider. If the model has learned to identify controlled effects the masks produced should only focus on the agent and not on the spider.

Fig. 5 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.

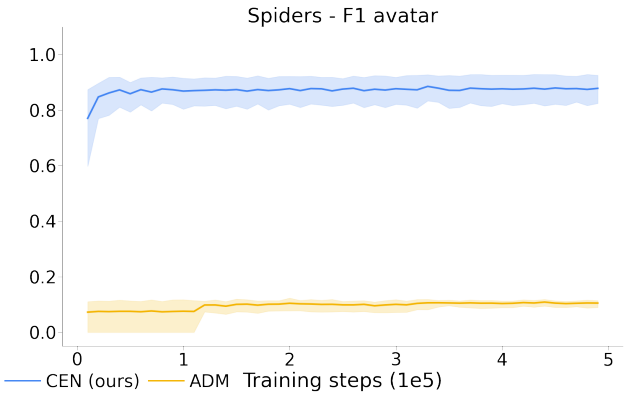

Figure 5. Pixel F1 scores for CEN and the baseline in the Spiders environment on the task of identifying the controlled effects produced by the avatar. CEN can correctly disentangle the agent from the randomly moving spider.

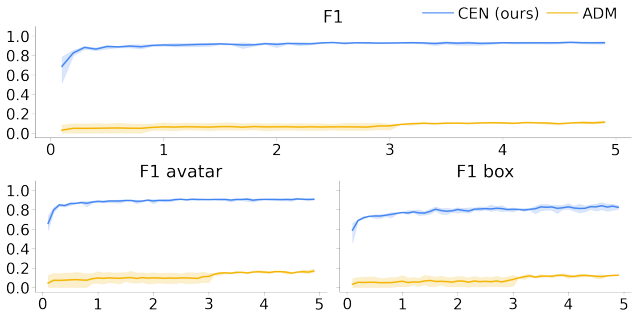

Figure 6. F1 pixel score on the Clusters environment. CEN is able to model not just the agent but also close controlled effects like the movement of boxes. Top) F1 score for every effect; Bottom-left) F1 score when the avatar moves; and Bottom-right) F1 score when there is a box moving. X axis refers to the training steps (1e5).

4.1.2 Close 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 "close" controlled effect to an effect that happens near the agent, like the box's movement. To evaluate CEN on this kind of effects we use the Clusters environment. In this environment an agent needs to move

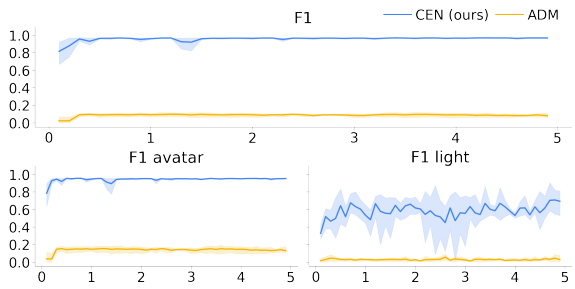

Figure 7. F1 score on the Lights environment. CEN can model distant effects, even when the agent is not part of the controlled effect. Top) F1 score for every effect; Bottom-left) F1 score when the avatar moves; and Bottom-right) F1 score when a light turns on. X axis refers to the training steps (1e5).

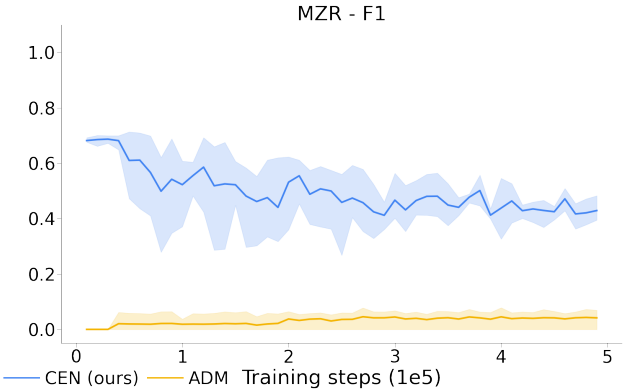

Figure 8. Montezuma's Revenge F1 pixel score. CEN outperforms the baseline even in an environment with more complex dynamics.

colored boxes to their corresponding fixed colored blocks.

Fig. 6 shows that CEN can accurately model the agent and boxes effects better than the baseline. 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 Distant controlled effects

Similarly 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 will use the Lights environment. Here the environment presents two buttons of different color that, when pressed, turn on their corresponding light of the same color. Lights are relatively far away from their corresponding buttons thus making it difficult to model them. As show in Fig. 7, CEN is able to model this kind of controlled effects. Although there is a clear decrease in F1 score, we believe that this is due to the more complex shape of the 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 the agent makes Montezuma's Revenge 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. Additionally, Fig. 9 (last column) shows examples of the masks generated by CEN and our baseline for this environment. Although the F1 score is lower than in the other environments, Atari has more complex shapes which make the model have lower F1 score even when modeling the controlled effect.

4.2 Controlled effects at attribute-level

In the second set of experiments, we want to analyze if CEN can predict controlled effects of attributes present on the environment's state, e.g. the agent's (x, y) location or if a light turned on. In these experiments, we employ a probing technique similar to the one described in Anand et al. (2019). We freeze trained versions of the CEN and baseline networks and use them to train a classifier per each attribute in the environment's state. We use the frozen networks to produce a binary mask of controlled effects, as in the previous experiments, and applying the resulting mask to occlude the perceived effects. These occluded effects are then used to train each probing classifier to predict if there was a positive, negative or none effect. Note that the classifier needs to predict any effect, not just controlled. This way the classifier should only be able to predict accurately controlled attributes such as the agent's position but should fail at predicting the spider's location. Furthermore, the model does not need to produce a perfect mask for the classifier to be able to predict these attributes. 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.

The results in Table 1 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 uncon-

Environment	Attribute	F1	
		CEN (ours)	ADM
Spiders	Agent	**1.0±0.00**	0.47±0.23
	Spider ↓	0.35±0.03	**0.25±0.03**
Clusters	Agent	**0.76±0.41**	0.28±0.08
	Box	**0.78±0.37**	0.32±0.19
Lights	Agent	**0.97±0.01**	0.33±0.15
	Button	**0.93±0.05**	0.33±0.01
	Light	**0.93±0.04**	0.41±0.14
MZR	Agent	**0.66±0.08**	0.42±0.23
	Skull ↓	**0.19±0.03**	0.20±0.08

Table 1. F1 score for the predicted state attributes. CEN outperforms the baseline and is able to model effects distant from the agent. Note that the locations of the spider and skull are not controlled by the agent thus, the worse it can be predicted the better.

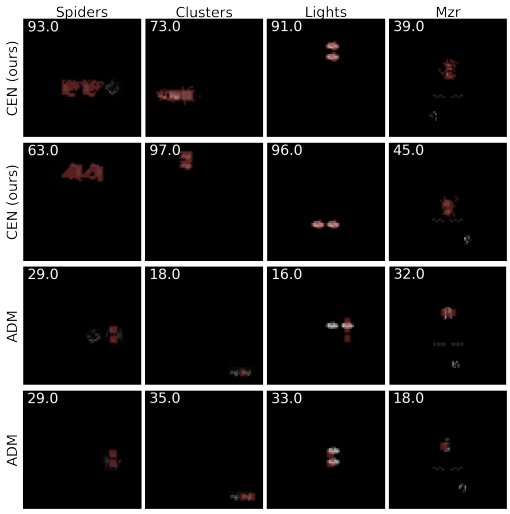

Figure 9. Samples of the masks generated by CEN and the baseline on each environment. Samples were picked for both algorithms based on their high F1 score and diversity. The original perceived effect is marked in black and white; the generated masks are marked in red. F1 score is overlaid in white.

trolled 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.

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). 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. Their method generates a mask with all possible controllable areas of an image and uses it as part of the policy's input. In this work, we instead identify controlled effects using causal concepts of normality and blame.

6 Conclusions

The identification of controlled effects is a central task in RL. This work proposes an unsupervised approach to this problem named Controlled Effect Network (CEN). CEN computes counterfactual worlds to assign blame to the agent's actions for what happened. The presented experiments show that, despite of being unsupervised, this method learns to identify controlled effects successfully. Furthermore, we show that popular methods based on action prediction do not model controlled effects besides the agent.

Understanding the world in terms of controlled effects, instead of actions, is a promising avenue. This approach narrows down the large amount of possibilities an agent faces and can be used to prioritize the next action to take based on how its consequences align with the agent's goals. The expressive power of effects ought to be explored to achieve better prioritization of effects. 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. This study relies on the notion that effects can only be controlled at the next time step and that the environment controls follow up effects. For example, a ball pushed by the agent will be identified as controlled the moment the agent pushes it but it will be governed by the environment's dynamics thereafter. A useful extension to this work would be to identify the consequences of an action taken N steps ago. We believe that the fast pace of the graph neural network ecosystem allows for this extension. These improvements would represent a major leap towards more efficient RL agents.

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.

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.

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.

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

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.

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 Experiments

A.1 Qualitative results

Here we provide additional qualitative results for our first set of experiments. CEN constantly find the agent and other controlled entities. Errors in Griddly environments are typically due to not recognizing the movement of the agent or predicting it will move when it did not. Montezuma's Revenge shows more complex patterns which the model fails to find a mask that perfectly suits the ground truth, even then the pixels outside the ground truth are near the agent.

A.2 Inverse model and latent probing

In this experiment, we evaluate the inverse model used in (Badia et al., 2020b). More precisely, we evaluate CEN and this new baseline on the task of predicting controlled attributes. The main difference with experiment 4.2 is that the latent representation is used to predict these attributes instead of the masked effects. Note that ADM does not have a clear latent representation of what is controlled. Similarly to the previous experiment, we train a linear probe to predict controlled attributes from the latent space of these models. In the case of CEN, we use the output of the controlled branch. For the inverse model, we concatenate the features of current and next observations. 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.

Results are presented in Table 2 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.

		F1	
ENVIRONMENT	ATTRIBUTE	CEN (OURS)	INVERSE
SPIDERS	AGENT	**0.97±0.01**	0.67±0.05
	SPIDER ↓	**0.41±0.01**	0.44±0.02
CLUSTERS	AGENT	**0.97±0.01**	0.56±0.09
	BOX	**0.95±0.02**	0.77±0.00
LIGHTS	AGENT	**1.0±0.01**	0.84±0.08
	BUTTON	**0.99±0.0**	**0.99±0.0**
	LIGHT	**1.0±0.0**	0.99±0.0
MZR	AGENT	**0.91±0.02**	0.88±0.02
	SKULL ↓	**0.61±0.03**	0.61±0.04

Table 2. F1 score for the predicted state attributes. CEN outperforms the baseline and is able to model effects distant from the agent. Note that the locations of the spider and skull are not controlled by the agent thus, the worse it can be predicted the better.

B Environments

B.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 present spikes. There are two different colors, blue and red. 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.

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.

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: $m \cdot e_a^p$.

B.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.

C Training

C.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.

C.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.

C.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 3. 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 4. ADM hyperparameter sweeps and final values used.

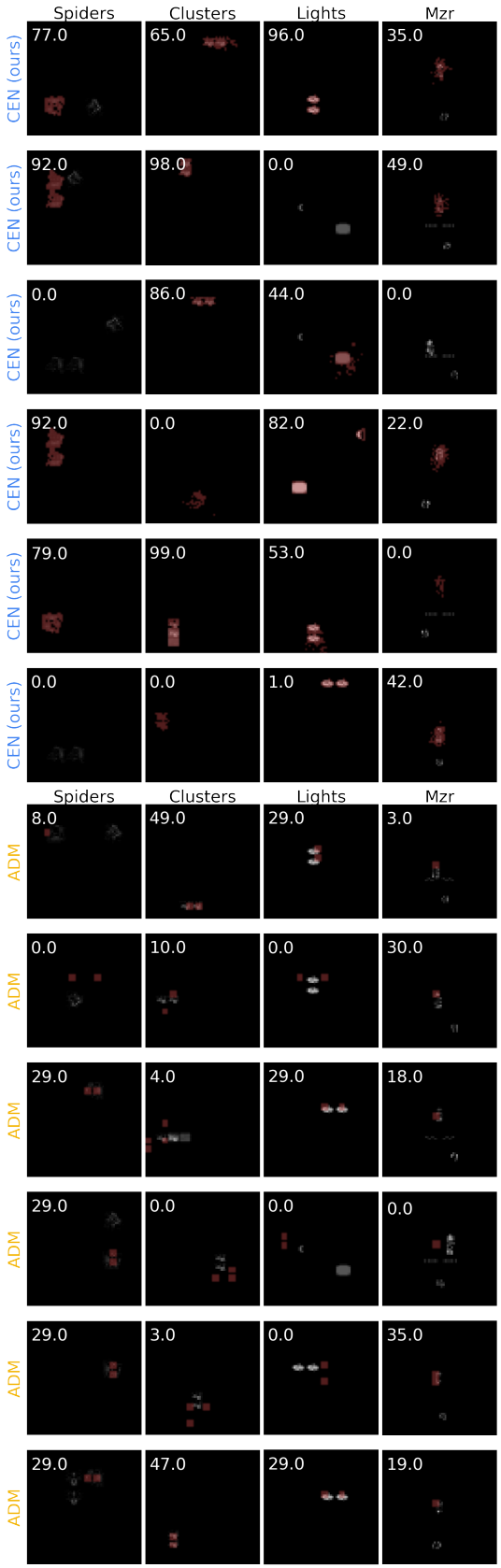

Figure 10. Additional examples of masks generated by CEN (left) and ADM (right).