DISENTANGLING CAUSAL EFFECTS FOR HIERARCHICAL 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

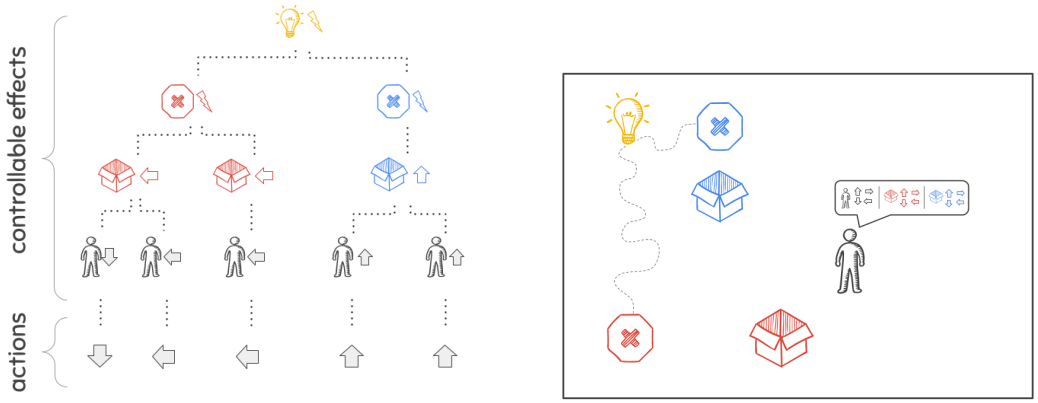
Exploration and credit assignment under sparse rewards are still challenging problems. We argue that these challenges arise in part due to the intrinsic rigidity of operating at the level of actions. Actions can precisely define how to perform an activity but are ill-suited to describe what activity to perform. Instead, causal effects are inherently composable and temporally abstract, making them ideal for descriptive tasks. By leveraging a hierarchy of causal effects, this study aims to expedite the learning of task-specific behavior and aid exploration. Borrowing counterfactual and normality measures from causal literature, we disentangle controllable effects from effects caused by other dynamics of the environment. We propose CEHRL[1], a hierarchical method that models the distribution of controllable effects using a Variational Autoencoder. This distribution is used by a high-level policy to 1) explore the environment via random effect exploration so that novel effects are continuously discovered and learned; and to 2) learn task-specific behavior by prioritizing the effects that maximize a given reward function. In comparison to exploring with random actions, experimental results show that random effect exploration is a more efficient mechanism, and that by assigning credit to few effects rather than many actions, CEHRL learns tasks more rapidly.

1 Introduction

Value-based methods for reinforcement learning (RL) (Sutton and Barto, 1998) have achieved impressive results in environments with dense rewards (Mnih et al., 2013). These methods learn by estimating the causal effects of actions on the reward function. This type of learning is particularly ineffective in environments with sparse rewards where no rewards are given for long periods of time, and thus requiring to collect vast amounts of experiences; each providing little to no learning. A promising solution is to use hierarchical RL methods (Sutton et al., 1999) to learn reusable skills. Skill discovery research has focused on information theory based methods (Florensa et al., 2017; Eysenbach et al., 2018; Sharma et al., 2019) or intrinsic motivators (Kulkarni et al., 2016; Nachum et al., 2018).

In contrast, we are motivated by studies in the field of developmental psychology, indicating that children use the effects of their actions to learn different ways of controlling their environment (Goodman et al., 2007; Buchsbaum et al., 2012, 2015). We conceptualize skills as useful changes on the environment, thus we propose to learn by estimating the causal effects of actions on the state. Intuitively, these effects describe controllable ways of changing the environment. Since not every change on the state is controllable, we adopt counterfactual and normality measures (Pearl, 2009; Halpern, 2016) from causal literature to disentangle effects caused by the agent, from effects caused by other dynamics, e.g. other agents or dynamic objects. A key aspect of controllable effects is that

[1] pronounced 'ciril'.



(a) Temporally abstract effects (b) Exploration with effects

Figure 1: **a)** A complex effect like turning on a light requires simpler effects like activating a switch, all the way down to actions; creating an implicit hierarchy of effects. **b)** To get a reward, the agent needs to turn a light on by moving boxes to their switches. Instead of performing random actions until the light turns on, CEHRL's exploration is based on performing controllable effects (e.g. moving boxes) to discover and learn novel and more complex effects. Continuously learning and performing controllable effects makes it more likely to find how to turn the light on.

they are composable, i.e. to perform a complex effect, an agent needs to combine simpler effects, which are also composed of more simple effects, all the way down to actions. Figure 1(a) illustrates the compositional nature of controllable effects, making them temporally abstract, i.e. effects may take several, possibly variable, number of actions to be performed (Sutton et al., 1999).

To take advantage of this compositionality, we introduce CEHRL (Causal Effects for Hierarchical RL); a method that builds an implicit hierarchy of controllable effects that servers a twofold purpose. First, instead of using random actions to explore the environment, CEHRL relies on random effect exploration. Random effect exploration is based on the idea that by combining basic effects the agent can discover and learn more complex effects, continuously enriching the hierarchy in a task-agnostic manner. To illustrate this type of exploration, consider the scenario presented in Figure 1(b) where an agent has already learned some basic effects like moving the agent or the boxes. Finding how to turn the light on is more likely by combining these effects than performing random actions. Second, we use this hierarchy to learn task-specific behavior by estimating how controllable effects would affect the reward function. CEHRL is composed of: a Variational Autoencoder (VAE) (Kingma and Welling, 2013) that approximates a distribution of discovered controllable effects; a task policy that uses this generative model to decide which effect should be performed next. This policy performs random effect exploration to learn in a task-agnostic setting, and trains a state-effect value function to learn task-specific behavior from a reward function using DQN with prioritized experience replay (Schaul et al., 2015). Finally, an effect-conditioned policy is trained to translate effects into actions using DQN with prioritized experience replay and HER-like (Andrychowicz et al., 2017) data augmentation.

Our main contributions are:

- A procedure to disentangle controllable effects from other dynamics in the environment.
- A task-agnostic method to continually learn a hierarchy of effects via random effect exploration.
- An algorithm that uses this hierarchy to rapidly learn task-specific behavior from rewards.

2 Disentangling causal effects

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"*. For example, if the life of an agent relies on eating food, eating food has a causal effect on the agent's life. Actual causality proposed in Halpern (2016) studies causal relations between individual events of X and Y. In this case, we would like to know if a particular action a would have an effect on the agent's life at a given state.

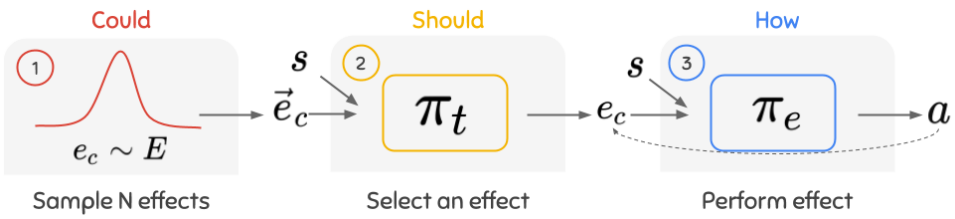


Figure 2: CEHRL is composed of three components: a learned distribution of controllable effects that models what **could** be done on the environment, a task policy that decides which effect **should** be performed next; and an effect-conditioned policy that decides **how** to perform an effect. Note that different choices of π_t can promote exploratory or task-specific behavior.

Figure 3 shows the causal graph of a typical RL setting, where a state s has a causal effect on both the agent's choice of action and the next state s'. Similarly, an action has an effect on the next state. Our goal is to unravel these causal relations to explore the environment and learn behavior from a reward function. We assume that $s \in \mathcal{S}$ is a p-dimensional vector and $a \in \mathcal{A}$ a discrete action. We define the total effect $e_t \in \mathbb{R}^p$ as the change in the environment's state when transitioning from s to s' due to taking action a i.e. $e_t(s,a) \equiv s' - s$. We disentangle total effects into controllable effects $e_c \in \{0,1\}^p$ caused by the agent's action; and dynamics effects $e_d \in \{0,1\}^p$ caused by other dynamics in the environment. Our goal is to identify if there was an effect or not, ignoring the magnitude and direction of the effect. Note that controllable effects represent single-step changes on the environment's state, thus effects caused by actions from previous steps pertain to dynamics effects. For example, the effect of pushing a ball is a controllable effect at the first step but the movement during subsequent steps is attributed to the dynamics of the environment.

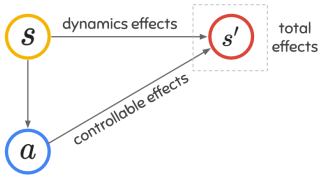


Figure 3: Total effects are disentangled into controllable and dynamics effects.

The causal effect of a variable X on another variable Y can be measured by comparing counterfactual worlds (Pearl, 2009)

$$Y_x \neq Y_{\tilde{x}} \,, \tag{1}$$

where Y_x reads as "what would the value of Y be if X had taken value x". Similarly, $Y_{\tilde{x}}$ describes the value of Y when X does not take the value of x. Intuitively, Eq. 1 compares the world where the event x happened against an alternative world where event x had not happened. Since this last world does not exist, it needs to be imagined. Halpern and Hitchcock (2014) propose to compare what happened with what normally would happen by constructing a normal world and comparing it to the counterfactual world. We can use this formulation to compute the causal effect of an action on the return G at a given state

$$\begin{aligned} e_G(s,a) &= G_a - G_{\tilde{a}} \\ &= Q(s,a) - V(s) \\ &= A(s,a). \end{aligned} \tag{2}$$

G_a is the return the agent would get if action a were to be taken. Usually, G_a is estimated using a state-action value function $Q(s,a)$ and the choice of normality for $G_{\tilde{a}}$ is to estimate the expected return with the state-value function $V(s)$. This choice leads us to the advantage function $A(s,a)$ which estimates the causal effect of an action on the return. Note that contrary to Eq. 1, this formulation uses the magnitude and direction of the effect, thus computing the difference between worlds. We give Eq. 1 another use, our goal is to identify what changed in the state due to an action i.e. we want to disentangle controllable effects from total effects

$$e_c(s,a) \equiv \left(e_t\left(s,a\right) \neq e_t\left(s,\tilde{a}\right)\right), \tag{3}$$

where as before, $e_t(s,\tilde{a})$ needs to be imagined. Note that defining $\tilde{a}$ as a special *do-nothing* action would not work since even doing nothing does something e.g. doing nothing when a bullet is flying towards the agent has an effect on the agent's life. Our choice of normality is to compute $e_t(s,\tilde{a})$ as

the most common world among every possible world, i.e. we compute $e_t(s, \tilde{a})$ as the mode over all actions and consequently $e_c(s, a)$ as

$$e_c(s,a) \equiv \left(e_t\left(s,a\right) \neq \underset{a_i \in A}{\text{mode}} \left(e_t\left(s,a_i\right)\right)\right). \tag{4}$$

In practice, we do not have access to every world and cannot compute Eq. 4 directly. We approximate $e_t(s, a)$ using a neural network $\hat{e}_t(s, a)$ trained in a self-supervised manner using experiences (s, a, s') and loss

$$\mathcal{L}_{\hat{e}_t} = \text{MSE}\left(\hat{e}_t\left(s,a\right), \left(s' - s\right)\right). \tag{5}$$

We provide more details of the training of $\hat{e}_t$ in section 3.1. Note that to compute Eq. 4, we need to discretize the output of this network. In our experiments $\mathcal{S} \subset \mathbb{N}^p$, and thus we discretize total effects to its nearest integer when computing Eq. 4 but train $\hat{e}_t$ with its continuous output.

3 CEHRL

We could use controllable effects to choose the next action, for example to take actions that lead to larger effects but this would make the agent myopic, i.e. it would consider changes at a single step of time. Instead, we want to learn temporally abstract effects so as to operate on an extended time horizon. We propose CEHRL (see Figure 2 for an overview of its components), a hierarchical method composed of a generative model E to provide the different ways an agent could change the environment; a task policy $\pi_t(e_c^{\text{goal}}|s, \vec{e}_c)$ that decides what effect to perform next; and an effect-conditioned policy $\pi_e(a|s, e_c^{\text{goal}})$ that learns how to translate an effect into actions. Next, we describe how to use random effect exploration for task-agnostic learning in section 3.1; and outline how to learn task-specific behavior from a reward function in section 3.2.

3.1 Controllable effects for task-agnostic learning

CEHRL uses random effect exploration to continuously discover novel controllable effects for task-agnostic learning. Random effect exploration is based on the idea that by combining controllable effects, an agent can discover novel and more complex effects. This method selects effects from the learned distribution and performs them on the environment. The resulting controllable effects are used to train this distribution further, creating a continuous learning cycle. We divide this process into *acting*, where the agent performs effects on the environment to collect experiences; and *learning*, where these experiences are used for training.

Acting: to perform random effect exploration, we first sample N candidate effects $\vec{e}_c = (e_c^1, \ldots, e_c^N)$ from the distribution of effects E; then the task policy selects one of these effects as goal e_c^{goal}; and finally, we unroll the effect-conditioned policy until either the controllable effect for the current step matches the goal or more than K actions have been performed. We store experiences of the form $(s, e_c^{\text{goal}}, a, s', e_c(s, a), r', d')$ in a prioritized replay memory $\mathcal{D}$ (Schaul et al., 2016). Instead of using the environment's terminal state d, we terminate the episode if either the environment's episode ends, the number of steps trying the goal exceeds a limit K or the goal is reached. We ignore the extrinsic reward provided by the environment and give a reward of 1 if the goal is reached or a punishment P otherwise. Experiences are augmented with a similar method to HER (Andrychowicz et al., 2017). Additionally, we deal with class imbalance between basic and complex effects by using a function $g(e_c)$ to compute the rarity of an effect and set it as initial priority to each experience.

Learning: here we describe CEHRL's components, the data they require and their training.

- *Total effects model:* to be able to compute $e_c(s, a)$ from Eq. 4, we train the network $\hat{e}_t(s, a)$ with the loss defined in Eq. 5 using samples (s, a, s') from the replay memory.
- *Distribution of effects:* we train a Variational Autoencoder (Kingma and Welling, 2013) to approximate the distribution of controllable effects present in the replay memory. This model is not conditioned on state nor actions, making effects reachable from any state.
- *Effect-conditioned policy:* we train a neural network $Q_e(s, e_c^{\text{goal}}, a)$ with experiences from the replay buffer to learn the state-effect-action value function using DQN with prioritized experience replay (Schaul et al., 2015), Double Q-learning (van Hasselt et al., 2015) and Dueling Networks

(Wang et al., 2015); and use this network as effect-conditioned policy

$$\pi_e(a|s, e_c^{\text{goal}}) = \arg\max_{a \in A} Q_e(s, e_c^{\text{goal}}, a). \tag{6}$$

- *Task policy:* to bias exploration as little as possible, our choice of task policy is to choose effects following a uniform distribution

$$\pi_t(e_c^{\text{goal}}|s, \vec{e}_c) = \text{uniform}(\vec{e}_c). \tag{7}$$

To coordinate the training of these networks, we use a scheduling function $f_M : t \mapsto M$ that specifies which models M to train at each training step t. We perform a single update for each model in M using batches sampled from the replay memory. In our experiments, we start by training the total effects model, then we incorporate the distribution of effects to the training, and finally the effect-conditioned policy. See algorithm 1 and the appendix B.1 for more details on the training schedule, hyperparameters and network architectures used in our experiments.

3.2 Controllable effects for learning task-specific behavior

In the previous section we chose to implement the task policy as a uniform sampling over effects so our exploration is as unbiased as possible. Here, we want to learn how to bias the agent so as to maximize a reward function. We do this by learning a Q-value function that estimates the value of state-effect pairs, thus the task policy is implemented as

$$\pi_t(e_c^{\text{goal}}|s, \vec{e}_c) = \arg\max_{e_c \in \vec{e}_c} Q_t(s, e_c). \tag{8}$$

We train $Q_t(s, e_c)$ using DQN and a prioritized replay buffer with experiences of the form $(s, e_c^{\text{goal}}, s', \vec{e'}_c, r', d)$ collected using random effect exploration (acting only) described in section 3.1. Here, d is the terminal state provided by the environment and $\vec{e'}_c$ are the candidate effects produced by the effect distribution for the next state. The effect-conditioned policy and task policy work at different time scales, i.e. an "effect step" is composed of multiple "environment steps", therefore states s and s' are consecutive effect steps but may not be consecutive environment steps. Since rewards are provided at every environment step, we accumulate rewards given between effect steps into r'. In this phase, we use pre-trained frozen models $\hat{e}_t, E$ and π_e learned in the task-agnostic phase. See algorithm 2 and appendix B.2 for more details.

Algorithm 1 Task-agnostic learning

Require: $K; \mathcal{D}$
  Initialize $\hat{e}_t$; E; π_e; $s = s_0$; $t = t_{\text{train}} = 0$; $d' = 1$; $C = \{1\}$
  while keep training **do**
    if $d' = 1$ **then**
      $\epsilon \sim \text{uniform}(C)$
      $\vec{e}_c \sim E$
      $e_c^{\text{goal}} = \pi_t(s, \vec{e}_c)$
    end if

    $a = \text{get\_action}(\pi_e, s, e_c^{\text{goal}}, \epsilon)$
    $s', d = \text{step\_env}(a)$
    $e_c^{\text{step}} = e_c(s, a)$
    $r' = r_{\text{exp}}(s, a, e_c^{\text{goal}})$
    $d' = (d \text{ or } (t > K) \text{ or } h(e_c^{\text{step}}, e_c^{\text{goal}}))$
    add($\mathcal{D}$, $g(e_c^{\text{step}})$, $(s, e_c^{\text{goal}}, a, e_c^{\text{step}}, s', r', d')$)
    add($\mathcal{D}$, $g(e_c^{\text{step}})$, $(s, e_c^{\text{step}}, a, e_c^{\text{step}}, s', r' = 1, d' = 1)$)
    $s = s'$
    $t = (t + 1)(1 - d')$

    if should train **then**
      $M = f_M(t_{\text{train}})$
      $C = f_C(t_{\text{train}}, M)$
      for all $m \in M$ **do**
        $b \sim \mathcal{D}_m$
        priorities $= \text{train}(m, b)$
        update_priority($\mathcal{D}_m$, b, priorities)
      end for
      $t_{\text{train}} = t_{\text{train}} + 1$
    end if
  end while

Algorithm 2 Task-specific learning

Require: $K; \mathcal{D}$
  Load pre-trained $\hat{e}_t$; E and π_e
  Initialize π_t; $s = s_0$; $s_g = s$; $d' = 1$; $r' = 0$; $t = 0$
  while keep training **do**
    if $d' = 1$ **then**
      $\vec{e}_c \sim E$
      if $h(e_c^{\text{step}}, e_c^{\text{goal}})$) **then**
        add($\mathcal{D}$, $(s_g, e_c^{\text{goal}}, e_c(s, a), s, \vec{e}_c, r', d)$)
      end if

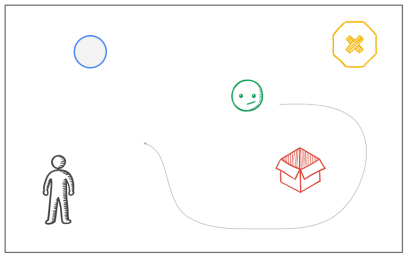
      $r' = 0$
      $e_c^{\text{goal}} = \pi_t(\vec{e}_c, s)$
      $s_g = s$
    end if

    $a = \pi_e(s, e_c^{\text{goal}})$
    $s', r, d = \text{step}(a)$
    $e_c^{\text{step}} = e_c(s, a)$
    $r' = r' + r$
    $d' = (d \text{ or } (t > K) \text{ or } h(e_c^{\text{step}}, e_c^{\text{goal}}))$
    $s = s'$
    $t = (t + 1)(1 - d')$

    if should train **then**
      $b \sim \mathcal{D}$
      priorities $= \text{train}(\pi_t, b)$
      update_priority($\mathcal{D}$, b, priorities)
    end if
  end while

4 Experiments

Our experiments provide empirical answers to the following questions: **a) credit assignment** - can controllable effects ease the task of credit assignment? **b) exploration** - How does exploration with random effects differ from using random actions? **c) cost** - how expensive is to operate with effects? First, we show how using a hierarchy of controllable effects can help with the credit assignment and exploration problems. Secondly, we show how the cost of building this hierarchy is amortized when the number of tasks or their complexity increases.



To answer the above questions, our environment presents multiple objects with different properties and ways of interaction. These objects are: a ball that can be picked and dropped, a chest where the agent can store the ball and a special target location where the agent can step into. Additionally, we introduce a demon that has its own dynamics to help us evaluate if our method can disentangle controllable effects, see appendix A.1 for more experiments on the disentanglement of effects. We provide three tasks where reaching the special target location becomes increasingly difficult:

- *T:* go to the target location.
- *BT:* go to the target location while carrying a ball
- *CBT:* pick ball, put it in the chest, and go to the target.

To implement this environment we use MiniGrid 2D (Chevalier-Boisvert et al., 2018). Agents are provided with discrete, entity-centric representation of the world as in Baker et al. (2019) and give a reward of one discounted by the time it took to solve the task, see appendix C.1 for more details. Although we list the set of possible effects in our results, these effects are not given to the agent, the agent has to learn them. Note that the duration of a single training step is different for each of the components in CEHRL and for the baseline, we report the training time instead of training steps to have a fairer comparison.

4.1 Can controllable effects ease the task of credit assignment?

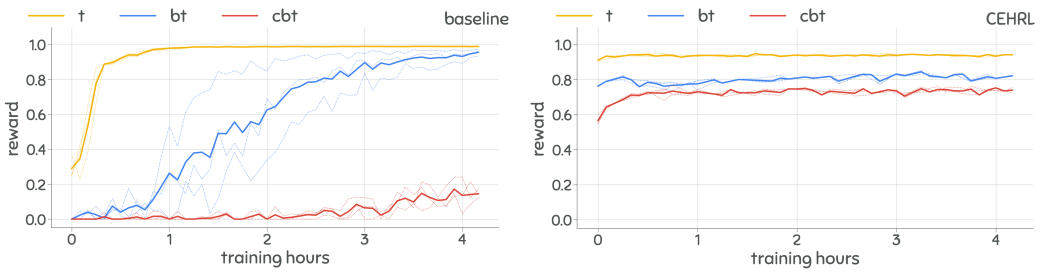
In this experiment, we evaluate if CEHRL can ease the credit assignment problem by using a few effects instead of numerous actions. We use the setting described in section 3.2 and train the task policy on the three variants of the environment using pre-trained models $\hat{e}_t$, E and π_e. We evaluate how CEHRL scales with the complexity of the task by comparing it to a DQN-based baseline.

Figure 4(a) shows how the baseline manages to achieve high reward in the first two tasks but cannot complete the more complex task. Moreover, it requires increasingly more time to complete each task, not scaling well with the task's complexity. On the other hand, CEHRL achieves high reward in the three tasks and needs 30 minutes to complete each task, scaling better with the task's complexity. We conjecture that learning the value of individual actions at every state requires large number of samples, whether using effects as abstractions reduces the space of values to model. Note that since we do not fine-tune π_e further, CEHRL does not achieve optimal reward.

In all three tasks, CEHRL learns to use effects from the top of the hierarchy leading to shorter episodes and consequently larger reward, these results are shown in appendix A.2. These results suggest that using a hierarchy of effects simplifies the credit assignment problem and enables solving long-horizon tasks more efficiently. This is possible due to reusing previously learned effects. Experiment 4.3 analyses the cost of learning these effects.

4.2 How does exploration with random effects differ from using random actions?

To analyze how random effect exploration differs from random action exploration, we record the effects performed by each method during 500K steps. We do not perform any training and use pre-trained models. Note that some effects are more difficult to perform than others, e.g. putting the ball in the chest vs moving forward. We use the following categorization: **basic effects** require usually one or two actions to be performed (these are usually effects on the agent), **simple effects**



(a) baseline (b) CEHRL

Figure 4: **a)** Reward achieved by the baseline in the three variants of our environment. **b)** Reward achieved by CEHRL. By using already learned effects, CEHRL can scale better with the task complexity. Bold lines denote mean reward over three seeds while dotted lines indicate each individual seed.

require more than one basic effect and **complex effects** require multiple simple effects. This choice is arbitrary but helps us better understand what each method is exploring.

An unbiased exploration method would perform effects as uniformly as possible. Note that a perfectly uniform exploration is not possible due to the compositional nature of effects i.e. to perform a complex effect the agent needs to perform multiple simple or basic effects. The results in Figure 5(a) show that random effect exploration performs simple and complex effects almost three times more often than with random actions. The effect "nothing" is not needed to perform other effects, therefore it happens less often. In contrast, other basic effects are central to simpler effects.

4.3 How expensive is to operate with effects?

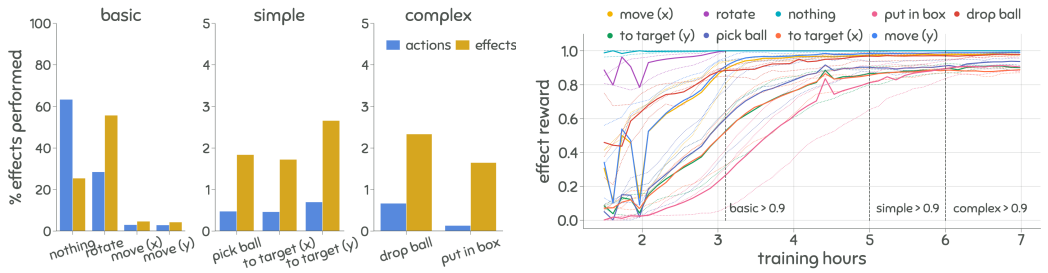
We have shown that exploration and credit assignment benefit from operating with effects. Here, we study the cost of creating this hierarchy using the method described in section 3.1. For this, we train models $\hat{e}_t$, E and π_e and record the reward achieved by each individual effect.

Figure 5(b) shows that CEHRL can perform every basic effect (reward higher than 0.9) after three hours of training and that simple and complex effects take five and six hours respectively. Comparing these results with experiment 4.1, we can see that CEHRL learns to perform every effect in six hours but takes five hours for the baseline to learn tasks *T* and *BT*, and cannot learn task *CBT*. These results indicate that the cost of learning a hierarchy of effects is quickly amortized when the number of tasks or their complexity increases. Moreover, using effects enables the learning of tasks that the baseline struggles to learn. We conjecture that this is due to the implicit curriculum of learning causal effects, i.e. the learning of simpler effects makes easier to learn complex effects. As with animals, the more rare an effect is, the harder it is to learn. It is important that their complexity increase gradually. Note that there is 1.5 hours of warmup where the total effects and distribution models are trained but the effect-conditioned policy is not; see supplementary materials for more training details.

5 Related work

Intrinsic rewards: a popular solution to the exploration and credit assignment problems is to use intrinsic rewards (Singh et al., 2005; Pathak et al., 2017; Burda et al., 2018; Song et al., 2019; Choi et al., 2019; Badia et al., 2020). This approach promotes exploratory behavior by rewarding curiosity. Burda et al. (2018) use an untrained neural network to estimate surprise and reward for it. Choi et al. (2019) and Badia et al. (2020) reward for the discovery of controllable areas. We consider these methods complementary to our work and can be incorporated to CEHRL.

Hierarchical RL: Dayan and Hinton (1993) proposed Feudal RL where a hierarchy of managers work at different granularity by controlling the information and rewards transferred to lower levels of the hierarchy. Vezhnevets et al. (2017) gave a deep learning implementation of this framework where a worker performs actions to accomplish the goal set by a manager. Sukhbaatar et al. (2018) on the other hand, divided the training of the manager and worker into exploration and task-dependent



(a) random effect vs random action exploration

(b) effects cost

Figure 5: **a)** Comparison between random action and random effect exploration. Note that basic effects are needed to perform simple and complex effects. **b)** Time taken to learn each effect. Each vertical line indicates when a group of effects reached a reward of 0.9.

stages. These methods rely on extrinsic rewards to learn representations between levels making them ill-suited for environments with sparse rewards.

The options framework proposed by Sutton et al. (1999) adopts a more decentralized approach to temporal abstraction. Each option represents a skill and autonomously decides if it should be started or finished based on the current state. A well-known problem is to discover useful skills (Bacon et al., 2016; Florensa et al., 2017; Eysenbach et al., 2018; Nair et al., 2018; Held et al., 2018; Jegorova et al., 2018; Sharma et al., 2019). A common practice is to optimize the mutual information between skills to make them easily distinguishable and diverse. Instead, we consider every way an agent can modify the environment as a potentially useful skill.

Generative RL: generative models like Variational Autoencoders (VAE) or Generative Adversarial Networks (GAN) have proven to be extremely useful in image or audio generation. Nonetheless, these methods are gaining popularity in RL (Held et al., 2018; Nair et al., 2018; Nair and Finn, 2019). Nair and Finn (2019) used a VAE to produce intermediate goal-images that facilitate the planning of actions to achieve a final goal. Using images as goals in environments with dynamics makes difficult to decide whether a goal has been achieved. Instead, CEHRL removes dynamic effects, only using controllable effects. Additionally, using effects instead of states reduce the size of the distribution to model, e.g. in a 1D infinite environment a generator that works with states would need to produce infinite states to achieve the effect of moving forward. In contrast, CEHRL would generate the same effect over and over. Similarly, Held et al. (2018) used GANs to produce a curriculum of goals to ease the learning of a specific task.

Causality: Incorporating concepts from causality to deep learning in general (Bengio et al., 2019; Chattopadhyay et al., 2019; Ke et al., 2019) and reinforcement learning in particular (Buesing et al., 2018; Jaques et al., 2018; Dasgupta et al., 2019; Goyal et al., 2019; Nair et al., 2019) has shown to be an important research avenue that can benefit a wide range of tasks. Badia et al. (2020) use an inverse model to identify controllable aspects of the environment. Although efficient, this approach is local to the agent; ignoring changes faraway from the agent. By using counterfactual measures we identify a broad set of controllable effects.

6 Conclusion and future work

We presented CEHRL, a hierarchical method that leverages causal tools to continuously learn a hierarchy of controllable effects by performing random effect exploration. We showed that this hierarchy can be used to efficiently learn task-specific behavior in a scalable manner. Our experiments also show that the cost of building this hierarchy can be quickly amortized when either the number of tasks or their complexity increases. To avoid confounding our experiments, CEHRL has been applied to entity-centric representations. In future work, we want to incorporate methods for unsupervised state representation learning (Burgess et al., 2019; Anand et al., 2019). Additionally, to avoid biasing exploration the task policy for exploration uniformly samples effects but we could introduce intrinsic motivators by giving different priorities to each effect. Another interesting avenue is to use the hierarchy of effects as a temporally abstract model of the world by conditioning it to the state, so only

effects achievable in certain number of steps are selected. The use of effects ought to be explored in multi-agent RL for: coordination between agents with dissimilar action spaces; teacher-student imitation learning; or to create social norms by extending counterfactual and normality concepts.

Acknowledgments

The authors would like to thank Jaan Aru, Daniel Majoral, Ardi Tampuu, Tambet Matiisen and Meelis Kull for insightful comments on the manuscript. This work was supported by the AWS Cloud Credits for Research program and by the University of Tartu ASTRA Project PER ASPERA, financed by the European Regional Development Fund.

References

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

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *CoRR*, abs/1707.01495, 2017.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture, 2016.

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, 2020.

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

Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms, 2019.

Daphna Buchsbaum, Sophie Bridgers, Deena Skolnick Weisberg, and Alison Gopnik. The power of possibility: Causal learning, counterfactual reasoning, and pretend play. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1599):2202–2212, 2012. ISSN 14712970.

Daphna Buchsbaum, Thomas L. Griffiths, Dillon Plunkett, Alison Gopnik, and Dare Baldwin. Inferring action structure and causal relationships in continuous sequences of human action. *Cognitive Psychology*, 76:30 – 77, 2015. ISSN 0010-0285. doi: https://doi.org/10.1016/j.cogpsych.2014.10.001.

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.

Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. MONet: Unsupervised Scene Decomposition and Representation. 2019.

Aditya Chattopadhyay, Piyushi Manupriya, Anirban Sarkar, and Vineeth N Balasubramanian. Neural Network Attributions: A Causal Perspective. 2019.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

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.

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.

Peter Dayan and Geoffrey Hinton. Feudal Reinforcement Learning. *Advances in neural information processing systems*, pages 271–278, 1993. ISSN 1049-5258.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is All You Need: Learning Skills without a Reward Function. 2018.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic Neural Networks for Hierarchical Reinforcement Learning. pages 1–17, 2017.

Noah D Goodman, Vikash K Mansinghka, and Joshua B Tenenbaum. Learning grounded causal models. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 29, 2007.

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.

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.

David Held, Xinyang Geng, Carlos Florensa, and Pieter Abbeel. Automatic Goal Generation for Reinforcement Learning Agents. *35th International Conference on Machine Learning, ICML 2018*, 4:2458–2471, 2018.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. *CoRR*, abs/1803.00933, 2018.

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.

Marija Jegorova, Stephane Doncieux, and Timothy Hospedales. Behavioural Repertoire via Generative Adversarial Policy Networks. 2018.

Nan Rosemary Ke, Olexa Bilaniuk, Anirudh Goyal, Stefan Bauer, Hugo Larochelle, Chris Pal, and Yoshua Bengio. Learning neural causal models from unknown interventions, 2019.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

Tejas D. Kulkarni, Karthik R. Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning, 2018.

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual Reinforcement Learning with Imagined Goals. 2018.

Suraj Nair and Chelsea Finn. Hierarchical Foresight: Self-Supervised Learning of Long-Horizon Tasks via Visual Subgoal Generation. (1):1–16, 2019.

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

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2015.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-Aware Unsupervised Discovery of Skills. 2019.

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, and Mai Xu. Mega-Reward: Achieving Human-Level Play without Extrinsic Rewards. Technical report, 2019.

Sainbayar Sukhbaatar, Emily Denton, Arthur Szlam, and Rob Fergus. Technical report, 2018.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 1(1999):181–211, 1999. ISSN 1098-6596. doi: 10.1017/CBO9781107415324.004.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. FeUdal Networks for Hierarchical Reinforcement Learning. 2017. ISSN 1938-7228.

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

A Additional experiments

We provide extra experiments to show specific characteristics of working with effects.

A.1 Should we disentangle controllable effects from total effects?

Throughout this work, we have proposed to disentangle controllable effects from total effects i.e. we deliberately ignore effects due to the dynamics of the environment. The intuition behind this choice is that controllable effects reduce the effect space to model i.e. they cluster total effects that are controlled by the agent in the same way together. Disentangling effects is costly, thus it must be worth doing.

This experiment compares how controllable effects differ from total effects when learning to pick up a ball. The main difference between total and controllable effects is the changes due to the dynamics of the environment. Thus, in this environment we increase the number of demons and the complexity of their dynamics to measure if controllable provide a benefit by ignoring the dynamic part of the state. We define three different dynamics: **static** where the demon does not move; **horizontal** where demons only move in a horizontal line; and **circular** where the demon describes a circular movement in the arena. Note that since controllable effects are binary vectors, we use $\mathbb{1}\{e_t \neq 0\}$ as total effects for a fairer comparison. Additionally, we do not train the total effects model when using total effects. We report the average number of actions taken to pick the ball. Since there are many total effects that map to a single controllable effect for picking up the ball, we report the average among all of them.

The first row in Figure A.1 shows that in the absence of dynamics, total effects learn more efficiently than controllable effects. There is an overhead to train a forward model that is not perfect when using controllable effects. By comparing the use of controllable against total effects column wise, we can see that the performance when using total effects decreases the more demons there are. On the other hand, the agent using controllable effects has similar performance no matter the number of demons used. Furthermore, the agent using total effects does not learn how to pick the ball in the horizontal and circular variants with neither two and three demons. In contrast, second and third rows show that the increase in the complexity of the dynamics does not impact the performance of either agent significantly.

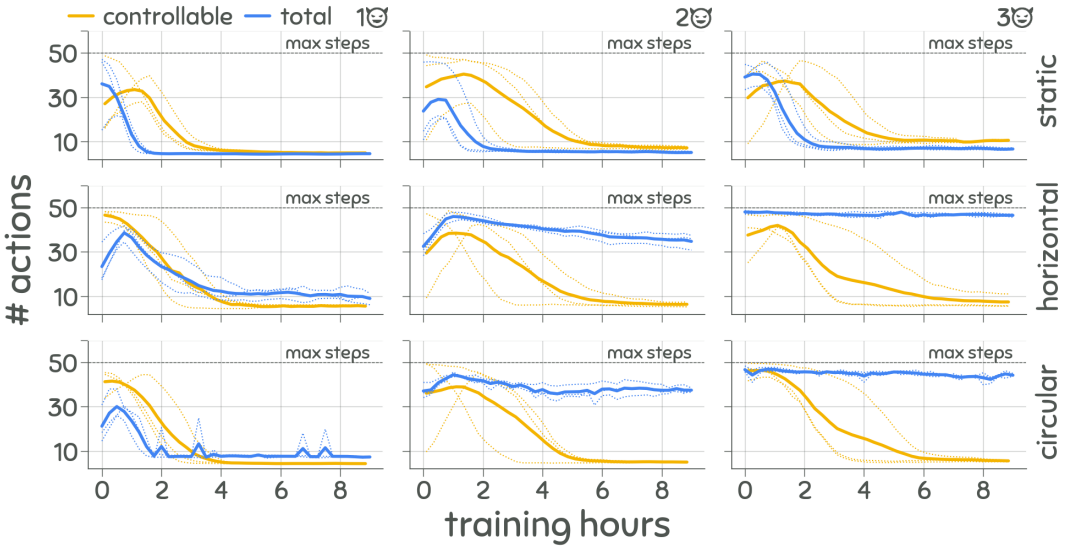


Figure 6: Comparison of the number of actions needed to pick a ball when using total effects or controllable effects. Each column increases the number of demons from 1 to 3 and each row increases the complexity of their dynamics. Each effect is attempted for $K = 50$ steps. The more dynamic objects, the harder to learn this effect using total effects. Controllable effects are robust to these changes since they ignore dynamics.

A.2 How does the task policy use the hierarchy of effects?

This experiment evaluates how the task policy uses the hierarchy of effects i.e. does it use abstract effects or does it use effects closer to actions? For this, we use the variant "T" of the environment where the agent needs to go to a target location. This variant can be solved in multiple ways, for example, the task policy could have learned to use only basic effects to change the x or y coordinates of the agent one step at a time; it could use more complex effects like go-to-target; or a mix of these two. Ideally, the use of higher levels of the hierarchy is preferred since these provide more abstract effects. An obvious benefit of this is when transferring such a hierarchy to another agent with different set of actions or when finding a better effect-conditioned policy, the task policy would not need to be relearned. The results in Figure 7 show how the policy learns to alternate between the

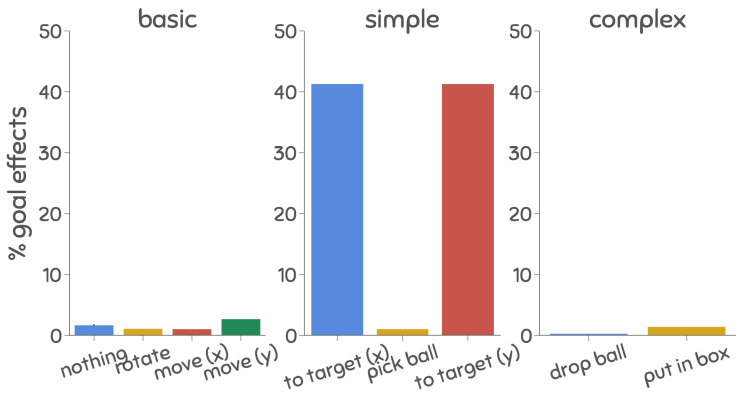


Figure 7: By comparing the three panels, the task policy learns to solve task "T" using abstract effects instead of low-level effects.

two high-level effects that lead to complete the task. By working at a higher time scale (as described in section 3.2 of the paper), together with the environment's reward function, the agent is motivated to use more abstract effects. An alternative to using the environment's reward is to motivate the agent with intrinsic rewards to promote using less number of effects to complete a task.

B Implementation

Here we describe more details related to the training and random effect exploration of CEHRL and how they are used for task-agnostic and task-specific learning.

B.1 Implementation details of task-agnostic learning

In this section, we provide additional details to CEHRL's training for task-agnostic learning. To decide when a goal is reached, we define a function $h(e_c, e_c^{\text{goal}})$ that compares if two controllable effects match at each dimension

$$h(e_c, e_c^{\text{goal}}) \equiv \mathbb{1}\left\{e_c = e_c^{\text{goal}}\right\}. \tag{9}$$

Dealing with effect imbalance: The replay memory is dominated by basic effects e.g. moving forward happens more often than turning the light on. This makes rare effects difficult to learn. To alleviate this imbalance, each experience is added to the buffer with an initial priority proportional to how rare an effect is

$$g(e_c) = \min\left\{10^5, \frac{1}{p(e_c|\mathcal{D}) + \epsilon}\right\}, \tag{10}$$

where $p(e_c|\mathcal{D})$ is approximated by a running average of the seen effects, and the final value is capped.

Shared data, individual priorities: Every component shares the same experiences stored in the replay memory $\mathcal{D}$ but we use an dedicated priority queue for each component. We use the same initial priority but each component updates its priorities based on their individual errors.

Fixed ϵ-greedy exploration: The effect-conditioned policy produces a sequence of actions to perform an effect on the environment. This policy relies on ϵ-greedy exploration to find the right action sequence. Typically, the exploration rate ϵ decays over time. Instead, we adopt an approach similar to Ape-X (Horgan et al., 2018) where this value is fixed throughout the training. Since CEHRL does not work in a distributed manner, we define a set of candidate epsilons $C = \{\epsilon_1, \dots, \epsilon_L\}$ and fix one randomly for each episode. Additionally, we use a warmup period where $C = \{1\}$. This period is used during the initial training of the total effects model and the distribution of effects, see Table 1 for more details. We use a function $f_C : t, M \to C$ that decides which epsilons to use based on the current training step and the set of models to be trained.

Hindsight learning: To speed up the learning of effects, we augment the collected data in a similar way to Hindsight Experience Replay (HER) by Andrychowicz et al. (2017). Every experience $(s, e_c^{\text{goal}}, a, s', e_c(s,a), r', d')$ is augmented with an additional experience where the goal effect has been replaced with the reached controllable effect i.e. $(s, e_c(s,a), a, s', e_c(s,a), r' = 1, d' = 1)$.

B.2 Implementation details of task-specific learning

The task policy is trained by performing random effect exploration. This method sets controllable effects as goals where an effect-conditioned policy tries to perform them on the environment. Unfortunately, this last policy will not be perfect and may fail to perform some goals, thus we do not add them to the replay buffer for training.

In contrast to fixed ϵ-greedy, we use the usual ϵ-greedy decay to explore different goal sequences before relying only on the learned value function. Note that even with $\epsilon = 0$ CEHRL still explores the environment, this is possible thanks to random effect exploration.

C Experimental setup

Here we specify the architectures used for CEHRL and the baseline. We also provide the hyperparameters used in our experiments and the search range of hyperparameters done. Furthermore, we provide more specifics on the environment used.

C.1 Environment

The environment is implemented using MiniGrid 2D by Chevalier-Boisvert et al. (2018). The agent can move forward, turn left or right, pick up different objects, and put objects into boxes. The variants T, BT, and CBT of the environment have all the same number of objects but the task rewarded for is different. Agents are provided with a discrete, entity-centric representation of the world as in Baker et al. (2019). This representation is a $J \times I$ matrix with the J objects in the arena (including the agent) and the following I attributes per object: type of object, x and y coordinates, color for objects or direction for the agent, and the carrying object. If the agent achieves a task, the environment gives a reward using the following function

$$r = \begin{cases} 1 - \frac{0.9 * t}{T} & \text{solved} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where t is the current time step and T is the maximum time steps per episode.

C.2 Architecture

We implement every network as an MLP with fully connected layers and ReLU activation functions.

State encoder: We use a network common to other components to encode the state. Each attribute type in the state is encoded into the same embedding space. We concatenate the resulting embeddings, and process them with a two fully connected layers to create a reduced latent space. See Figure 8.

Effect encoder: Similarly, the effect encoder creates a low-dimensional latent space by using the effect as input of two fully connected layers. See Figure 9.

Total effects model: To predict the dynamics of the environment, we provide this component with two consecutive states. We implement this component with two state encoders to encode each

consecutive state into a latent vector. These two latent vectors are then processed by four fully connected layers to output the predicted total effects for each of the actions. See Figure 10.

Distribution of effects: This component is implemented as a VAE with four fully connected layers to encode controllable effects into a latent space and three fully connected layers to decode latent vectors to controllable effects. Note that the last layer of the decoder has a Sigmoid activation function so as to output a binary vector. Consequently, we replace the typical mean-squared-error reconstruction loss of the VAE with a binary cross-entropy loss. See Figure 11.

Effect-conditioned Q-value function: This component uses state and effect encoders to create a latent representation of its input. This latent representation is then passed to a four fully connected layer network with an additional Siamese layer to compute state-value and advantage functions, as used in dueling networks. See Figure 12.

Task Q-value function: This is the same network as the effect-conditioned Q-value function but without the Siamese network since we do not have a limited set of controllable effects. See Figure 13.

Baseline: We use the same network as the effect-conditioned Q-value network but without the effect encoder, and train it using DQN with prioritized experience replay, double DQN and dueling networks. This baseline uses the fixed ϵ-greedy exploration described above. Since the baseline is composed of only a policy, we do not do any warmup phase.

C.3 Hyperparameters

Table 1 provides the training schedule used to implement function f_M and f_C. Hyperparameters common to every training and models are shown in Table 2. Hyperparameters for task-agnostic learning are shown in Table 3. Hyperparameters for task-specific learning are shown in Table 4. Hyperparameters for the baseline are shown in Table 5. Note that each table also provides the search range we used to fix each hyperparameter, if applicable.

Models	Steps	Use warmup C
$\{\hat{e}_t\}$	[90000, 0]	True
$\{\hat{e}_t, E\}$	[30000, 0]	True
$\{\hat{e}_t, \pi_e\}$	[30000, 0]	False
$\{\hat{e}_t, E\}$	[10000]	False
$\{\pi_e\}$	[60000]	False

Table 1: Training schedule used by f_M and f_C. Every X steps the function f_M selects the next row in the scheduling table. Once the function reaches the end, it starts again from the first row on the second (or last) Steps field.

Hyperparameter	Value	Search range
Batch size	128	(32,64,128,256)
Priority replay buffer α	1.0	N/A
Priority replay buffer β	0.01	N/A
Discount factor	0.85	(0.8,0.85,0.9,0.95,0.99)
Training frequency	5	N/A
N	20	N/A
P	-0.02	N/A
K	50	N/A

Table 2: Common hyperparameters.

Hyperparameter	Value	Search range
Task-agnostic replay capacity	500K	(200K,500K,1M)
C[warmup]	{1}	N/A
C	{0.8, 0.6, 0.4, 0.2, 0.1, 0.01}	N/A
State encoder units	128	(64,128,256)
State encoder latent	32	(8,12,16,32)
Effect encoder units	256	(64,128,256)
Effect encoder latent	12	(8,12,16,32)
Q_e units	512	(64,128,256,512,1024)
Q_e learning rate	0.0001	(1e-3,5e-4,1e-4,5e-5,1e-5)
Q_e target update	15K	(1K,5K,10K,15K,20K)
E encoder units	(256-128-64)	(512,256,128)
E decoder units	(64-128-256)	(512,256,128)
E latent	8	(4,8,16,32)
E learning rate	0.001	(5e-3,1e-3,5e-4,1e-4,5e-5)
$\hat{e}_t$ learning rate	0.0005	(1e-3,5e-4,1e-4)
$\hat{e}_t$ units	32	(32,64,128,256)

Table 3: Hyperparameters for task-agnostic training.

Hyperparameter	Value	Search range
Task-specific replay capacity	100K	N/A
Q_t units	32	(8,16,32,64,128,256)
Q_t learning rate	0.001	(1e-3,1e-4,5e-4,5e-5)
Q_t target update	2K	(1K,2K,5K,10K,15K)
Q_t epsilon start	1.0	N/A
Q_t epsilon end	0.0	N/A
Q_t epsilon steps	50K	N/A

Table 4: Hyperparameters for task-specific training.

Hyperparameter	Value	Search range
Replay capacity	500K	N/A
C	{0.8, 0.6, 0.4, 0.2, 0.1, 0.01}	N/A
State encoder units	256	N/A
State encoder latent	12	N/A
Effect encoder units	256	N/A
Effect encoder latent	12	N/A
Q_b units	256	(32,64,128,256,512)
Q_b learning rate	0.00005	(1e-3,1e-4,5e-4,5e-5)
Q_b target update	15000	(1K,5K,10K,15K,20K)

Table 5: Hyperparameters for the baseline.

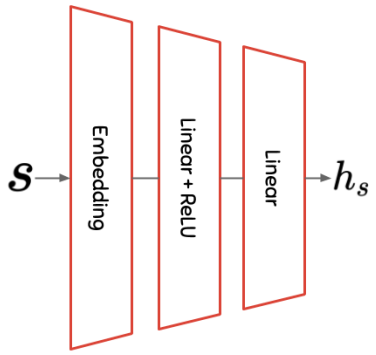


Figure 8: State encoder architecture

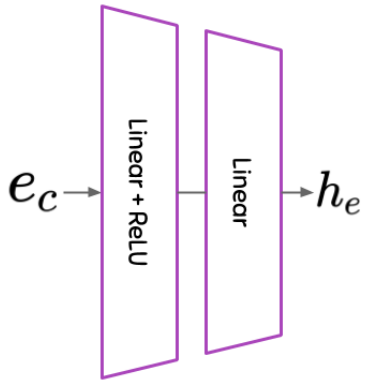


Figure 9: Effect encoder architecture

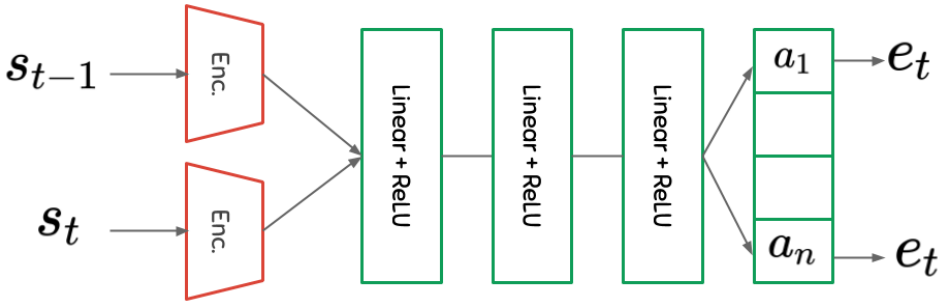


Figure 10: Total effects model architecture

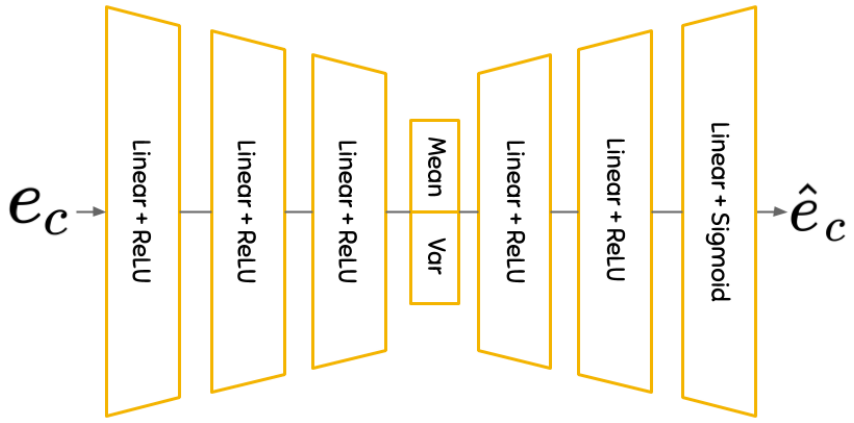


Figure 11: VAE architecture

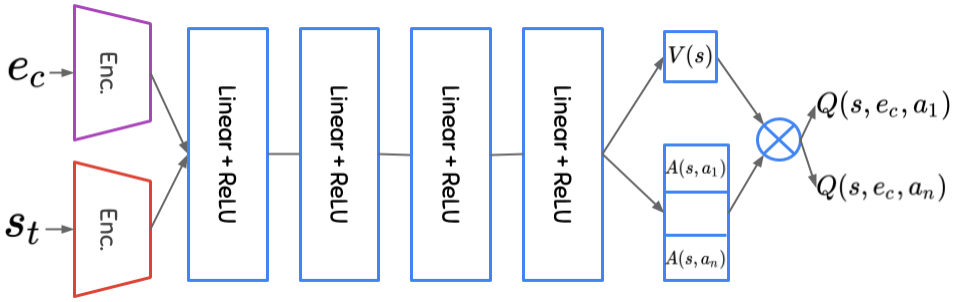


Figure 12: Effect-conditioned policy's q-value architecture

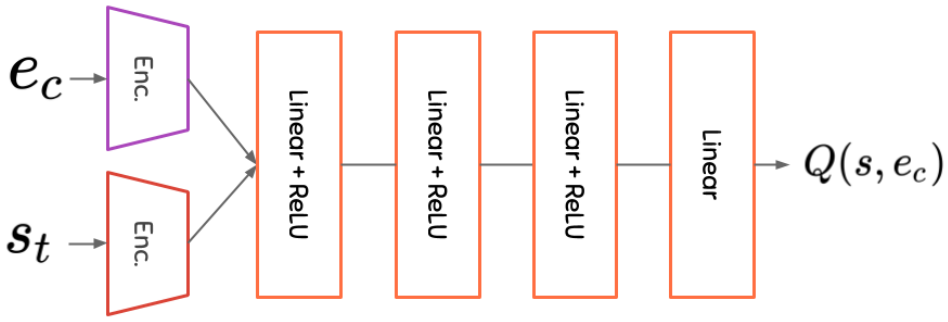


Figure 13: Task policy's q-value network architecture for task-specific learning