Proceedings of Machine Learning Research 148:236–254, 2021 NeurIPS 2020 Preregistration Workshop

Context-Adaptive Reinforcement Learning using Unsupervised Learning of Context Variables

Hamid Eghbal-zadeh[†] HAMID.EGHBAL-ZADEH@JKU.AT
LIT AI Lab & Institute of Computational Perception, Johannes Kepler University, Linz, Austria
Florian Henkel[†] FLORIAN.HENKEL@JKU.AT
Institute of Computational Perception, Johannes Kepler University, Linz, Austria
Gerhard Widmer GERHARD.WIDMER@JKU.AT
LIT AI Lab & Institute of Computational Perception, Johannes Kepler University, Linz, Austria

Abstract

In Reinforcement Learning (RL), changes in the context often cause a distributional change in the observations of the environment, requiring the agent to adapt to this change. For example, when a new user interacts with a system, the system has to adapt to the needs of the user, which might differ based on the user's characteristics that are often not observable. In this Contextual Reinforcement Learning (CRL) setting, the agent has to not only recognise and adapt to a context, but also remember previous ones. However, often in CRL the context is unknown, hence a supervised approach for learning to predict the context is not feasible. In this paper, we introduce **C**ontext-**A**daptive **R**einforcement **L**earning **A**gent (CARLA), that is capable of learning context variables in an unsupervised manner, and can adapt the policy to the current context. We provide a hypothesis based on the generative process that explains how the context variable relates to the states and observations of an environment. Further, we propose an experimental protocol to test and validate our hypothesis; and compare the performance of the proposed approach with other methods in a CRL environment. Finally, we provide empirical results in support of our hypothesis, demonstrating the effectiveness of CARLA in tackling CRL.

Keywords: Contextual Reinforcement Learning, Disentangled Representations

1. Introduction

In Reinforcement Learning, an agent interacts with an environment through receiving observations, executing actions, and receiving rewards. The goal of the agent is to maximise the cumulative reward that is defined based on the task at hand. In some scenarios however, the behaviour of the environment as well as the distribution of the observations can change over time. Under certain conditions, the change in the observation distribution is caused by some variability that changes the *context* of the environment. Therefore, a change in context affects the distribution of the environment's observations. As such changes may occur numerous times, not only does the agent have to adapt to the new contexts, but it also has to *remember* the previous ones. This problem is known as Contextual Reinforcement Learning (CRL).

As an example, consider a setting where users interact with a website, and the goal of the website is to adapt to the user's needs, which might change depending on the current

[†] Equal contribution.

user. However, the behaviour of the user – the environment – is actually affected by some *unobserved* parameters such as age and gender. If the goal of the agent – the website – is to adapt to the needs of the user, it is often helpful to be able to infer the user's characteristics and adapt to them. Another example is a robot that sees the world through a camera, where the time of the day (day/night) or the surrounding location can affect how the robot perceives its environment. Hence, it is crucial that an agent can detect a context, and be able to adapt to it.

Several approaches have been proposed to address this problem. For example, models that can adapt to the changes of the environment by having better exploration strategies (Gregor et al., 2016; Pathak et al., 2019), have been used to tackle environments with changing dynamics. As the context changes, the exploitation of the current policy is no longer as effective, and the agent's previous policy will no longer be suitable to tackle the changes in the environment. Hence, the agent needs to explore new observations, in order to accumulate more reward. Another approach to adapt to new contexts, is to use options in a hierarchical reinforcement setting, where a meta-policy switches between a set of available policies (Achiam et al., 2018; Eysenbach et al., 2018). Hallak et al. (2015) define a Contextual Markov Decision Process (CMDP), as a constrained Partially Observable Markov Decision Process (POMDP), where each context is parameterised as an MDP. In this setting, they propose a solution to tackle CRL assuming a fixed observation space over different contexts, and the agent picking a suitable policy, given the available context. In contrast to the Contextual MDPs as a special case of POMDPs, Jiang et al. (2017) propose a generalisation of MDPs and POMDPs known as Contextual Decision Processes (CDPs), where there is a general context space that the observations are drawn from. Although this formulation is quite general, this work focuses on problems with low Bellman ranks, which corresponds to MDPs with low-rank transition matrix, or small observation space.

In this paper, we provide a definition for Contextual Reinforcement Learning that assumes changing the context, affects the distribution of the states of the environment, resulting in a change in the distribution of the observations. Our definition is motivated by the generative process in a contextual world, where the context variables affect the states of the generative model of the world. Given this definition, we provide a solution using unsupervised learning of the context variable that allows for a better adaptation of the policy based on the context. More generally, in this work we are trying to answer the following questions:

1. Does knowing the context variable help the policy to better adapt to different contexts?

2. What characteristics does a predictive model need to predict context from observations?

3. Can our learnt context variable help the policy to better adapt to different contexts?

In order to answer these questions, we conduct a set of experiments to test the performance of an agent with and without knowing the context variable. Additionally, we conduct experiments to investigate whether disentanglement is actually helpful for estimating the context. Further, using our proposed approach, we estimate the context variable in an unsupervised manner, and compare the performance of agents with and without this estimated variable.

2. Related work

Contextual RL: Contextual settings have been mainly explored in Multi-armed bandits (Langford and Zhang, 2007). Hallak et al. (2015) propose contextual MDPs (CMDPs), extending the standard MDP formulation with multiple contexts that change the underlying dynamics. They introduce an algorithm that is able to detect different contexts and optimize the CMDP. However, their work is focused on low-dimensional observation-spaces and, only a small number of fixed contexts is considered. In contrast, our work is proposed for high-dimensional observation-space such as images, and can deal with a variable number of contexts as it incorporates a continuous multivariate context variable. Another work formulates contextual decision processes as a generalization of MDPs and POMDPs (Jiang et al., 2017), where the observations themselves or their history, respectively, form the context. Our approach differs from this formulation by explicitly distinguishing between context and observations, and having a generative view on the observations based on states that depend on a context.

Eysenbach et al. (2018) propose to use mutual information between the context and the observations as a learning signal, and the entropy of the policy over different contexts as a regularisation term to improve exploration, and better adapt to the change of context. This approach assumes the context variable is known to the policy. Achiam et al. (2018) propose VALOR and use a variational auto-encoder (VAE) that first encodes context to trajectory via policy, and subsequently decodes the trajectory back to the initial context using a probabilistic recurrent decoder that assigns high probabilities to trajectories that are unique to a context. Their approach also assumes that the context variable is known, and is used as a supervised signal to train the decoder. A different model-based approach is explained in (Pathak et al., 2019). An ensemble of dynamic models is trained to predict next observation given the current observation and action. The variance over the output of this ensemble is used as intrinsic reward to train the policy. This approach improves the exploration, which is helpful in contextual settings as the agent better adapts to new contexts.

Representation Learning for RL: Recently, several approaches to learn better representations for RL have been proposed. Higgins et al. (2017b) propose DARLA following a two stage learning approach. First, an agent learns disentangled state representations using β-VAEs (Higgins et al., 2017a) from a high dimensional observation space. Second, based on the previously learned disentangled state representation, the agent has to learn a policy to solve a given task. In contrast to our work, the learned disentangled state representations are not explicitly used to infer different contexts, and the policy is directly learned using the disentangled features, while as we will explain, in our work the disentanglement is only used for learning the context variable, and the agent can learn an unconstrained representation from the observations, in addition to the context variable. Similarly, Stooke et al. (2020) propose to decouple representation learning from the RL task. By applying image augmentation (Kostrikov et al., 2020) and a contrastive loss for learning state representations from raw pixel-observations, they are able to outperform end-to-end trained RL agents on various environments. Such a contrastive learning approach was also applied in (Srinivas et al., 2020).

3. Problem definition

In this section, we provide a generative view to Contextual Reinforcement Learning (CRL), and detail the relation between context variables and the states of the environment. Based on this view, we provide a solution for CRL that can automatically recognise the change in the states of the environment, accordingly predict the new context, and adapt the policy to the new context.

3.1. Contextual Reinforcement Learning

A Partially Observable Markov Decision Process (POMDP) is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O})$, with $\mathcal{S}$ being the state space, $\mathcal{A}$ the action space, $\mathcal{P}$ the transition probabilities, and $\mathcal{R}$ the reward function. In this setting, the agent does not directly observe the true states of the environment, but receives observation $o \in \Omega$. This observation is generated from the underlying system state s and the received action a, according to the probability distribution $o \sim \mathcal{O}(o \mid s, a)$.

In this work we consider finite-horizon episodic Contextual POMDPs (CPOMDPs). At the beginning of each episode an agent will encounter a specific POMDP depending on a randomly sampled context $c \in \mathcal{C}$, which we assume to not change over time within an episode. While for regular POMDPs, the goal of an RL agent is to learn a policy $\pi(a \mid o)$ that maximizes the expected cumulative reward, in CPOMDPs the agent has to learn a policy $\pi(a \mid o, c)$ that further depends on a context c.

3.2. A Generative View on Contextual Reinforcement Learning

Generative Process: We assume a generative process is in place such that everything within the environment is happening in a two-step generative process. First, a multivariate latent random variable z is sampled from a distribution $P(z)$, where z corresponds to semantically meaningful factors of variation of the observations (e.g, shape, colour of the objects; density of objects). Second, the observation x is sampled from a conditional distribution $P(x \mid z)$. We assume that the observation space has higher dimensionality than the semantic space, hence, the data space can be explained with substantially lower dimensional and semantically meaningful latent variable z, and is mapped to the high dimensional observation space x.

Generative Process in Contextual Reinforcement Learning: In Contextual Reinforcement Learning, we assume that the environment $\mathrm{E}_z(o_t, a_t)$ generates the next observation o_{t+1}, given the current observation o_t and action a_t, i.e., $o_{t+1} = \mathrm{E}_z(o_t, a_t)$, with z being a variable controlling its statics (e.g, shape or size of objects). In our generative view, the observations of an episode are generated from a generative model $\mathrm{E}_z(o_t, a_t)$ in 3 steps as follows. In the first step, a multivariate latent random variable $c \in \mathcal{C}$ is sampled from a distribution $P(c)$, where c corresponds to a context. In the second step, a multivariate latent random variable z is sampled from a conditional distribution $P(z \mid c)$, where z corresponds to the state of the environment that controls the statics, defining how the environment generates the next observation, given he current observation and action during an episode. In the third step, the next observation o_{t+1} is generated from the environment's generative model $\mathrm{E}_z(o_t, a_t)$.

4. Proposed Approach

In this section, we propose **C**ontext-**A**daptive **R**einforcement **L**earning **A**gent (CARLA), which is capable of adapting to new contexts in an environment, without any supervision or knowledge about the available contexts.

CARLA consists of two parallel networks: a *context network*, and a *representation network*. The context network aims at learning the context variable, while the representation network is aiming at learning a suitable representation from the environment. The output of these two networks are then further feed into the policy network, where an adaptive policy is formed given the environment variables and the context variable. The policy network then adapts the current policy, based on the context variable. A block-diagram of CARLA is provided in Figure 1 (left).

As detailed in Section 3.2, our assumption in the generative process is that the context variables define the statics of the environment, which in turn defines the distribution of the observations within an episode. The aim of the context network is to reverse this process and estimate the context vector given the observations. As shown in Figure 1 (right), it contains two main modules: a feature disentanglement module and a context learning module. The context network first estimates the environment's statics, and further uses it to learn the context variable. This context factor is then feed to the policy network, in order to adapt the policy to the current context.

The feature disentanglement module is an encoder part of a Variational Autoencoder (VAE) (Kingma and Welling, 2014), which is trained with annealing the Kullback Leibler (KL) Divergence term of the Evidence Lower Bound (ELBO). The VAE is trained using random samples drawn from an experience replay buffer (Lin, 1992). The context learning module is trained online given the observations received in each episode, along with the representation network and the policy network by optimizing the RL objective. This module learns upon the disentangled states of the environment, extracted using the feature disentanglement module explained above.

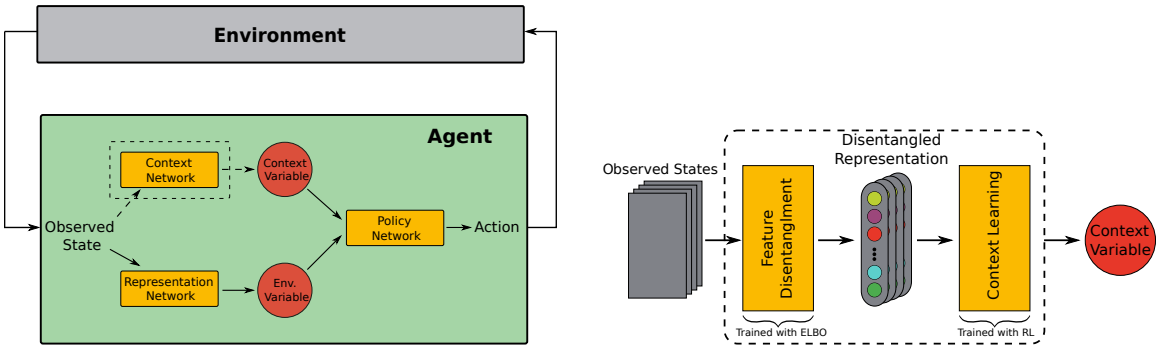


Figure 1: left) Block diagram of CARLA. right) Block diagram of the Context Network. Hidden variables are shown with circles, while processing units are shown with squares.

The graphical models for various approaches in CRL is provided in Figure 2. As can be seen, our model has a different graphical model than DARLA (Higgins et al., 2017b) and

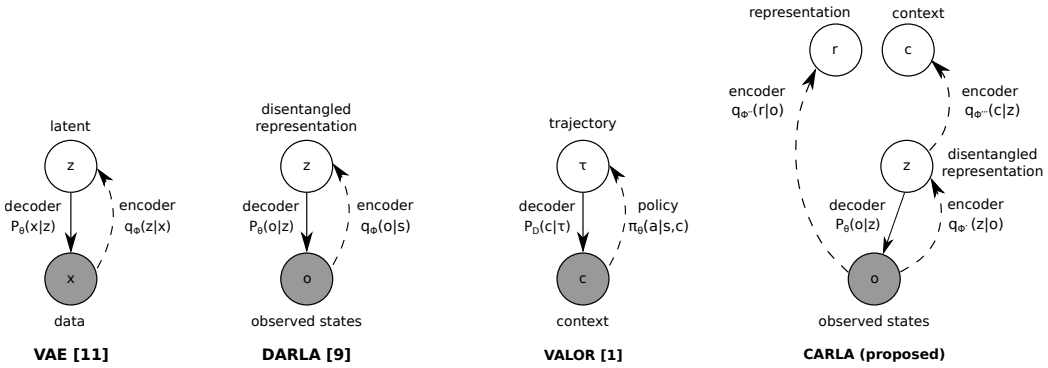


Figure 2: Comparison of the graphical models in different approaches. The solid lines represent generation, and the dashed line represent inference. Gray circles represent observed variables, while white circles represent hidden variables.

VALOR (Achiam et al., 2018). The main idea in CARLA, is to learn disentangled factors from the environment using pre-selected training data, in a similar manner to DARLA. In contrast, CARLA uses a recurrent context network that can build the sequential relationship for the disentangled factors, to predict the context variable, which might be useful in a partially-observable setting to infer the dynamics. Further, CARLA allows the agent to learn an unconstrained representation from the environment during training the agent via interacting with the environment. Although VALOR uses a sequential decoder, it differs from CARLA in various ways. For example VALOR assumes an observed context, in contrast to CARLA which estimates the context variable using a sequence of disentangled factors.

5. Experimental Protocol

In this section, we detail our experimental setup and evaluation strategy, in order to demonstrate the effectiveness of the proposed approach in tackling CRL. In our evaluation, we are testing several hypotheses to answer the following questions:

Does knowing the context help the performance of the agent?
To investigate this, we conduct an experiment testing whether adding a context variable as an additional input to the policy network helps the policy to better adapt to context changes. For evaluation, we will compare the performance in terms of the cumulative reward of a baseline agent that does not use the context information, to an agent that has access to this additional information.

What characteristics does a predictive model need to *predict* context from observations?
In this experiment, we evaluate how well a context can be learned from the observations. Our goal is to determine on the one hand which representations capture the most information about the context in an unsupervised manner and on the other hand which modelling technique (feed-forward vs. sequential) is more suitable to learn the context variable. To this end, we investigate whether feature disentanglement helps in learning the context,

by training unsupervised VAEs. We compare a vanilla VAE (Kingma and Welling, 2014) that does not perform feature disentanglement, to an annealed VAE (Burgess et al., 2018) that incorporates it. Subsequently, given the features extracted by each of the VAEs, we compare a feed-forward classifier to a recurrent one, for learning the context from a single, or a sequence of representations, extracted by the VAEs from observations, respectively. A train-test split on the observations and their respective context label is used to evaluate the generalisation of the context classifiers.

Can our learnt context variable help the policy to better adapt to different contexts?
Finally, to test our full setup, we compare CARLA with the context variable being jointly trained using the RL objective, against two baseline agents in terms of the accumulated reward. The first agent does not have the additional context information, which basically is CARLA without the context network. For the second agent, we remove the representation network and the context learning from CARLA leaving only the feature disentanglement, which is thus similar to DARLA.

For all our experiments, we use a modified dynamic obstacle gridworld environment (Chevalier-Boisvert et al., 2018a,b) as follows. The task of an agent will be to reach a goal position, while collecting and avoiding certain objects. The agent receives a reward of +1 and -1 for *good* and *bad* objects, respectively. Whether an object is good or bad will be based on a certain context, e.g., a specific configuration of differently colored and shaped objects that allow for a clear distinction between contexts. We consider a fully and a partially observable variant of this environment to properly compare feed-forward and recurrent context learning, by either showing the whole grid or a subset. For the VAEs, we use the architecture from (Higgins et al., 2017b). However, as shown in (Burgess et al., 2018), annealing the KL term provides a better disentanglement than the β-VAE, which was used in (Higgins et al., 2017b). Hence, we use the annealing technique proposed in (Burgess et al., 2018) for training the VAE. Similar to (Higgins et al., 2017b), we use an experience replay buffer to draw i.i.d. samples for optimizing the VAE objective. For collecting the observations we follow two different strategies. First, we will store observations that are received by the agent during training in an online fashion. Second, as this might cause the VAE to overfit to its recent experience and not generalize across all possible observations, we will use a different agent to collect the observations that simply avoids all objects and moves around in the world, similar to what is proposed in (Higgins et al., 2017b). To train the RL agents, we use vanilla policy gradient as well as the hyperparameters reported in (Achiam et al., 2018). For the context network, we compare two architectures: a 1-layer LSTM (64 neurons), and a 1-layer MLP (64 neurons), and the policy network is always a 2-layer MLP (64 neurons). For all feed-forward hidden layers with non-linearities, we apply ReLU activation (Nair and Hinton, 2010).

6. Experimental Results

In the following, we first briefly introduce the changes applied to the experimental protocol and provide more details on the used environment as well as the evaluation. Afterwards, we present the results to the questions posed in Section 5.

6.1. Modification to the experimental protocol

Due to computational limitations, the following changes are applied to the experimental protocol. Instead of considering both a partial and a fully-observable setting, we limit ourselves to a fully-observable environment. As a consequence we do not need to investigate the use of a sequential model to learn context variables, allowing us to focus on feed-forward neural networks. Regarding the hyperparameters we had to deviate from some of those reported in (Achiam et al., 2018), which we explain in more detail in Section A.1 in the Appendix.

Additionally, model evaluation will not be based on the cumulative reward only, but we further consider a metric specifically targeted at assessing the object collection-avoidance behaviour of the trained agents.

6.2. Environment and Evaluation Details

As explained in Section 5, we train and test our RL agents on a simplistic contextual gridworld environment, where the task of the agent is to reach a goal position while collecting and avoiding certain objects in the world. In particular, the environment is specifically designed for a contextual setup which requires the agent to infer the current context, and based on that decide what objects should be collected or avoided.[1]

Contexts are define using colour combinations for obstacle and goodie/agent. As shown in Figure 3, for each colour combination there are two available contexts forming a context pair. Overall we use 90 contexts (45 context pairs), where at the beginning of each episode a random context is chosen. The colours used to create the contexts are provided in Table 3 in the Appendix.

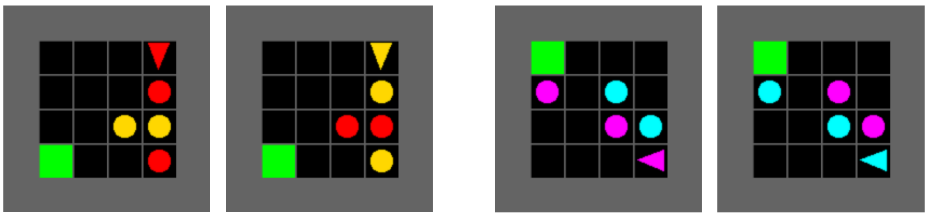


Figure 3: Visualization of two context pairs in our contextual gridworld environment based on (Eghbal-zadeh et al., 2021). The goal is always encoded as a green square and randomly placed in a corner. The context is defined as the colour of the agent (given as a triangle) and the colours are shared across a context pair, i.e., if the agent has a yellow colour in one context it should collect yellow object (given as circles), while avoiding red ones. However, in the second context the agent is red and therefore has to collect red and avoid yellow objects.

Besides using the reward to evaluate the performance of the agents, we separately examine their behaviour in terms of the ability to distinguish between goodies and obstacles as well as to reach the goal. It is important to mention that because we are in a contextual setting, distinguishing between different contexts, especially similar contexts, is of

1. Our environment can be found at: http://eghbalz.github.io/carla/

high importance to the success of the agent. In order to evaluate this important aspect, we introduce additional evaluation measures. In non-contextual RL, these aspects are often ignored, and agents are solely evaluated based on cumulative reward.

Similar to (Eghbal-zadeh et al., 2021), we use a *Goodie/Obstacle Discrimination* (*GOD*) score for each context c:

$$\text{GOD}^c = \max\left(0, OL^c - GL^c\right), \tag{1}$$

where OL^c and GL^c indicate the obstacles and goodies left after an episode ends, averaged across 50 episodes. OL and GL are normalized to $[0, 1]$ by dividing with the number of goodies and obstacles, respectively. Additionally, we consider a *goal reached* GR^c score indicating whether the goal was reached by the agent or not (again averaged over 50 episodes for each context c). This is further extended to a *per context pair* (*PCP*) score ratio for the set of all context pairs $(c_i, c_j) \in C$, by considering only if both contexts in a pair are solved, based on a threshold value t:

$$\text{GOD-PCP} = \frac{1}{|C|} \sum_{(c_i, c_j) \in C} \mathbb{1}_{\min\left(GOD^{c_i},\ GOD^{c_j}\right) \geq t}, \tag{2}$$

and

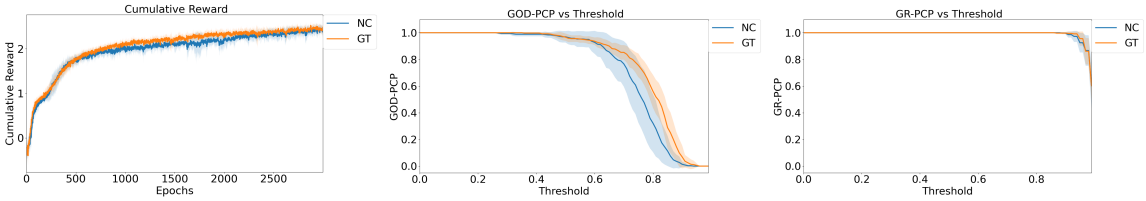
$$\text{GR-PCP} = \frac{1}{|C|} \sum_{(c_i, c_j) \in C} \mathbb{1}_{\min\left(GR^{c_i},\ GR^{c_j}\right) \geq t}, \tag{3}$$

with $\mathbb{1}_{\min(\cdot,\cdot) \geq t}$ yielding 1 if $\min(\cdot, \cdot) \geq t$ and 0 otherwise.

6.3. Results and Discussion

Does knowing the context help the performance of the agent? The first research question we address is whether knowing the context improves the performance of an RL agent. To this end, we train an agent that does not have any contextual information (NC) and one where the policy has an additional input indicating the ground truth context (GT). Figure 4 summarizes the results of this experiment. Comparing the cumulative reward during training in Figure 4(a), we observe that the GT agent is marginally better than NC. However, the difference becomes more evident when we compare the agents in terms of the *goodie-obstacle-discrimination*, as shown in Figure 4(b). As can be seen, the improvement in *goodie-obstacle-discrimination* using GT information is significant, and the agent gains significant abilities in distinguishing between similar contexts, as measured by our metric. These results suggest that having additional access to contextual information can indeed improve the performance of an agent. Since it is not feasible to always provide ground truth information of a context, we next investigate whether contextual information can be inferred in an unsupervised manner.

What characteristics does a predictive model need to *predict* context from observations? In order to evaluate the characteristics of the context-encoders used in our experiments, we conduct a set of experiments to evaluate different VAEs, and choose the best-performing model. We choose 3 VAE models, namely GECO (Rezende and Viola, 2018), Annealed-VAE (Burgess et al., 2018), and Beta-VAE (Higgins et al., 2017a) (with 3 different β values; the VAE with $\beta = 1$ represents the Vanilla-VAE discussed in Section 5).



(a) (b) (c)

Figure 4: Non contextual (NC) agent vs. an agent that is provided with additional ground truth information (GT). Figure 4(a) is the cumulative reward over training epochs, Figure 4(b) is the goodie-obstacle discrimination ratio and Figure 4(c) the goal reached ratio averaged across 8 different random seeds. The shaded area indicates the standard deviation.

In order to compare the VAE models, we conduct two sets of experiments: 1) context classification, and 2) disentanglement evaluation.

For context classification, we first uniformly draw 50k samples from the 90 different contexts used in our environment, then encode them using the VAE's encoder. We then train a classifier on the embeddings of the VAE encoder, using the context labels. Subsequently, the classifier has to predict the context, given an image drawn from the environment. We use a 2-layer MLP, as well as a linear model for classifying context using the VAE embeddings. A random 80%training-20%testing split is incorporated to test the generalisation of the context classifier for each VAE. The results of context classification are provided in Table 1. As can be seen, GECO achieves the best performance among different VAEs.

In order to compare the disentanglement performance of different VAEs, we conduct disentanglement evaluation based on the β-VAE score introduced in (Higgins et al., 2017a). These results can be found in Table 2. It can be again observed that GECO performs best in disentanglement among all other VAEs. Based on these results, we choose GECO as the VAE model, to be used in our proposed agent. The detailed procedure of the disentanglement evaluation alongside with reconstructed examples, reconstruction error and KL divergence of different VAEs is provided in Appendix B.

Can our learnt context variable help the policy to better adapt to different contexts? In Figure 5 we compare our proposed method (*CARLA*) to the non contextual (*NC*) and ground-truth (*GT*) agents from before as well as to an agent that uses a pre-trained VAE to extract state features, as done in *DARLA* (Higgins et al., 2017b). Both *CARLA* and *DARLA* use the same pre-trained VAE which has been trained using GECO. A comparison using Beta and GECO VAE is shown in Figure 7 in the Appendix. For *CARLA*, we additionally analyze two different ways of combining the contextual information: 1) simply concatenating the contextual information to the encoded state, and 2) using the *Feature-wise Linear Modulation* (FiLM) layer (Perez et al., 2018) which we refer to as *CARLA-CG* and *CARLA-FG*, respectively. A discussion about the different ways of combining contextual information is provided in Section A.1 in the Appendix. We also

Table 1: Context classification accuracy on a train-test split for a linear and MLP classifier. Higher is better, best results are marked bold.

VAE	Classifier	Train	Test	Classifier	Train	Test
GECO		**0.7708**	**0.6968**		**0.5481**	**0.5003**
Annealed		0.6906	0.6363		0.3044	0.2581
Beta$^{\beta=10}$	MLP	0.2061	0.1885	Linear	0.0302	0.0227
Beta$^{\beta=2.5}$		0.5141	0.4087		0.1552	0.1121
Beta$^{\beta=1}$		0.7164	0.6798		0.4806	0.4391

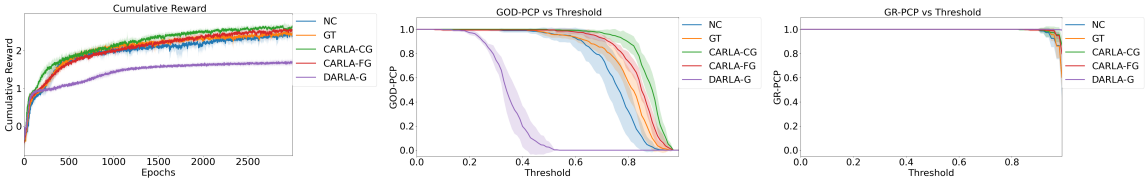
Table 2: Evaluating disentanglement in VAEs. Higher is better, best results are marked bold.

β-VAE Score	Train	Test
GECO	**0.90**	**0.90**
Annealed	0.86	0.87
Beta$^{\beta=10}$	0.54	0.54
Beta$^{\beta=2.5}$	0.82	0.83
Beta$^{\beta=1}$	0.83	0.84

investigated the effect of updating the VAE encoders on the performance of the agents, and provide the results and the discussion in Section C of the Appendix. While all methods seem to be equally good in reaching the goal (as visualized in Figure 5(c)), we see a clear difference in the goodie-obstacle discrimination shown in Figure 5(b). *DARLA*'s performance seems to be the lowest among all methods, in terms of the cumulative reward and GOD, indicating that disentangled state-features alone are not sufficient to solve this task. Both *CARLA* variants outperform the *NC*, *GT*, and *DARLA* agents, with *CARLA-CG* achieving a higher *GOD-PCP* score than *CARLA-FG*.

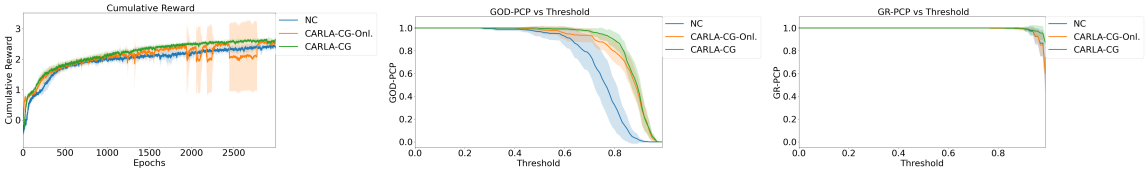
We also investigate an online version of *CARLA*, where the policy and context network (the VAE) are trained from scratch, during training, and no pre-training was used. For that, we maintain a queue of 50k samples from the latest observed states to train the VAE. The VAE is only trained with the VAE objective, and the RL objective updates the state features, policy and value function. A pseudo-code of our *Online CARLA* algorithm can be found in Algorithm 1 in the Appendix. In these experiments, we update the VAE 30 times more than the policy ($m = 30$ in Algorithm 1). These results are provided in Figure 6. As can be seen, *Online CARLA* achieves similar performance to *CARLA* using a pre-trained VAE, outperforming non-contextual agents. We observe an increase in variance in cumulative reward in some epochs, which can be resolved by further adjusting the update frequency of the VAE.

The details of hyperparameters and architectures used in our experiments are provided in Section A in the Appendix.



(a) (b) (c)

Figure 5: Comparison of non contextual (NC), ground truth (GT), DARLA and CARLA agent. DARLA and CARLA use GECO VAE, and for CARLA we use concatenation (CARLA-CG) as well as FiLM (CARLA-FG) to incorporate contextual information. Figure 5(a) is the cumulative reward over training epochs, Figure 5(b) is the goodie-obstacle discrimination ratio and Figure 5(c) the goal reached ratio averaged across 8 different random seeds. The shaded area indicates the standard deviation.



(a) (b) (c)

Figure 6: Comparison of Online and Offline CARLA. Figure 6(a) is the cumulative reward over training epochs, Figure 6(b) is the goodie-obstacle discrimination ratio, and Figure 6(c) the goal reached ratio averaged across 8 different random seeds. The shaded area indicates the standard deviation.

7. Conclusion

Although a non-contextual agent could still achieve high rewards, by carefully evaluating the behaviour of the agents and using our proposed GOD measure, we discovered that non-contextual agents cannot very well distinguish between similar contexts. Furthermore, we demonstrated that by providing context information as ground-truth to the agent, this issue can be alleviated and the ability of an agent to distinguish between similar context can be significantly improved. Additionally, we showed that we can provide the context information to the agent by using disentangled features. However, we have observed that providing the agent only with this information is not enough, and the agent needs to additionally learn state features itself and adapt to the changes in the environment. All in all, our results

suggest that incorporating context information using disentangled features is a very effective way of improving the ability of agents to distinguish between different contexts and, as a result, perform better.

In order to have an agent that can dynamically learn from the environment without the need of offline pre-training, we further investigated an online version of our contextual agent, namely *Online CARLA*. As discussed, our results demonstrated the effectiveness of *Online CARLA* in achieving better performances in terms of distinguishing between different contexts, outperforming non-contextual agents, and achieving performances on par with *CARLA* that uses a pre-trained VAE, but without the need of any pre-training.

While in this paper, we focus on a setting where the context can be directly inferred from single states, in future work we would like to address problems in partially-observable settings where this not possible. To that end, a sequential model can be incorporated in the CARLA architecture for learning the context variables. Furthermore, it will be important to see whether the proposed architecture can be used in more complex environments.

Acknowledgments

This work has been partially supported by Google Cloud, and the European Research Council (ERC) under the EU's Horizon 2020 research and innovation program (grant agreement number 670035). The LIT AI Lab is supported by the Federal State of Upper Austria. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU used for this research. We would also like to thank Carlo D'Eramo and Davide Tateo of the Intelligent Autonomous Systems group, at the University of Darmstadt, and José A. Arjona-Medina from the Machine Learning Institute, at the Johannes Kepler University of Linz and Dynatrace Research, for the fruitful discussions throughout this project.

References

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018a.

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

Hamid Eghbal-zadeh, Florian Henkel, and Gerhard Widmer. Learning to infer unseen contexts in causal contextual reinforcement learning. In *Self-supervision for Reinforcement Learning (SSL-RL) Workshop, ICLR*, 2021.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017a.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. DARLA: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017b.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E. Schapire. Contextual decision processes with low Bellman rank are PAC-learnable. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, 2007.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *Proc. of the 30th International Conference on Machine Learning (ICML)*, 2013.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. *arXiv preprint arXiv:1906.04161*, 2019.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence*, pages 3942–3951, 2018.

Danilo Jimenez Rezende and Fabio Viola. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. *arXiv preprint arXiv:2009.08319*, 2020.

Supplementary Material

Appendix A. Experimental Details

A.1. Architectures and Hyperparameters

All agents are trained using two separate Adam optimizer (Kingma and Ba, 2015) with a learning rate of $7e^{-4}$ and $1e^{-3}$ for policy and value function, respectively. The discount factor γ is set to 0.97 and entropy regularization is used with a coefficient of 0.01 to encourage exploration. Updates are performed every 4096 transitions which are collected by 8 parallel actors (512 per actor). For each policy update, we perform 30 value function updates which showed to improve training stability and mitigate convergence to sub-optimal behaviour. Overall the agents are trained for 3000 epochs, where one epoch corresponds to 4096 transitions.

State Encoder: The state encoding network consists of four 4×4 convolutional layers with 16, 16, 32 and 32 channels and a stride of 2. ReLU activation function (Nair and Hinton, 2010) is applied between all layers.

Context Encoder: If applicable, agents use a separate context encoding network which is the encoder of a Convolutional Variational Autoencoder (VAE) as described in Appendix B. Unless specified, this encoder is pre-trained and not updated during training of the RL agents.[2]

Policy and Value Networks: The policy and value networks use two fully-connected layers of size 64, ReLU activation and a linear output layer of size 3 for the policy and 1 for the value function. State and context encoder are shared across both.

Combining Contextual Information: In order to incorporate context information captured by the VAE encoders, we investigate two different approaches. In the first approach, we simply concatenate the VAE embeddings with the flattened state encoder embeddings, and further pass it to the policy/value networks. While this approach is fairly

2. VAEs are only updated in *Online CARLA*, and those specified with *Upt.*

simple and does not introduce any additional parameters, it allows the policy to adapt to the contextual information.

Additionally, as specialized layers have been developed for combining different sources of information e.g, in a multi-modal RL setting, we incorporate such modules into our agent. In particular, we apply a *Feature-wise Linear Modulation* (FiLM) layer (Perez et al., 2018), using disentangled features to control a weight and a bias term for the state features, followed by a normalisation layer as follows:

$$\text{FiLM}(\mathbf{x}) = \text{norm}(\gamma(\mathbf{c}) \cdot \mathbf{x} + \beta(\mathbf{c})), \tag{4}$$

with $\mathbf{x}$ being the state encoder embeddings, $\mathbf{c}$ being the VAE embeddings, and γ and β being two linear layers. For normalization, we use batch-norm (Ioffe and Szegedy, 2015). We report results using both concatenation and FiLM layer in our *CARLA* agent. Based on our results, it seems that while both techniques are successful in incorporating contextual information, the simple concatenation performs better, while introducing no additional parameters.

Algorithm 1: Online Carla

Initialise policy π, value v, state encoder ϕ, context encoder/decoder ψ_{enc} and ψ_{dec}
Initialise empty VAE-Buffer $\mathcal{B}$ with length $\mathbf{L}$
for $i = 1$ **to** n **do**
  collect trajectory τ of length T following policy π
  add observations from τ to $\mathcal{B}$ (overwrite oldest sample if max length $\mathbf{L}$ is reached)
  update π, v and ϕ using the RL objective with trajectory τ
  for $j = 1$ **to** m **do**
    update ψ_{enc} and ψ_{dec} using VAE loss with samples from $\mathcal{B}$
  end
end

A.2. Defined Contexts

To create the context for our experiments we use 11 distinct colours as shown in Table 3.[3] One of them is used as a fixed goal colour, while the remaining 10 allow us to create 45 unique colour combinations. These colour combinations are then used to generate context pairs, e.g., given the combination *red*/*gold*, we create two contexts, one where the obstacle is *red* and the goodie and agent is *gold*, and one where obstacle is *gold* and goodie and agent is *red*. Overall, this results in 90 distinct contexts (45 context pairs).

Appendix B. VAE Details

B.1. Training and Architecture

A Convolutional Variational Autoencoder (VAE) (Kingma and Welling, 2014) architecture is used with four convolutional layers with channels $32, 32, 64, 64$, kernels $8, 4, 3, 3$, and strides

3. Colour selection based on https://mokole.com/palette.html

Table 3: Used colours to create the contexts.

Goal	Obstacles/Goodies
lime	saddlebrown, forestgreen, steelblue, darkblue, red gold, aqua, fuchsia, moccasin, hotpink

4, 2, 1, 1, and no padding for the encoder. The decoder uses the reversed architecture of the encoder, with transposed convolutional layers instead of convolutional layers. The *Leaky Rectified Linear Units* (Maas et al., 2013) with a negative slope of 0.002 were used as the non-linearity for the hidden layers of the VAE. The model was trained using Adam optimizer with learning rate of $5e^{-4}$ and batch-size of 128, for 100 epochs, on a dataset comprised of 50k random samples from the environment using the train contexts, which were collected and used as the training data. All VAEs incorporate a Gaussian prior, using *MSE* as reconstruction error. Both KL and reconstruction losses are first summed across all dimensions, then divided by the batch-size, before being combined. In GECO, we set the target reconstruction to 6, and in Annealed-VAE, we aim for achieving the final KL of 21, which are the hyperparameters that we found to work best for each method. We train 3 different β-VAEs with beta values of $\beta = 1$ (Vanilla VAE), $\beta = 2.5$, and $\beta = 10$. Random samples from the dataset, and their reconstructions using our fully trained VAE are provided in Table 5.

Table 4: VAE reconstruction and KL comparison. In each mini-batch, KL and MSE are summed over all dimensions, then are divided by the batch-size. Lower is better, best results are marked bold.

	MSE	KL
GECO	**5.89**	21.15
Annealed	9.83	21.03
Beta$^{\beta=10}$	56.02	**1.66**
Beta$^{\beta=2.5}$	23.69	8.77
Beta$^{\beta=1}$	8.12	17.09

B.2. Evaluating Disentanglement

In order to evaluate the disentanglement in our VAEs, we follow the approach proposed in (Higgins et al., 2017a), and we adapt this measure to our environment. We define a ground-truth factor as a 9-dimensional vector, modelling agent, goodie, and the goal. Each of these entities have 3 values: one value for colour, one for the x position, and one for the y position. Given this 9-dimensional vector, we can then render an image with the agent, the goal, and the goodie, with specified locations, and colours for each of these entities.

The *β-VAE score* is calculated by using a ground-truth factor, and altering one of the following factors: (1) goal location, (2) agent location, (3) agent colour, (4) goodie colour, and (5) goodie location.

We then generate samples from both the original, and the altered ground-truth factor, further compute the embedding of the two generated samples using a VAE encoder. The difference between these two embeddings are then calculated and is used as a feature, along with the factor itself as ground-truth label, to train a linear classifier. This classifier is then evaluated on a set of unseen *difference vectors*. A higher accuracy indicates better disentanglement of the encoder.

To reduce the variance, we create 3 different versions of the altered ground-truth, from a given ground-truth factor, and calculate 3 difference-vectors from it. The average of these 3 difference-vectors are then used as feature for training the linear classifier, as recommended in (Higgins et al., 2017a).

B.3. VAE Reconstruction Examples

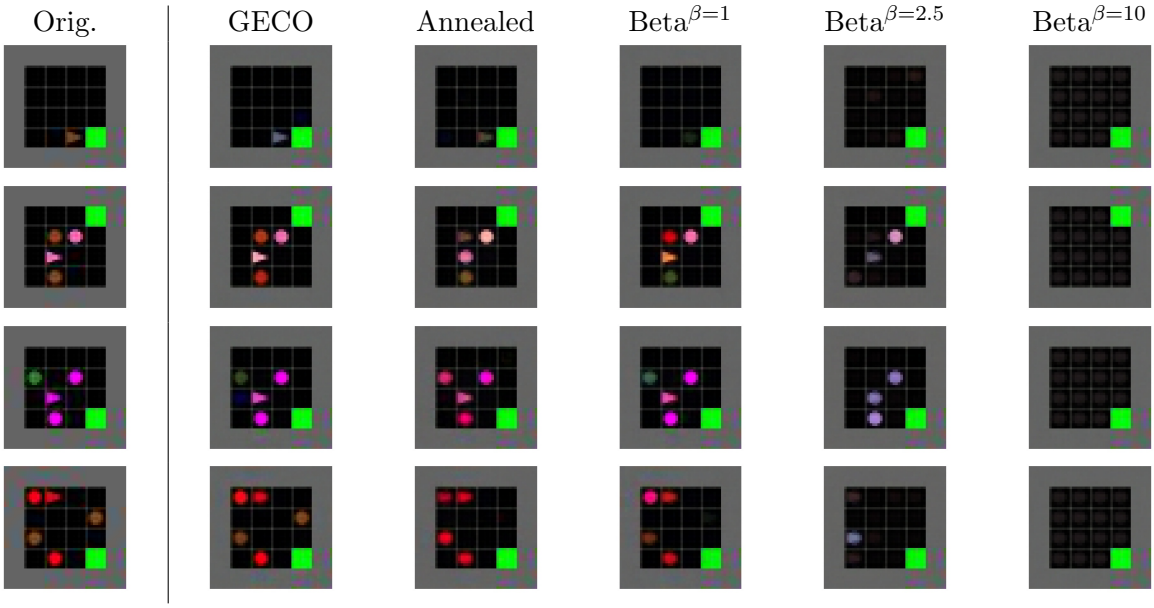
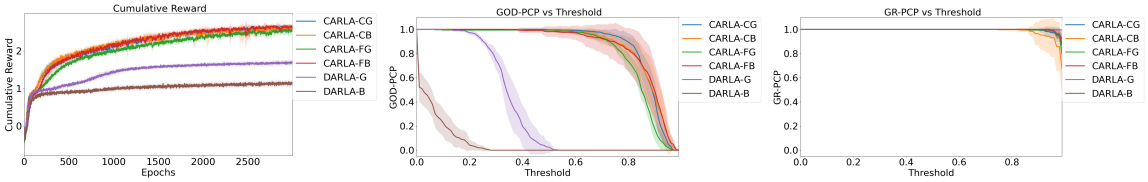


Table 5: Original (left), and reconstructed samples from different VAEs.

Appendix C. Additional Results

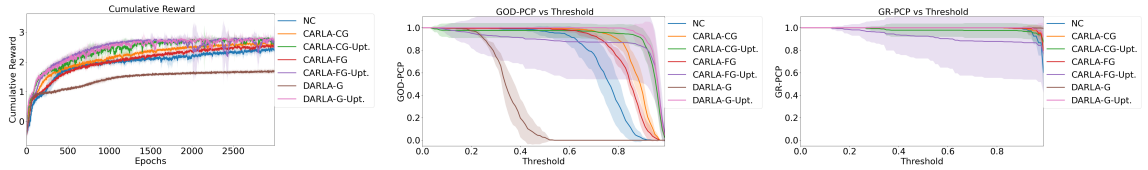
Comparing different VAEs in RL agents: In Figure 7 we compare the performance of *DARLA* and *CARLA* trained with two different VAE setups – Beta VAE (with $beta = 2.5$), and GECO. While *DARLA* fails to solve the task with both VAEs, we observe a higher performance using GECO. For *CARLA* the difference is not as distinct. When using FiLM to incorporate contextual information, Beta VAE (*CARLA-FB*) outperforms its GECO counterpart (*CARLA-CG*). When concatenation is used (*CARLA-CB*/*CARLA-CG*) the performance is very similar, with Beta VAE being slightly better for higher threshold values.



(a) (b) (c)

Figure 7: Comparison of DARLA and CARLA with Beta (B) and GECO (G) VAE. For CARLA, we additionally compare concatenation (C) and FiLM (F) to incorporate contextual information. Figure 7(a) is the cumulative reward over training epochs, Figure 7(b) is the goodie-obstacle discrimination ratio and Figure 7(c) the goal reached ratio averaged across 8 different random seeds. The shaded area indicates the standard deviation.

VAE pre-training as warm-up: In Figure 8 we compare the performance of DARLA and CARLA agents, where the pre-trained VAE encoder was allowed to be updated via the RL loss, during the training. We can observe that by allowing the encoder to update via the RL loss, the features can be better aligned with the policy, which results in performance improvements in both CARLA and DARLA agents. Specially, in DARLA we can observe a significant performance improvement when the encoder is allowed to be updated. This suggests that aligning the state/context features with the policy during training is beneficial to improve the performance of the agents.



(a) (b) (c)

Figure 8: Comparison of non contextual (NC), DARLA and CARLA with GECO VAE. For DARLA and CARLA we include variants where the VAE encoder is frozen or updated during training, respectively. For CARLA we additionally compare concatenation (C) and FiLM (F) to incorporate contextual information. Figure 8(a) shows the cumulative reward, Figure 8(b) shows the goodie-obstacle discrimination ratio, and Figure 8(c) the goal reached ratio averaged across 8 different random seeds. The shaded area indicates the standard deviation.