A Proof for Proposition1

Proof. The value function of CMDP is defined in the feasible region $\Pi^c = \{\pi \in \Pi^c : \sum_{t=1}^{H} c(s_t, a_t) \leq C, a_t = \pi(s_t), s_{t+1} = \mathcal{T}(s_t, a_t)\}$, where $\pi \in \Pi^c$ and

$$V_c^\pi(s) = \sum_{t=1}^{H} r(s_t, a_t), \text{ where } \quad a_t = \pi(s_t), s_{t+1} = \mathcal{T}(s_t, a_t) \tag{7}$$

The learning objective is to find $\pi \in \Pi^c$ such that

$$V_c^*(s) = \max_{\pi \in \Pi^c} V_c^\pi(s). \tag{8}$$

The value function for ET-MDP is defined similarly as normal MDP by

$$V_{ET}^\pi(s) = \sum_{t=1}^{H} r'(s_t, b_t, a_t), \text{ where } a_t = \pi(s_t), s_{t+1} = \mathcal{T}'(s_t, b_t, a_t), b_{t+1} = b_t + c_t. \tag{9}$$

For any $\pi \in \Pi^c$, the trajectories are the same in the ET-MDP and its counterpart. We have $r'(s_t, b_t, a_t) = r(s_t, a_t)$ for all $t \leq H$. Therefore, we have $V_c^\pi = V_{ET}^\pi$ for $\pi \in \Pi^c$.

The optimal value function of ET-MDP is defined over its optimal policy

$$V_{ET}^*(s) = \max\{\max_{\pi \in \Pi^c} V_c^\pi(s), \max_{\pi \notin \Pi^c} \sum_{t=1}^{h_\pi \leq H} r(s_t, a_t) + r_e\}, \tag{10}$$

where h_π is the step at which the constraint is violated. Therefore, $V_{ET}^*(s) = V_c^*(s)$ for sufficiently small r_e and the optimal state values are achieved for the same optimal policy $\pi^* \in \Pi^c$ □

B Detailed Pseudo-Code of the Proposed Method

Algorithm 1 MOPA for ET-MDP

Initialize critic networks Q_{w_1}, Q_{w_2}, actor network π_θ
Initialize context models $\mathcal{C}_{w_a}$,$\mathcal{C}_{w_c}$ for the actor and critic networks separately with recurrent networks.
Initialize target networks $w_1' \leftarrow w_1$, $w_2' \leftarrow w_2$, $\theta' \leftarrow \theta$
Initialize replay buffer $\mathcal{B} = \{\}$
Initialize a context queue $\mathcal{Z}_L$ with length L by $\mathcal{Z}_L = [\mathbf{0}_s, \mathbf{0}_a, \mathbf{0}_r] \times L$, maintain a copy $\mathcal{Z}_L' \leftarrow \mathcal{Z}_L$
for $t = 1, 2, ...$ **do**
  Interact with environment and get transition tuple (s, a, r, c, s'), $r \leftarrow r + r_e$ if $c > 0$.
  Update context queue with $\mathcal{Z}_L$, append (s, a, r), and store $(s, a, r, s', \mathcal{Z}_L', \mathcal{Z}_L)$ in $\mathcal{B}$, update $\mathcal{Z}_L' \leftarrow \mathcal{Z}_L$
  Sample a batch of transitions $\{(s, a, r, s', \mathcal{Z}_L', \mathcal{Z}_L)\}$ from $\mathcal{B}$
  Calculate context variable for actor and critic with $z_a = \mathcal{C}_{w_a}(\mathcal{Z}_L')$, $z_c = \mathcal{C}_{w_c}(\mathcal{Z}_L')$, and context variable for calculating the next action and next value $z_a' = \mathcal{C}_{w_a}(\mathcal{Z}_L)$, $z_c' = \mathcal{C}_{w_c}(\mathcal{Z}_L)$
  Calculate perturbed next action by $\tilde{a} \leftarrow \pi_{\theta'}(s', z_a') + \epsilon$, ϵ is sampled from a clipped Gaussian.
  Calculate target critic value y and update critic networks:
    $y \leftarrow r + \gamma \min_{i=1,2} Q_{w_i'}(s', \tilde{a}, z_c')$
    $w_i \leftarrow \arg\min_{w_i} \mathbf{MSE}(y, Q_{w_i}(s, a, z_c))$
  Update w_c, the context model for critic through
    $w_c \leftarrow \arg\min_{w_c} \mathbf{MSE}(y, Q_{w_i}(s, a, \mathcal{C}_{w_c}(\mathcal{Z}_L')))$
  Update θ by the deterministic policy gradient, with learning rate η:
    $\theta \leftarrow \theta - \eta \nabla_a Q_{w_1}(s, a, z_c)|_{a=\pi_\theta(s, z_a)} \nabla_\theta \pi_\theta(s, z_a)$
  Update w_a, the context model for actor according to Eqn.(6)
  Update target networks, with $\tau \in (0, 1)$:
    $w_i' \leftarrow \tau w_i + (1 - \tau) w_i'$; $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$
  Break this episode if constraint is broken.
end for

C Reproduction Checklist

Network Structure Our implementation of Context TD3 is mainly based on the code of [18]. The hyper-parameters of TD3 are the same as the authors recommend in the paper. In our Context TD3, we also use 3-layer MLPs for both actor and critic networks (with 256 hidden units).

We find in our experiments using separated context networks that trained through gradients of actor and critic will benefit learning. Details of ablation study on the network structure are provided in Appendix F.3

Value of r_e In our analysis, the value of r_e can be selected as any sufficiently small number. However selecting too small value may lead to over-conservative behavior. In our experiments reported in the main text, we find in experiments that $r_e = -1$ works fairly well. Ablation studies on the selection r_e are provided in Appendix F.1.3.

Batch Size In our experiments we follow Fujimoto et al. [18] to use a mini-batch size of 256. In PPO, CPO and PPO-Largrangian, we use a batch size of 1000 and mini-batch size of 256 for the short-horizon games (e.g., Maze, PointGather, both with $T \leq 32$), so that there are around 1000 episodes in training. For the long-horizon games where $T \sim 1000$, we collect 10 trajectories for each episode for better training stability [5, 56].

Hardware and Training Time We experiment on a server with 8 TITAN X GPUs and 32 Intel(R) E5-2640 CPUs. Experiments take 0.5 (the maze environment with 0.1M interactions) to 10 hours (the safety-gym with 1M interactions) to run. The training of Context TD3 will introduce higher computation expense as additional context models need to be trained.

D Environments Details

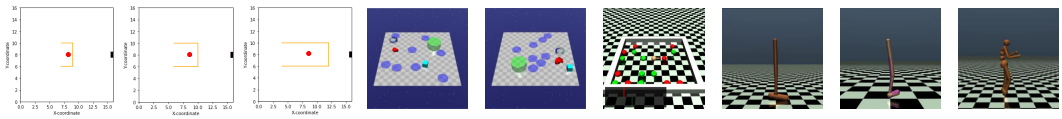

Figure 5: Examples of the tested environments: The first three figures show the DangerZone tasks with different level (different mazes); the following three figures show the budget tasks where agents control a point or a car to collect reward without hitting cost regions too many times; the last three figures show loose-constrained tasks where agents need to learn to move forward without falling.

D.1 DangerZone Environment

In the DangerZone environment, an agent needs to navigate in a maze for a goal point without stepping into the lava. The input of the agent is the coordinate of current state, and permitted action is limited to $[-1, 1]$. We generate four different level of tasks. In all experiments the size of the maze is set to be 16×16, and episodic length is set to be 32, which is two times of the side length. In each episode, the agent is initialized in the center of the maze. Stepping into the target position will result in a $+30$ reward, and stay in the position will continuously gain that reward. A tiny punishment of -0.1 is applied for every timestep, otherwise.

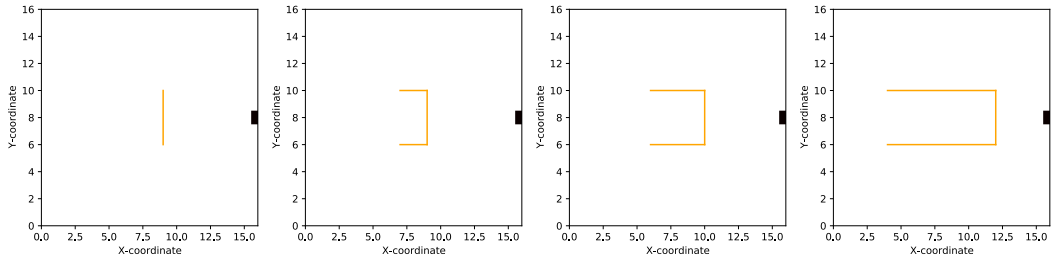

Figure 6: The four DangerZone environments with different levels in our experiments. From left to right: Maze-Level-1 (DangerZone Easy), Maze-Level-2 (DangerZone Medium), Maze-Level-3, Maze-level-4 (DangerZone Hard). The regions with orange color are dangerous region where the agent should not step into. For each game, the agent is initialized at center of the map, therefore the difficulty of finding a solution without violating the constraints becomes harder and harder from Level-1 to Level-4.

E Missing Learning Curves

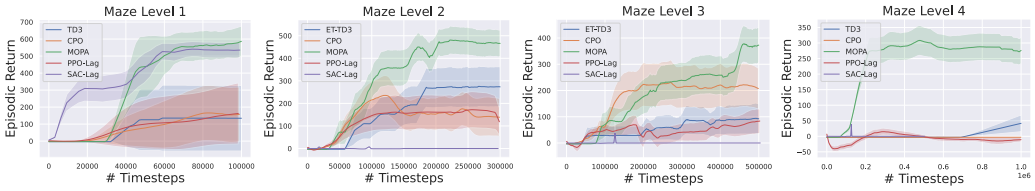

Figure 7: Learning curves of the Maze environments. As the constraints are binary, any reward gained when the constraints are violated is not taken into consideration. Thereafter, only episodic return curves are shown in the figures. i.e., All of those rewards are gained *without breaking any constraint*.

F Additional Empirical Studies

F.1 Sensitivity to Hyper-Parameter

F.1.1 Value of Historical Horizon

We experiment on the DangerZone environments to show how the proposed method work with different length of historical horizon in the context model. Results are shown in Figure 8. **Context** 1 means we only include the past state, action, reward in the computation of context variables, while **Context** 7 indicates the past 7 steps of transitions are leveraged in generating the context variables. We find the context model with historical horizon 3 achieve fairly well performance in all levels of environments.

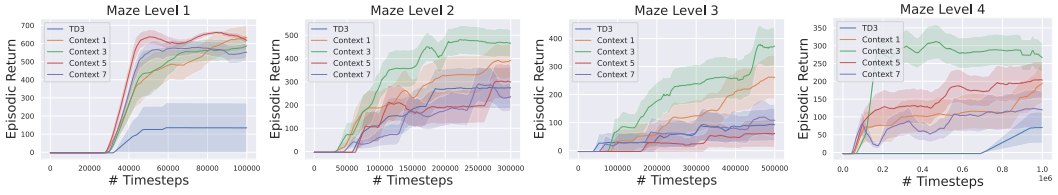

Figure 8: Ablation studies on the selection of different length of historical horizon. All corresponding costs are zero and omitted under our ET-MDP settings.

F.1.2 Number of Hidden Units in GRUs

We experiment on the selection of different number of hidden units used in GRUs. We compare the results with 30 hidden units (reported in the main text, denoted as **MOPA** in Figure 9) with the results with 120 hidden units (denoted as **MOPA-Large** in Figure 9). We find using 30 hidden units is enough to achieve improved performance, and in the same time balance the computational cost. And using too much hidden units may lead to reduction on learning efficiency (in the Humanoid-Not-Fall-v0 environment).

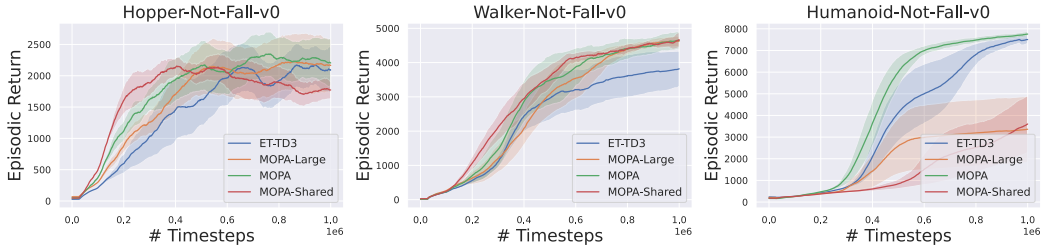

Figure 9: Ablation studies on the number of hidden units used in GRU, and comparison on different selection of network structure: shared v.s. separated context model.

F.1.3 Value of r_e

We show experimental results on the selection of different value of the ending reward r_e in this section. Figure 10 shows the results on the CarGoal, PointGoal and PointGather environments. In both TD3 and Context TD3 working in ETMDP, smaller r_e's result in more conservative policies that achieve lower cost and lower primal task reward.

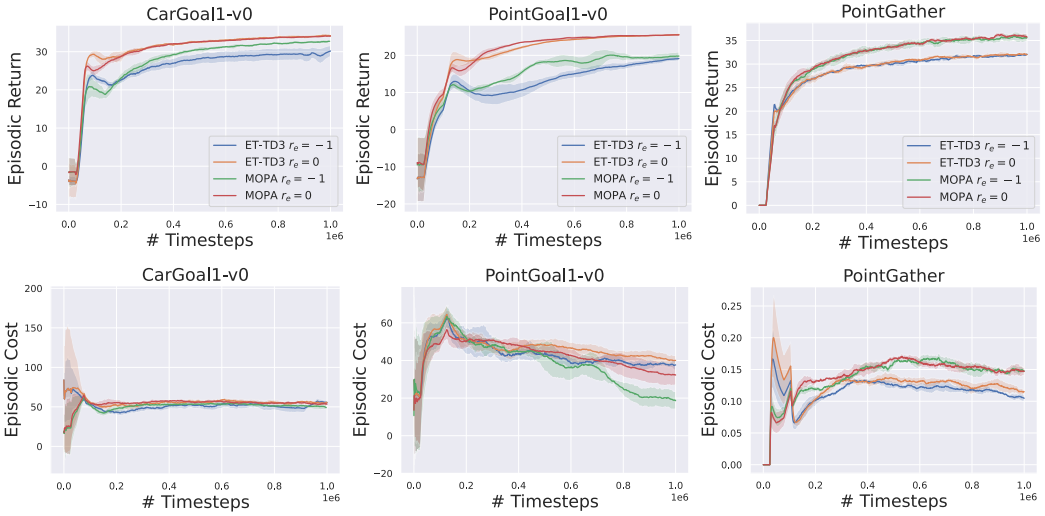

Figure 10: Ablation studies on the selection of r_e, the value of absorbing reward. The first line shows the episodic return curves of each methods in different environments while the second line shows the corresponding episodic costs. Using smaller r_e will lead to more conservative behavior, i.e., slightly lower return and lower cost.

F.2 More Experiments

F.2.1 Validation of Syllogism 4.5 on Other Benchmarks

In this section we show our experiments on various MuJoCo and DeepMind Control benchmarks to show the superiority of the Context TD3 over the vanilla TD3 in sample-efficient learning in MDP

tasks. Figure 11 shows the experiment results. In most environments, Context TD3 achieves better asymptotic performance while being able to converge faster. We use the same hyper-parameter of historical horizon $= 7$ in all experiments. Elaborated searching for hyper-parameters may result in even better performance.

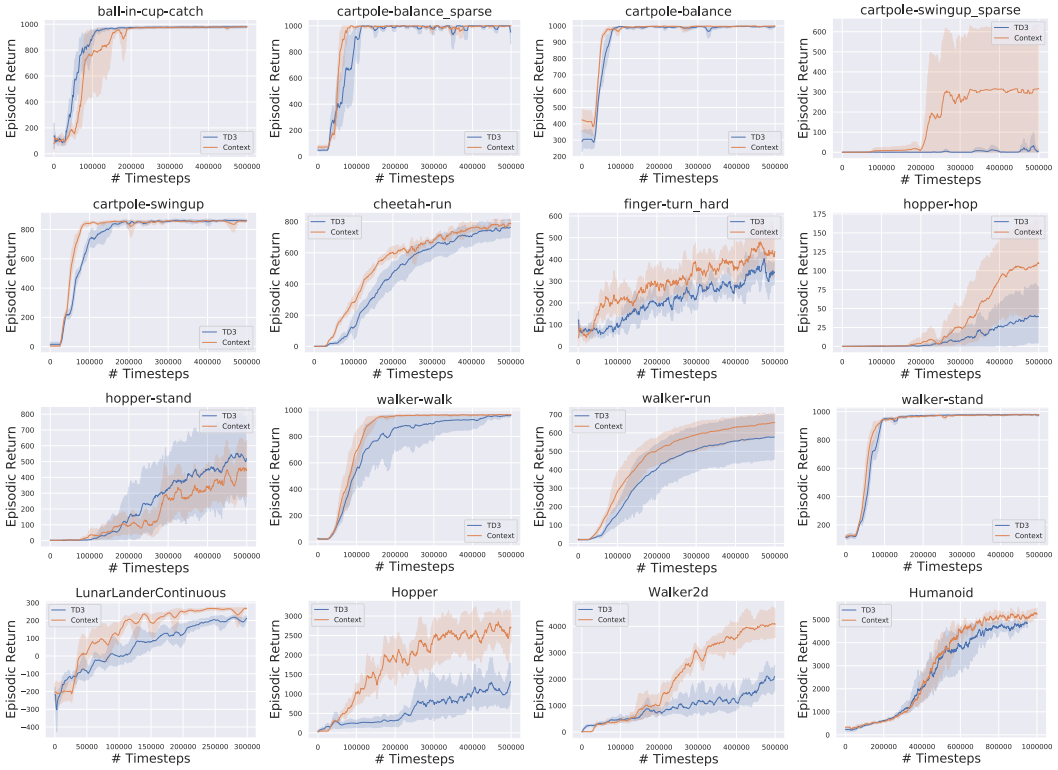

Figure 11: Experiment results on the DeepMind Control Suite. In all 16 benchmark environments we experimented on, context models outperforms normal TD3 in most environments (13 out of 16), showing the superiority of context models in improving learning efficiency.

F.3 Model Structure

F.3.1 GRU v.s. Transformer

In this section we provide ablation studies on the choice of context models: we compare the results of Context models based on GRUs and based on recent advances of self-attention based models [57]. The results are shown in Figure 12, where we find leveraging the transformer models can not result in better performance.

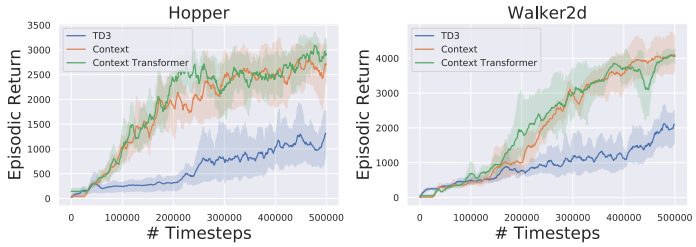

Figure 12: Ablation studies on the selection of context models.

F.3.2 Shared v.s. Separated Context Variables

In the work of [30], the context model is trained only through the learning of critic networks. Differently, in our experiments we find training context models separately for the actor and critic can result in better performance. **MOPA-Shared** in Figure 9 denotes the results when the context model is shared by actor and critic as recommended in the Meta-RL literature [30].

G Qualitative Results

We also include demo videos in the supplemental materials. Where the performance of agents trained with different algorithms in the PointGoal1-v0 and CarGoal1-v0 safe-navigation tasks are shown qualitatively.