Supplementary Material

In this supplementary material, we provide further information about our implementation details and ablation studies.

1 Truncation Error

The update procedure of the Euler Equation

$$y_{i+1} = y_i + hf(x_i, y_i), \tag{1}$$

and the accurate update equation is

$$y_{i+1} = y_i + hf(x_i, y(x_i)) + \frac{h^2}{2} f''(\xi_i), \quad \xi_i \in (x_i, x_{i+1}). \tag{2}$$

The lost item $\frac{h^2}{2} f''(\xi_i)$ is the local truncation error of the Euler method. The global truncation error is defined as

$$\begin{aligned} e_{i+1} &= y(x_{i+1}) - y(i+1) \\ &= (y_i + hf(x_i, y(x_i)) + \frac{h^2}{2} f''(\xi_i)) - (y_i + hf(x_i, y_i)) \\ &= e_i + h(f(x_i, y(x_i)) - f(x_i, y_i)) - \frac{h^2}{2} f''(\xi_i) \end{aligned}$$

In Euler method $f(x, y)$ satisfy the Lipschitz condition for y and denote $T_{i+1} = -\frac{h^2}{2} f''(\xi_i)$ is the local truncation error and $T = \max_i |T_i| = O(h^2)$

$$\begin{aligned} |e_{i+1}| &\leq |e_i| + h|f(x_i, y(x_i)) - f(x_i, y_i)| + |T_{i+1}| \\ &\leq |e_i| + h|y(x_i) - y_i| + |T_{i+1}| \\ &\leq |e_i|(1 + hL) + T \\ &\leq (1 + hL)^{i+1}(|e_0| + \frac{T}{hL}) \\ &\leq e^{(i+1)hL}(|e_0| + \frac{T}{hL}) \\ &= O(h) \end{aligned}$$

where $e^0 = y(x - 0) - y_0 = 0$.

2 Enivronment Specification

We incorporate the occlusion benchmark proposed by VRM and replace the deprecated roboschool with PyBullet, as the official GitHub repository suggests. We transform the original MDP task into a POMDP version by removing all position/angle-related entries in the observation space for "-V" environments and velocity-related entries for "-P" environments.

{Pendulum, CartPole, AntBLT, WalkerBLT, HopperBLT}-P. The "-P" denotes environments that retain position-related observations by eliminating velocity-related observations.

{Pendulum, CartPole, AntBLT, WalkerBLT, HopperBLT}-V. The "-V" denotes environments that retain velocity-related observations by eliminating position-related observations.

3 Implemnetaion Details

We use the PyTorch framework for our experiments. Some basic hyperparameters about the network architectures are listed below

Also, there are some unique parameters in TD3 and SAC, we report them together

Table 1: Environment information

Environment name	$dim(o)$	$dim(a)$	Maximum step
Pendulum-P(position only)	1	1	200
Pendulum-V(velocity only)	1	1	200
CartPole-P(position only)	2	1	1000
CartPole-V(velocity only)	2	1	1000
AntBLT-P(position only)	17	8	1000
AntBLT-V(velocity only)	11	8	1000
WalkerBLT-P(position only)	13	6	1000
WalkerBLT-V(velocity only)	9	6	1000
HopperBLT-P(position only)	9	3	1000
HopperrBLT-V(velocity only)	6	3	1000
HalfCheetah-Dir	17	6	200

Table 2: Hyperparameters

Hyperparameters	Description	Value
γ	Discount factor	0.99
lr	Learning rate	0.0003
λ	Balance weight between KL divergence and RL loss	0.5
τ	Fraction of updating the target network each gradient step	0.005
dqn_layers	MLP layer sizes of value function	[256,256]
policy_layers	MLP layer sizes of policy network	[256,256]
sampled_seq_len	The number of steps in a sampled sequence for each update	64
batch_size	The number of sequences to sample for each update	64
buffer_size	The number of saved transitions	1e6
action_embedding_size	Embedding dimension of action	16
observ_embedding_size	Embedding dimension of observation	32
reward_embedding_size	Embedding dimension of reward	16
gru_hidden_size	Hidden layer size of recurrent encoder	128

3.1 Encoding procedure of GRU-ODE

The input observations x_t is the concatenation of observations, actions and rewards, as we mentioned above. The time increment dt is set to a fixed value in regular observation environments. In this paper, we set $dt = 0.1$ for all POMDP tasks.

3.2 Integrating with TD3

We use separate recurrent encoders for the actor and critic in order to improve the performance of our method. Owing to the policy network of TD3 updating less frequently than the value function. We modify the iteration equation of the policy network to:

$$\mathcal{L}(\phi_a, \theta, \psi) = - \mathop{\mathbb{E}}_{o \sim \mathcal{D}} [q_{\psi_1}(o, \pi_\theta(o, z), z)] + \lambda_a \mathcal{K}(\phi_a), \tag{3}$$

Algorithm 1 The GRU-ODE algorithm.

Input: Observations and time difference between observations $(x_t, dt)_{t=1..T}$
h_0 = **0**
for t in 1, 2, . . . , T **do**
  $\tilde{h}_t = \text{GRUCell}\,(h_{t-1}, x_t)$ {Update hidden state}
  $h_t = \text{ODESolve}\left(f_\theta, \tilde{h}_t, dt\right)$ {Solve ODE}
$z_t = \text{MLP}(h_t)$ for all $t = 1..T$
Return: $\{z_t\}_{t=1..T}; h_t$

Table 3: Hyperparameters of SAC and TD3

Method	Hyperparameters	Value
SAC	entropy_alpha	0.2
SAC	automatic_entropy_tuning	True
SAC	alpha_lr	0.0003
TD3	exploration_noise	0.1
TD3	target_noise	0.2
TD3	target_noise_clip	0.5

$\mathcal{D}$ denotes the replay buffer. The state s is replaced with observation o owing to the partially observable environments.

The update of the value function is modified to:

$$\begin{aligned} y(r, o', z') &= r + \gamma \min_{i=1,2} Q_{\psi_i, targ}(o', \pi_{\theta, targ}(o'), z'), \\ \mathcal{L}(\psi_i) &= \mathop{\mathbb{E}}_{(o,a,r,o')\sim\mathcal{D}} \left[(Q_{\psi_i}(o, a, z) - y(r, o', z'))^2 \right], \\ \mathcal{L}(\phi_c, \psi) &= \mathcal{L}(\psi_1) + \mathcal{L}(\psi_2) + \lambda_c \mathcal{K}(\phi_c), \end{aligned} \tag{4}$$

where subscript $1, 2$ denotes two different critic networks implemented in TD3 and $targ$ denotes the target network. The context variable is computed by $g_\phi(z_t|\tau_{:t})$ as the policy network and value function update, we omit this procedure to simplify the equation.

3.3 Integrating with SAC

Similar to TD3, we implement our GRU-ODE in SAC. The training loss of the policy network is modified to

$$\mathcal{L}(\phi_a, \theta, \psi) = - \mathop{\mathbb{E}}_{o\sim\mathcal{D}} \left[\min_{j=1,2} q_{\psi_j}(o, \pi_\theta(o, z), z) - \alpha \log \pi_\theta(a|o) \right] + \lambda_a \mathcal{K}(\phi_a), \tag{5}$$

and the value function becomes

$$\begin{aligned} y(r, o', z') &= r + \gamma \min_{j=1,2} (Q_{\psi_j, targ}(o', \pi_{\theta, targ}(o'), z') - \alpha \log \pi_\theta(\tilde{a}|o')), \tilde{a} \sim \pi_\theta(\cdot|o'), \\ \mathcal{L}(\psi_i) &= \mathop{\mathbb{E}}_{(o,a,r,o')\sim\mathcal{D}} \left[(Q_{\psi_i}(o, a, z) - y(r, o', z'))^2 \right], \\ \mathcal{L}(\phi_c, \psi) &= \mathcal{L}(\psi_1) + \mathcal{L}(\psi_2) + \lambda_c \mathcal{K}(\phi_c), \end{aligned} \tag{6}$$

where symbols have the same meaning as in Eq. 4. The parameter λ is set to 0.5 both in TD3 and SAC algorithms.

4 ODE Solver Comparison

In this ablation study, we ask two questions in relation to numerical integration. (i) how different numerical solvers influence the final performance of POMDP tasks, and (ii) the effect on the training runtime hours. We experimentally evaluate using different numerical methods as *ODESolve* for integrating $\tilde{h}_t$ in GRU-ODE. From Table 4, we find that there is no obvious relationship between the

Table 4: Final performance of different numerical integration methods

	Ant-P	Ant-V	Walker-P	Walker-V
Euler	1243 ± 43	998 ± 93	1487 ± 81	612 ± 27
RK4	1183 ± 67	1045 ± 112	1405 ± 64	683 ± 45
Heun	1123 ± 53	1023 ± 69	1457 ± 45	625 ± 41

complexity of the numerical solvers and the final performance of the PODMP tasks. However, the

RK4 and Heun need much more time for training. We train these methods on a server with NVIDIA TITAN Xp and Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz as GPU and CPU respectively. The following are rough estimates of average run times for the AntBLT-P environments.

- Ours (with Euler) 12 hours
- Ours (with RK4) 20 hours
- Ours (with Heun) 17 hours

As the results show for context variable coding, the simplest numerical Euler method can work well as *ODESolve* and save training time. We speculate that though the environment is complicated to solve, the context variables always reflect the essential dynamic information of the environments which are more refined than the observations. Thus, simple numerical solvers are enough.

5 Time Cost of various baselines

We evaluate the time costs of different baselines on Walker-P environments. From the results, we find our method does not increase the computation a lot compared to other baselines. We analyze that there are two reasons. (I). Within our implementation of ODESolve(), we employ the "Euler" method for computation. This offers substantial reductions in computation time when contrasted with the default "dopri5" approach (Runge-Kutta of order 5 of Dormand-Prince-Shampine) utilized in ODE-RNN. We observe that employing complex ODESolve() methods does not notably enhance performance. Instead, it increases computational time significantly. (II). We are working within the context of RL tasks. When training RL algorithms, it's not just the ODE-GRU that needs training, other networks such as policy networks and value networks also require training. During the training process, a significant amount of time is consumed by sampling transitions from the data buffer and updating the data buffer. Therefore, in terms of the overall training time of our approach, it doesn't significantly exceed the time required by the previous method that used an RNN as an encoder.

Table 5: Time Cost of different methods

Methods	Time Cost
Ours	12h
RMF	10h
VRM	18h
SAC-LSTM	6h
SLAC	3h

6 Further Performance Comparison

We add two baselines for performance comparison, the RMF (Recurrent Model-free) is a recent SOTA method on POMDP problems and the TD3-FPOW (TD3 with fixed previous observation window) is a modified version of TD3, which takes 3 concatenation frames as the input.

Table 6: Performance comparison

Environments	SLAC	VRM	SAC-LSTM	RMF	TD3-FPOW	Ours
Ant-P	950±129	1040±75	946±47	1048±74	974±43	**1243±43**
Ant-V	663±87	981±63	690±34	**1021±165**	903±54	998±93
Walker-P	277±121	1121±167	971±103	1123±176	1043±103	**1487±81**
Walker-V	138±25	551±30	491±38	586±52	482±32	**612±27**
Hopper-P	222±21	1851±61	1236±39	2133±326	1654±217	**2455±87**
Hopper-V	310±41	**1652±67**	890±43	1495 ± 38	1312±87	1545±104

The result shows that our method achieved the best performance in 4 out of 6 environments.

7 Additional ablation study

We study the influence of the shared/separate encoders, and different concatenations of inputs as the ablation study. Due to the time limitation, we conducted these experiments on the Walker-P and Walker-V environments.

Table 7: Performance comparison about shared/separate encoders

Environments	shared	separate
Walker-P	1336	1487
Walker-V	584	612

The result shows that separate context encoders for the policy network and value function improve the performance.

Table 8: Performance comparison different kinds of inputs

Environments	o	oa	or	oar
Walker-P	1276	1065	1396	1487

The result shows that taking the input x_t as the concatenation of o_t, r_t and a_{t-1} for the context encoder (GRU-ODE) improves the performance.