Hybrid Quantum-Classical Reinforcement Learning in Latent Observation Spaces

Dániel T. R. Nagy, Csaba Czabán, Bence Bakó, Péter Hága, Zsófia Kallus, and Zoltán Zimborás

Quantum Information National Laboratory HUNGARY

National Research, Development and Innovation Office Hungary

PROGRAM FINANCED FROM THE NRDI FUND

ELTE EÖTVÖS LORÁND UNIVERSITY

AIME 2024

Outline

- Motivation
- RL, PPO, QRL with PPO
- Latent-space QRL
- Numerical results
- Conclusions

Motivation

- **Quantum computational supremacy demonstrated on:**
  - Superconducting device by Google (2019) https://www.nature.com/articles/s41586-019-1666-5
  - Photonic
    - Xanadu, 2022: https://www.nature.com/articles/s41586-022-04725-x
    - Jiuzhang 3.0, 2023: https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.131.150601

Motivation

- **Quantum computational supremacy demonstrated on:**
  - Superconducting device by Google (2019) https://www.nature.com/articles/s41586-019-1666-5
  - Photonic
    - Xanadu, 2022: https://www.nature.com/articles/s41586-022-04725-x
    - Jiuzhang 3.0, 2023: https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.131.150601
- **NISQ: Noisy Intermediate-Scale Quantum Devices**
  - Today already 50-100 noisy qubits (NISQ)
  - Early versions of error correction
  - Approaching regime of potential practical quantum advantage

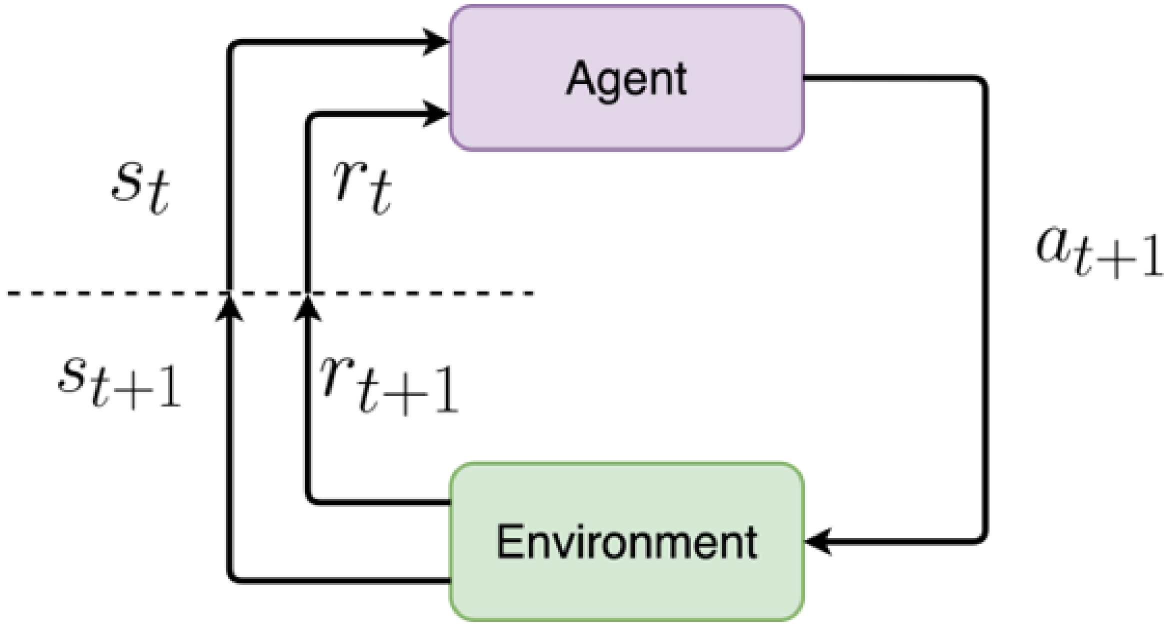
Motivation

- **NISQ-era candidates for practical quantum advantage:**
  - Simulation of quantum chemistry and many-body systems
  - Variational quantum optimization methods like QAOA
  - **Quantum Machine Learning (Includes Quantum Reinforcement Learning)**
  - **Hybrid Quantum-Classical methods enabled by classical HPC**

Motivation

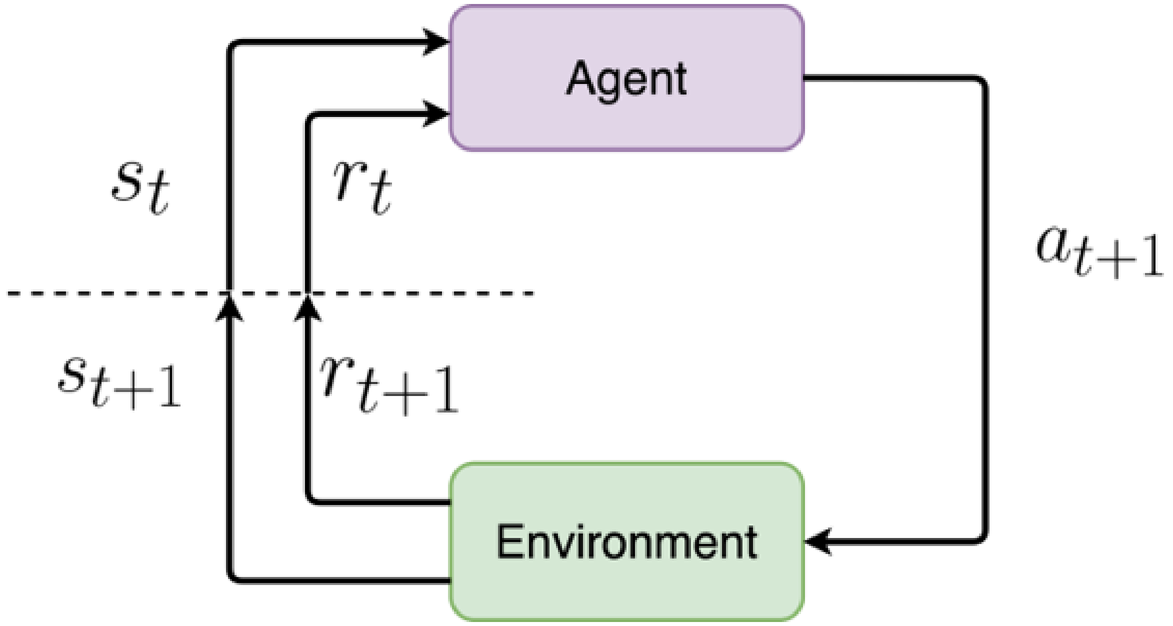
- **QRL is limited by the available QPU sizes**
  - Many RL environments have high dimensional state spaces (e.g. visual data)
  - We would need large scale QPUs to encode raw features into quantum states
  - **Proposal: use latent features extracted by classical algorithms**

Reinforcement Learning



- **Reinforcement Learning (RL)** is a method designed to optimally solve a control problem in a simulated or real-world environment.
- In RL, an **Agent** is **observing the state** of the **environment** and choses **actions** accordingly.
- After the agent performs the action, the **environment** returns a **reward** and the **next state.**

Reinforcement Learning



- The goal is to train an agent which maximizes the discounted cumulative reward,

$$R = \sum_t \gamma^t r_t$$

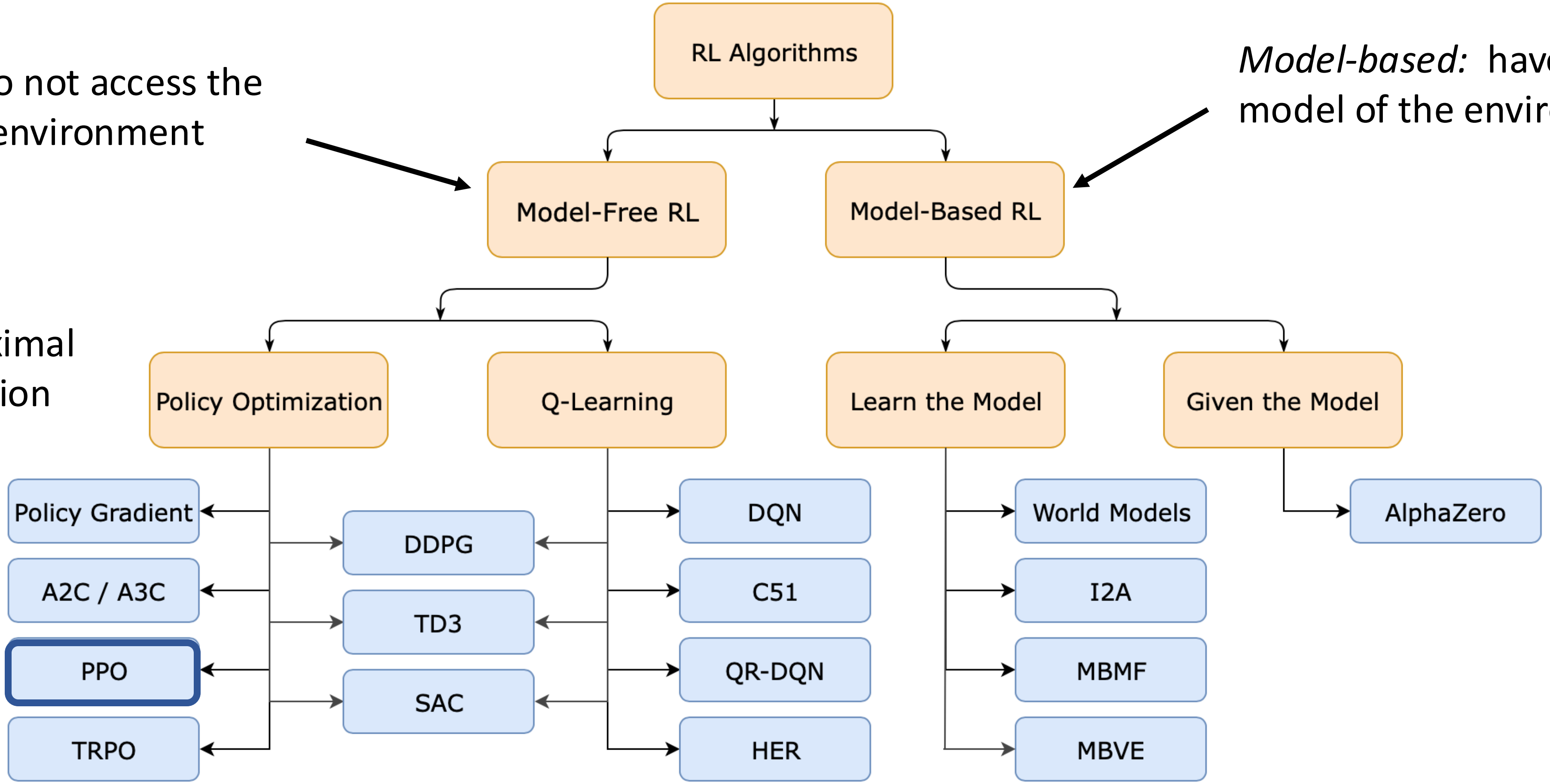
- Such Agents are usually implemented as NNs.

Reinforcement Learning

Model-free: do not access the model of the environment

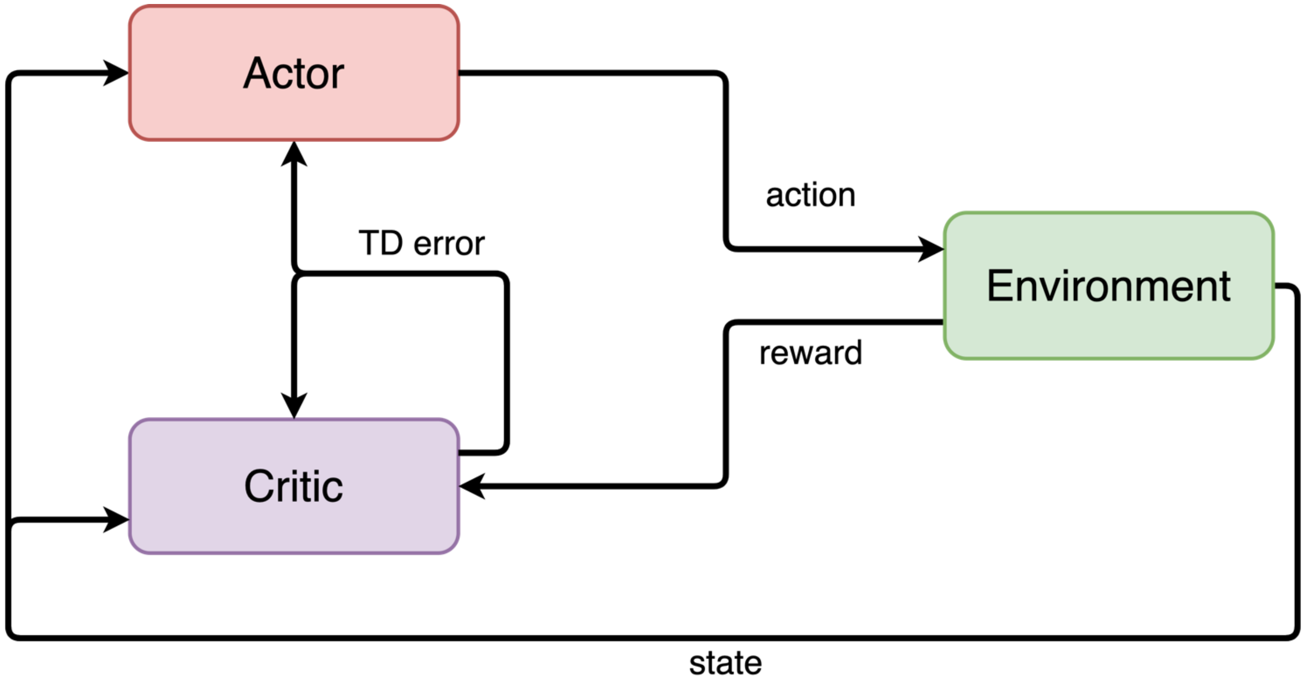
Model-based: have access to a model of the environment

We will use Proximal Policy Optimization



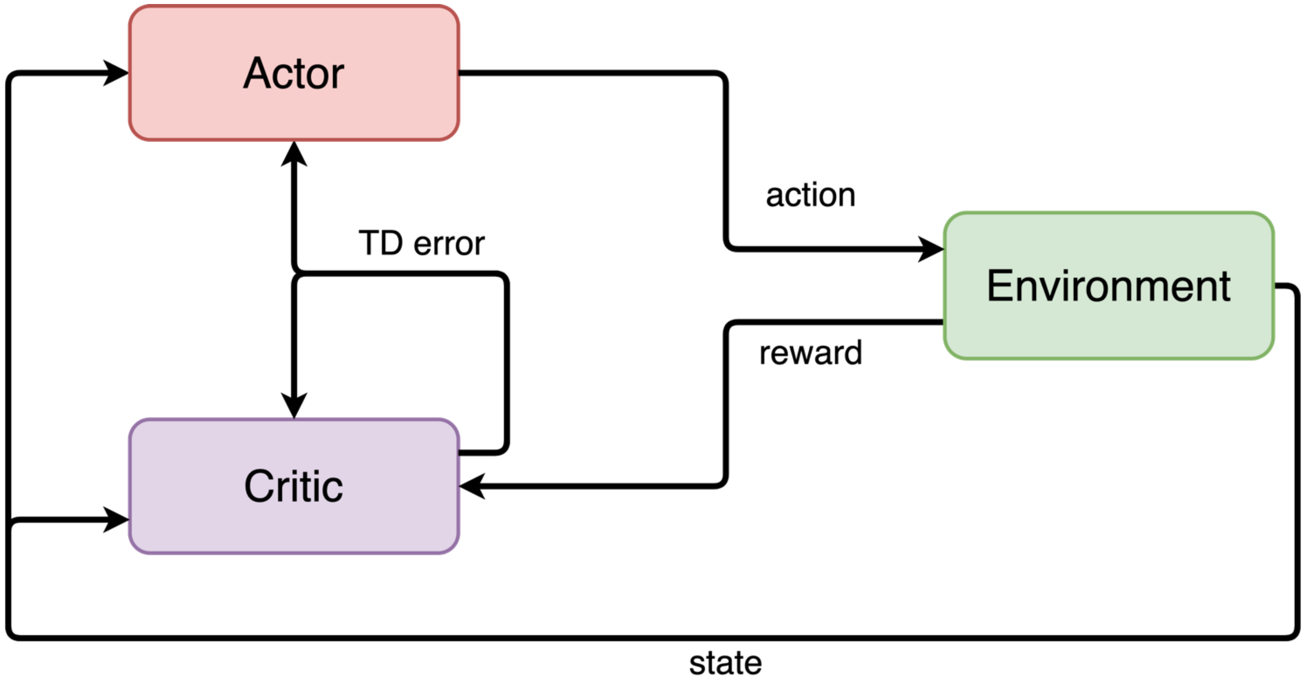
Source: https://spinningup.openai.com/

Proximal Policy Optimization (PPO)



- PPO uses two function approximators (NNs): an **Actor** and a **Critic**.
- **Actor:** choses an action according to a policy π.

Proximal Policy Optimization (PPO)



- **Critic** : receives state & reward and calculates the temporal difference error.
- The TD Error is used to update both Actor and Critic networks.

Proximal Policy Optimization (PPO)

s_t , a_t , r_t are the state, action & reward at timestep t.

$\boxed{\pi_\theta(\cdot|s)}$ is the policy, where theta are the tunable parameters.

$r_t(\theta) = \pi_\theta / \pi_{\theta_{\text{old}}}$ is the ratio of the new and old policies.

$V^\pi(s)$ is the value function used by the Critic.

$\hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l}$ is the estimated advantage with $\delta_t = r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$

The advantage function estimates the extra reward that could be obtained by the agent by taking that particular action.

Proximal Policy Optimization (PPO)

Critic Loss:

$$\mathcal{L}^{VF} = \mathbb{E}_{t}\left[\left(V^{\pi}(\mathbf{s}_t) - V^{\pi}_{\text{targ}}(\mathbf{s}_t)\right)^2\right]$$

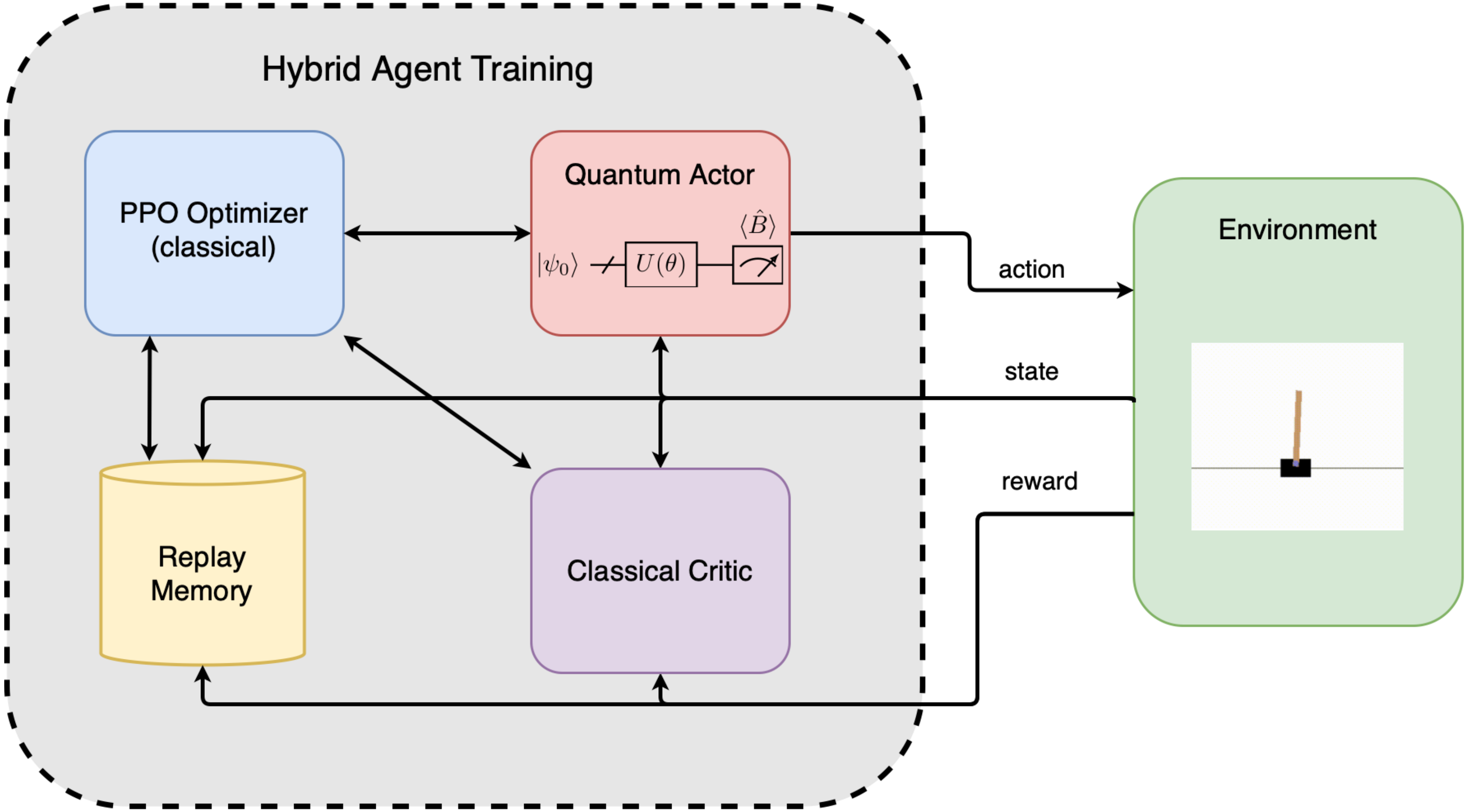
Clipped Surrogate Objective:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), \epsilon\right)\hat{A}_t\right)\right]$$

PPO Objective:

$$\boxed{\mathcal{L}^{\text{PPO}} = \mathcal{L}^{\text{CLIP}}(\theta) + c_1 S\left[\pi_\theta\right] + c_2 \text{Reg}(\theta)}$$

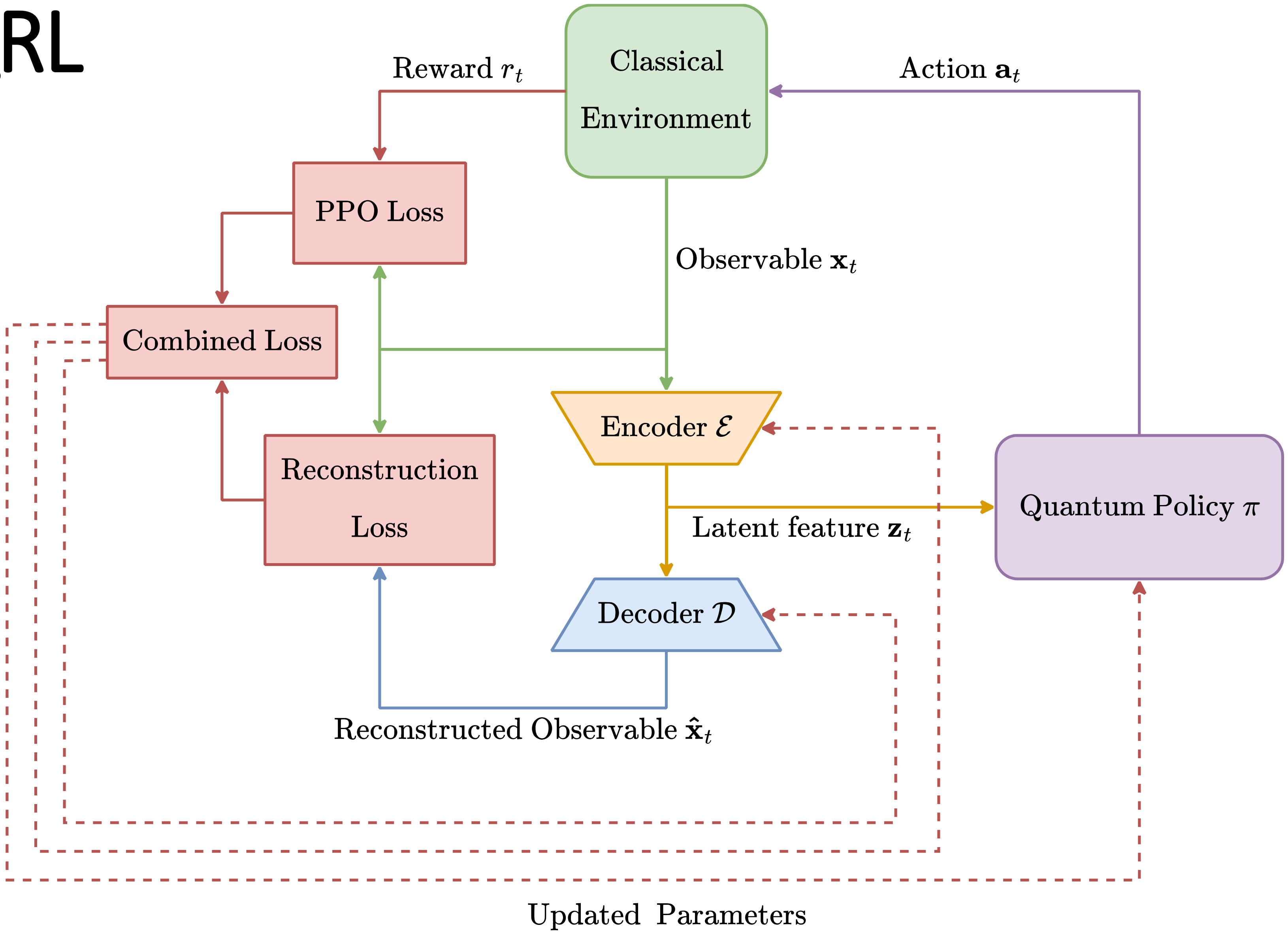
Quantum Reinforcement Learning with PPO



- Substitute the classical policy with a QNN
- Encode states into q-states, compute actions from measurements
- The rest of the system is classical
- Optimize the QNN policy parameters via gradient descent

Latent-space QRL

- As mentioned: environments often have high dimensional observables
- We use a classical AE for feature extraction, and encode latent features
- Classical AE: may be pretrained & frozen or trained together with the agent.

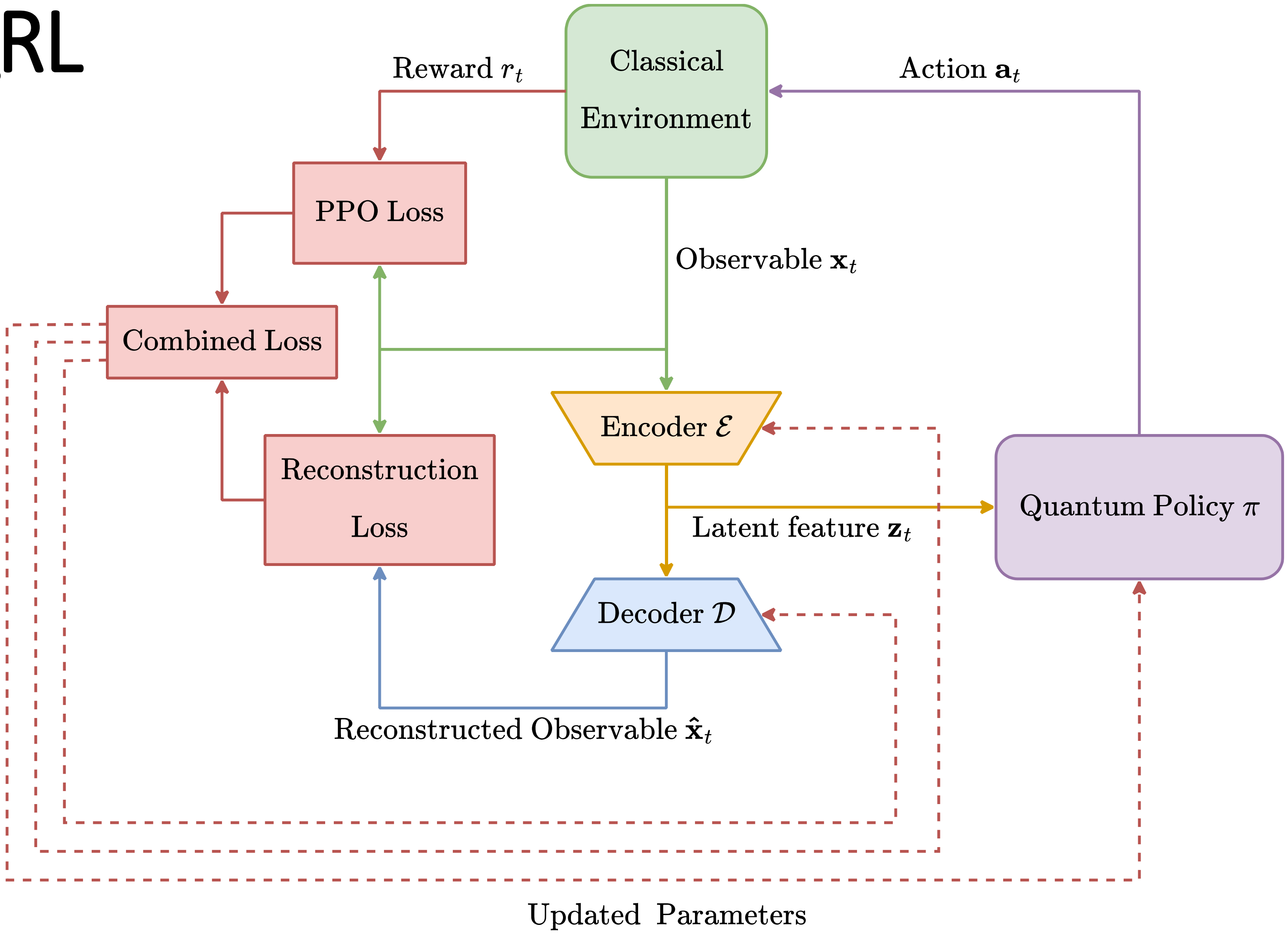


Latent-space QRL

- We optimize the hybrid system via a combined loss function:

$$\boxed{\mathcal{L}^{(PPO+AE)} = \mathcal{L}^{(PPO)} + c_{\mathrm{ae}}\mathcal{L}^{(AE)}}$$

- Optionally, the AE can be pre-trained



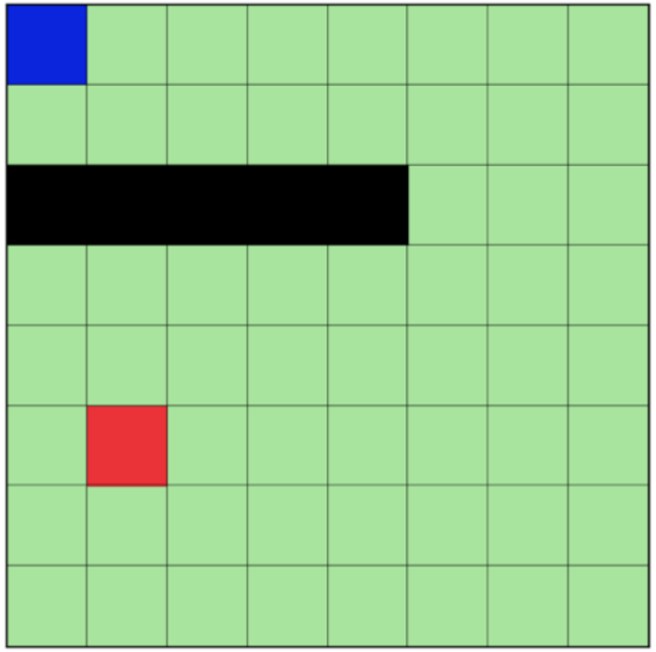
Numerical experiments

- We tested this approach with various configurations:
  - Two environments: Cartpole-v1 and Maze-v0
  - Various AE sizes, and various number of QNN layers
  - Both qubit-based and photonic QNNs
  - Compared with fully classical baselines

Numerical experiments

(a) Cart-pole balancing problem

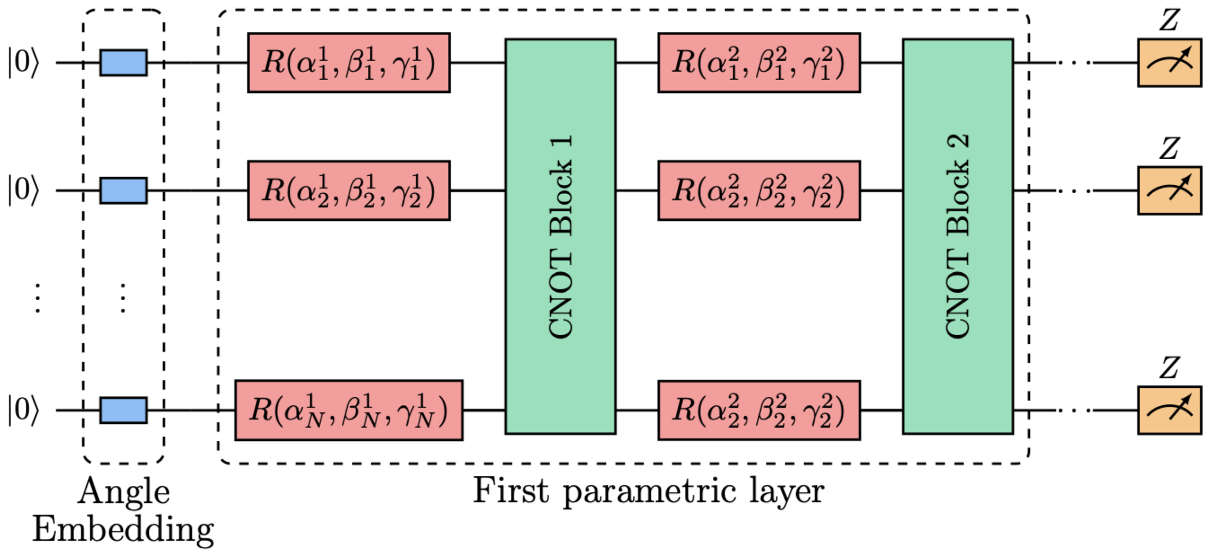
- CartPole-v1 environment
- dim(4) real-valued vector
- 2 possible actions: left, right
- Keep the vertical deviation less than 15 degrees

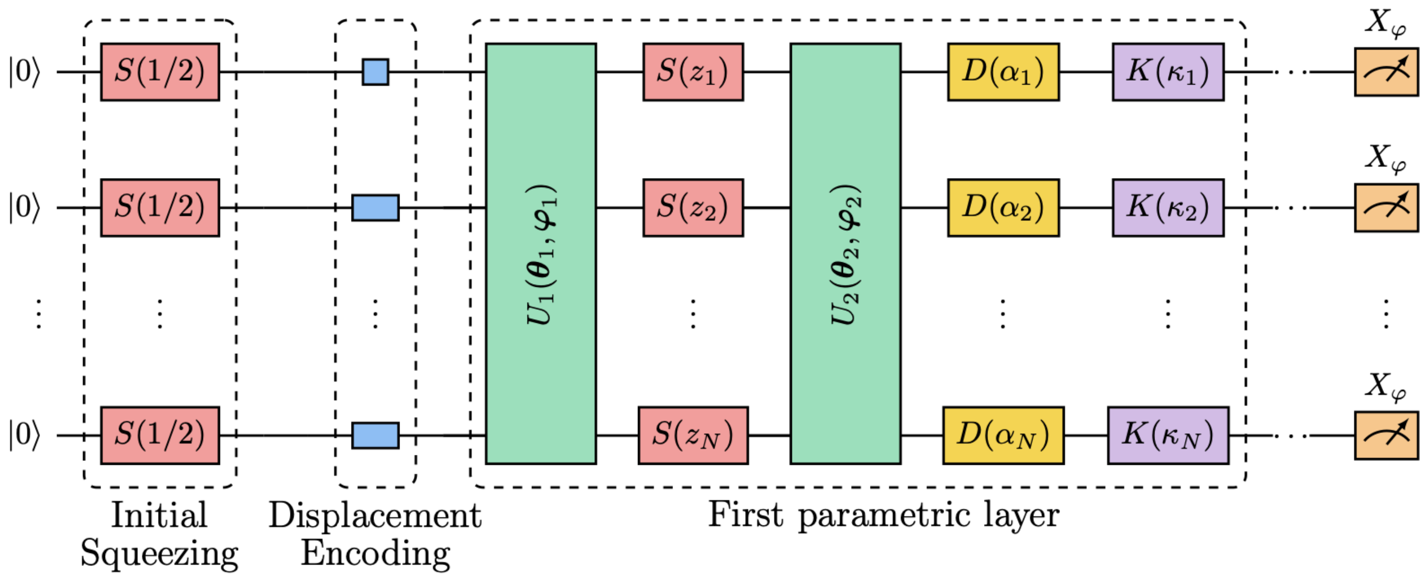


(b) Visual navigation problem

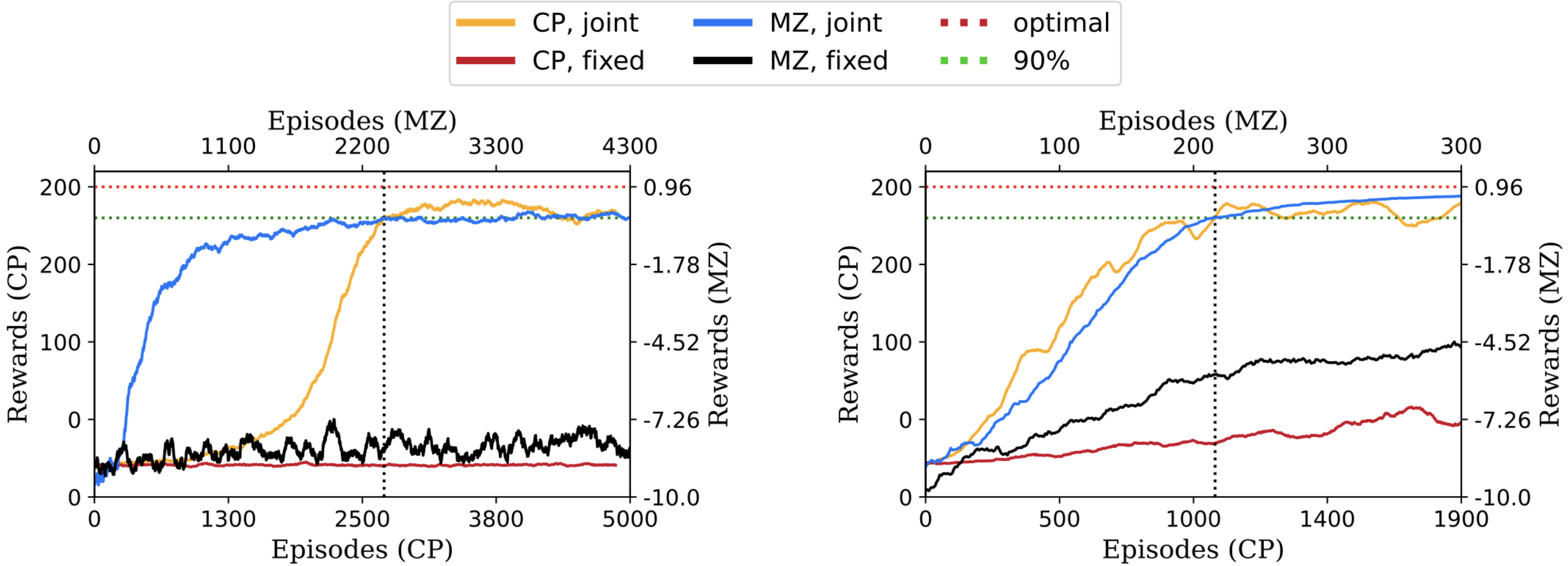
- Maze-v0 environment
- 48x48 grayscale image
- 4 possible actions: up, down, left, right
- The agent (blue) needs to find the target (red)

Numerical experiments





Numerical experiments

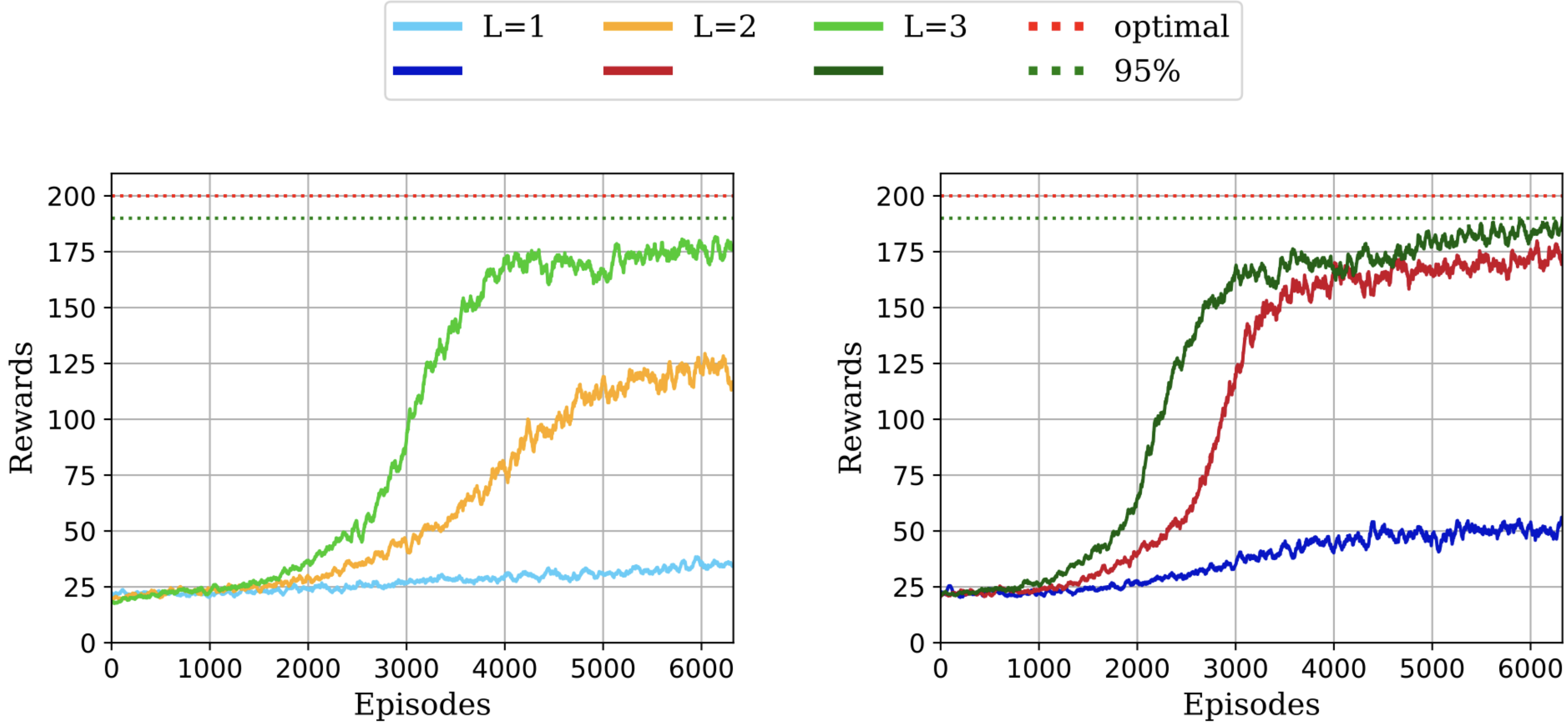


(a) Normalized learning curves for qubit based agents

(b) Normalized learning curves for photonic agents

Each curve is a smoothed average over five agents run in parallel

Numerical experiments

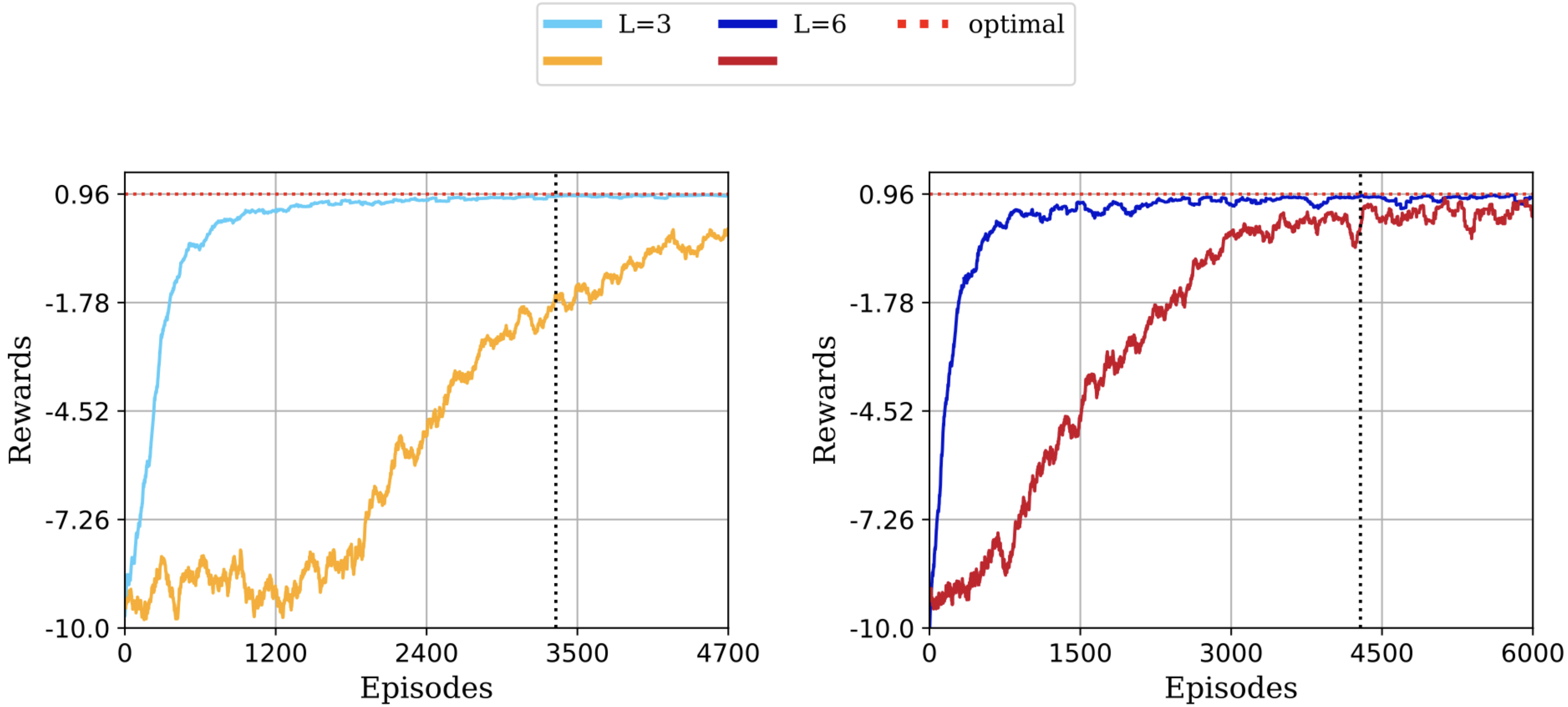


(a) Increasing number of QNN layers trained jointly with small AE

(b) Increasing number of QNN layers trained jointly with large AE

We compare the effect of increasing AE expressibility for different number of QNN layers. Experiments were run using the CartPole-v1 environment and qubit-based agents.

Numerical experiments



(a) Comparing photonic hybrid QRL agent with 3 layers and fully classical RL agent

(b) Comparing photonic hybrid QRL agent with 6 layers and fully classical RL agent

We compare 3- and 6-layer photonic AE+QNN agents (blue) with fully classical agents of similar parameter count (orange, red). Experiments were run on the Maze-v0 environment.

Conclusions

- We demonstrated that the AE+QNN method enables the application of QRL for high dimensional environments.
- We showed that jointly training the classical AE and a QRL agent is necessary for fast convergence.
- We see a tradeoff between AE size and QNN layer count
- Using photonic agents, the AE + QNN method can outperform the fully classical approach on the Maze-v0 environment

Conclusions

arxiv:241018284

Code on Github

Thank You

HUN REN | wigner

ERICSSON

Quantum Information National Laboratory HUNGARY

National Research, Development and Innovation Office Hungary

PROGRAM FINANCED FROM THE NRDI FUND

ELTE EÖTVÖS LORÁND UNIVERSITY

References

[1] Dunjko, V., Taylor, J.M., Briegel, H.J.: Advances in quantum reinforcement learning. In: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 282–287 (2017)

[2] Hoof, H., Chen, N., Karl, M., Smagt, P., Peters, J.: Stable reinforcement learning with autoencoders for tactile and visual data. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3928–3934 (2016)

[3] Killoran, N., Bromley, T. R., Arrazola, J. M., Schuld, M., Quesada, N., & Lloyd, S. (2019). Continuous-variable quantum neural networks. Phys. Rev. Research, 1, 033063. doi:10.1103/PhysRevResearch.1.033063

[4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553):436–444, May 2015. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

[5] Proximal policy optimization algorithms
J Schulman, F Wolski, P Dhariwal, A Radford, O Klimov - arXiv preprint arXiv:1707.06347, 2017

[6] Nagy, D., Tabi, Z., Hága, P., Kallus, Z., and Zimborás, Z., "Photonic Quantum Policy Learning in OpenAI Gym arXiv:2108.12926.

Suppplimentary information

Environment	platform	QNN layers	QNN params
`CartPole-v1`	qubit	1	6
`CartPole-v1`	qubit	2	12
`CartPole-v1`	qubit	3	18
`CartPole-v1`	qubit	6	32
`CartPole-v1`	qumode	1	14
`CartPole-v1`	qumode	3	42
`CartPole-v1`	qumode	6	84

Environment	platform	QNN layers	QNN params
`Maze-v0`	qubit	1	24
`Maze-v0`	qubit	3	72
`Maze-v0`	qubit	5	120
`Maze-v0`	qumode	1	94
`Maze-v0`	qumode	3	282
`Maze-v0`	qumode	6	564

Platform	Number of layers	CNN param count	convAE + QNN param count
qumode	1	519	487 (94 + 172 + 221)
qumode	3	731	675 (282 + 172 + 221)
qumode	6	974	957 (564 + 172 + 221)
qubit	1	517	491 (24 + 210 + 257)
qubit	3	556	539 (72 + 210 + 257)
qubit	5	609	587 (120 + 210 + 257)