Quantum Reinforcement Learning for Beam Steering

V. Kain, K. Li, M. Schenk, S. Vallecorsa
CERN, Switzerland

M. Popa
Politehnica University of Bucharest, Romania

E. F. Combarro
University of Oviedo, Spain

Quantum Technology Initiative Forum: Quantum Computing

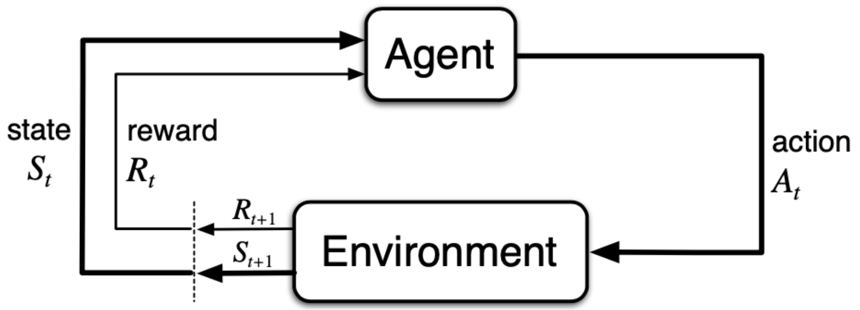
26. April 2021

Contents

- **Introduction:** RL in a nutshell
- **Motivation:** QBM vs DQN
- **Our project:** beam steering
- **Results:** DQN and QBM
- **Ongoing work:** QAOA and actor-critic

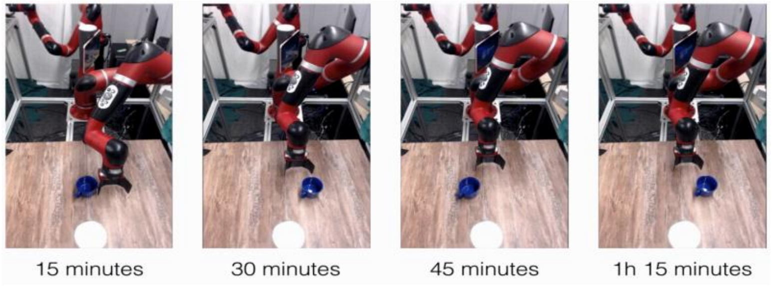
Reinforcement learning in a nutshell

RL book: Sutton & Barto



Agent interacts with environment

- **Receives reward after every action**
- Learns through **trial-and-error**



source

Decision making

- Agent follows certain **policy** $\boldsymbol{\pi}$: $S \to A$
- **Goal: find optimal policy** $\boldsymbol{\pi}^*$
- **Optimal ⬄ maximizing return:** $G_t = \sum_k \gamma^k R_{t+k}$

Expected return can be estimated through ***value function Q(s, a)***

- **"What's the best action to take in each state" => *greedy policy:*** *take action that maximizes Q(s,a)*
- **Not a priori known, but can be learned iteratively**
- **Q-learning** – learn Q(s, a) using **function approximator**
  - DQN: Deep Q-learning *(feed-forward neural network)*
  - QBM-RL *(Quantum Boltzmann Machine)*

Motivation

- **Why using QBM for RL?**
  - **Free energy based RL** (FERL): efficient for high-dim. spaces (https://www.jmlr.org/papers/volume5/sallans04a/sallans04a.pdf)
  - **Higher sample efficiency over Deep Q-learning** (https://arxiv.org/pdf/1706.00074.pdf)
  - **Quantum RL:** an exciting combination ☺
- **Objective:** apply to one of our RL problems: beam steering

Free energy-based reinforcement learning using a quantum processor

Anna Levit,[1] Daniel Crawford,[1] Navid Ghadermarzy,[1,2]
Jaspreet S. Oberoi,[1,3] Ehsan Zahedinejad,[1] and Pooya Ronagh[1,2,*]
[1]1QBit, 458-550 Burrard Street, Vancouver (BC), Canada V6C 2B5
[2]Department of Mathematics, The University of British Columbia, 121-1984 Mathematics Road, Vancouver (BC), Canada V6T 1Z2
[3]School of Engineering Science, Simon Fraser University, 8888 University Drive, Burnaby (BC), Canada V5A 1S6

Recent theoretical and experimental results suggest the possibility of using current and near-future quantum hardware in challenging sampling tasks. In this paper, we introduce free energy-based reinforcement learning (FERL) as an application of quantum hardware. We propose a method for processing a quantum annealer's measured qubit spin configurations in approximating the free energy of a quantum Boltzmann machine (QBM). We then apply this method to perform reinforcement learning on the grid-world problem using the D-Wave 2000Q quantum annealer. The experimental results show that our technique is a promising method for harnessing the power of quantum sampling in reinforcement learning tasks.

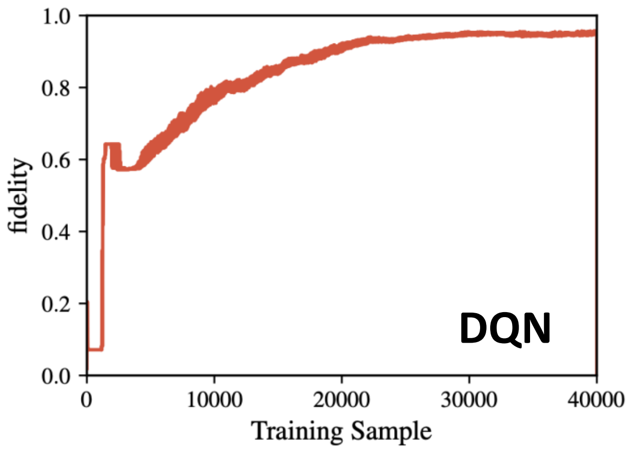
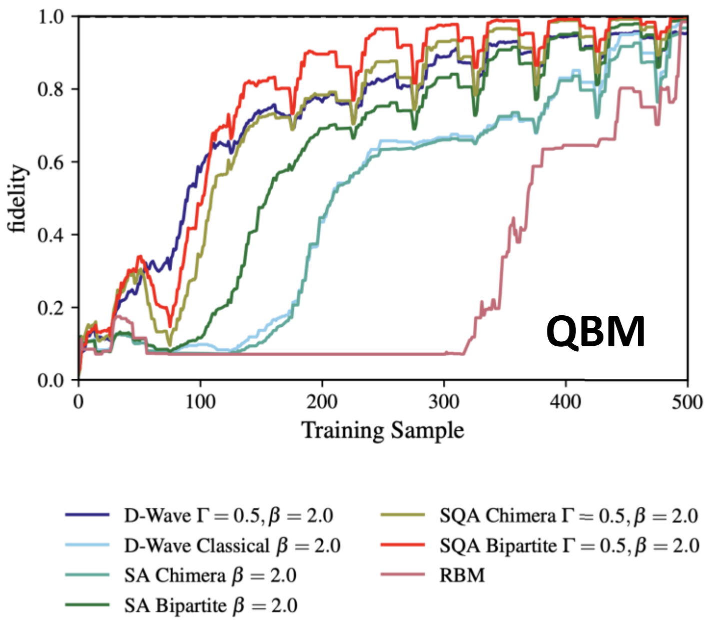


FIG. 4: The learning curve of a deep Q-network (DQN) with two hidden layers, each with eight hidden nodes, for the grid-world problem instance as shown in Fig. IV.



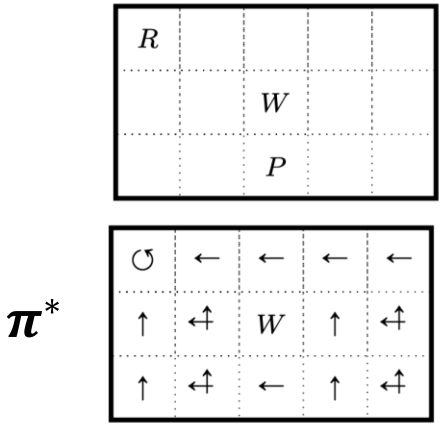


FIG. 3: (top) A 3×5 grid-world problem instance with one reward, one wall, and one penalty. (bottom) An optimal policy for this problem instance is a selection of directional arrows indicating movement directions.

Q-learning with QBM and DQN

FERL: clamped QBM

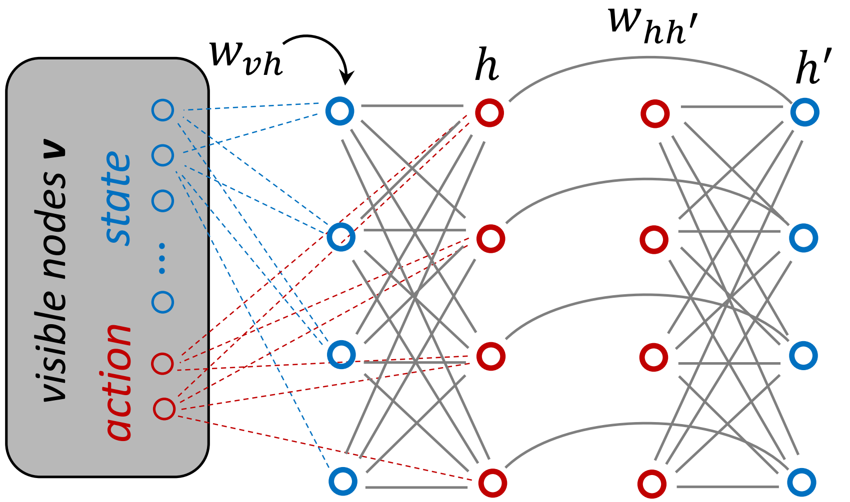
- **Network of coupled, stochastic, binary units** (spin up / down)
- $\hat{\boldsymbol{Q}}(\boldsymbol{s}, \boldsymbol{a})$ ≈ **negative free energy** of classical spin configurations c
- **Sampling** c using **(simulated) quantum annealing**
- **Clamped:** visible nodes not part of QBM; accounted for as biases
- Here **visible nodes are discrete, binary** *(restriction can be lifted)*
- **Using 16 qubits of D-Wave Chimera graph**

DQN: Q-net

- **Feed-forward, dense** neural network
- 2 hidden layers, 8 nodes each (≈ Chimera graph)

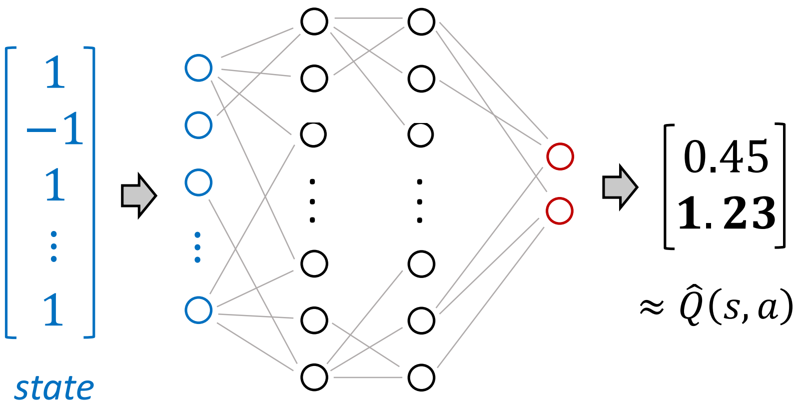
Learning: update Q by applying **temporal difference rule** to QBM and Q-net weights

Clamped QBM

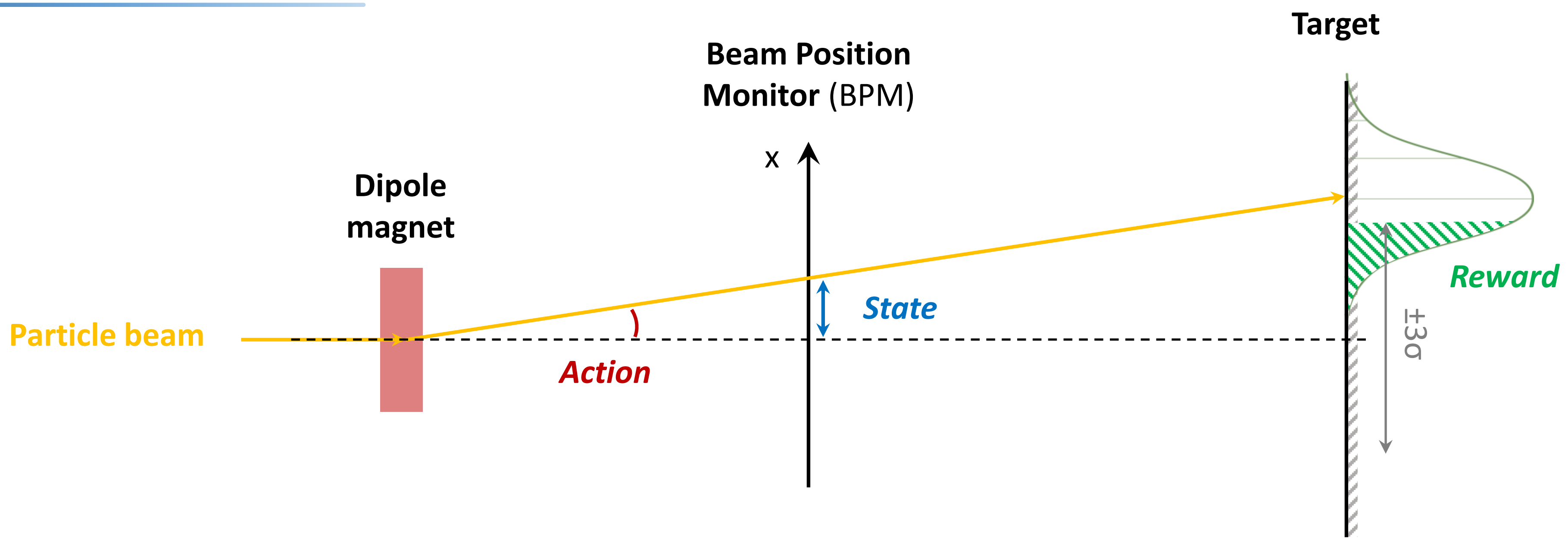


$$\hat{Q}(s,a) \approx -F(\boldsymbol{v}) = -\langle H_{\boldsymbol{v}}^{\mathrm{eff}} \rangle - \frac{1}{\beta} \sum_c \mathbb{P}(c|\boldsymbol{v}) \log \mathbb{P}(c|\boldsymbol{v})$$

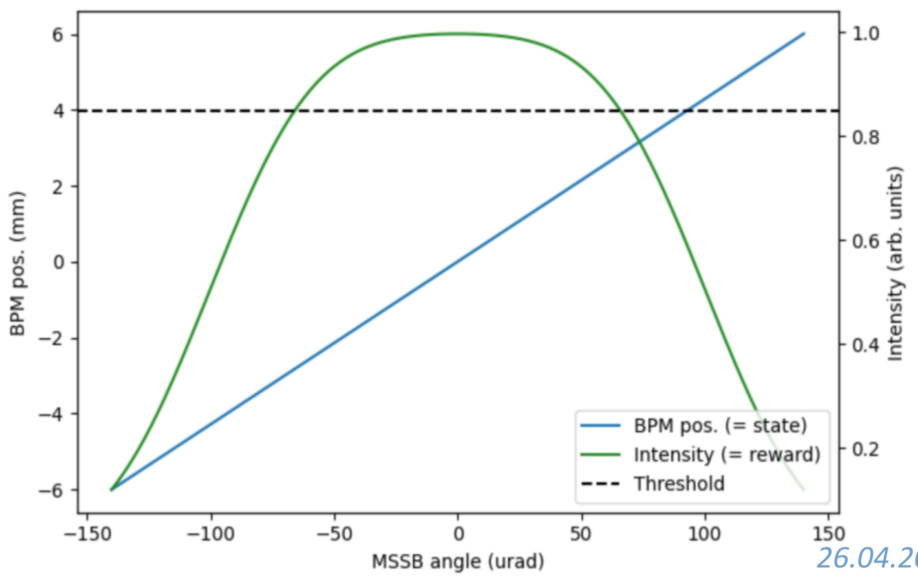
Q-net



Toy model: beam steering

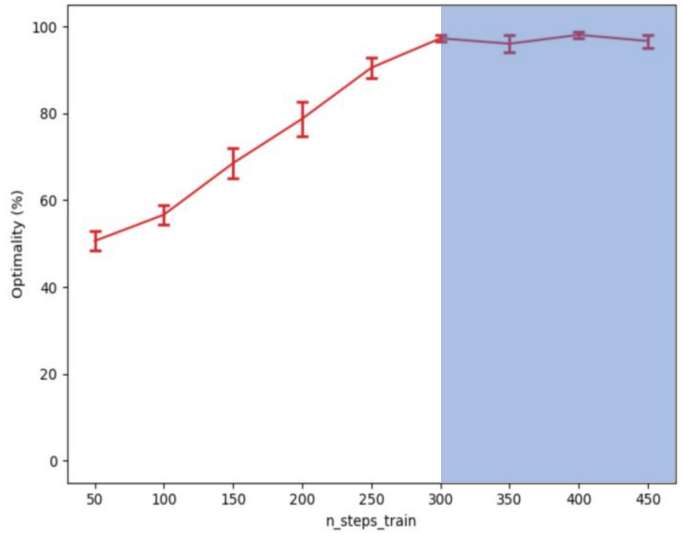


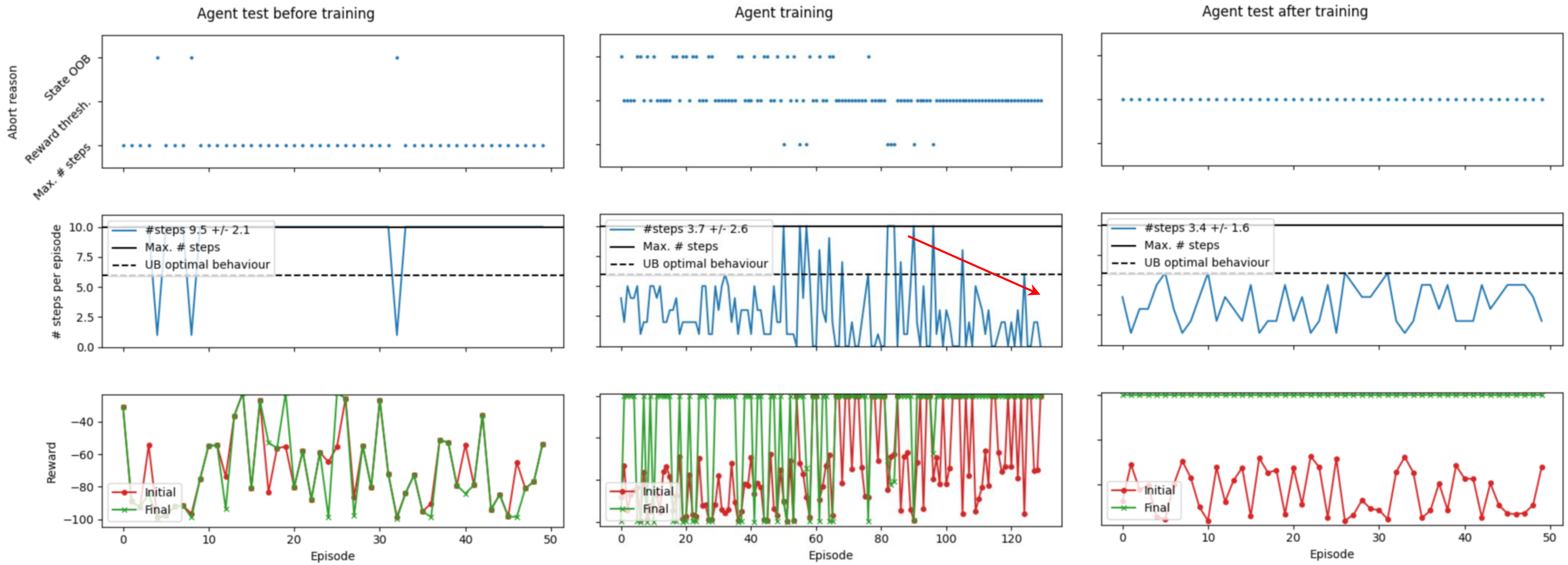
- **Toy model based on actual steering problem**, e.g. for fixed target experiments at CERN Super Proton Synchrotron
- [*OpenAI gym template*]
- **Action:** deflection angle
  - 2 possibilities: up or down by fixed amount
- **State:** beam position at BPM
- **Reward:** integrated beam intensity on target



DQN: discrete state space

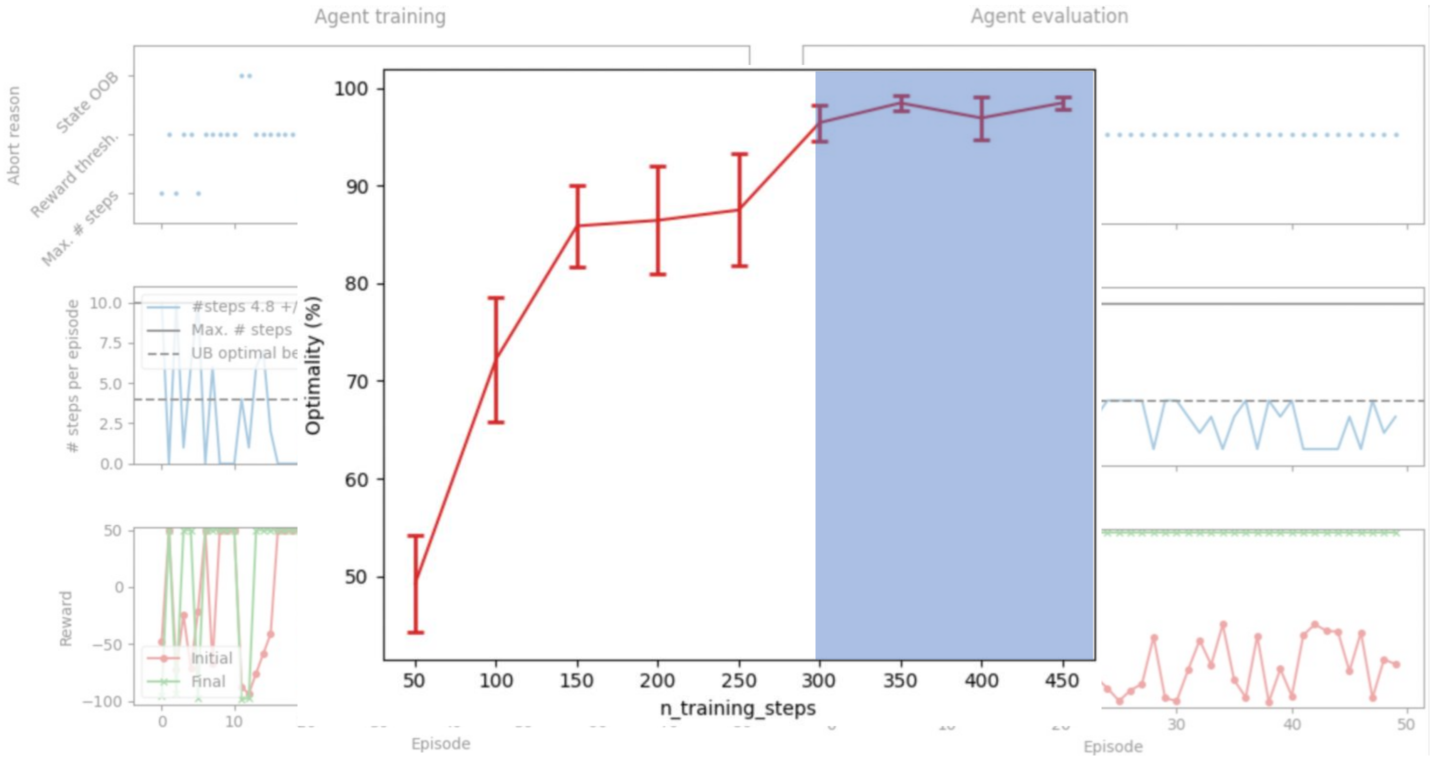
- *Stable-baselines3* implementation of DQN
- **Efficiency:** required **# training_steps** *after hyperparameter tuning*
- **300+** training steps: get optimal policy with nearly **100% success rate**

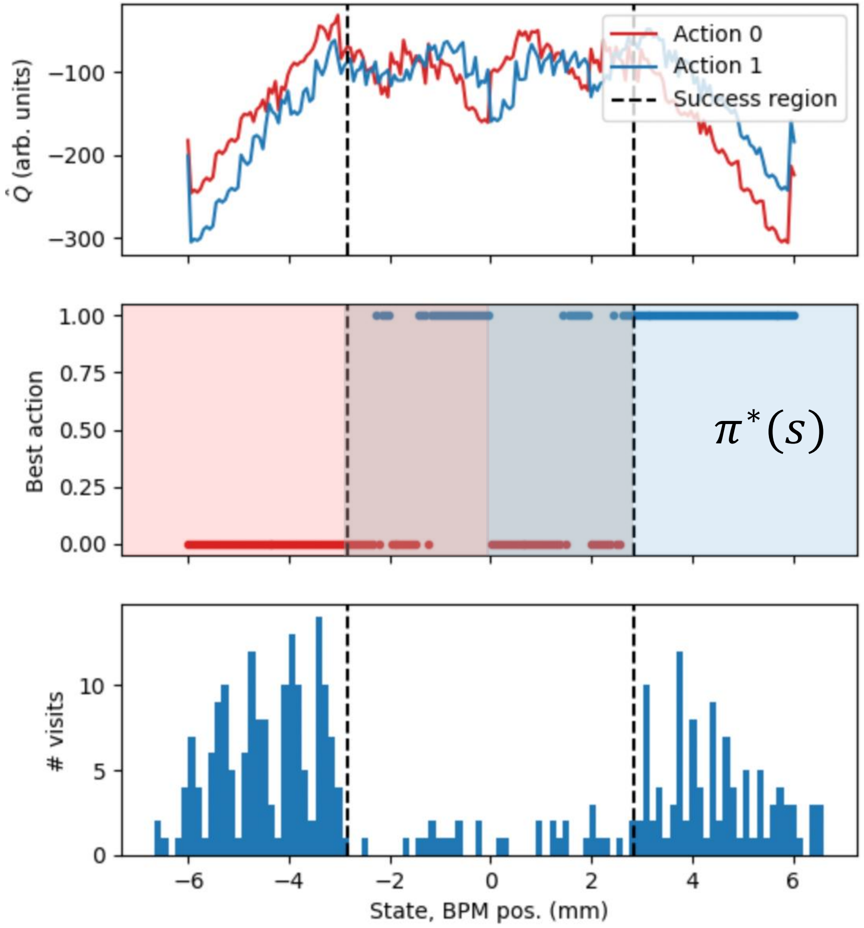




QBM: discrete state space, simulated quantum annealing

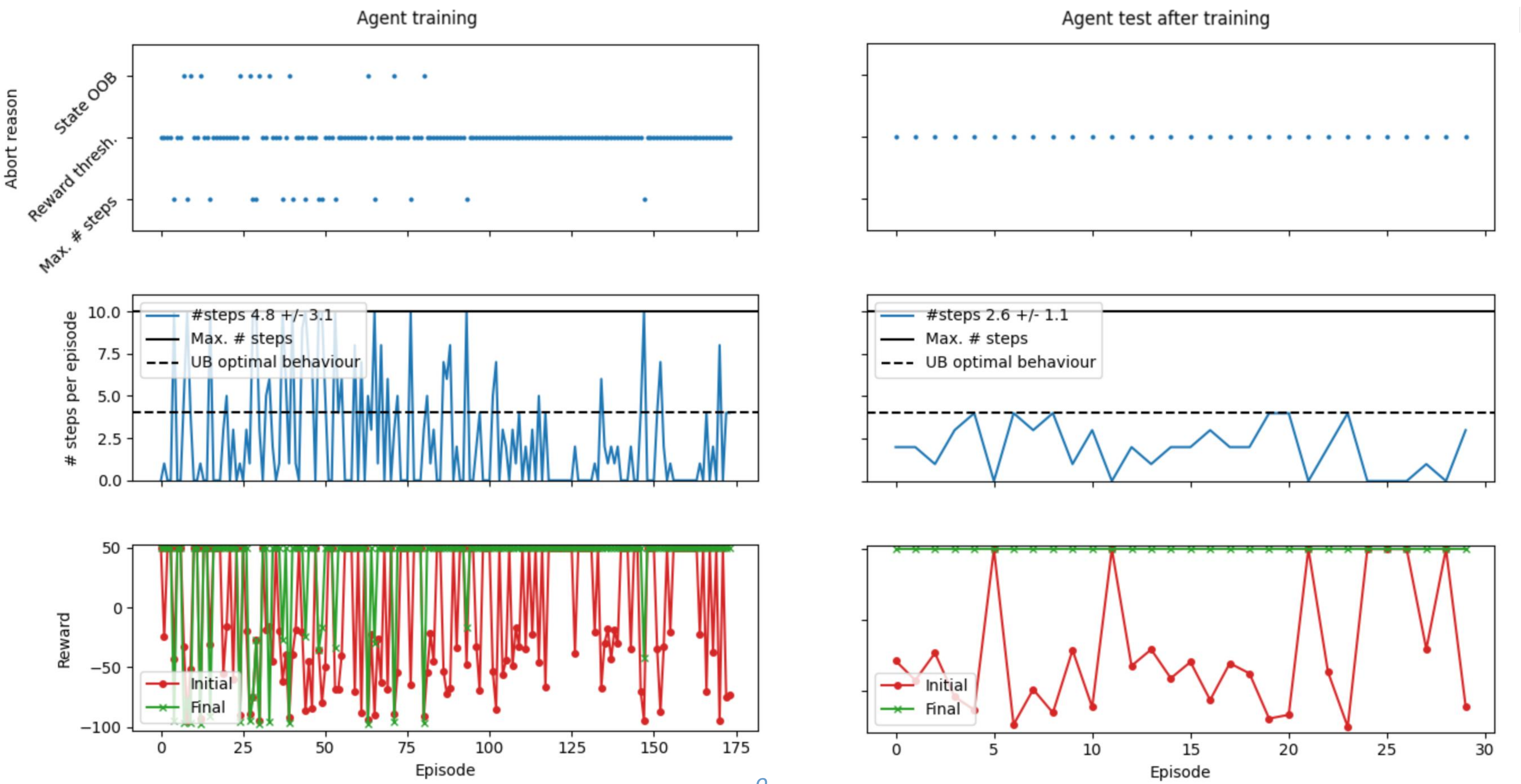
- **Tune QBM-RL with simulated quantum annealing** (SQA, *library: sqaod*) before moving on D-Wave QPU
- With some tuning: **successful training** *(300 iterations)*
- $\hat{Q}(s, a)$ leads to optimal policy
- **Similar efficiency to DQN**





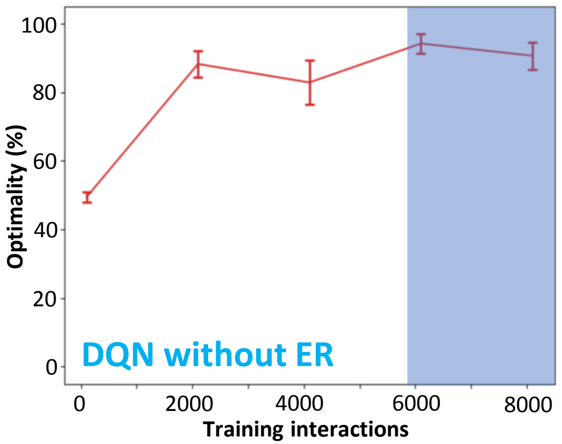
QBM: discrete state space, D-Wave 2000Q

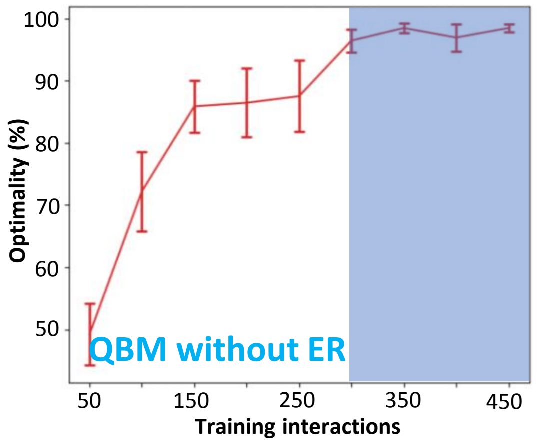
- **D-Wave training from scratch** *(600 iterations)* after hyperparameter tuning with SQA
- **Our first successful RL training on an actual QPU ☺ !**

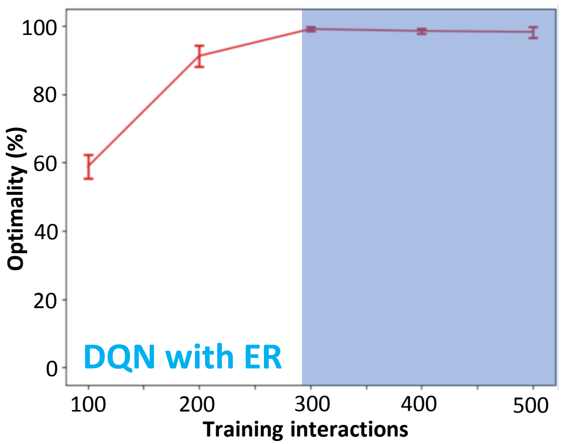


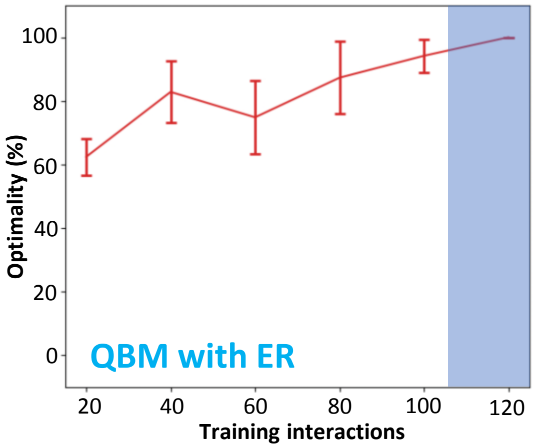
DQN vs QBM: effect of experience replay

- DQN vs QBM: **roughly same number of training interactions** required
- **Not consistent** with [original paper](#) *(40’000 vs. 500 interactions)*
- **Reason:** experience replay (ER)
  - **DQN:** 6000+ interactions (w/o ER) vs ~300 interactions (w/ ER)
  - **QBM:** ~300 interactions (w/o ER) vs ~120 interactions (w/ ER)



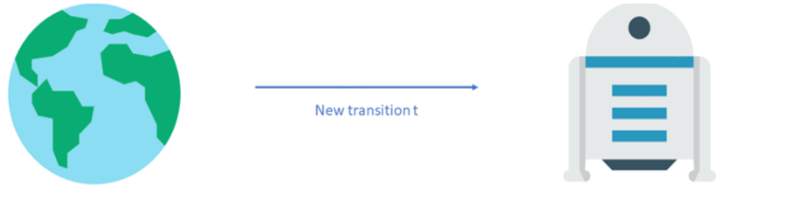






Online Learning

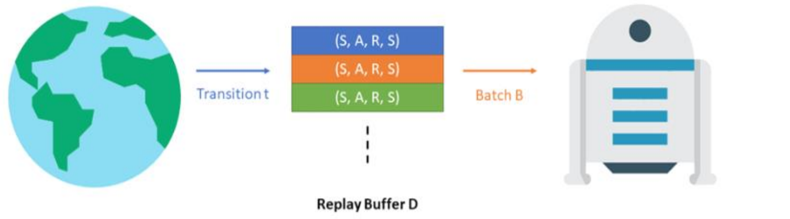
- Learn directly from latest experience
- Highly correlated data
- Agent learns from each interaction once and discards it immediately after



endtoend.ai

Experience Replay

- Save transitions into memory buffer
- Sample batch B from buffer to train agent at every step



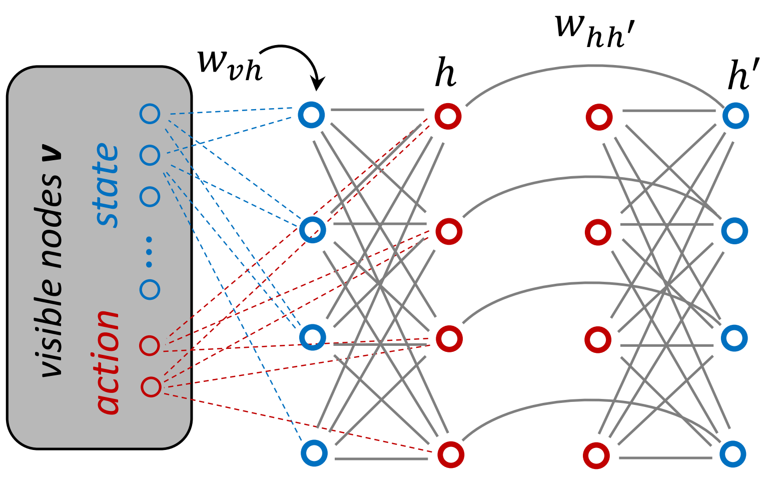
endtoend.ai

https://www.endtoend.ai/paper-unraveled/cer/

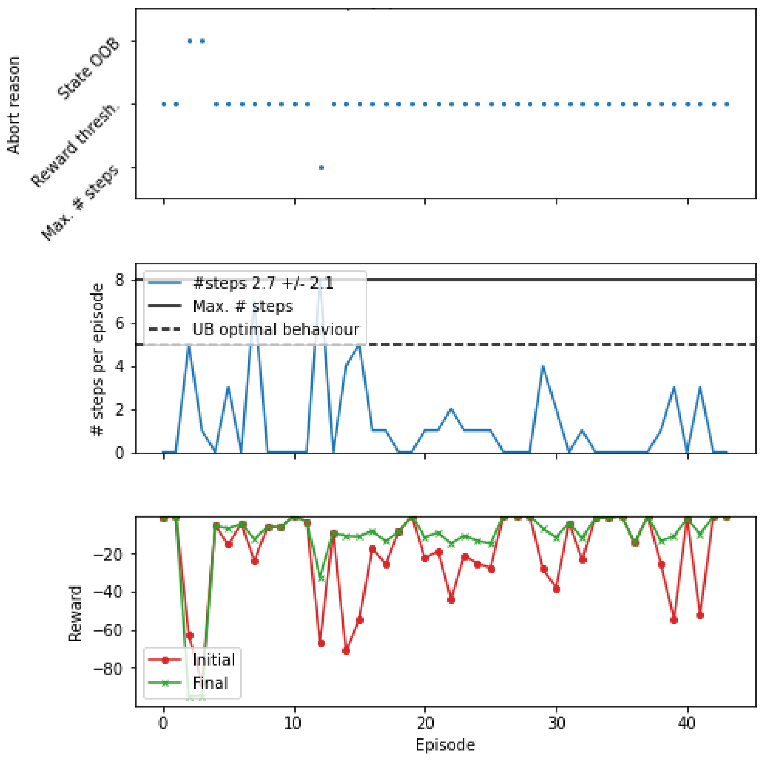
QBM: continuous state space

- **Visible nodes not** represented by **qubits => no need to be discrete, binary**
- **Training on D-Wave with continuous state space and ER**: ~120 interactions
- **Q functions more robust** thanks to smaller number of training weights
- **Opens doors for more complex and more practical applications**

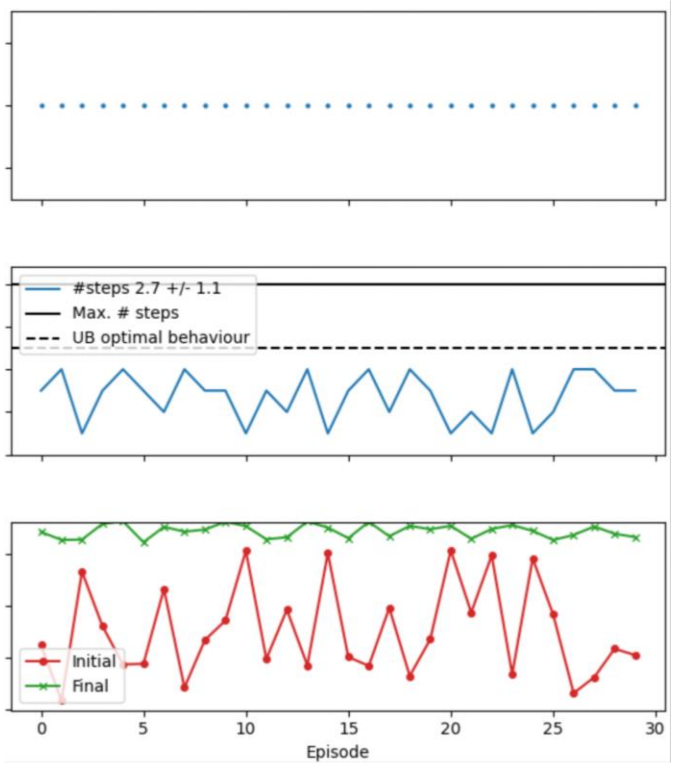
Clamped QBM



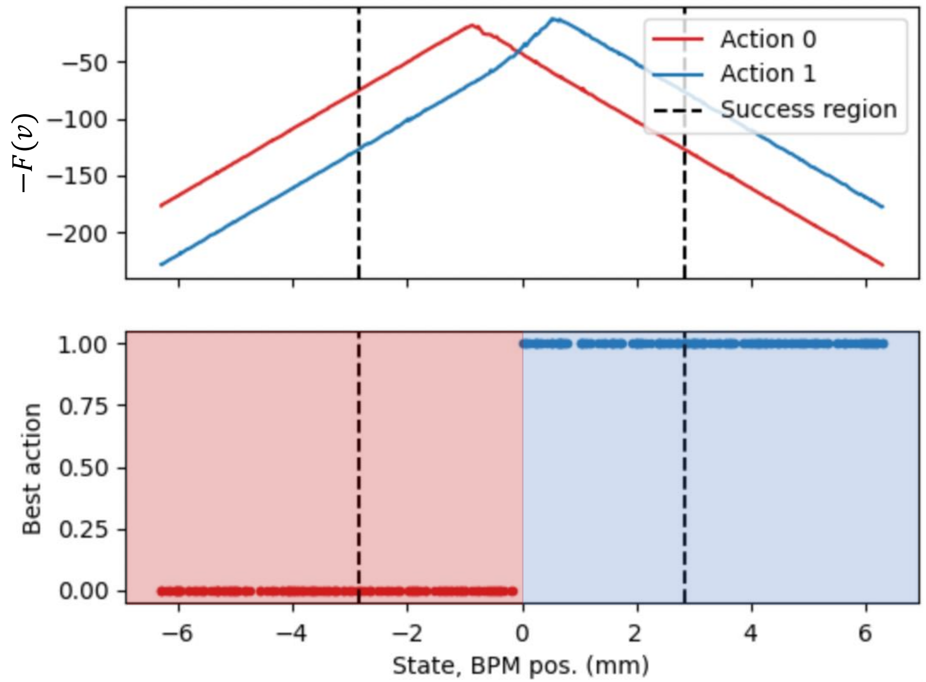
Training (D-Wave)



Evaluation (SQA)

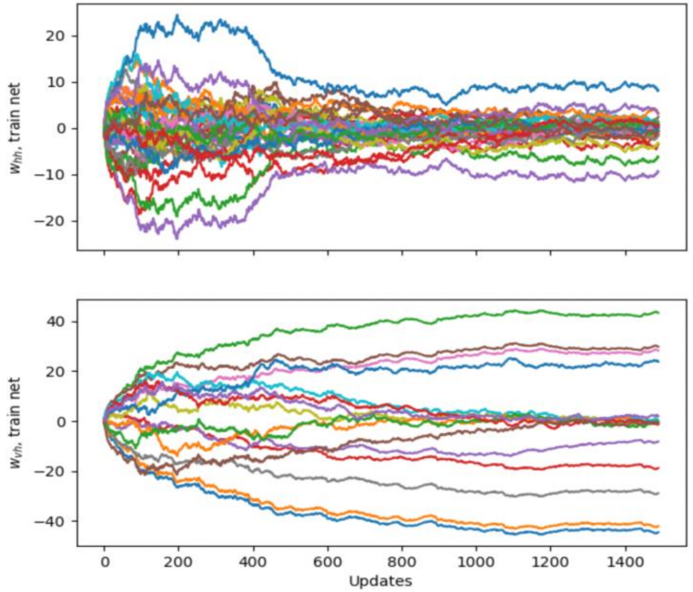


Q-functions (SQA)

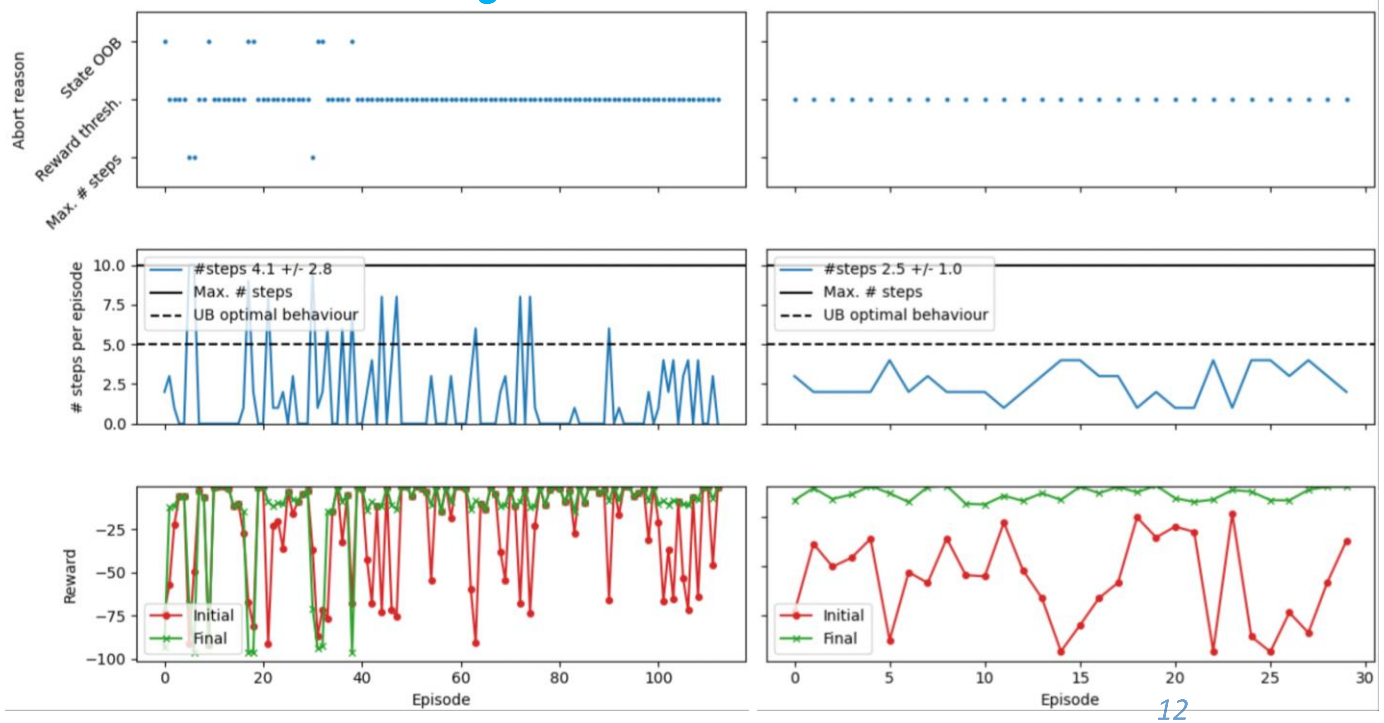


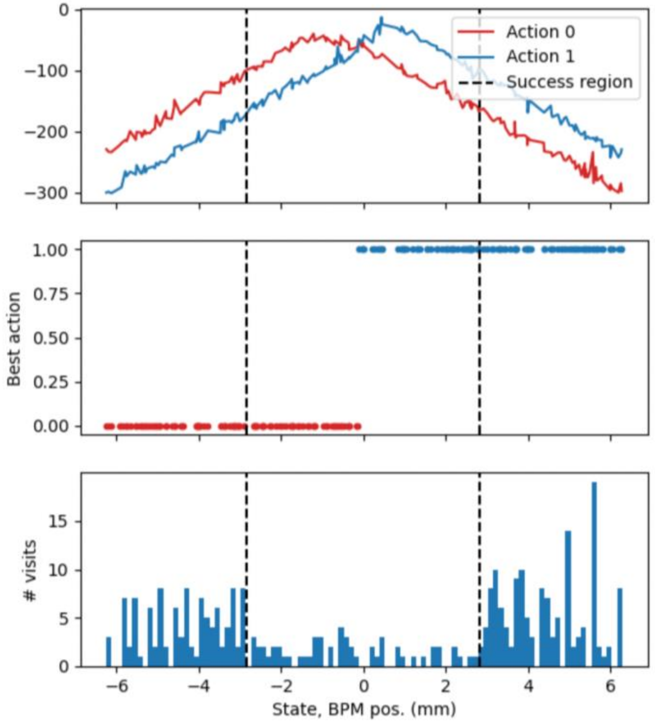
Ongoing work: QBM using QAOA

- [**QAOA:** Quantum Approximate Optimization Algorithm]
- **Solver** for combinatorial optimization problems: find spin configuration with minimum energy, **not based on annealing**
- **Works well,** but quite **compute-intensive** *(~5.5 h for 100 interactions)*
- On hardware (e.g. IBM): to be tested, could be affected by noise



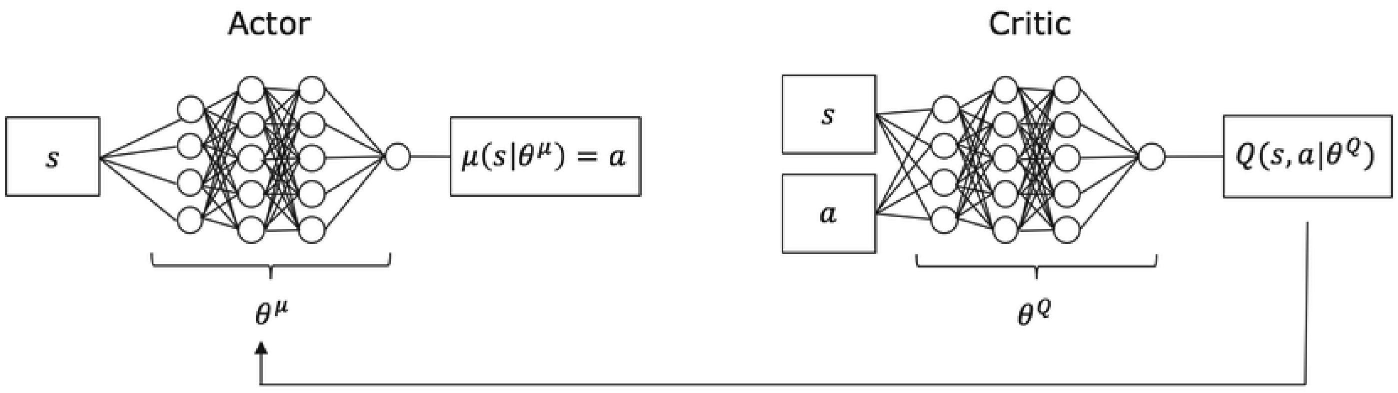
Training ***Evaluation***





Ongoing work: actor-critic

- **Goal: continuous state *and* action spaces** to tackle **real-world problems**
- **DQN not suitable:** only for discrete, low-dimensional action spaces
- **Actor-critic algorithm** *[Deep Deterministic Policy Gradient (DDPG)]*
  - **Actor** (= policy network): parameterized action function, mapping states to actions
  - **Critic** (= Q-net): similar to DQN, estimator for Q(s,a)



Policy Gradient: $\nabla_{\theta^\mu}\mu = \mathbb{E}_\mu[\nabla_{\theta^\mu}Q(s,\mu(s|\theta^\mu)|\theta^Q)] = \mathbb{E}_\mu[\nabla_a Q(s,a|\theta^Q)\cdot\nabla_{\theta^\mu}\mu(s|\theta^\mu)]$

- **Plan is to create hybrid:** replace Q-net by QBM; keep classical NN for actor

Summary and outlook

Summary

- **Comparison between Deep Q-learning** *(Q-net)* **and Free Energy Based RL** *(QBM)*
- **QBM** works for both **discrete and continuous state space**
- It can be trained successfully with **SQA, D-Wave hardware, and QAOA simulator**
- **Experience replay** has an **important impact** on the training efficiency *(here: factor ~3)*
- First steps made towards **continuous action space using DDPG**

Outlook

- Participate in *BQIT:21* workshop with poster presentation *(26.04. to 28.04.)*
- **Finish actor-critic implementation**
- **Continue studies with QAOA**
- Move to **more complex, higher dimensional environment**

Thank you !

Backup

FERL

https://www.jmlr.org/papers/volume5/sallans04a/sallans04a.pdf

- In RL: need to **estimate action-value functions in high dimensional state-action space** where not all state-action pairs can be visited (e.g. 2^{40})
- Can no longer use table: **use function approximator $\widehat{Q}(s, a)$**
- Conditions: need to be able to **calculate derivative of $\widehat{Q}$ wrt. its weights** to train using TD rule
- One option: **Product of Experts (PoE) models**
    - Combine **simple probabilistic models by multiplying** their probability distributions with each other
    - e.g. stochastic binary units of BM
- **Free energy of such models** can be used as approximator of value function, but needs training for different visible nodes (state-action pairs)
- Once trained, **sampling according to PoE will give probability distribution over actions** given a fixed state (Boltzmann exploration policy)

$$P(\mathbf{a}|\mathbf{s}) = \frac{e^{-F(\mathbf{s},\mathbf{a})/T}}{Z} \approx \frac{e^{Q(\mathbf{s},\mathbf{a})/T}}{Z}$$

- Intuition: **good actions sampled more likely than bad ones**
- **Probabilistic nature** provides advantage in large state-action spaces compared to traditional NN

FERL: Clamping

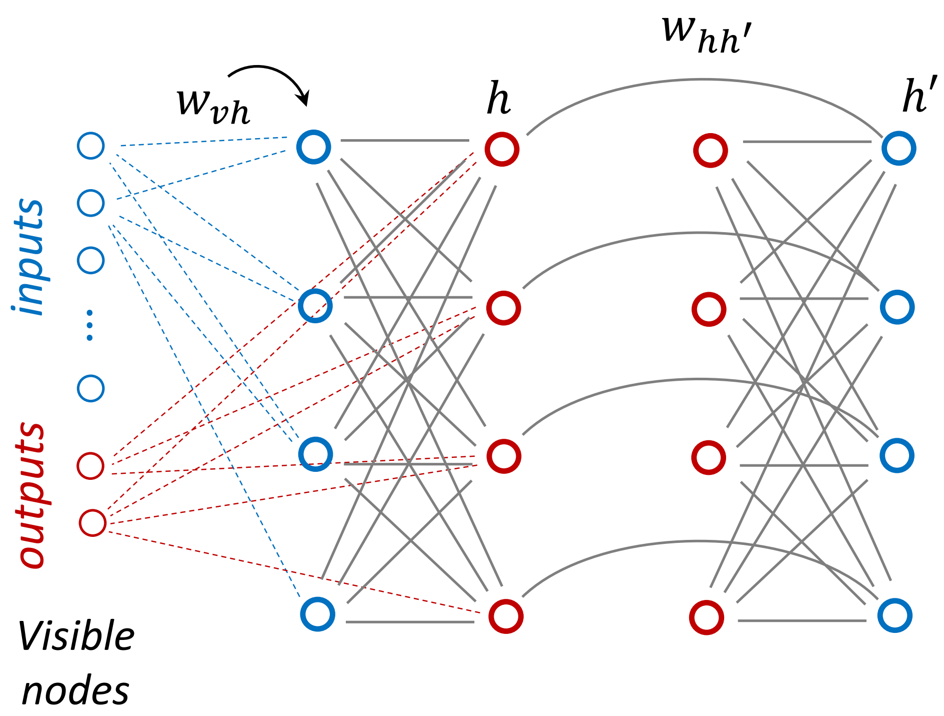
- **All nodes of QBM are hidden**
- **Clamping: add visible nodes as self-couplings** (biases) to hidden nodes they are connected to **and remove them from the graph**

- Every **spin configuration has specific energy** described by Hamiltonian of the transverse-field Ising model

$$\mathcal{H}_{\mathbf{v}} = -\sum_{v\in V,\, h\in H} w^{vh} v \sigma_h^z - \sum_{\{h,h'\}\subseteq H} w^{hh'} \sigma_h^z \sigma_{h'}^z - \Gamma \sum_{h\in H} \sigma_h^x$$

Γ: transverse field strength, $\sigma^{x,z}$: Pauli spin matrices

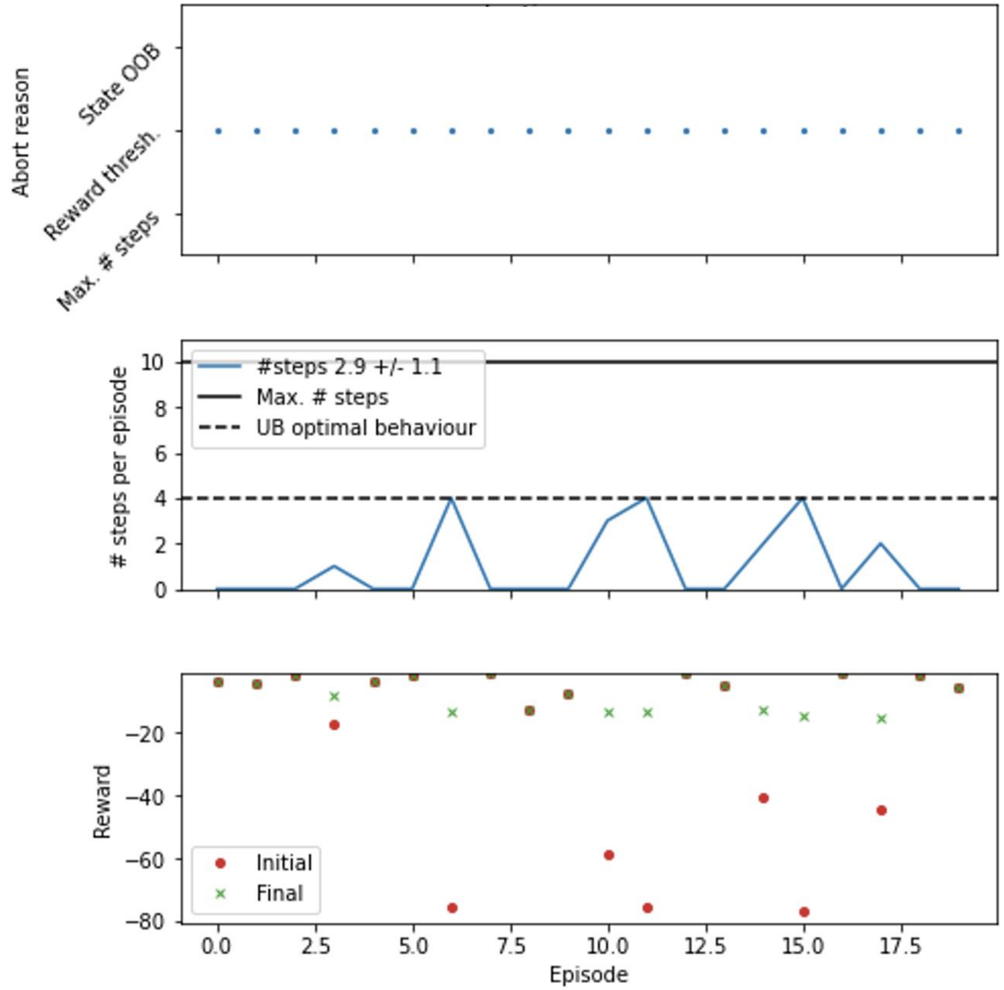
- Once **we measure spin in z direction**, we no longer have access to transverse component **=> cannot know system's energy**
- Can be fixed using replica stacking (Suzuki-Trotter expansion)
  see https://arxiv.org/pdf/1706.00074.pdf and refs. therein

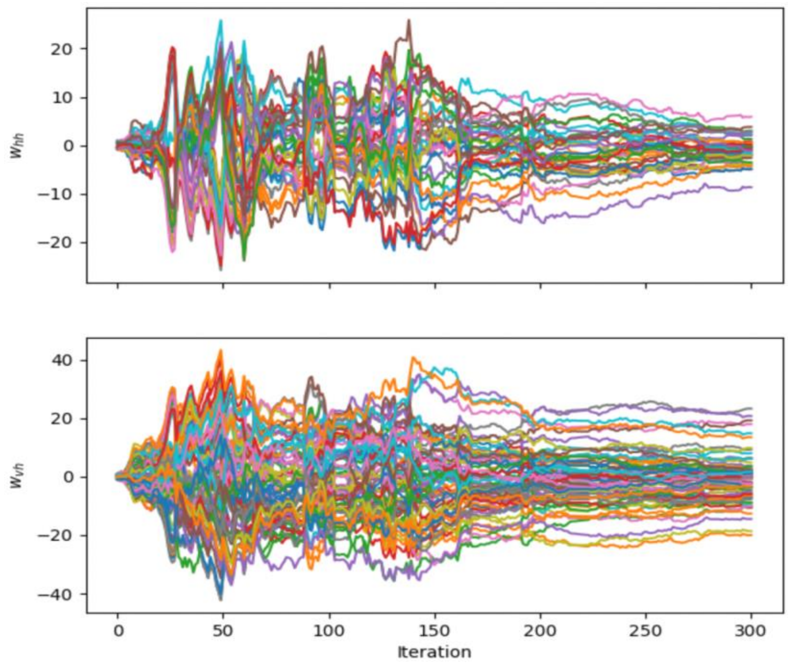


QBM: results on D-Wave 2000Q, part I

- AWS Braket platform: D-Wave 2000Q
- **First trainings not successful:** hyperparameter scans on hardware too expensive
- Train QBM with SQA and **reload trained weights on D-Wave**
- **Evaluation on D-Wave looks promising!**

SQA agent evaluation on D-Wave 2000Q



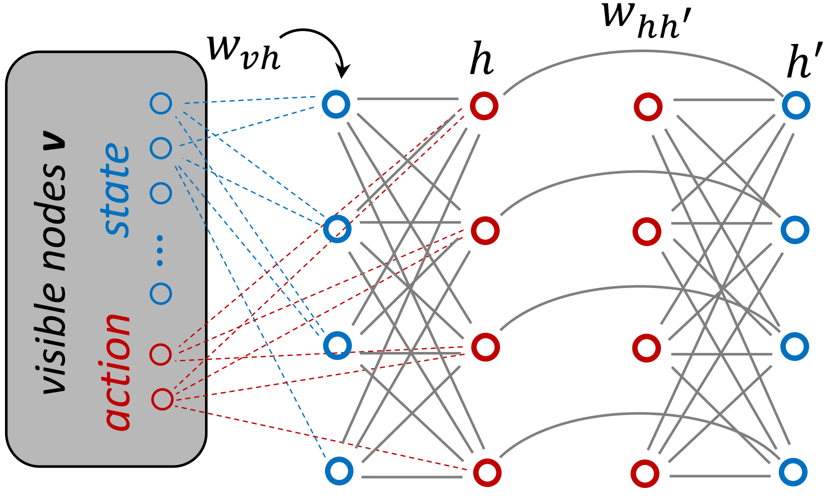


Evolution of QBM weights during training with SQA

QBM: continuous state space I

- **Major limitation: discrete, binary state space**
  - E.g. here we use 8 nodes => 256 bins
  - Limited resolution, limited state space dimension, large number of coupling weights, slow, training less robust
- **QBM is clamped**
  - Visible nodes are **not actually represented by qubits**, which are binary by definition (spin up / down)
  - They **enter system only as biases** => no need to be discrete, binary
- **Continuous state space possible**
  - Opens doors for more complex systems and more practical applications
  - Later today: actor-critic setup

Clamped QBM



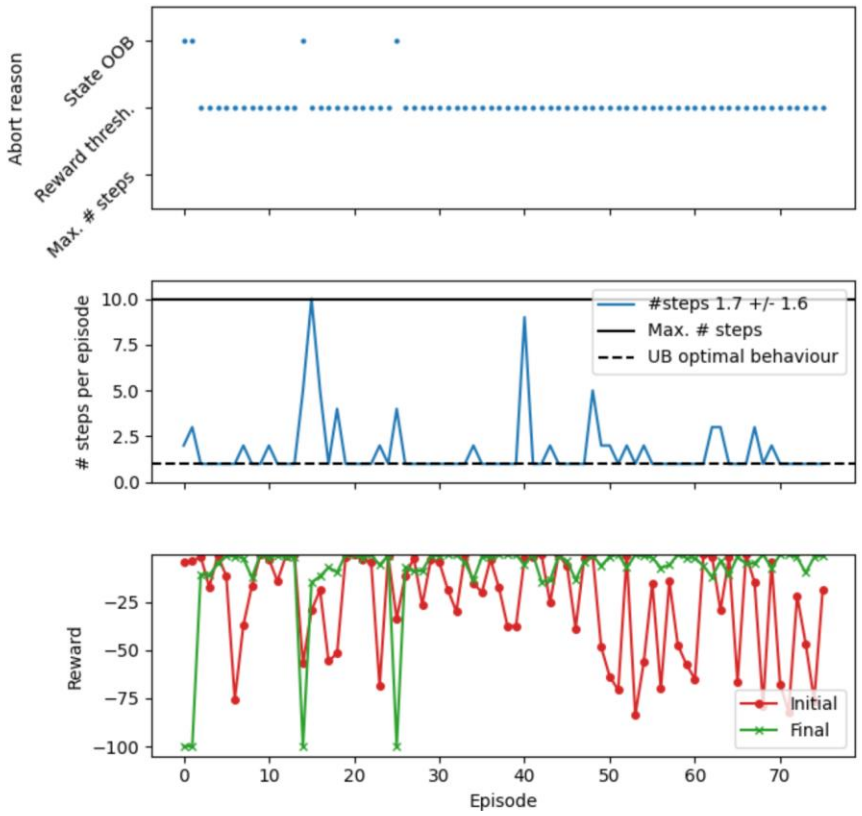
$$\hat{Q}(s,a) \approx -F(\boldsymbol{v}) = -\langle H_v^{\text{eff}} \rangle - \frac{1}{\beta} \sum_c \mathbb{P}(c|\boldsymbol{v}) \log \mathbb{P}(c|\boldsymbol{v})$$

$$H_v^{\text{eff}} = -\sum_{v \in V,\, h \in H} w_{vh}\, v\, \sigma_h^v - \sum_{h,h' \in H} w_{hh'}\, \sigma_h^z \sigma_{h'}^z$$

Ongoing work: actor-critic II

- **Step 1: test with our implementation of DDPG**
  - Already separates actor and critic for easier replacement of Q-net (step 2)
  - With continuous action space: **optimal behaviour means 1 step is enough** to solve the problem
  - **Works well**

Training



Evaluation