Quantum Reinforcement Learning for Beam Steering

V. Kain, K. Li, M. Schenk
BE-OP-SPS, CERN, Switzerland

E. F. Combarro[*], M. Popa, S. Vallecorsa
CERN Openlab, Switzerland
**also at University of Oviedo, Spain*

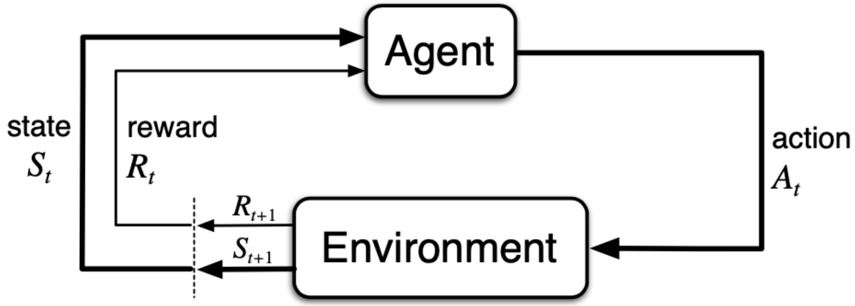
Contents

Reinforcement learning in a nutshell

RL book: Sutton & Barto

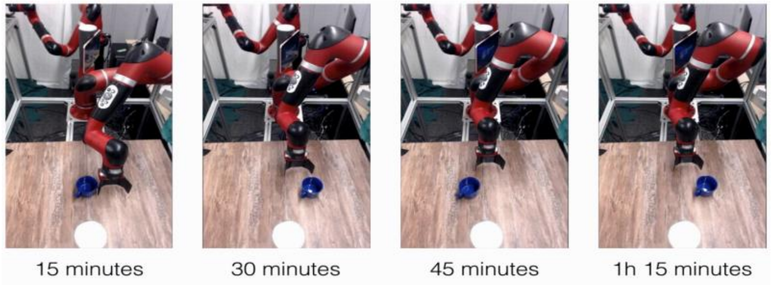
Agent interacts with environment

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



Decision making

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



source

Expected return can be estimated by ***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**
- *This talk:* **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.

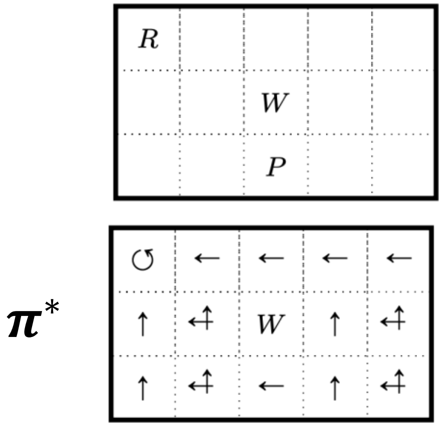


FIG. 3: (top) A 3×5 grid-world problem instance with one reward, one wall, and one penalty. (bottom) An optimal policy for this problem instance is a selection of directional arrows indicating movement directions.

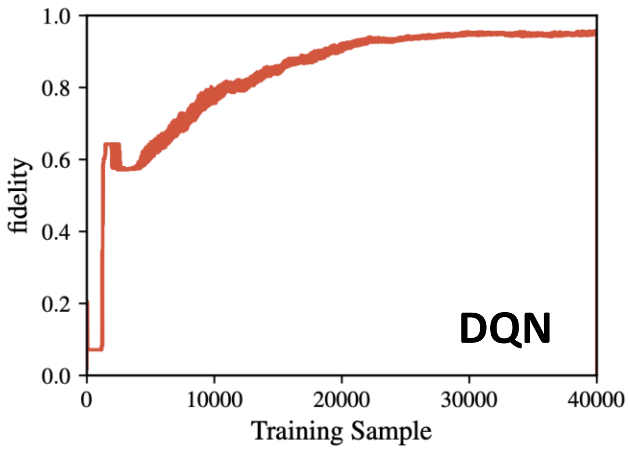
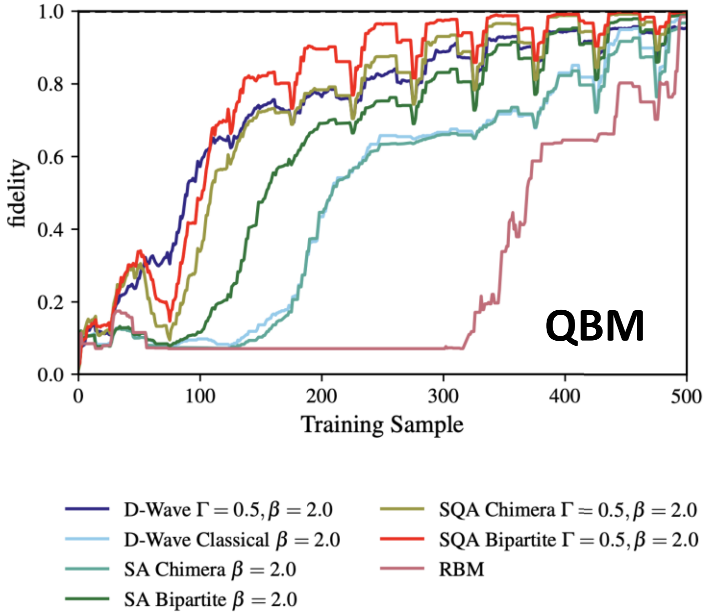


FIG. 4: The learning curve of a deep Q-network (DQN) with two hidden layers, each with eight hidden nodes, for the grid-world problem instance as shown in Fig. IV.

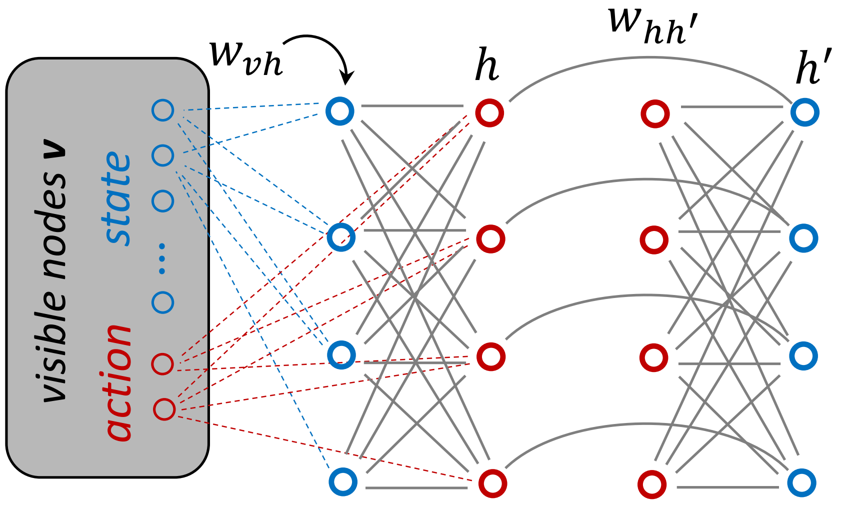


Q-learning with QBM and DQN

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
- **Using 16 qubits of D-Wave Chimera graph**
- **Discrete, binary-encoded** state and action spaces

Clamped QBM

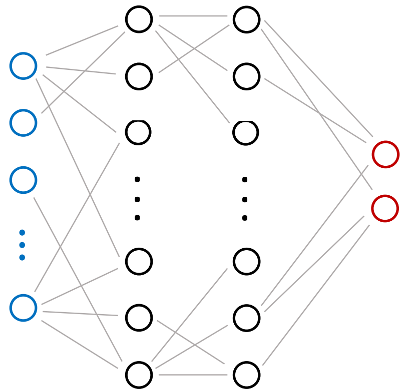


$$\hat{Q}(s,a) \approx -F(\boldsymbol{v}) = -\langle H_{\boldsymbol{v}}^{\text{eff}} \rangle - \frac{1}{\beta}\sum_{c} \mathbb{P}(c|\boldsymbol{v}) \log \mathbb{P}(c|\boldsymbol{v})$$

DQN: Q-net

- **Feed-forward, dense** neural network
- 2 hidden layers, 8 nodes each (≈ Chimera graph)
- Can handle **discrete, binary-encoded** state and action spaces

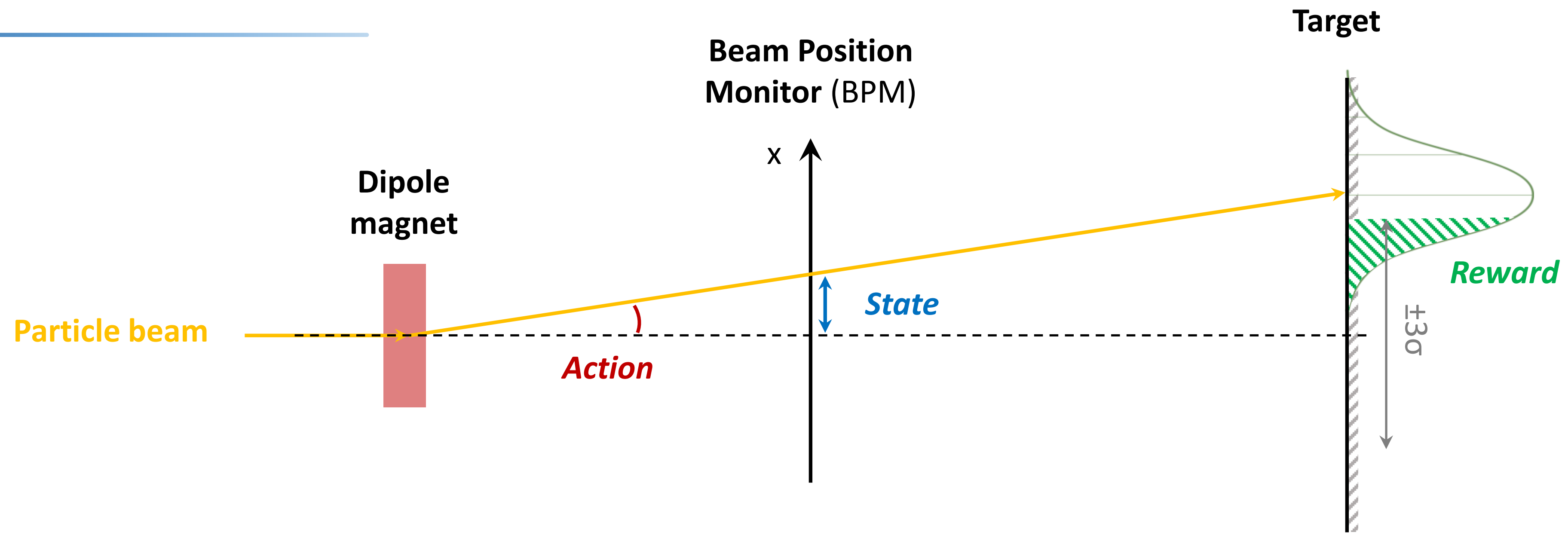
Q-net



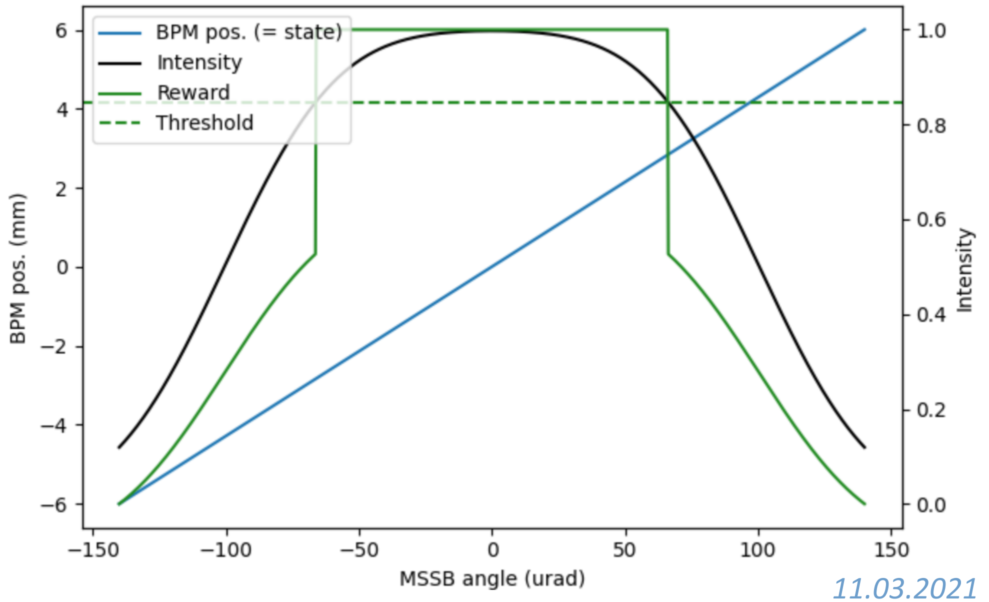
$$\begin{bmatrix} 1 \\ -1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \Rightarrow \cdots \Rightarrow \begin{bmatrix} 0.45 \\ \mathbf{1.23} \end{bmatrix} \approx \hat{Q}(s,a)$$

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

Our project: beam steering

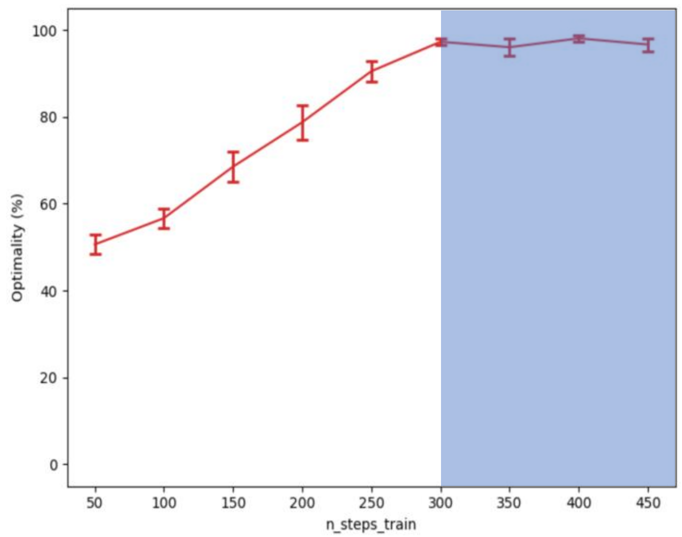


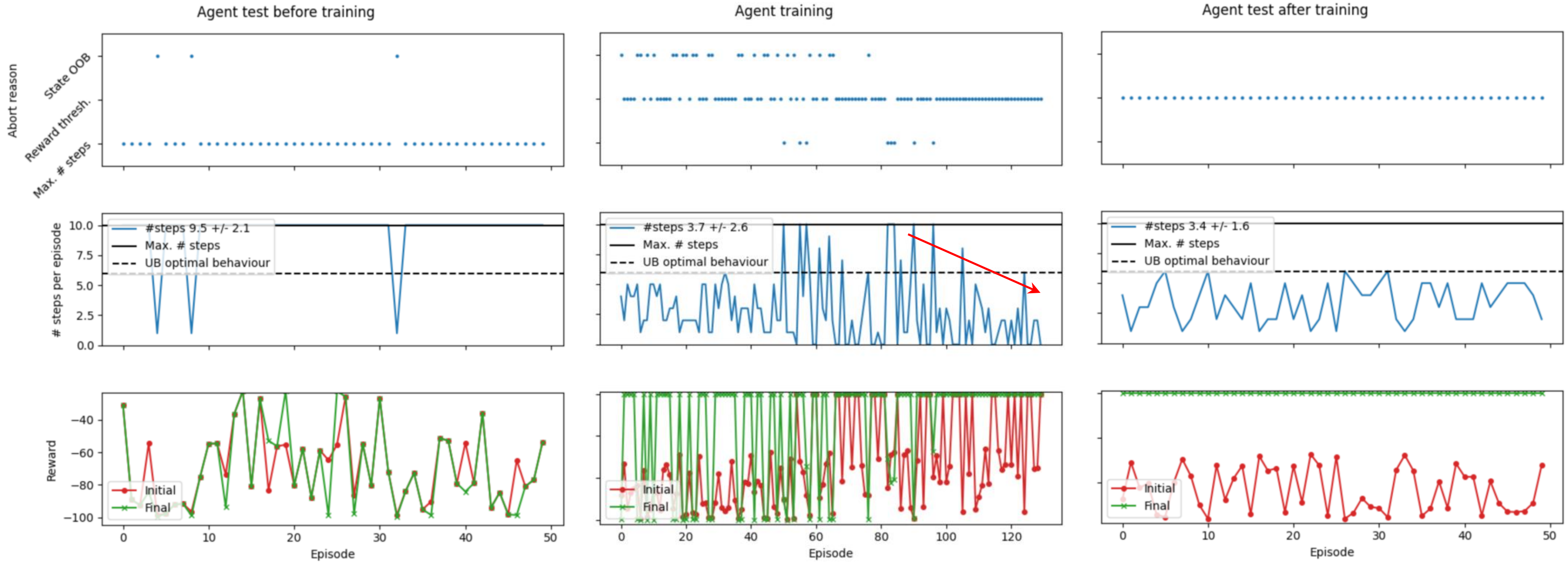
- **Toy model based on actual steering problem**, e.g. for fixed target experiments at CERN Super Proton Synchrotron
- *OpenAI gym template*
- **Action:** deflection angle
  - 2 possibilities: up or down by fixed amount
- **State:** beam position at BPM
- **Reward:** integrated beam intensity on target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



DQN: main results

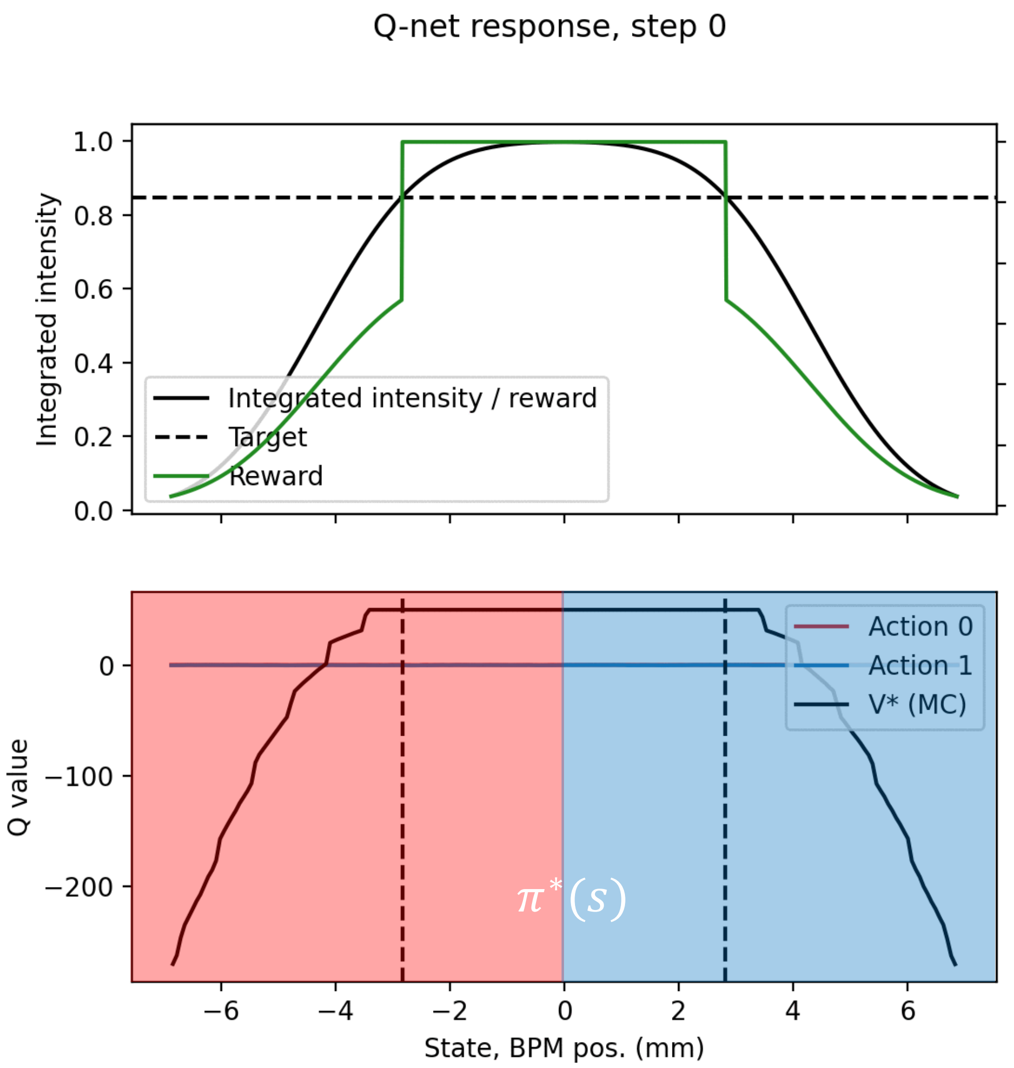
- ***Stable-baselines3*** **implementation of DQN**
- **Efficiency:** required **# training_steps** *after hyperparameter tuning*
- **300+** training steps: get optimal policy with nearly **100% success rate**
- **No need to visit every state-action pair:** *256 states* x *2 actions* = 512 >> n_{train}





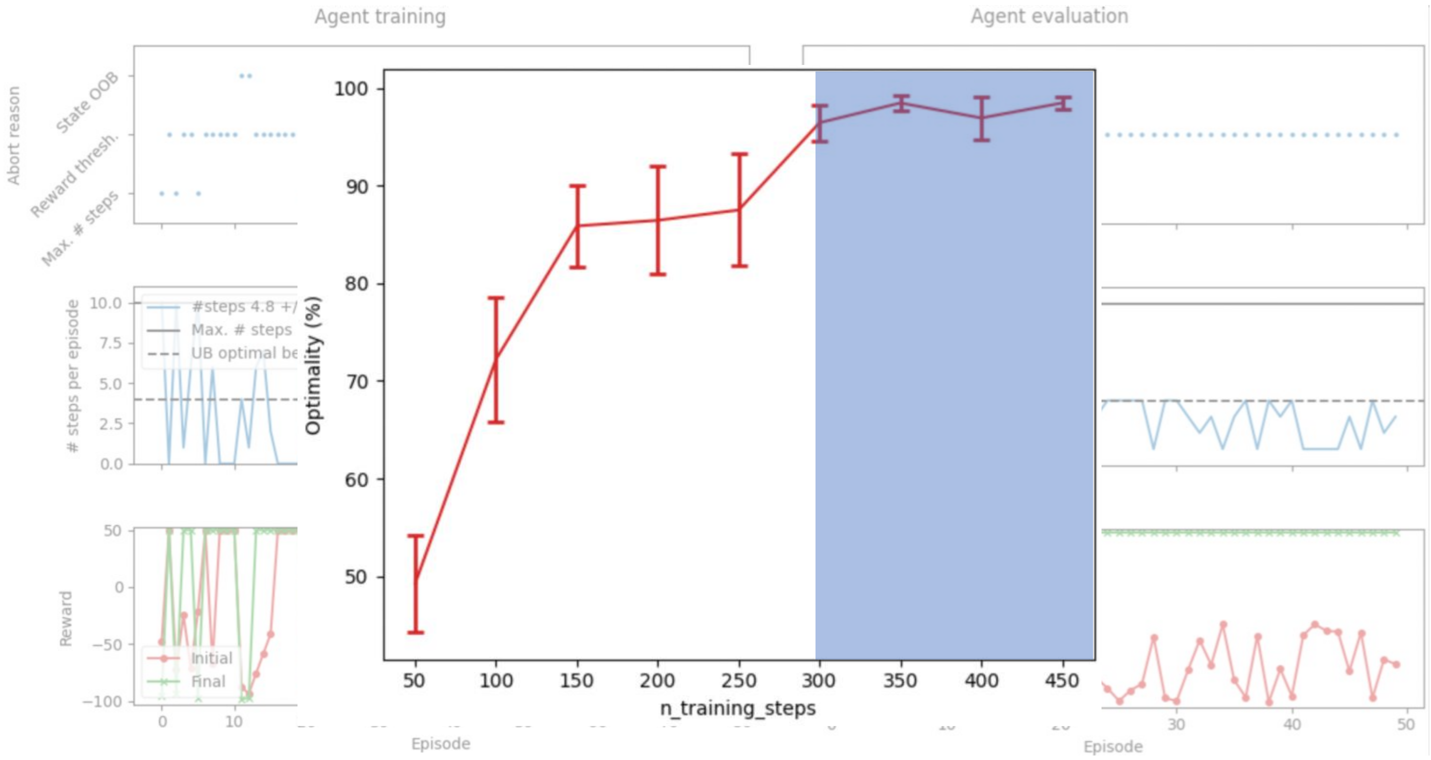
DQN: “looking inside the agent’s mind”

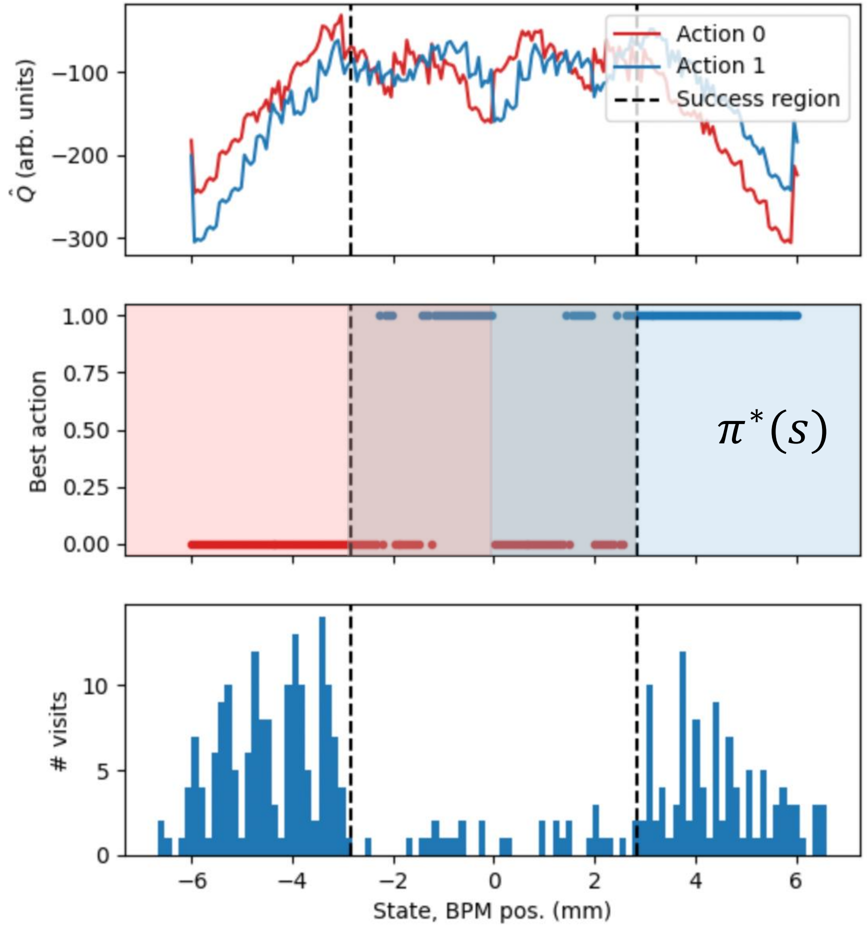
- **See how agent learns and makes decisions**
- Our simple environment: **optimal policy $\boldsymbol{\pi}^*(s)$ known**
- **Benchmark for convergence:** calculate optimal state-value function V^* using Monte-Carlo evaluation
- **Q-functions prove problem has been solved**
- Here: no need to train until convergence



QBM: results with simulated quantum annealing

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

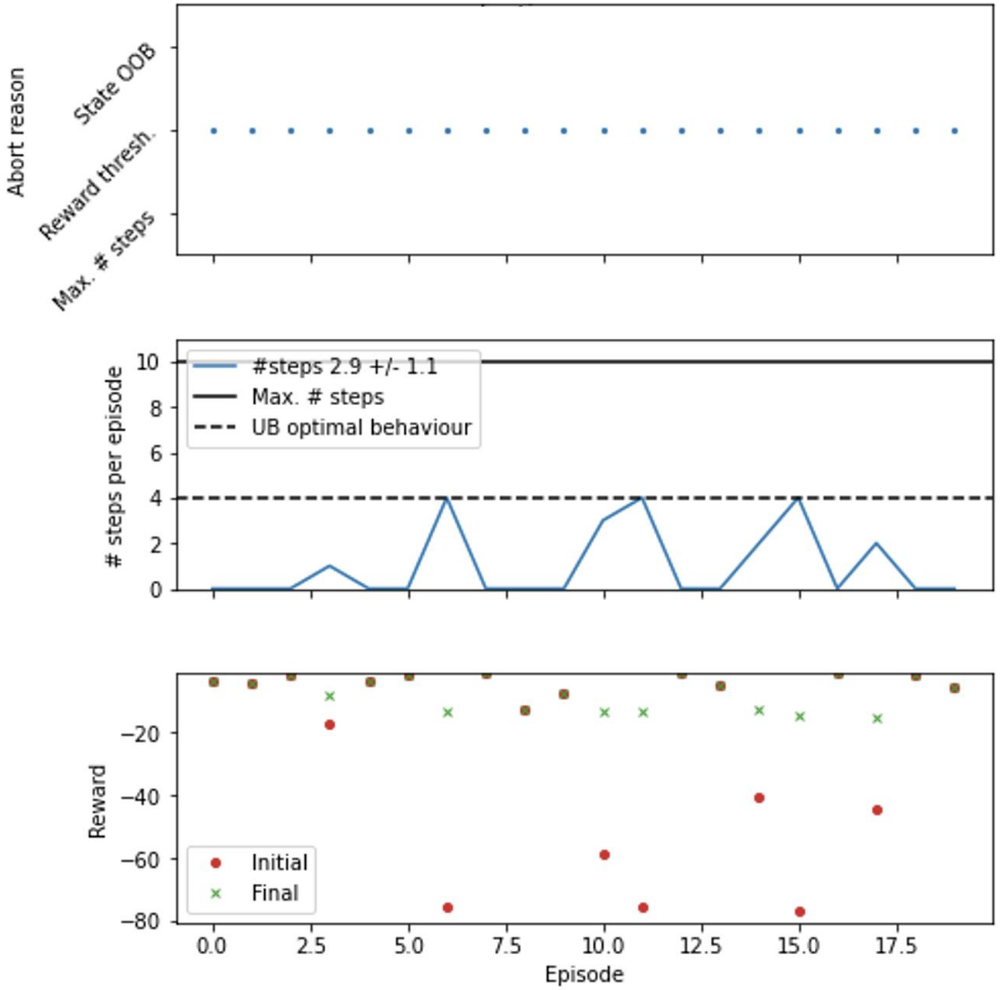




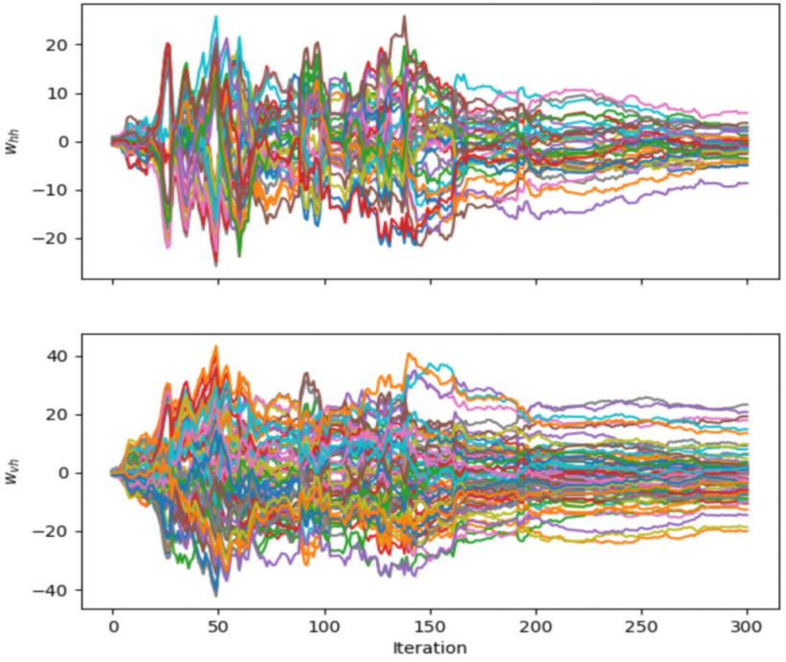
QBM: results on D-Wave 2000Q, part I

- 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

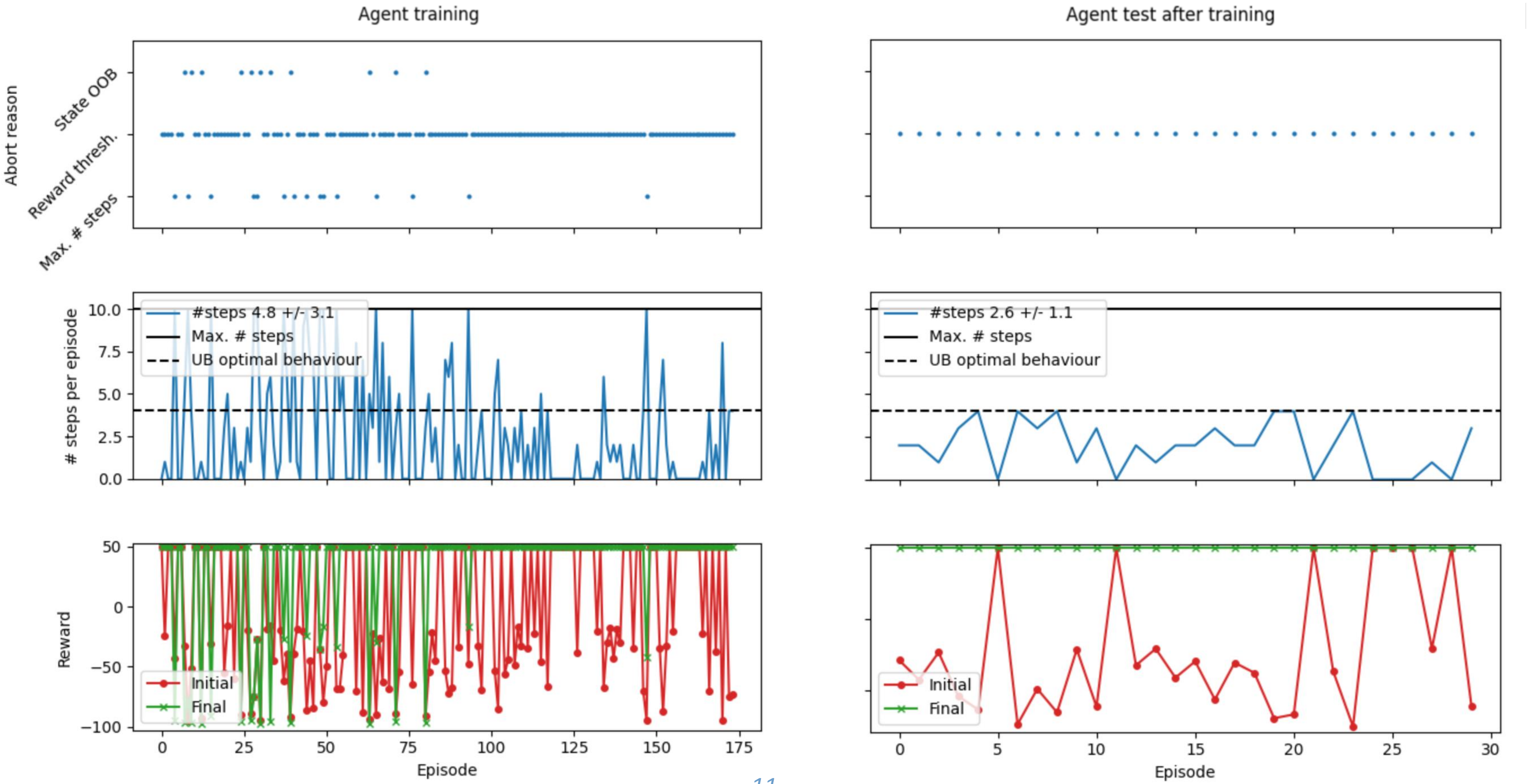


Evolution of QBM weights during training with SQA



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

- **D-Wave training from scratch** *(600 iterations)* after additional hyperparameter tuning with SQA
- **Successful RL training on real QPU ☺ !**



Summary

- **Comparison between Deep Q-learning** *(Q-net)* **and Free Energy based RL** *(QBM)*
- **Simple beam steering environment to start with**
- **Successfully trained both DQN and QBM**
  - Preliminary: similar sample efficiency between DQN and QBM
  - Advantage only apparent for larger state-action spaces?
- **QBM: training successful with simulated quantum annealing** (SQA) ***and*** **on D-Wave 2000Q**
  - Can exchange weights between agent trained with SQA and on D-Wave
  - Training on D-Wave 2000Q less effective likely due to lack of on-hardware parameter tuning
- **Outlook: consider more complex RL 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** $\boldsymbol{\widehat{Q}(s,a)}$
- Conditions: need to be able to **calculate derivative of** $\boldsymbol{\widehat{Q}}$ **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

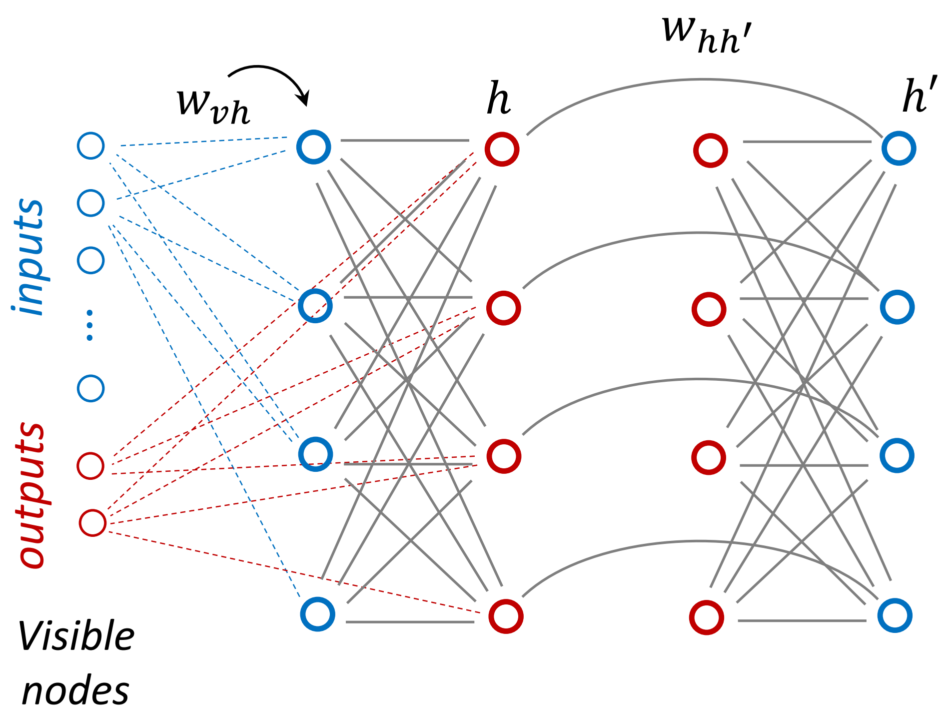
- **All nodes of QBM are hidden**
- **Clamping: add visible nodes as self-couplings** (biases) to hidden nodes they are connected to **and remove them from the graph**

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

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

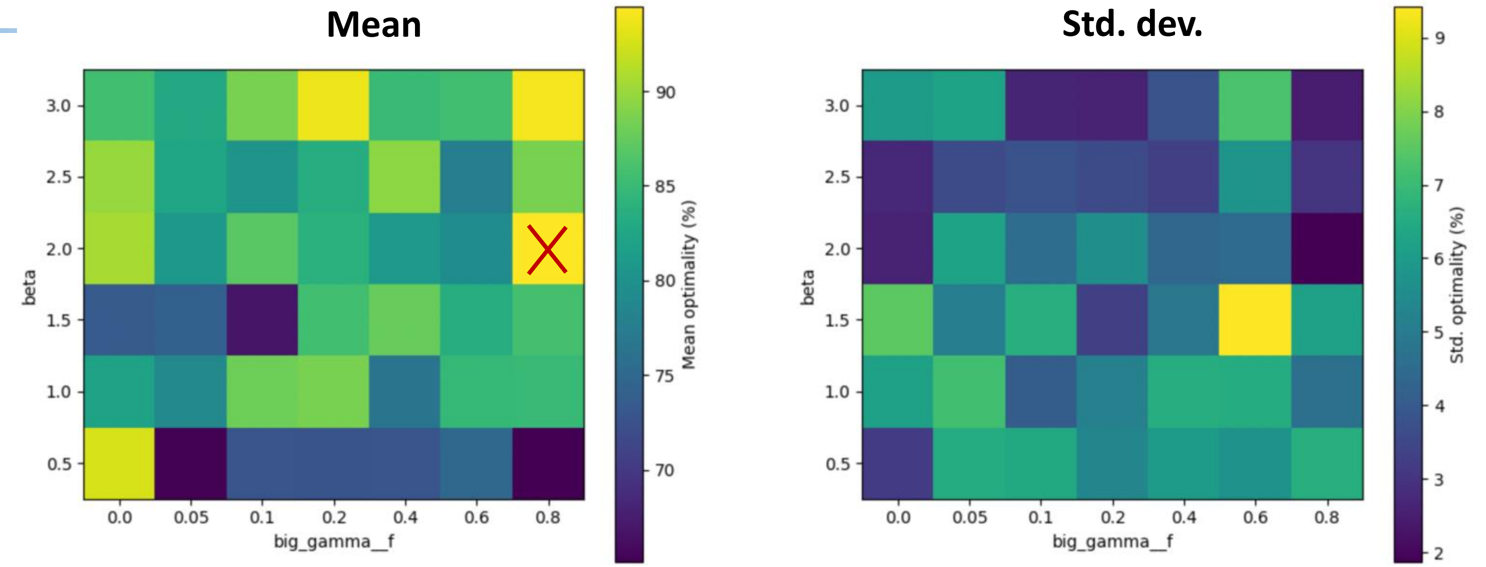
Γ: transverse field strength, $\sigma^{x,z}$: Pauli spin matrices

- 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



Results: QBM with SQA, 2D scans

Γ_f vs β



lr_{final} vs $lr_{initial}$