Deep reinforcement learning approach for fast qubit control

Artem Ryzhikov[1], Vlad Belavin[1], Alexey Kharlamov[1], Andrey Ustyuzhanin[1,2,3], Dmitry Negrov[2], Evgeniy Korostylev[2]

2018-11-05, CERN

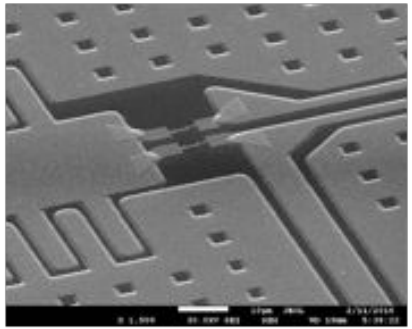
[1] HSE
[2] MIPT
[3] YSDA

SCHOOL OF DATA ANALYSIS

SRF
MIPT

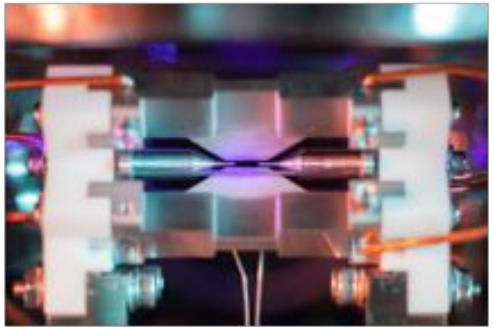
HIGHER SCHOOL OF ECONOMICS

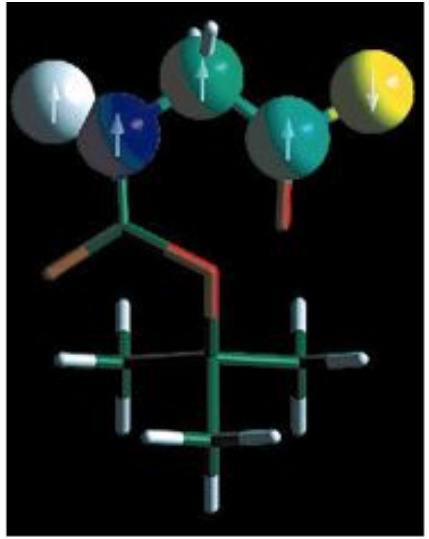
Quantum Qubit Implementations



Superconducting quantum circuit

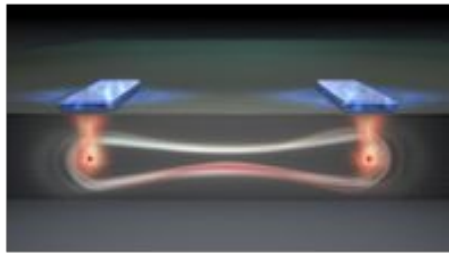
Atomic traps





NMR

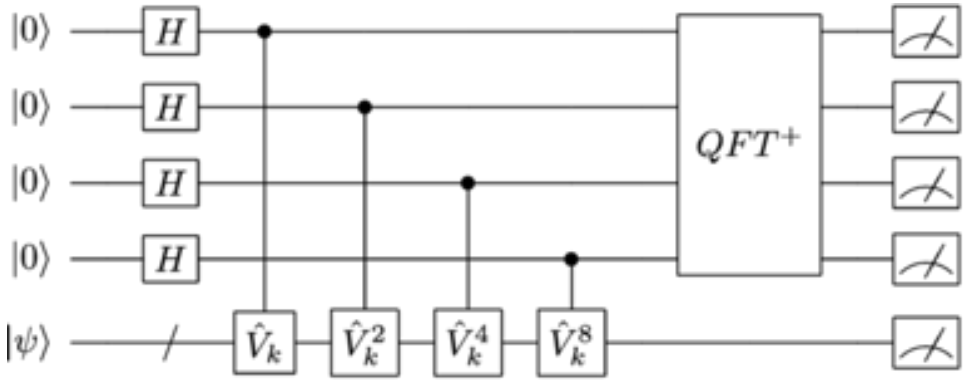
Kane-like



and many more ...

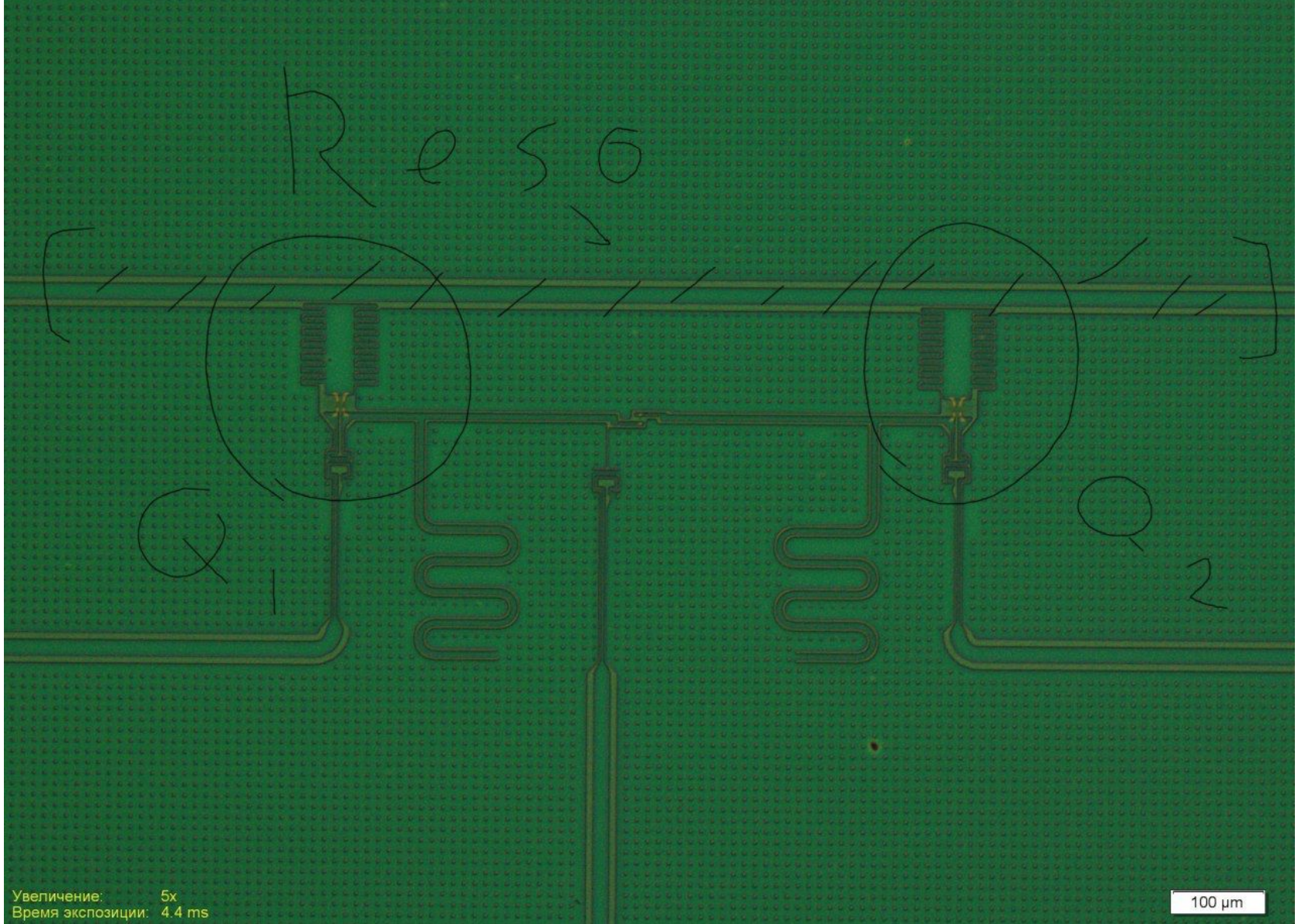
Gated Quantum Circuit Control

Sequential application of quantum gates to a number of qubits



Every gate is a unitary transformation:

Apply some voltage and current waveforms to stimulate gates


Reso
Q1
Q2
Увеличение: 5x
Время экспозиции: 4.4 ms
100 µm

Qubit Control Motivation

Goal:

- Change qubit from given state to target state

Difficulties:

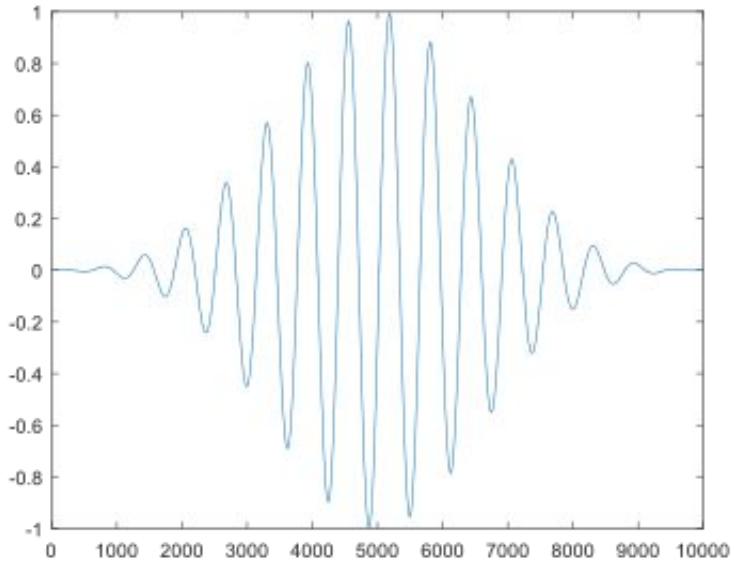
- Should be robust to noise
- No information on system is known a priori (no closed-form solution is applicable)
- Should be fast to prevent information leakage

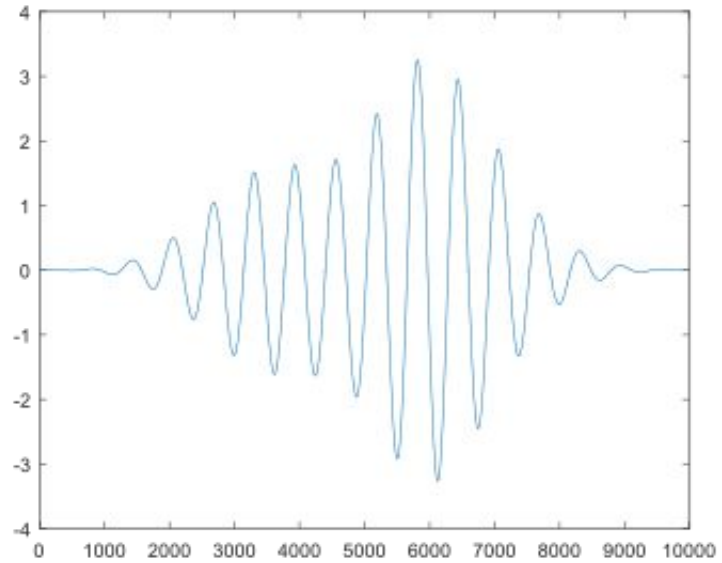
Sources of noise:

- States outside of computational basis
- There are parasitic coupling terms (e.g. small z-coupling when we expect only x-coupling)

Quantum Feedforward Control

- The simplest one is pulse shape optimization:



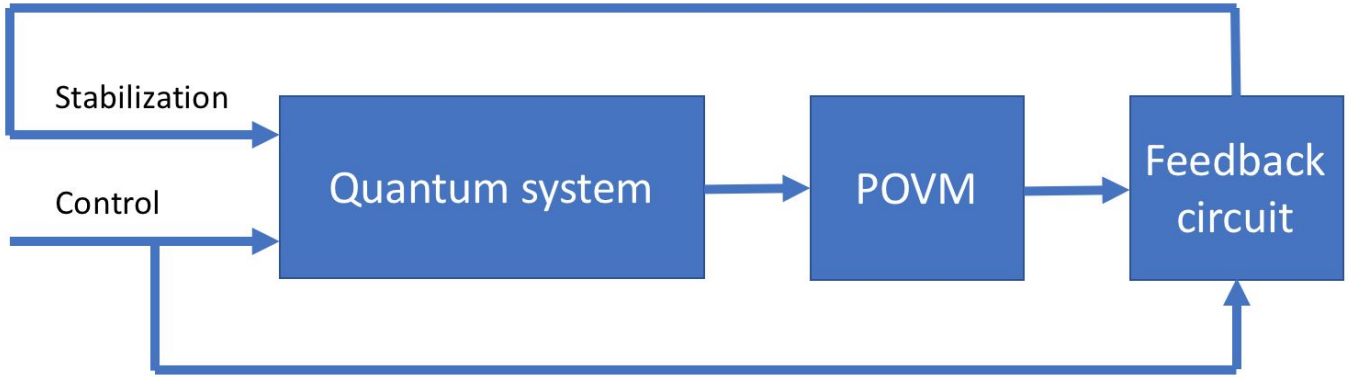


More advanced ones: CRAB / GRAPE / Stochastic Gradient

Con: scale poorly on larger systems

Quantum Feedback Control

A way to stabilize environment drifting parameters (e.g. local magnetic field) by extracting information from running system



Pro: gives better convergence
Con: not easy to engineer

Why Machine Learning?

- Quantum control problems cannot rely on closed-form solutions
- Quantum control problems are usually not convex
- High accuracy simulations are available
- Loss functions are easily definable from the first principles
- Scales to large systems

Challenges:

- build robust precise model
- and make it fast

Quantum Qubit

- **State:** two quantum vectors (Fock states) of qubit and resonator system

Time-independent hamiltonian part of the system:

$$H_0 = h\omega a^+ a + \frac{h}{2}\sum_n \epsilon_n \left|n\right\rangle \left\langle n\right| + hg(a^+ + a)\sigma_x$$

Time-dependent hamiltonian part:

$$H_t = \frac{eC_x(C_q + C_g)}{C_q C_g}\sigma_x \cdot G(t)$$

Quantum Qubit

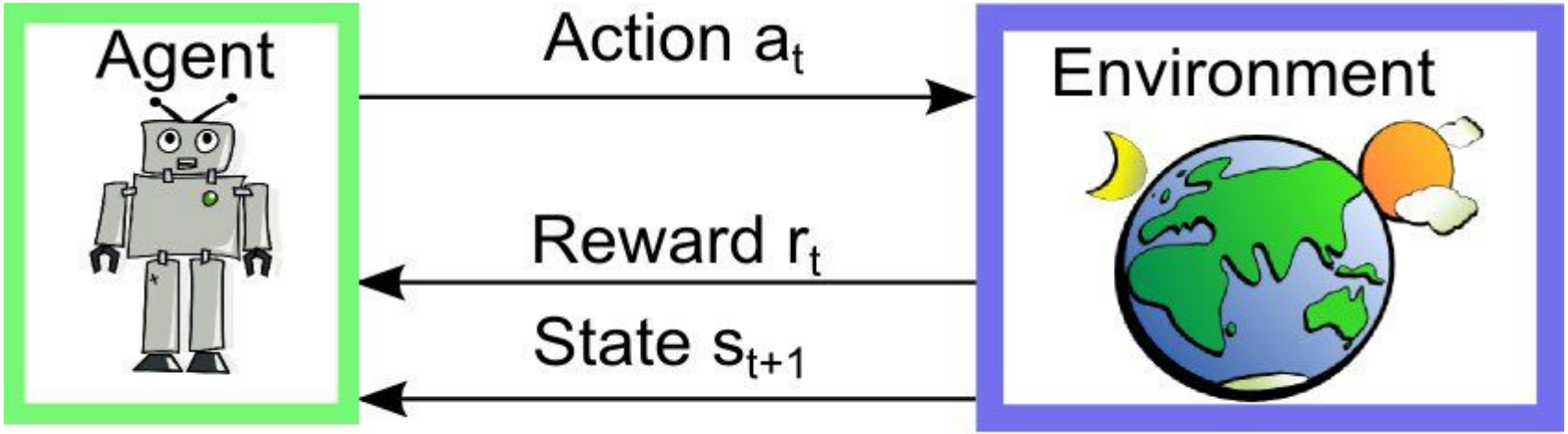
- **System state:**
  $\mathbf{a^+a^-}$ (particle number operator of resonator),
  $\boldsymbol{\sigma^+\sigma^-}$ (particle number operator of qubit),
  $\boldsymbol{\sigma^z}$ (projection of qubit spin on z-axis),
  $\boldsymbol{\sigma^x}$ (projection of qubit spin on x-axis),
  $\mathbf{a^+ + a^-}$ (field potential)
- **Control:**
  quadrature modulated $G(t)$ = NN(t)[0] * sin(φ t) + NN(t)[1] * cos(φ t)
- **FOM**: fidelity between current and target states:

$$F(\rho, \sigma) = |\langle \psi_\rho | \psi_\sigma \rangle|^2$$

Problem statement

- **Find best strategy to transform quantum resonator** from initial qubit state to final one:
  - get as close to the final state as possible (fidelity is ~1.0)
  - within minimum time
- No a priori information of the system is given,
  - besides simulator of the system

Reinforcement Learning



Reinforcement Learning Setup

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Environment

- Quantum resonator + qubit:
  - with 5D continuous state
  - continuous 2D actions
- Implemented on top of qutip (http://qutip.org) simulator
- Only N {2,3} bottom levels are taken into account
- Adapted to Open AI gym API

Agent

- Represented by neural network(s)
- Loss function defined by
  - Reward: $\text{fidelity}_{\text{current}} - \text{fidelity}_{\text{old}}$
- Training:
  - Policy Gradient approach
  - Ornstein-Uhlenbeck exploration (https://wiki2.org/en/Ornstein–Uhlenbeck_process)

Deep Q-Networks

Neural network (θ) that receives the environment state as input and generates as output the Q-values for each of the possible actions, using the loss function

$$\mathcal{L}(\theta) = \mathbb{E}\left[(y - Q(s, a; \theta))^2\right]$$

follows the gradient

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}\Big[\big(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\big) \nabla_\theta Q(s, a; \theta)\Big]$$

(Mnih et al., 2013, 2015)

Deep Q-Networks

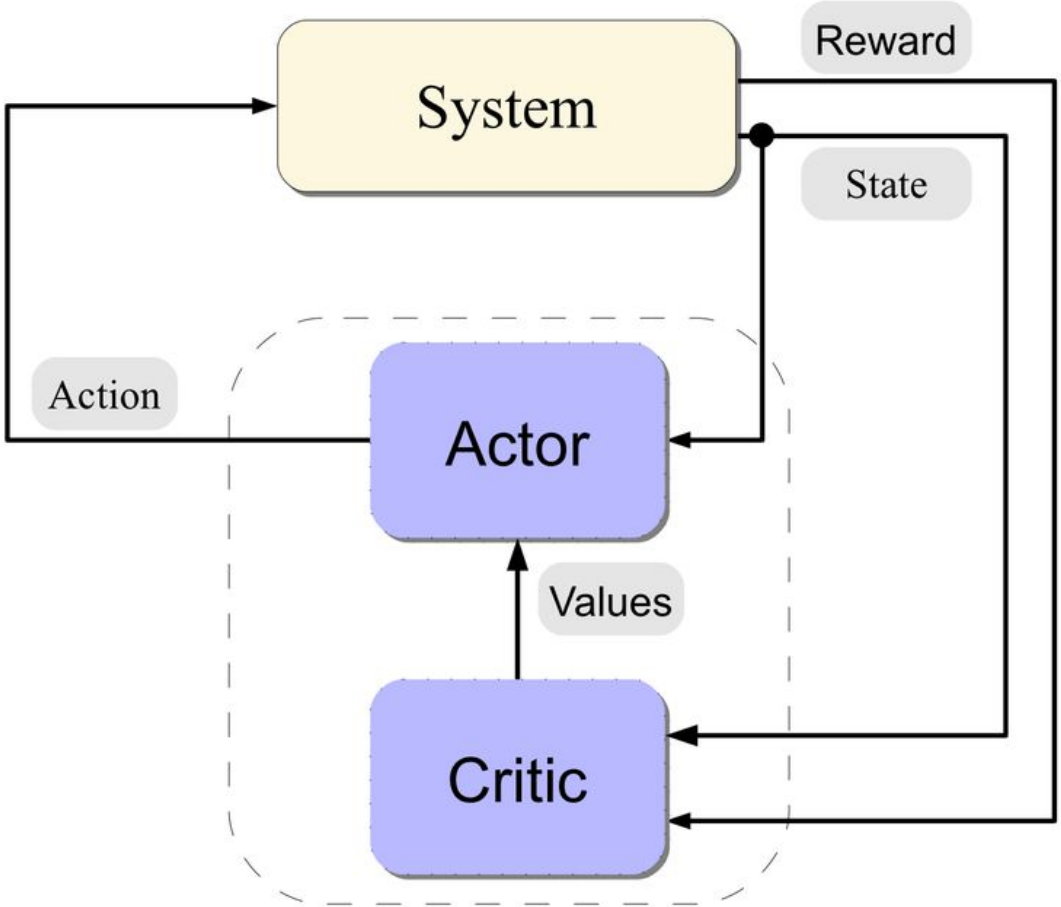
Issues of stability addressed by:

- Experience replay: keep a memory of past action-rewards
- Reward clipping: scale and clip the values of the rewards to the range [−1, +1]
- Target network: keep a separate DQN so that one is used to compute the target values and the other one accumulates the weight updates, which are periodically loaded onto the first one.

Deep Deterministic Policy Gradient

- Deals with continuous action-space
- combines the actor-critic classical RL approach (Sutton and Barto, 1998) with Deterministic Policy Gradient (Silver et al., 2014)
- Deterministic policy (as opposed to stochastic) approximated by a neural network **actor** $\pi(s;\theta^{\pi})$ and another separate network $Q(s,a;\theta^{Q})$ implementing the **critic**

Actor-Critic Architecture



Training

Critic update

$$Q(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1}} \left[r(s_t, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1})) \right]$$

Actor update:

$$\begin{aligned} \nabla_{\theta^\pi} \mathcal{L} &\approx \mathbb{E}_s \left[\nabla_{\theta^\pi} Q(s, \pi(s|\theta^\pi)|\theta^Q) \right] \\ &= \mathbb{E}_s \left[\nabla_a Q(s, a|\theta^Q)|_{a=\pi(s|\theta^\pi)} \nabla_{\theta^\pi} \pi(s|\theta^\pi) \right] \end{aligned}$$

Same stability tricks as in DQN

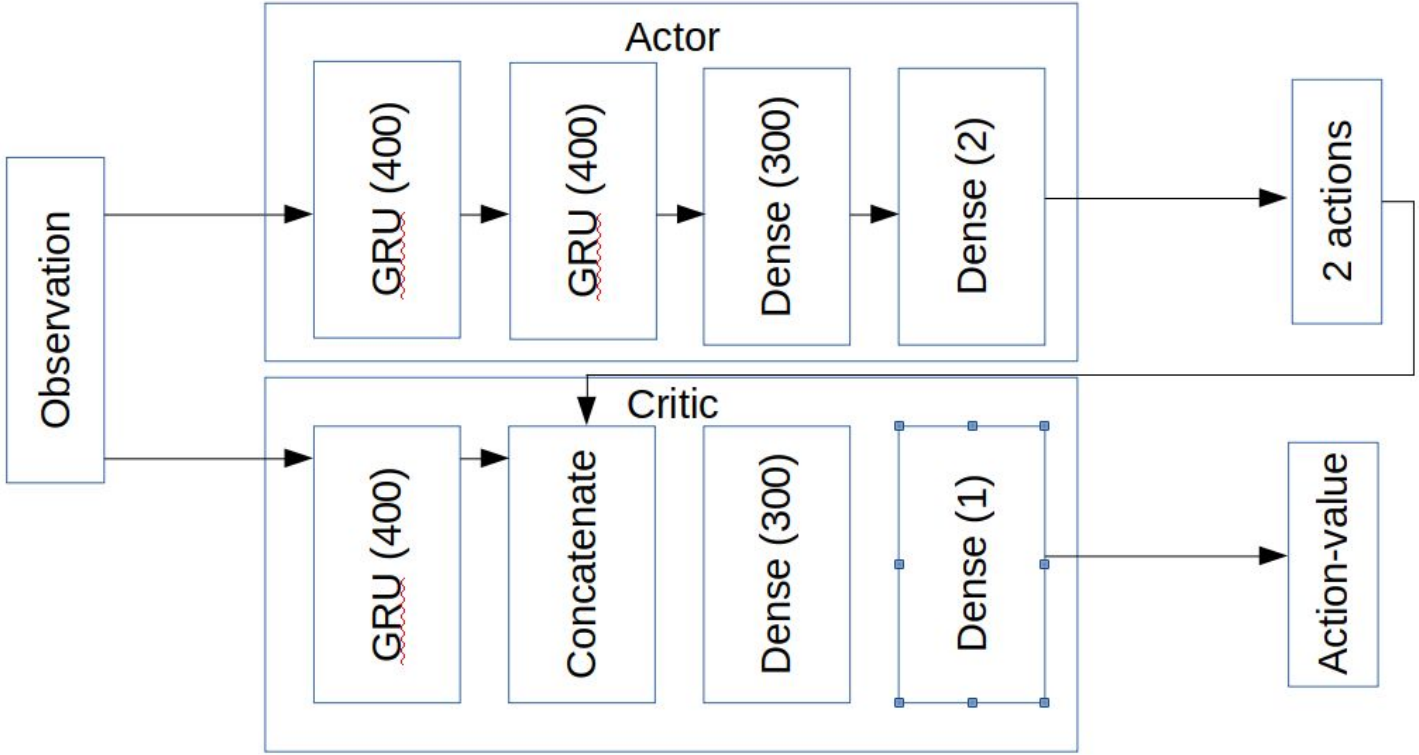
Algorithm 1 Deep Deterministic Policy Gradient algorithm

Randomly initialize weights of $Q(s, a|\theta^Q)$ and $\pi(s|\theta^\pi)$.
Initialize target net weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\pi'} \leftarrow \theta^\pi$.
Initialize replay buffer R.
for each episode **do**
  Initialize random process $\mathcal{N}$ for action exploration.
  Receive initial observation state s_1.
  for each step t of episode **do**
    Select action $a_t = \pi(s_t|\theta^\pi) + \mathcal{N}_t$.
    Execute a_t and observe reward r_t and state s_{t+1}.
    Store transition (s_t, a_t, r_t, s_{t+1}) in R.
    Sample from R a minibatch of N transitions.
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \pi'(s_i + 1|\theta^{\pi'})|\theta^{Q'})$.
    Update critic by minimizing the loss:
      $\mathcal{L} = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$.
    Update the actor using the sampled policy gradient:
      $\nabla_{\theta^\pi} \mathcal{L} \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q) \nabla_{\theta^\pi} \pi(s|\theta^\pi)|s_i$.
    Update the target networks:
      $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
      $\theta^{\pi'} \leftarrow \tau\theta^\pi + (1-\tau)\theta^{\pi'}$.
  end for
end for

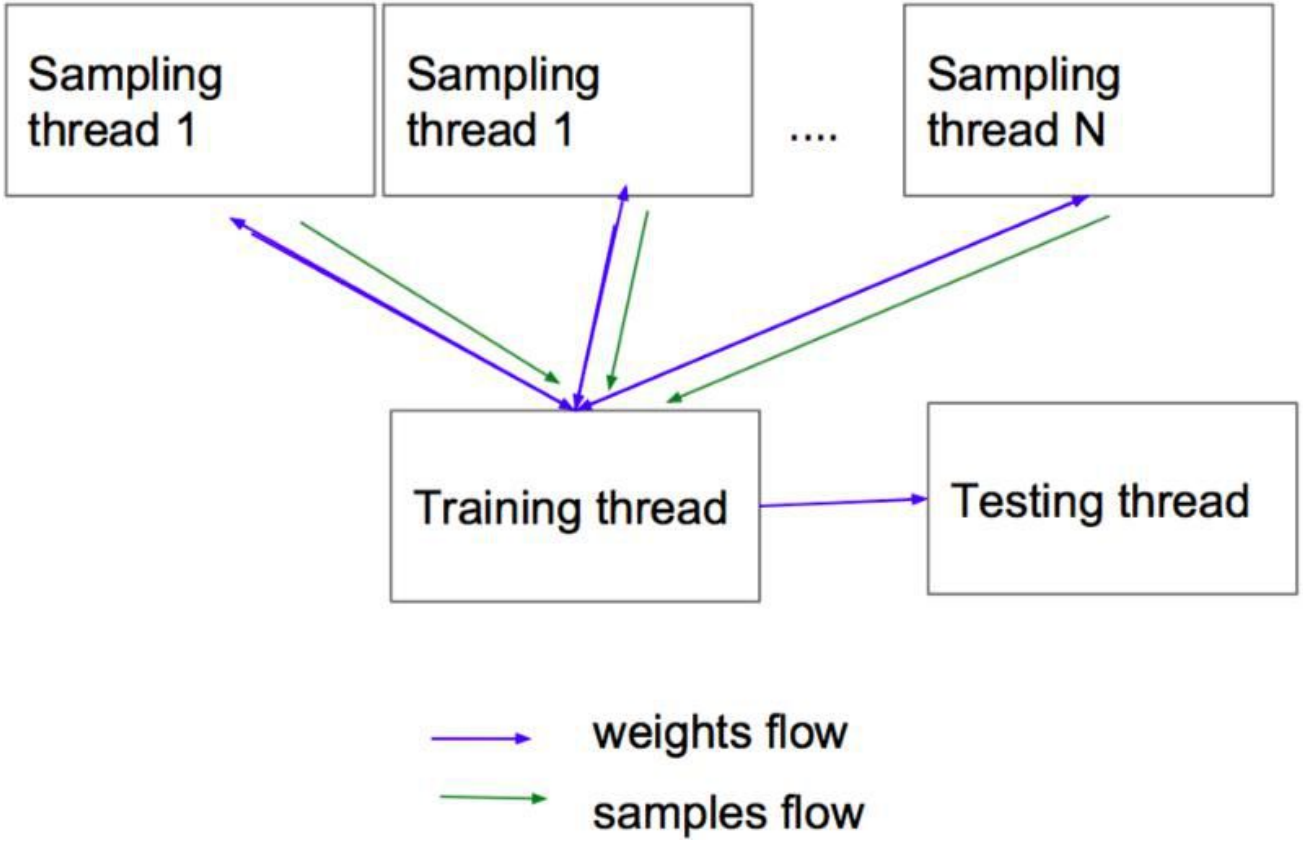
(Noe Casas, 2017)

Actor-Critic Architecture

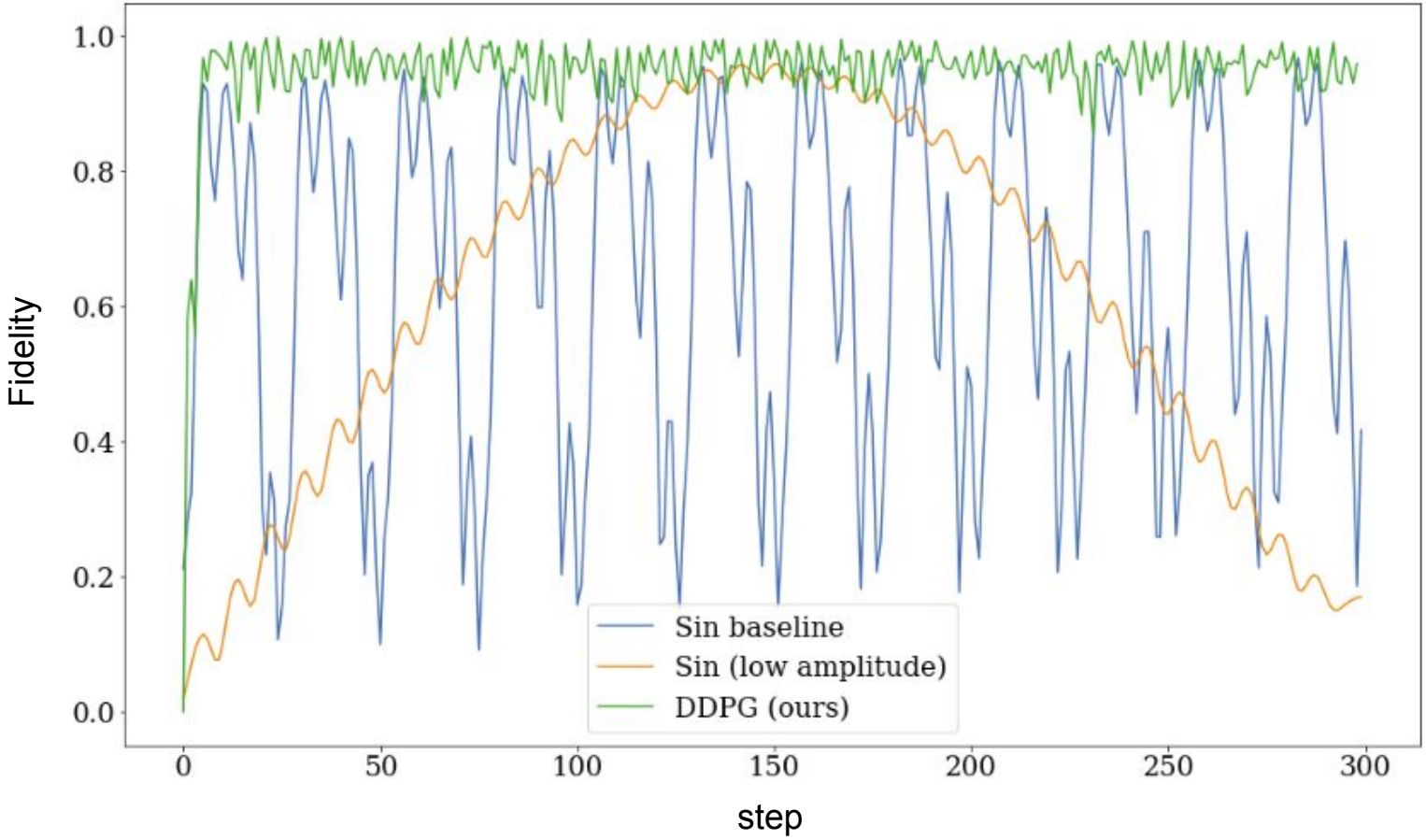
- Aplus
- Aminus
- SigmaX
- SigmaZ
- SigmaP
- few-hand-crafted features



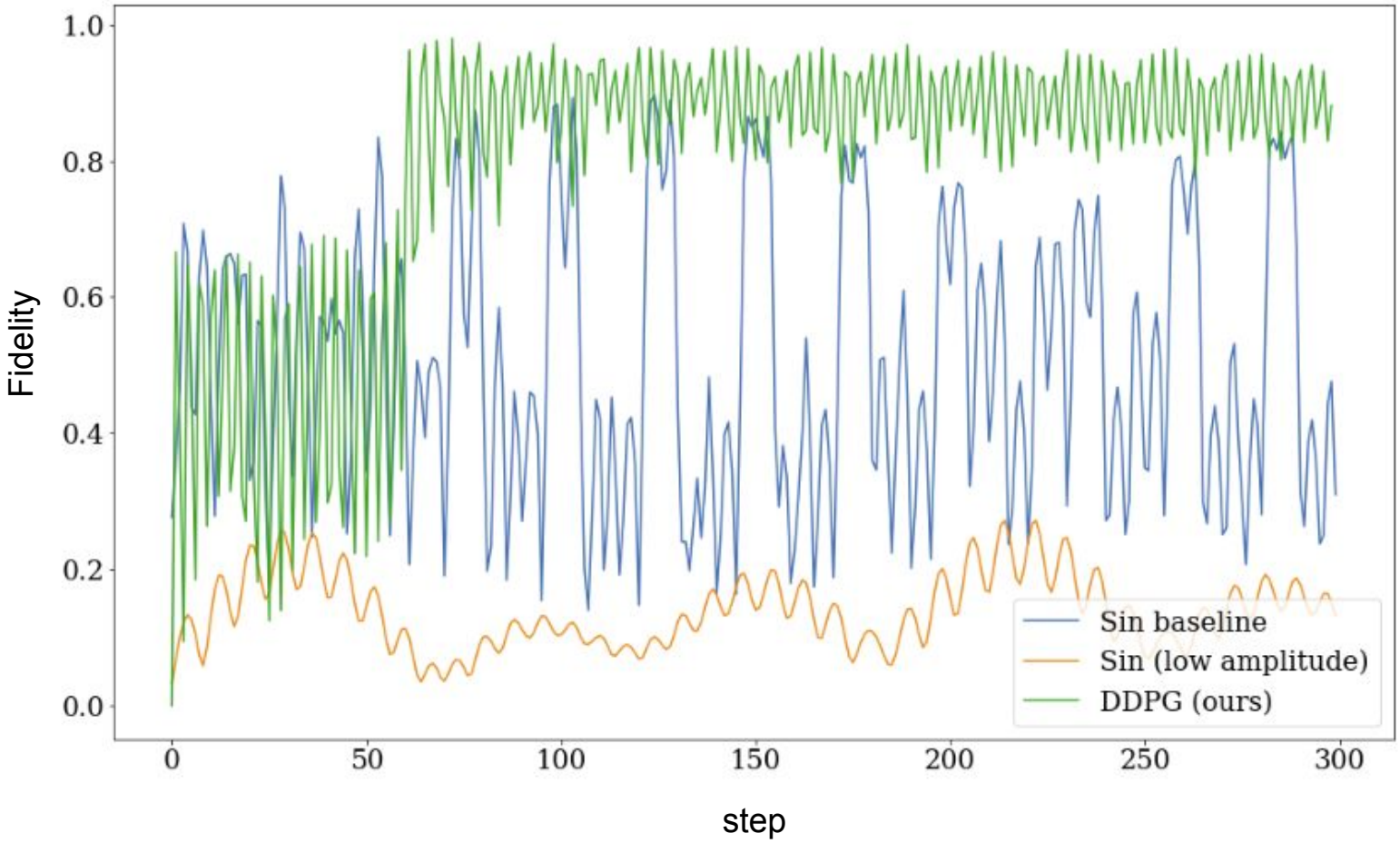
Training Parallelization



Results (2-level system)



Results (3-level system)



Results [WIP]

- Implemented the environment
  - based on qutip, tuned for real system, single qubit
  - continuous actions (realistic)
  - Open AI gym API compatible: https://github.com/HSE-LaMBDA/gym-qubit
- Implemented parallel-training scheme for CUDA
- First attempt:
  - Agent reaches ~0.98 fidelity
  - Faster convergence, more robust than baseline

Next Steps

- Add more degrees of freedom
  - more qubits
  - more state pairs
  - more operations (inverse, CNOT, etc)
- Play with different ML tricks:
  - ACKTR as agent to improve training
  - D4PG for off-policy concurrent training
  - Different optimization techniques
- Test efficiency on real qubit hardware
- Accelerate ML Community around Quantum Computing

Thank you!

Andrey Ustyuzhanin
austyuzhanin@hse.ru
anaderiRu @twitter

Backup

Reference

- Andrey Karpathy, Deep Reinforcement Learning: Pong from Pixels, http://karpathy.github.io/2016/05/31/rl/
- Noe Casas, *Deep Deterministic Policy Gradient for Urban Traffic Light Control*, arxiv:1703.09035
- Marin Bukov, Alexandre G. R. Day, Dries Sels, Phillip Weinberg, Anatoli Polkovnikov, and Pankaj Mehta, *Reinforcement Learning in Different Phases of Quantum Control*, Phys. Rev. X 8, 031086