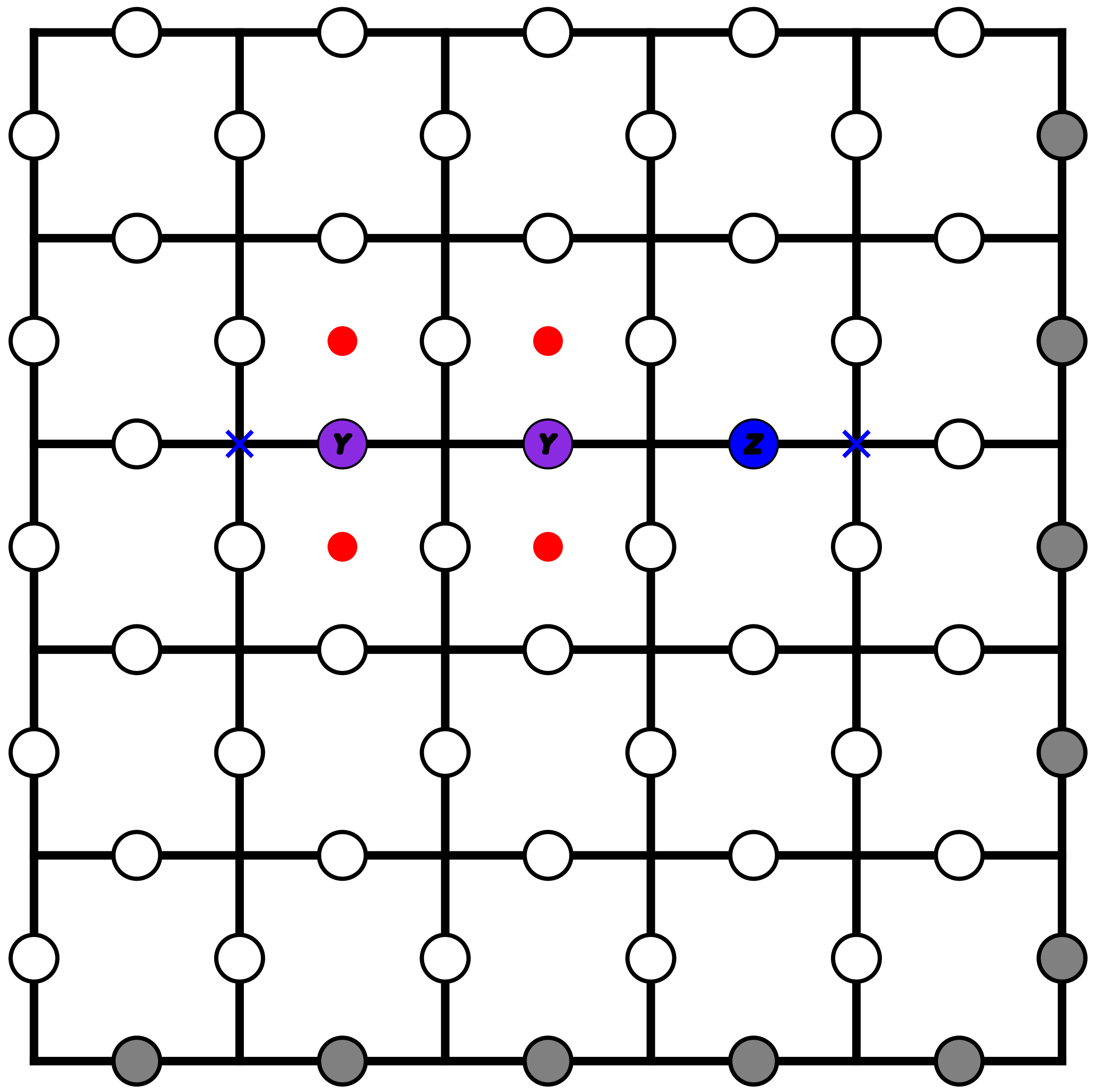
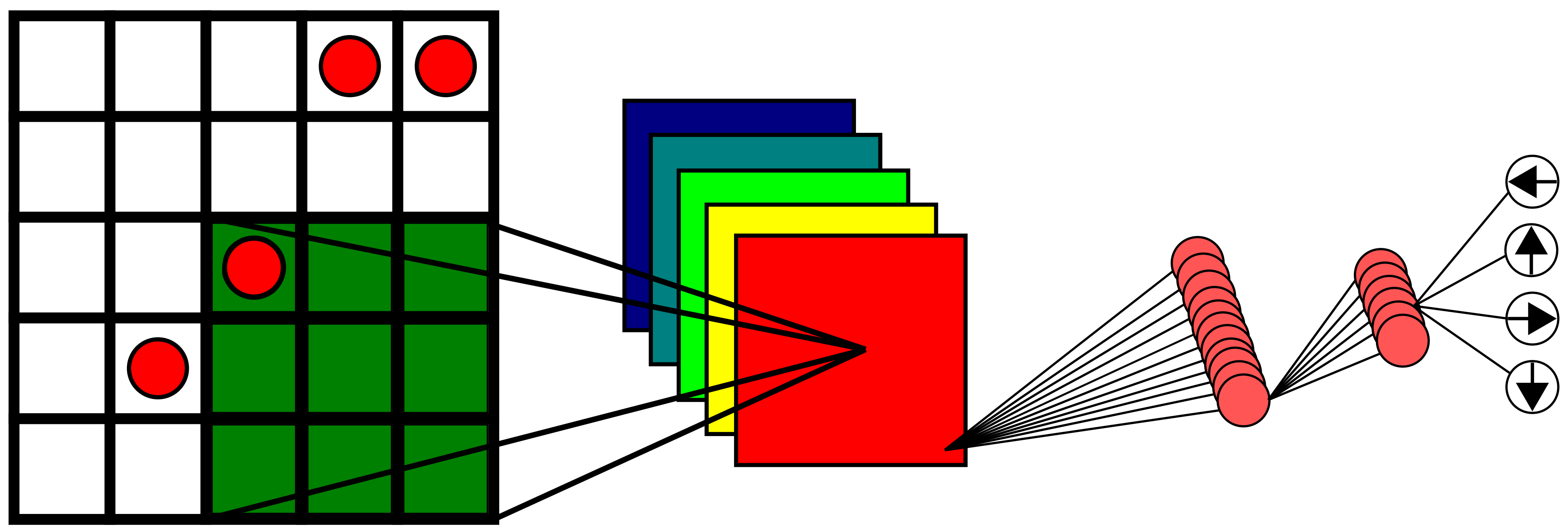
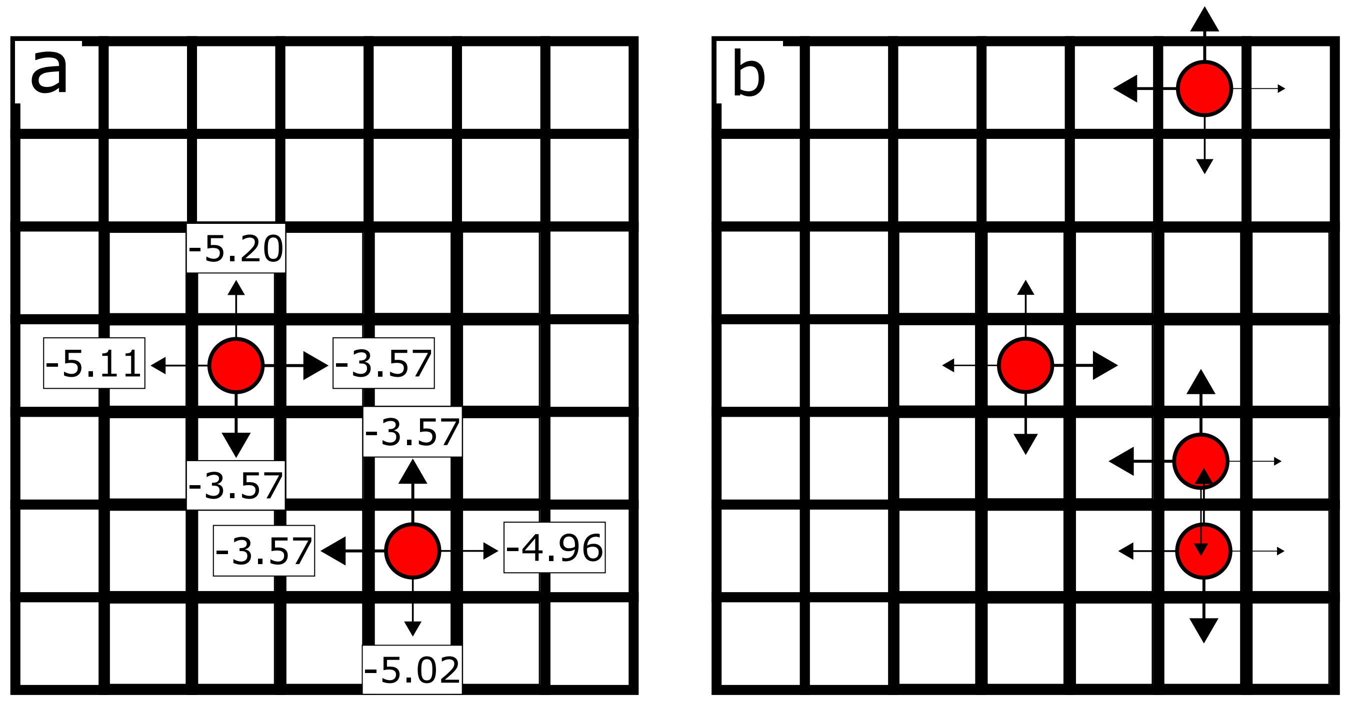
Quantum error correction for the toric code using deep reinforcement learning

Mats Granath
Department of Physics
University of Gothenburg

Machine Learning for Quantum Matter
Nordita
August 26, 2019







Philip Andreasson, Joel Johansson, Simon Liljestrand, MG, arXiv:1811.12338, accepted to Quantum

Mattias Eliasson, David Fitzek, MG, in progress

Outline

- Toric code/Quantum error correction
- Reinforcement learning/Q learning/Deep Q learning
- Toric code with bit-flip error only
- Toric code with depolarizing noise

Quantum error correction and topological codes

Fault tolerant quantum computation requires error correction

Scheme for reducing decoherence in quantum computer memory
Peter W. Shor
Phys. Rev. A 52, R2493 (1995)

Error Correcting Codes in Quantum Theory
A. M. Steane
Phys. Rev. Lett. 77, 793 (1996)

Fault-tolerant quantum computation by anyons

A.Yu. Kitaev*
L.D. Landau Institute for Theoretical Physics, 117940, Kosygina St. 2, Germany
Received 20 May 2002

JOURNAL OF MATHEMATICAL PHYSICS VOLUME 43, NUMBER 9 SEPTEMBER 2002

Topological quantum memory[a)]

Eric Dennis[b)]
Princeton University, Princeton, New Jersey 08544
Alexei Kitaev,[c)] Andrew Landahl,[d)] and John Preskill[e)]
Institute for Quantum Information, California Institute of Technology, Pasadena, California 91125

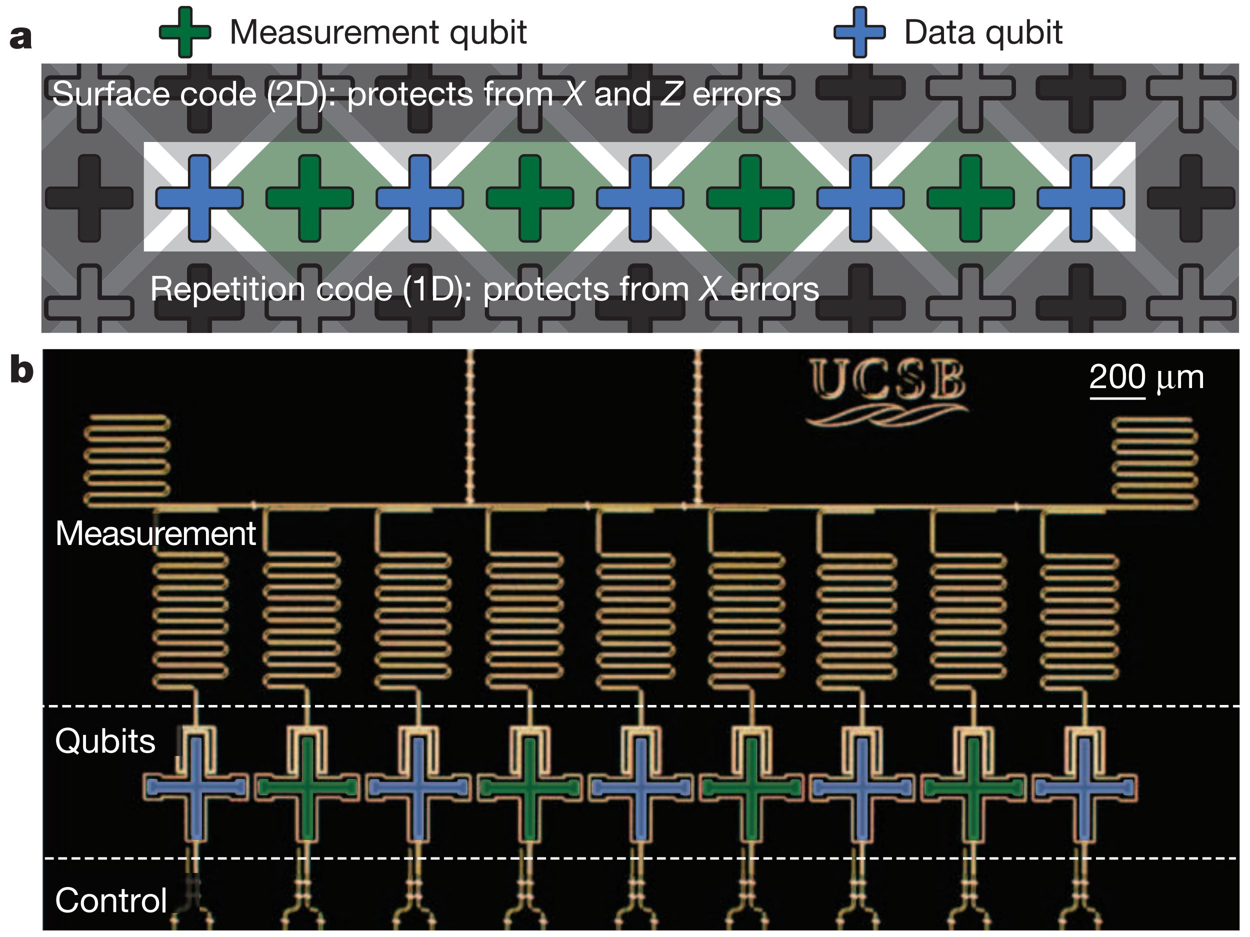
Error correcting codes starting to be built, but still far off technologically.

LETTER

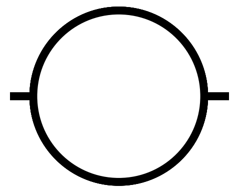
doi:10.1038/nature14270

State preservation by repetitive error detection in a superconducting quantum circuit

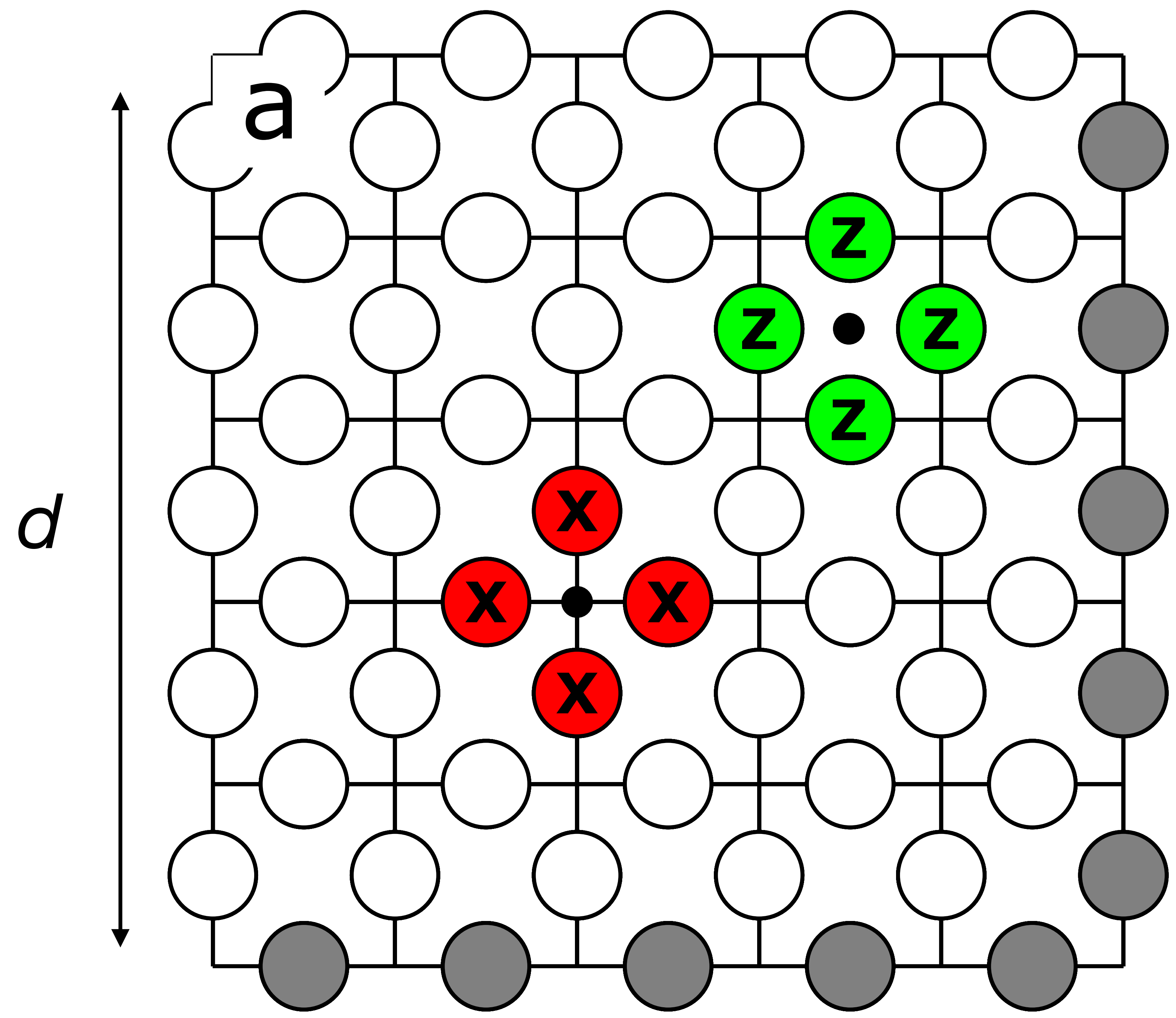
J. Kelly[1]*, R. Barends[1]†*, A. G. Fowler[1,2]†*, A. Megrant[1,3], E. Jeffrey[1]†, T. C. White[1], D. Sank[1]†, J. Y. Mutus[1]†, B. Campbell[1], Yu Chen[1]†, Z. Chen[1], B. Chiaro[1], A. Dunsworth[1], I.-C. Hoi[1], C. Neill[1], P. J. J. O'Malley[1], C. Quintana[1], P. Roushan[1]†, A. Vainsencher[1], J. Wenner[1], A. N. Cleland[1] & John M. Martinis[1]†



The toric code


$= \alpha|\uparrow\rangle + \beta|\downarrow\rangle$ **"Physical" qubit**

Josephson junction, nuclear spin, quantum dot, ...



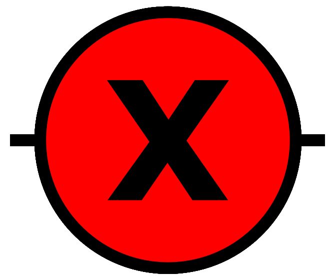
$$H = -\sum_{\alpha} \hat{P}_{\alpha} - \sum_{\nu} \hat{V}_{\nu}$$

$$\hat{P}_{\alpha} = \prod_{i \in \alpha} \sigma_i^z$$

$$\hat{V}_{\nu} = \prod_{i \in \nu} \sigma_i^x$$

Commuting **Plaquette** and **Vertex** stabilizers (parity checks)

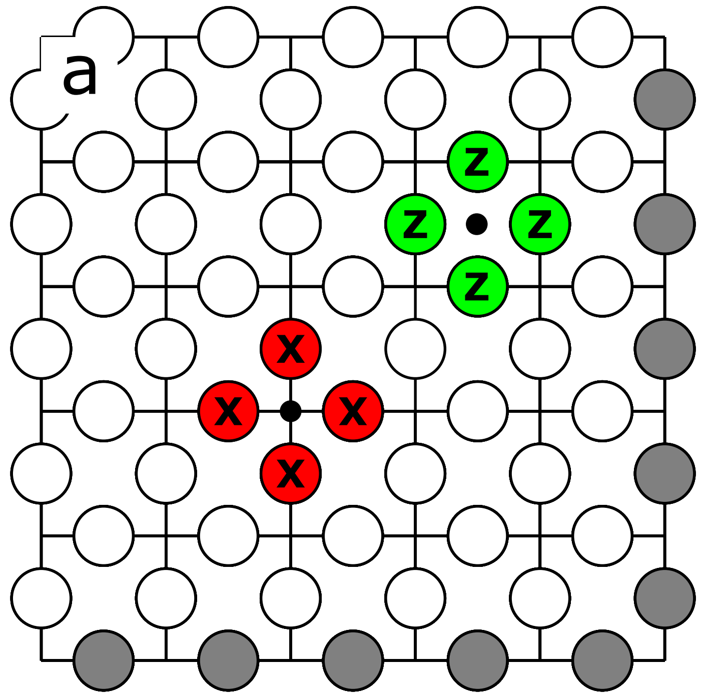
$2d^2$ physical qubits, $2d^2$-2 independent stabilizers



Bit flip $|\uparrow\rangle \Rightarrow |\downarrow\rangle$

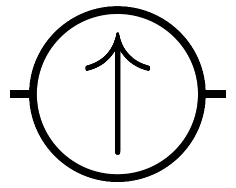
Phase flip $|\uparrow\rangle + |\downarrow\rangle \Rightarrow |\uparrow\rangle - |\downarrow\rangle$

Ground state



$$H = -\sum_{\alpha} \hat{P}_{\alpha} - \sum_{\nu} \hat{V}_{\nu}$$

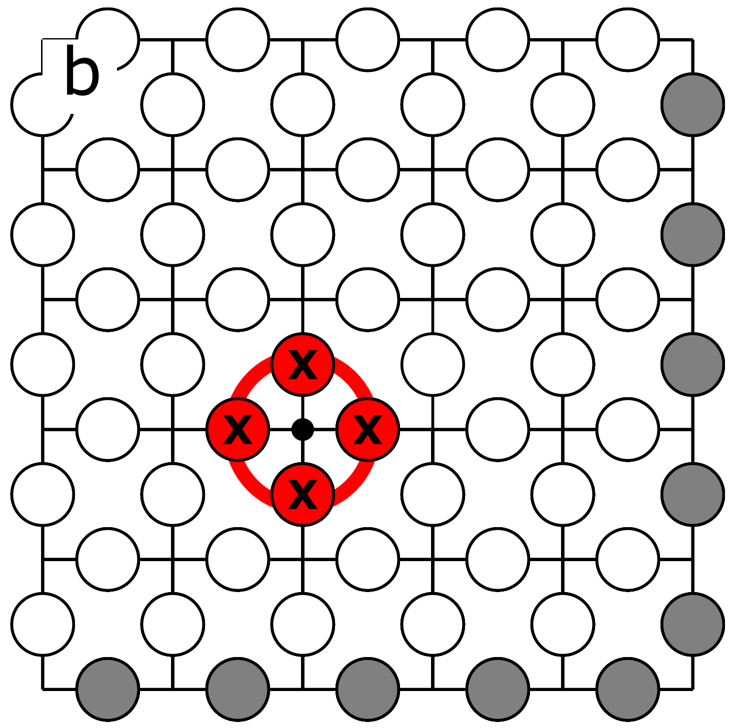
1) consider:

All 

$$| \uparrow\uparrow\uparrow \cdots \rangle$$

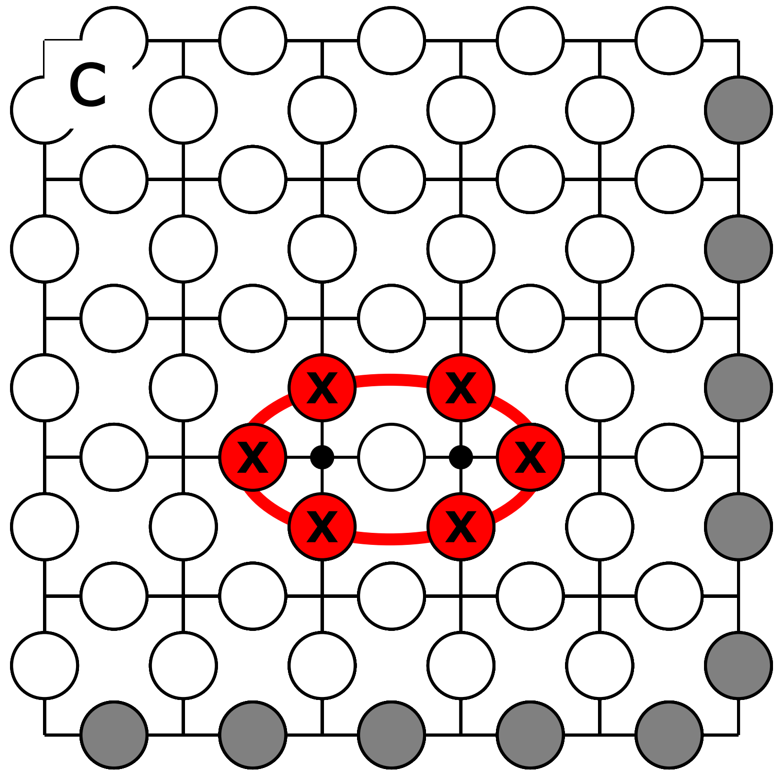
plaquette operator
ground state

2) act with vertex op:



still a plaquette
ground state

3) act with two vertex op:



still a plaquette
ground state

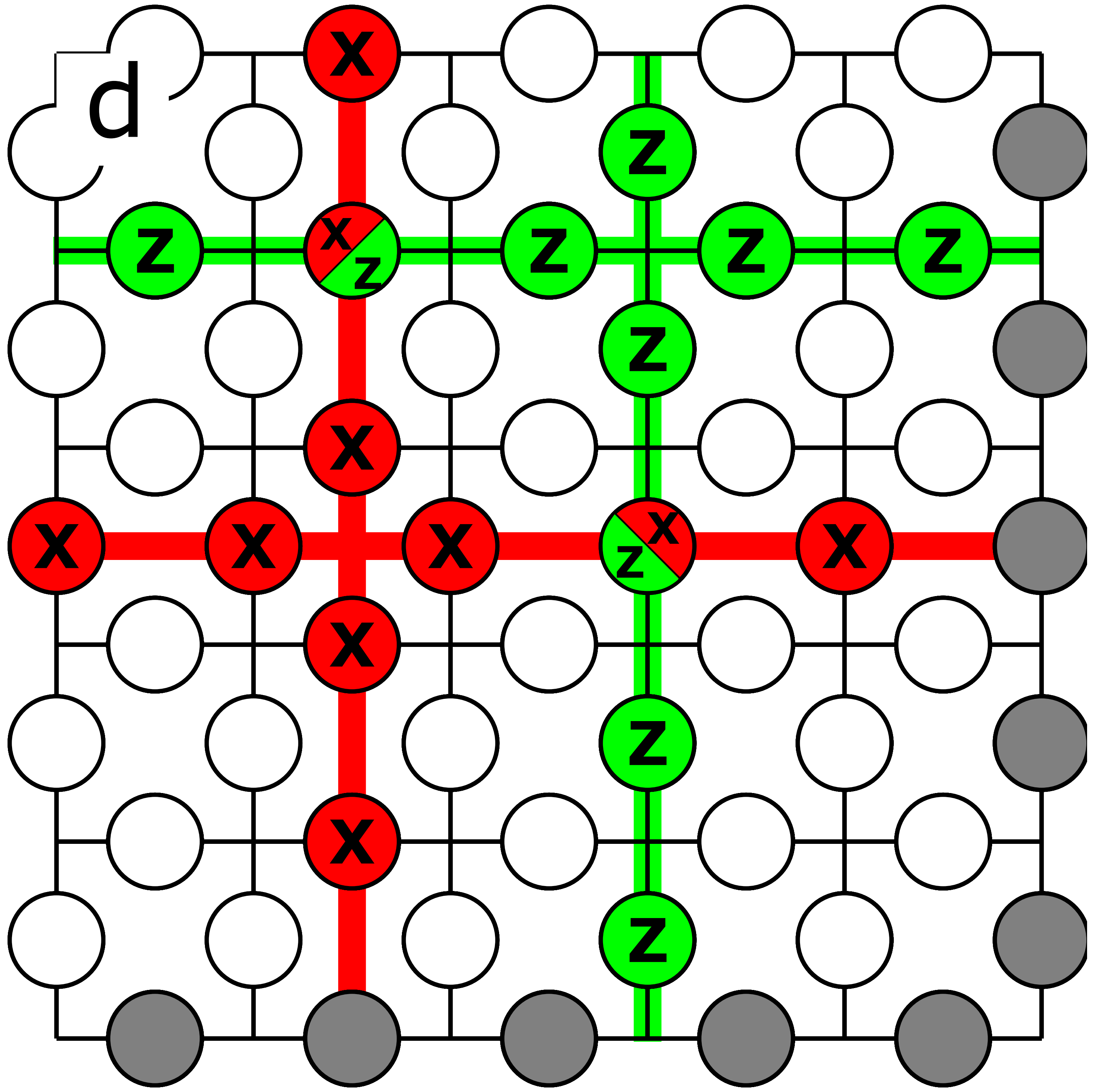
4) GS is symmetric superposition
of all *trivial* loops:

$$|\mathrm{GS}_0\rangle = \sum_{i \in \text{all trivial loops}} \mathrm{loop}_i | \uparrow\uparrow\uparrow \cdots \rangle$$

highly entangled

Ground state degeneracy

Non-trivial loops (encircling torus) X_1, X_2
are not products of vertex operators.



Four ground states/The logical qubit

$$\{|\mathrm{GS}_0\rangle, X_1|\mathrm{GS}_0\rangle, X_2|\mathrm{GS}_0\rangle, X_2X_1|\mathrm{GS}_0\rangle\}$$

Distinguished by ±1 eigenvalues of Z_1 and Z_2.

Corresponding to $2(d^2-1)$ independent stabilizers on $2d^2$ physical qubits.

Topologically protected qubit

Non-trivial loops=Logical bit-flip operators
Requires at least *d* physical bit-flip errors
code distance *d*

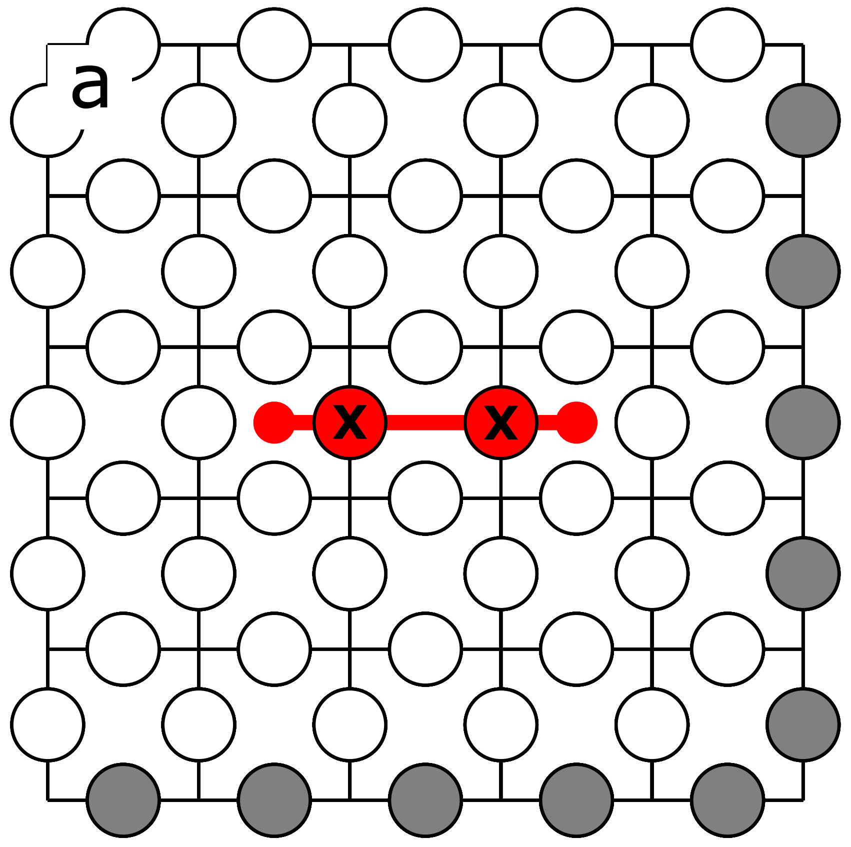
Error correction

consider bit-flip errors

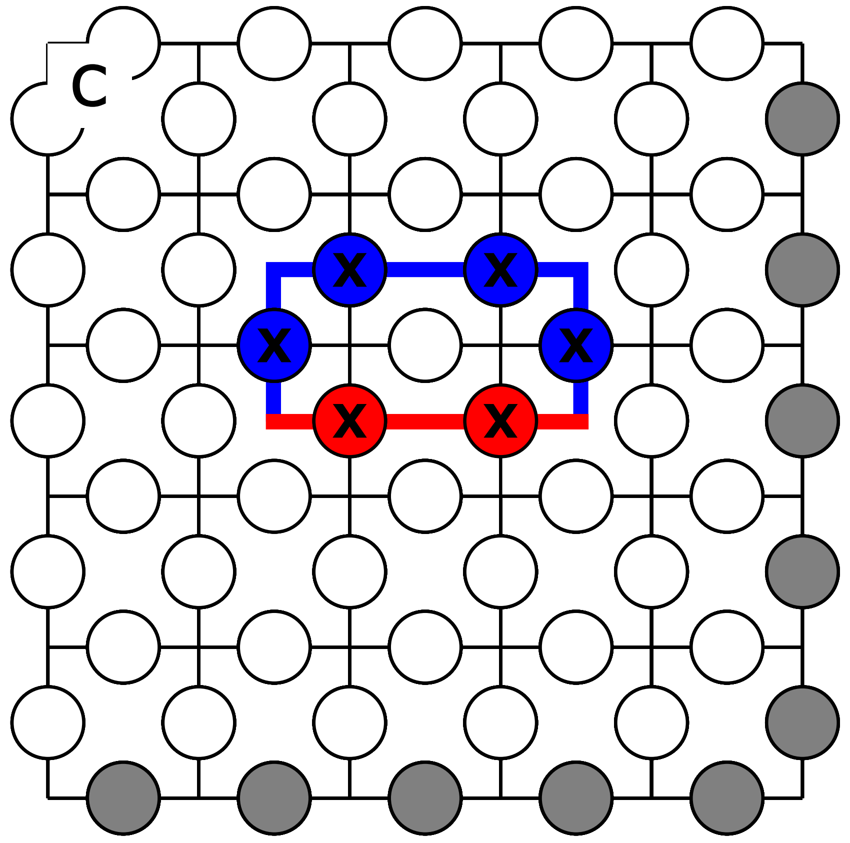
The ***syndrome*** (defects/bad plaquettes), given by quantum non-demolition measurement

Ex.

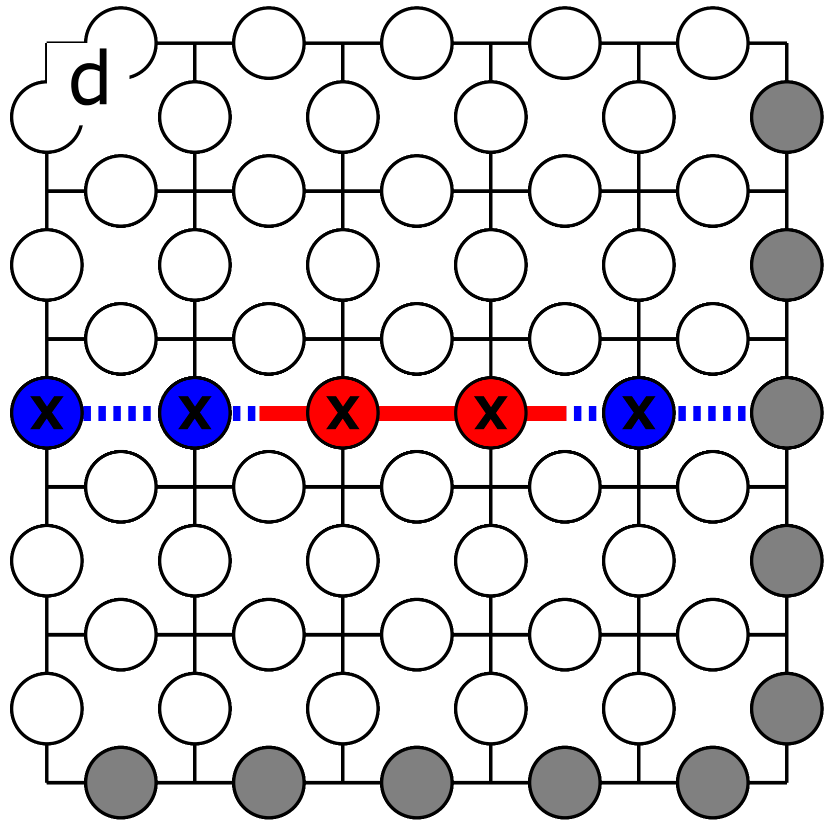
two neighbouring bit flip errors,
two defects



proper error correction
trivial loop



failed error correction
non-trivial loop



Same syndrome given by blue bit flip strings.
Need to learn the statistics of errors.

Standard algorithm to suggest error correcting strings:
Minimum Weight Perfect Matching (MWPM)/Blossom

J. Edmunds, 1965

Find shortest total correction string. (Which is the most likely)

Error models

Uncorrelated

- $(1-p)^2$ no error
- $p(1-p)$ X
- p^2 Y=XZ
- $p(1-p)$ Z

Bit- and phase-flip errors
(i.e. plaquette and vertex errors).
are independent. Corrected separately.

MWPM is (near) optimal

Look at this first

Depolarizing

- $(1-p)$ no error
- $p/3$ X
- $p/3$ Y=XZ
- $p/3$ Z

Plaquette and vertex errors are correlated.

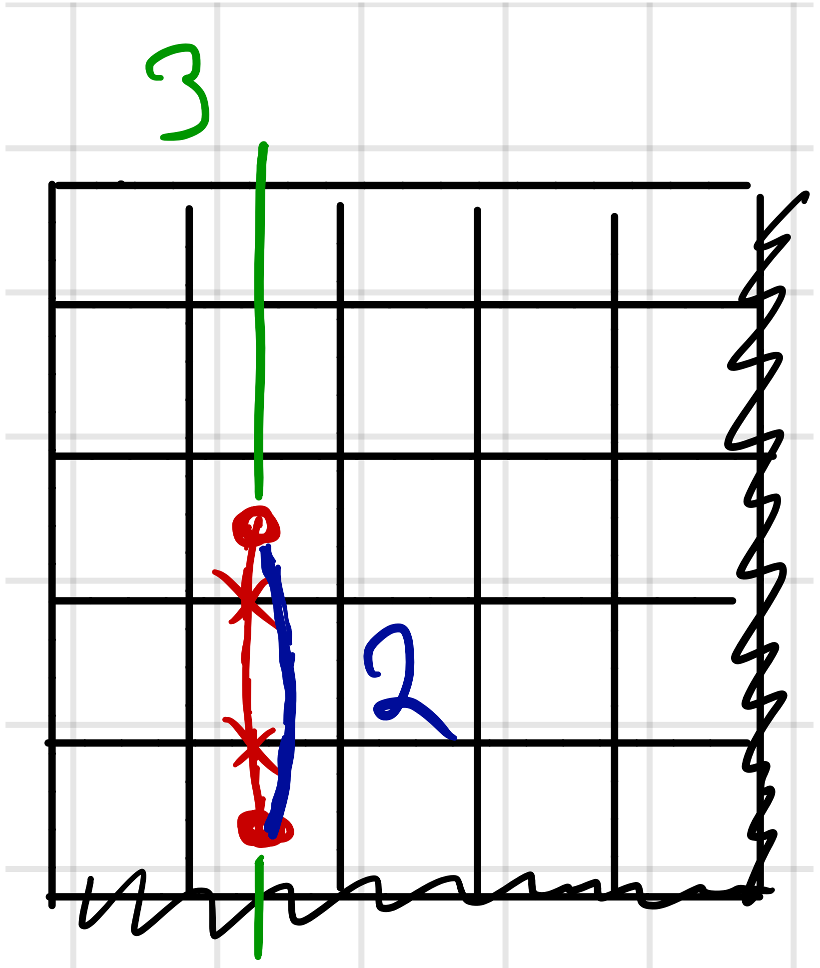
MWPM suboptimal

Minimum Weight Perfect Matching Low-p fail rate for bit-flip errors

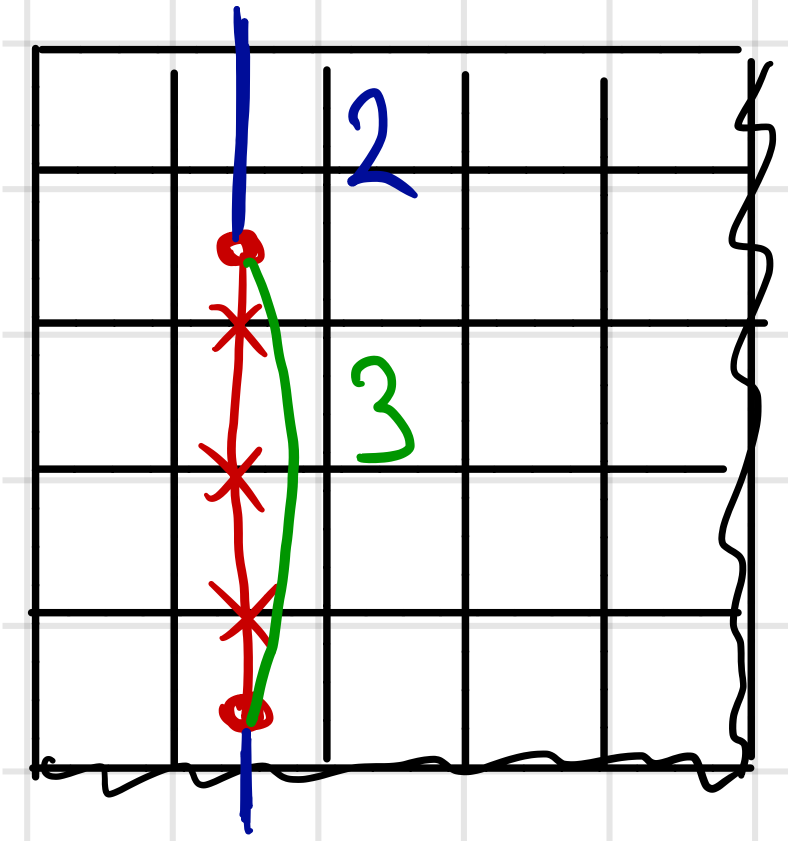
For p -> 0 we only need to consider error chains with minimal number of errors that can give failed error correction

Consider d=5:

Two errors is always corrected successfully

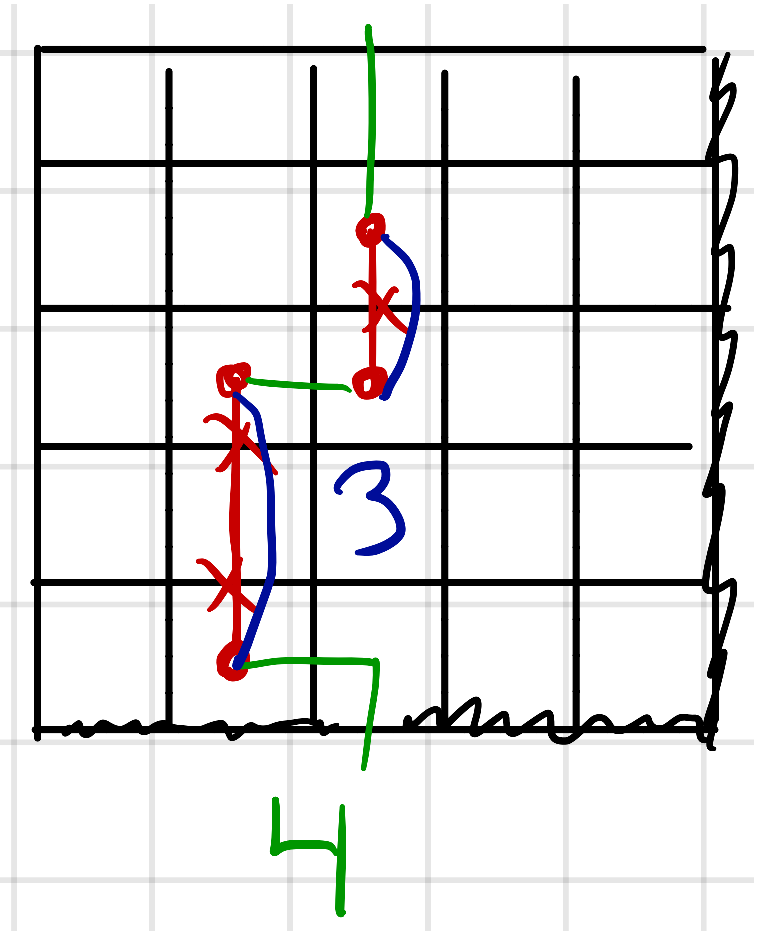


Three errors in a row always gives failed error correction



reward for RL?

Three errors not in a row always gives successful correction

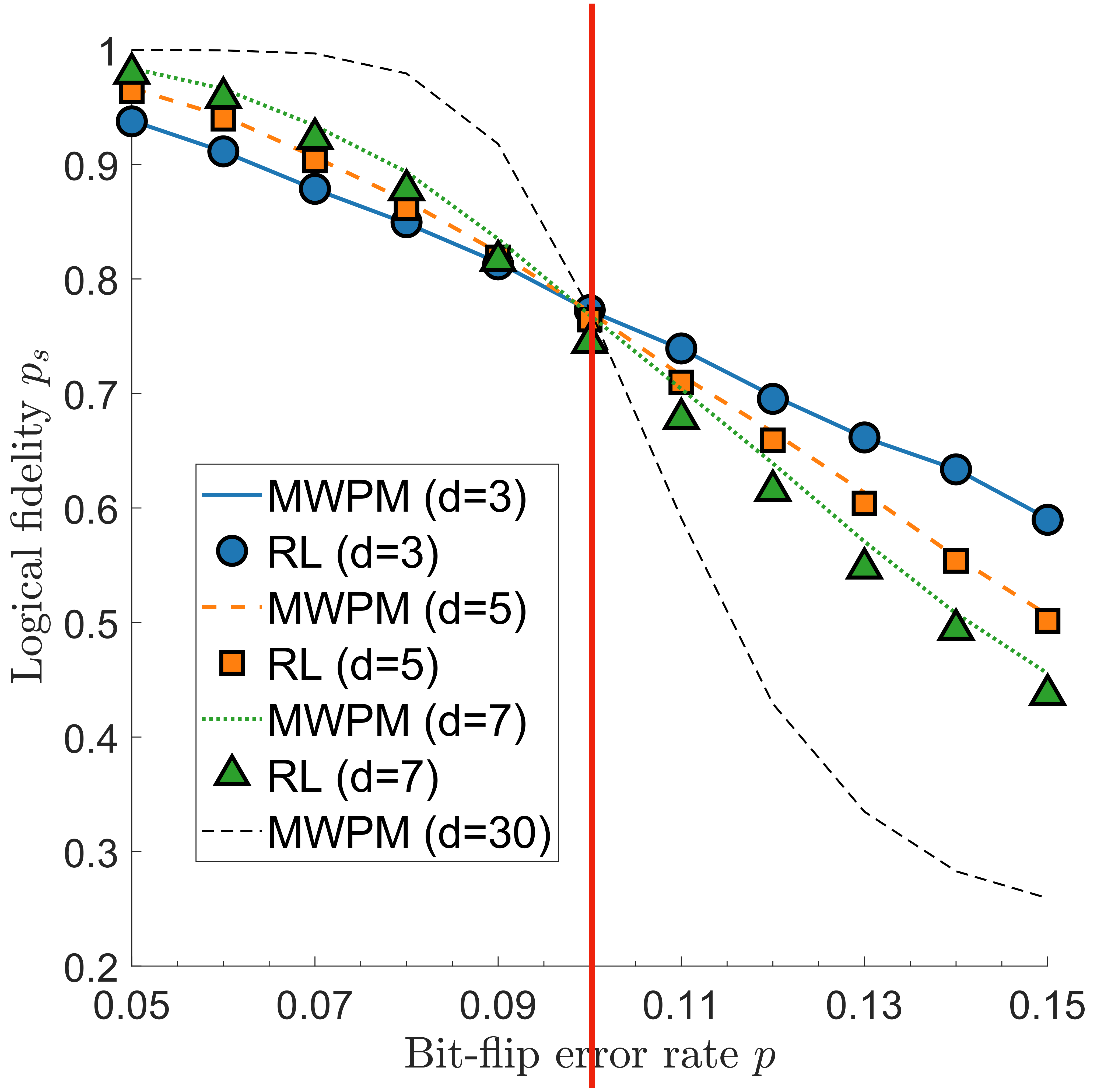


MWPM asymptotic (lowest order in p) fail rate is:

$$p_L = 2d \binom{d}{\lceil d/2 \rceil} p^{\lceil d/2 \rceil}$$

Mats Granath, MLQM, Nordita 2019

Threshold value for logical recovery rate versus error rate



Map to random bond Ising model
Phase transition for large d

For this problem MWPM is almost optimal

Threshold for Minimum Weight Perfect Matching decoder

Many decoder algorithms suggested

(non-exhaustive listing)

A renormalization group decoding algorithm for topological quantum codes

Guillaume Duclos-Cianci and David Poulin

PHYSICAL REVIEW A **89**, 022326 (2014)

Efficient Markov chain Monte Carlo algorithm for the surface code

Adrian Hutter, James R. Wootton, and Daniel Loss

Efficient algorithms for maximum likelihood decoding in the surface code

Sergey Bravyi, Martin Suchara, and Alexander Vargo
Phys. Rev. A **90**, 032326 – Published 25 September 2014

Cellular-automaton decoders for topological quantum memories

Michael Herold, Earl T Campbell, Jens Eisert & Michael J Kastoryano

npj Quantum Information **1**, Article number: 15010 (2015) | Download Citation

---

Neural Network Decoders for Large-Distance 2D Toric Codes

Xiaotong Ni
*QuTech, Delft University of Technology, P.O.Box 5046, 2600 GA Delft, The Netherlands.**
(Dated: September 19, 2018)

PRL **119,** 030501 (2017) PHYSICAL REVIEW LETTERS week ending 21 JULY 2017

Neural Decoder for Topological Codes

Giacomo Torlai and Roger G. Melko

Optimizing Quantum Error Correction Codes with Reinforcement Learning

Hendrik Poulsen Nautrup,[1,*] Nicolas Delfosse,[2] Vedran Dunjko,[3] Hans J. Briegel,[1,4] and Nicolai Friis[5,1]

Machine learning based decoders:

Machine-learning-assisted correction of correlated qubit errors in a topological code

P. Baireuther[1], T. E. O'Brien[1], B. Tarasinski[2], and C. W. J. Beenakker[1]

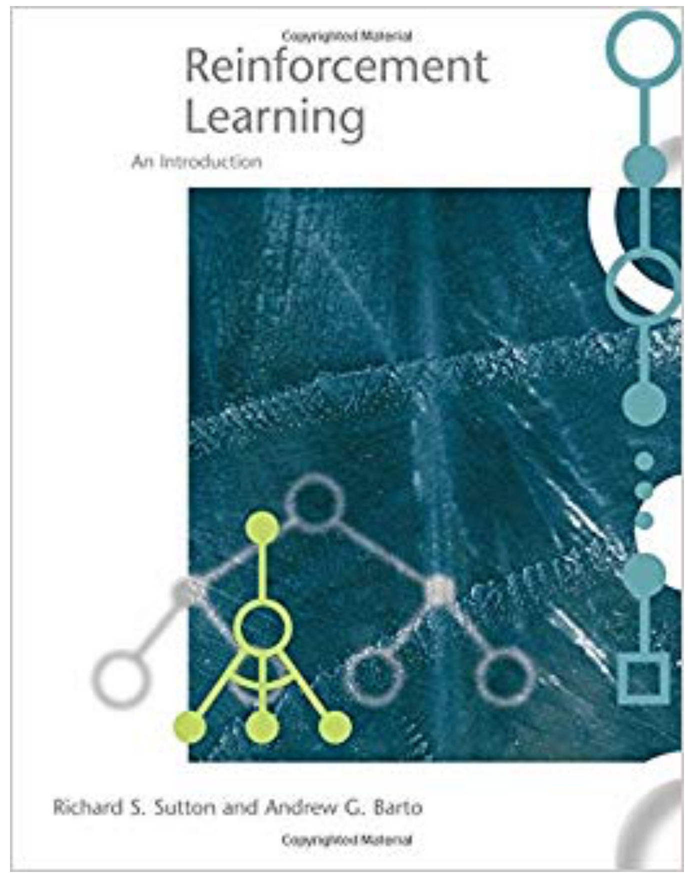
Reinforcement Learning Decoders for Fault-Tolerant Quantum Computation

Ryan Sweke,[1] Markus S. Kesselring,[1] Evert P. L. van Nieuwenburg,[2] and Jens Eisert[1,3]
[1]*Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany*
[2]*Institute for Quantum Information and Matter, Caltech, Pasadena, CA 91125, USA*
[3]*Department of Mathematics and Computer Science, Freie Universität Berlin, 14195 Berlin*
(Dated: October 18, 2018)

The Bitter Lesson

Rich Sutton

March 13, 2019



The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time

In computer chess, the methods that defeated the world champion, Kasparov, in 1997, were based on massive, deep search. At the time, this was looked upon with dismay by the majority of computer-chess researchers who had pursued methods that leveraged human understanding of the special structure of chess. When a simpler, search-based approach with special hardware

A similar pattern of research progress was seen in computer Go, only delayed by a further 20 years. Enormous initial efforts went into avoiding search by taking advantage of human knowledge, or of the special features of the game, but all those efforts proved irrelevant, or worse, once search was applied effectively at scale. Also important was the use of learning by self play to learn a value function (as it was in many other games and even in chess, although learning did not play a big role in the 1997 program that first beat a world champion). Learning

Message: High powered computations are key to progress!

Deep reinforcement learning/Deep Q-learning

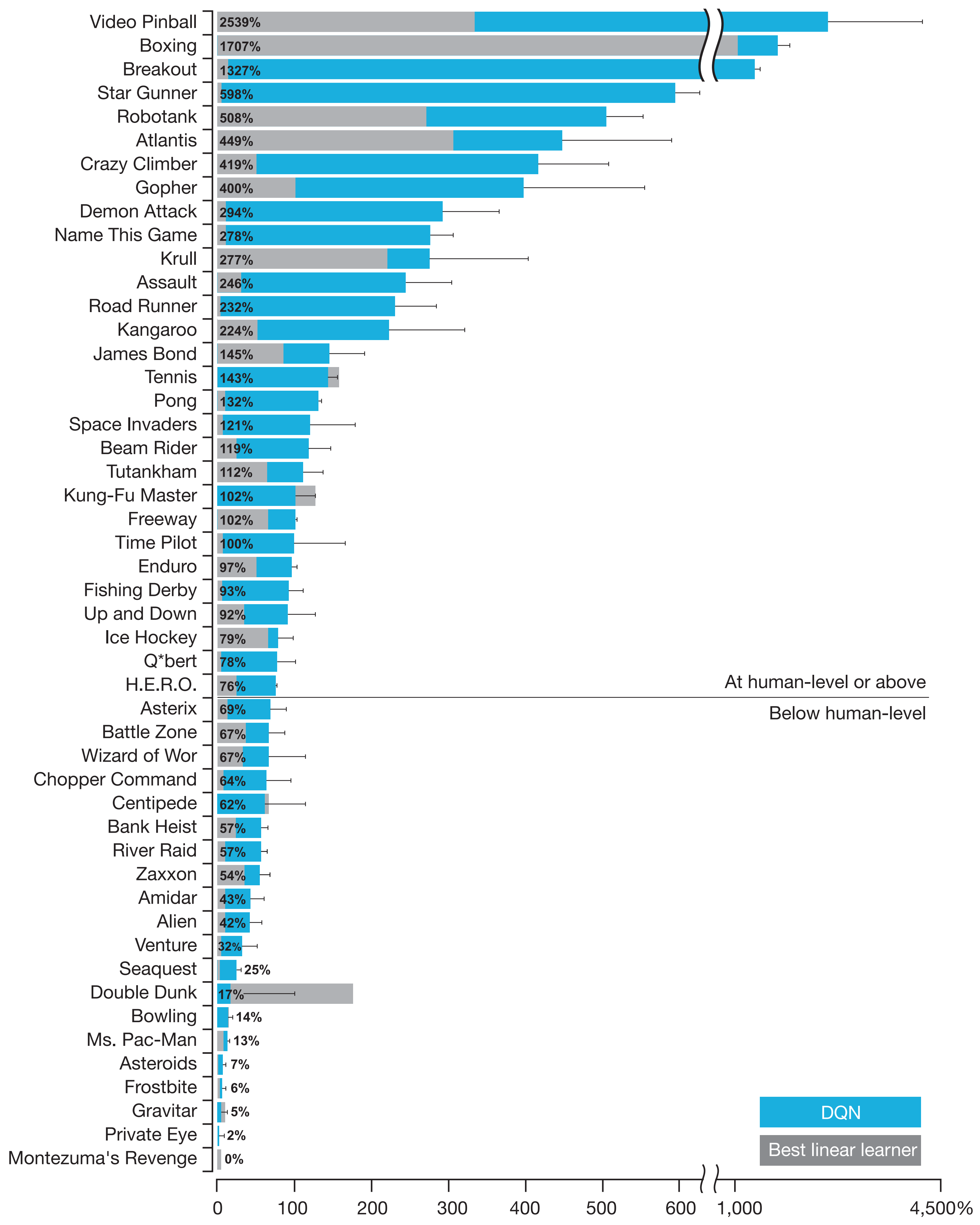
LETTER

2015

doi:10.1038/nature14236

Human-level control through deep reinforcement learning

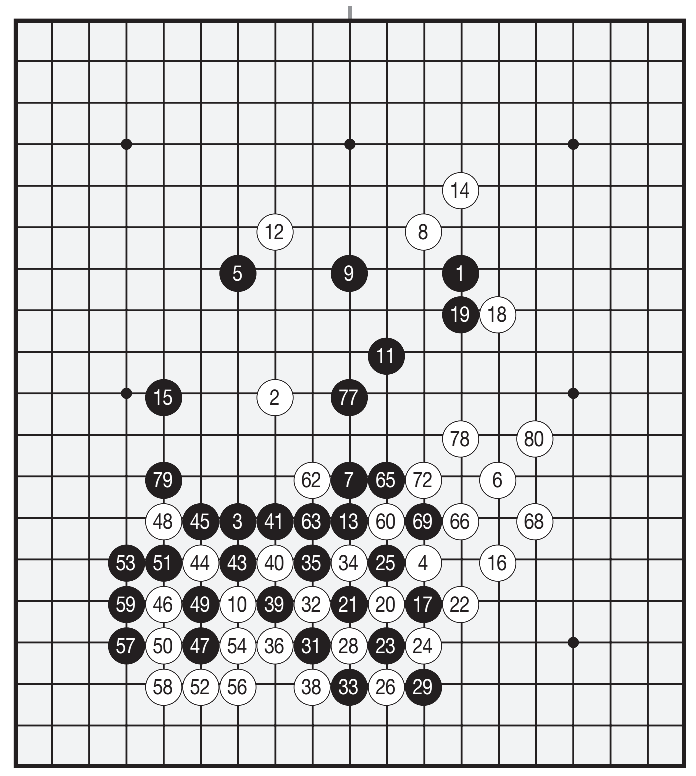
Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]



2017

Mastering the game of Go without human knowledge

David Silver[1]*, Julian Schrittwieser[1]*, Karen Simonyan[1]*, Ioannis Antonoglou[1], Aja Huang[1], Arthur Guez[1], Thomas Hubert[1], Lucas Baker[1], Matthew Lai[1], Adrian Bolton[1], Yutian Chen[1], Timothy Lillicrap[1], Fan Hui[1], Laurent Sifre[1], George van den Driessche[1], Thore Graepel[1] & Demis Hassabis[1]



AlphaStar 2019

Q-learning

- Agent in an environment described by a **state** *s*.
- Agent takes **actions** *a* to move between states, *s* -> *s*'.
- **Reward** (positive or negative) *r* is given depending on state/action.
- Agent learns **policy**, *π(s,a)*, to navigate environment for optimal accumulated reward (return) by exploring.

Q-function (action-value fcn) *Q(s,a)* **quantifies expected return from taking action** *a* **in state** *s* **and subsequently following the optimal policy.**

$$\boxed{Q(s,a) = r + \gamma \max_{a'} Q(s',a')}$$

γ<1 is discounting factor, better to get reward now than later

Explore to get reward and learn Q => optimal policy

Difficult if big world with many states and actions

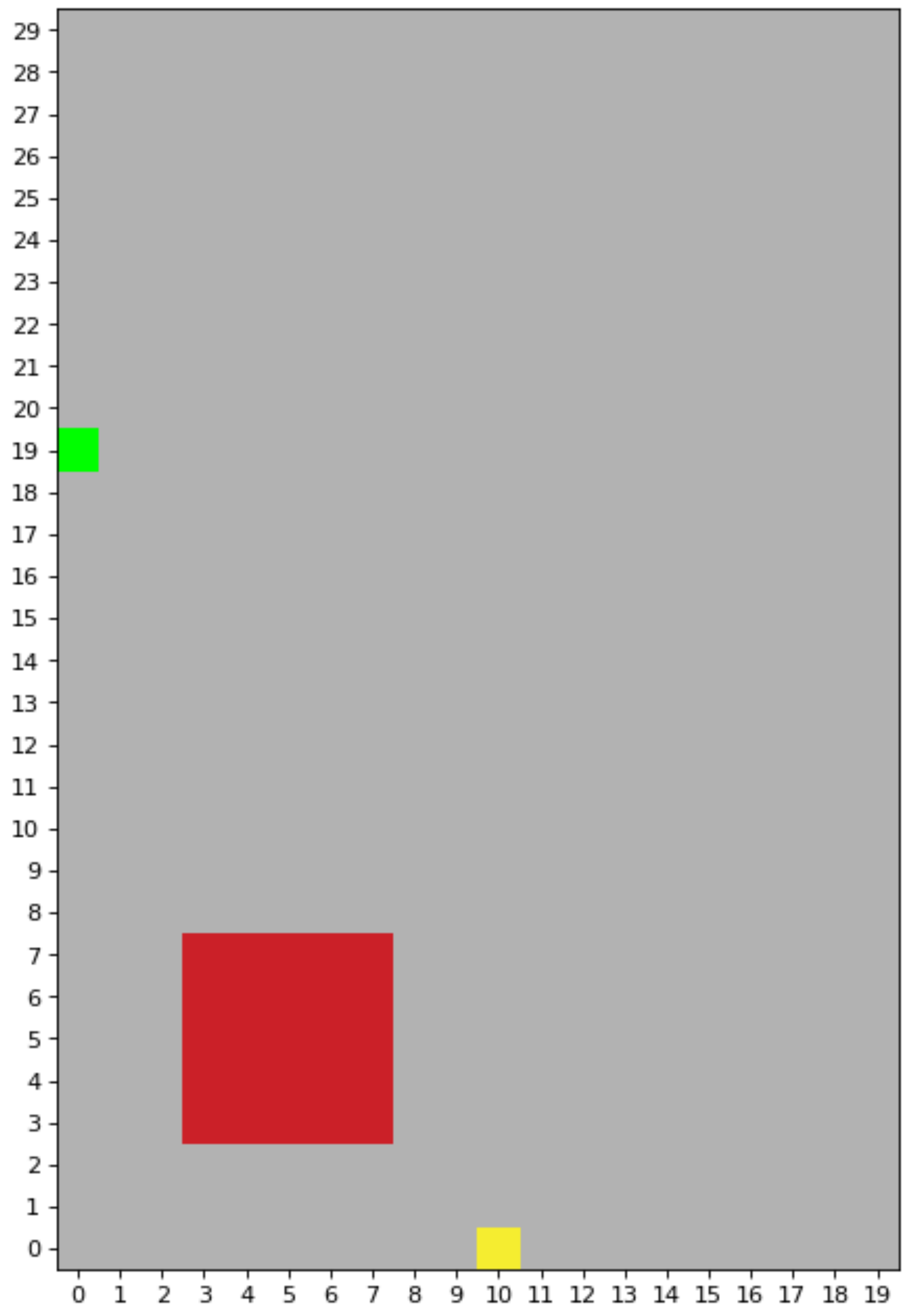
Use Artificial Neural Network to represent Q-function
Deep Q-learning

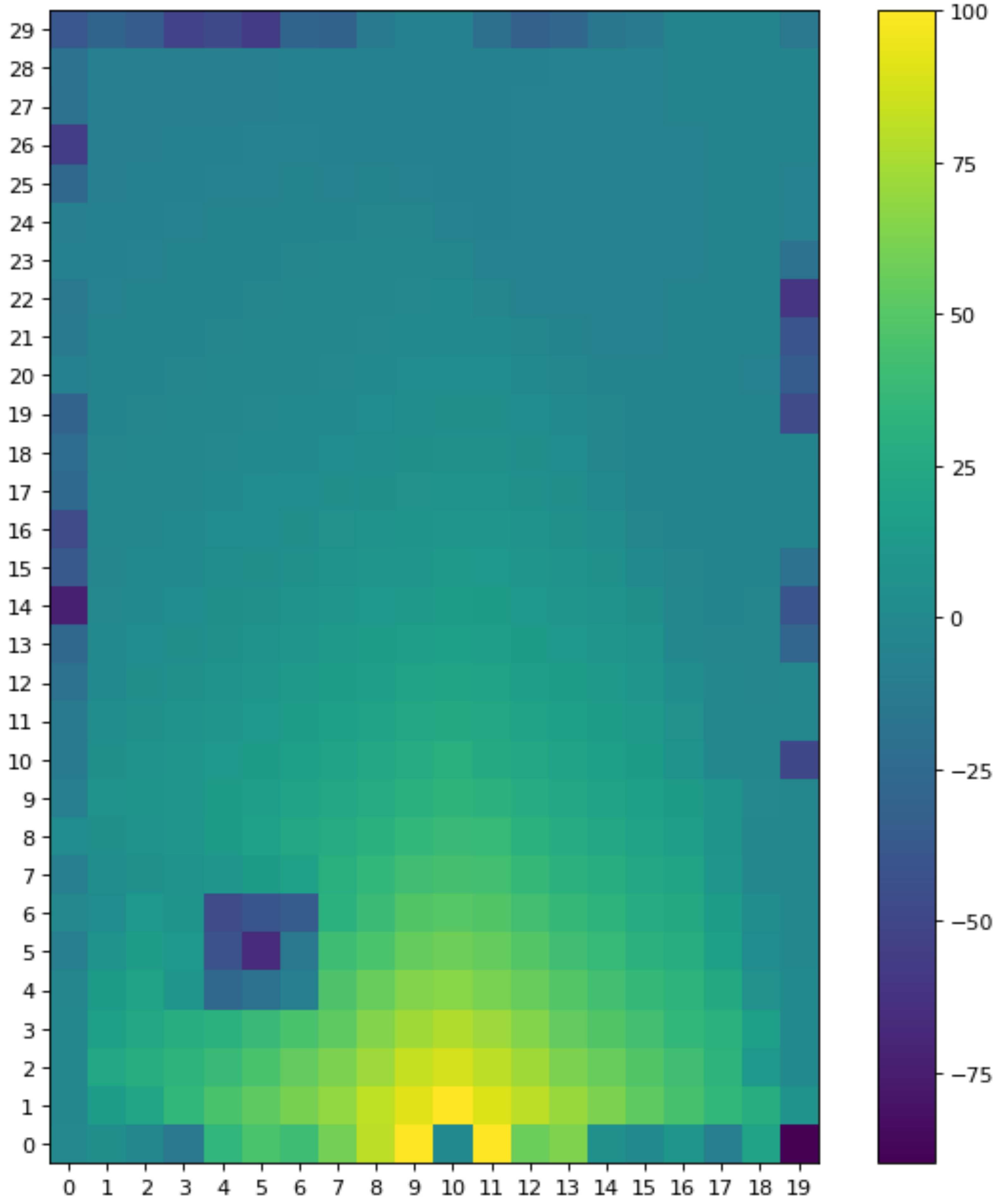
Q-learning example

from undergraduate course at Gothenburg University

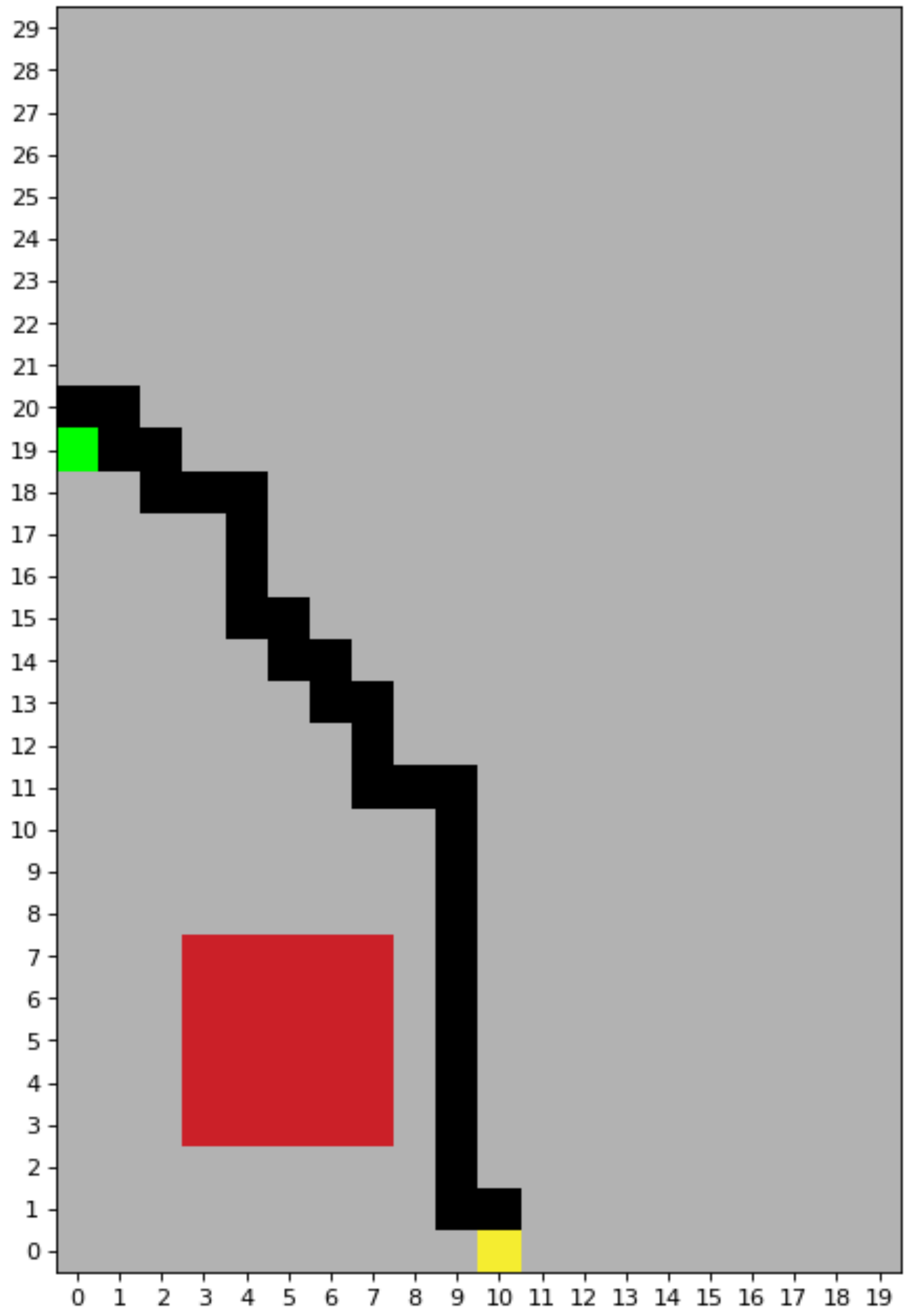
“grid-world” with **fire** (red) and **cliffs** on the side and **treacherous wind**

learn to move from green to yellow in as few steps





$$V(s) = \max_a Q(s,a)$$

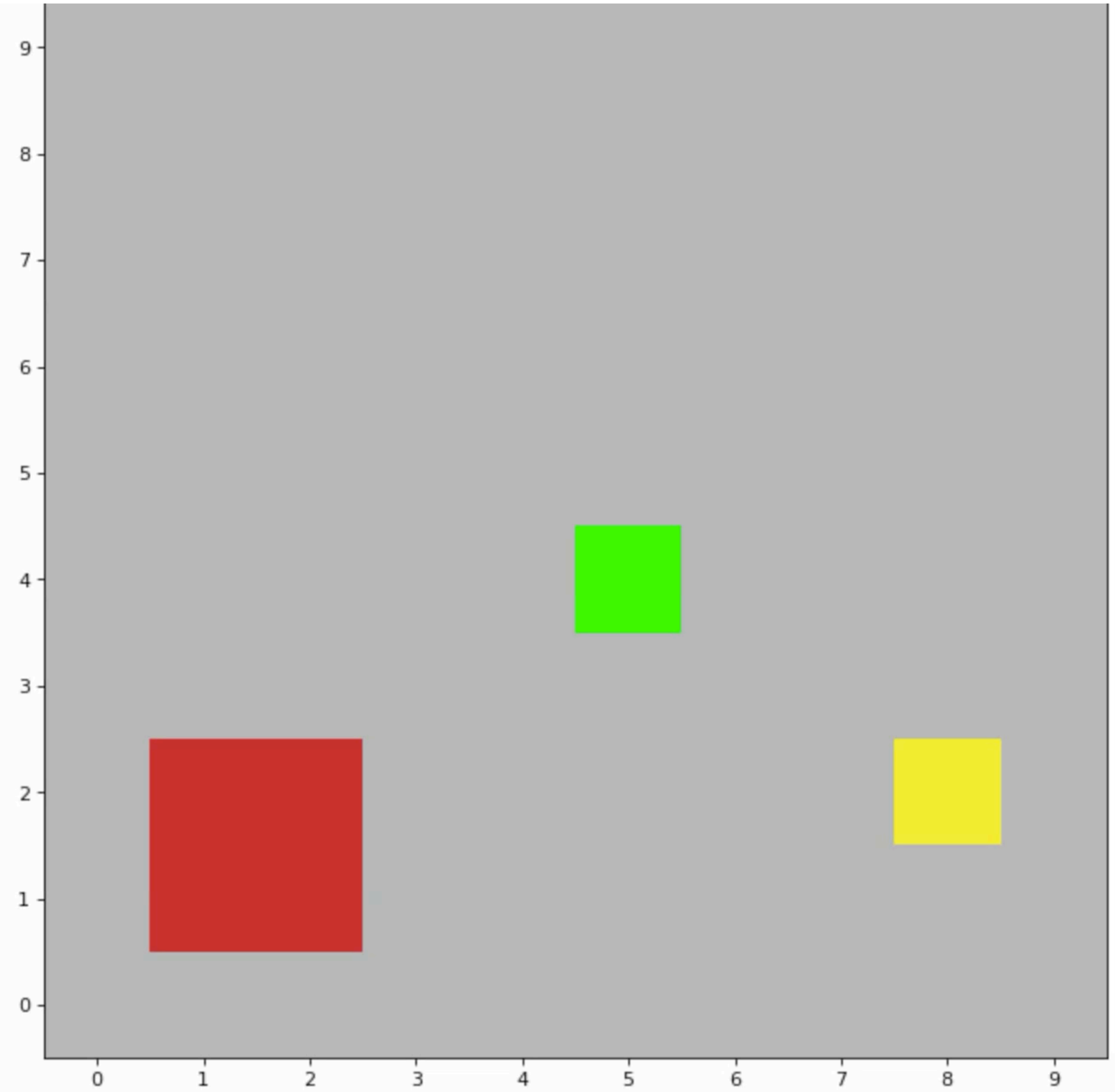


optimal policy path

Small state-action space. Easy to store.

Deep Q-learning

Dynamic fire gives huge state space, (2^{100})
Different Q-function for each configuration of fire.



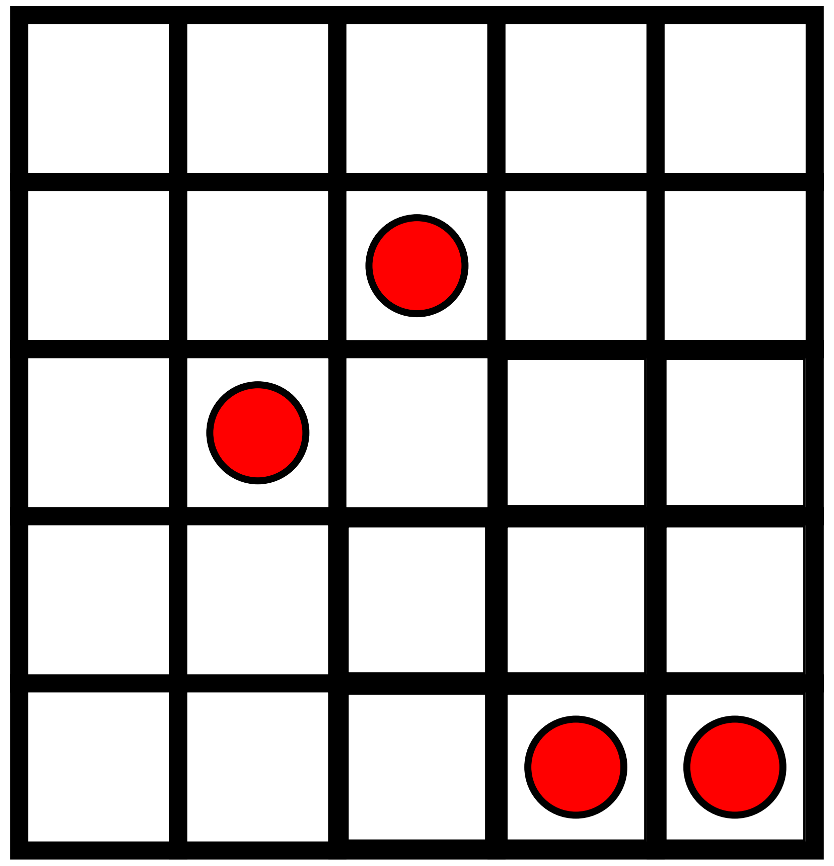
Fire spread


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 8, 8, 32)          896
_________________________________________________________________
batch_normalization_1 (Batch (None, 8, 8, 32)          128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 6, 6, 32)          9248
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 4612
=================================================================
Total params: 14,884
Trainable params: 14,820
Non-trainable params: 64
```

Network can generalize state-actions

Q-learning for the toric code with bit flip error

state is a syndrome
action is a bitflip=cardinal move of defect



State space is very big
number of ways of placing N_S defects on d^2 sites:

$$\binom{d^2}{N_s} \approx \binom{49}{20} \sim 10^{13}$$

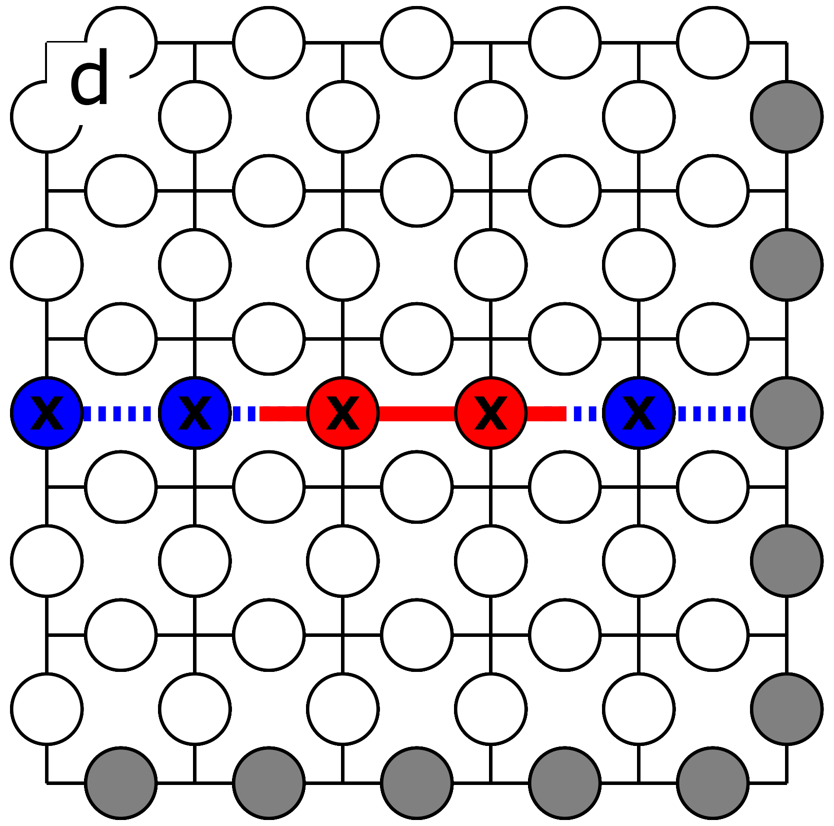
for d=7 and p=10%

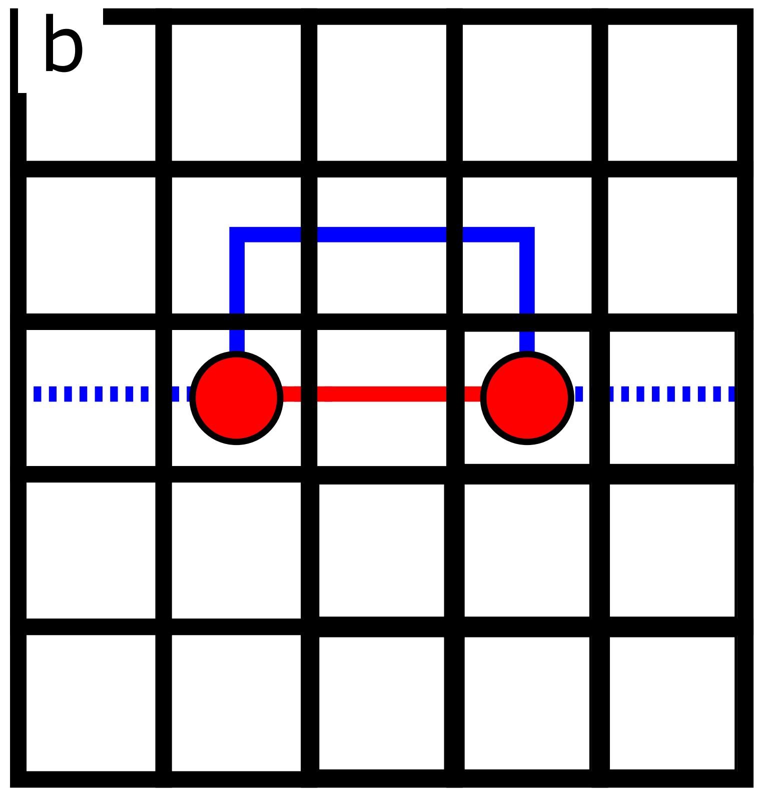
Use deep Q-learning

Reward scheme is a challenge

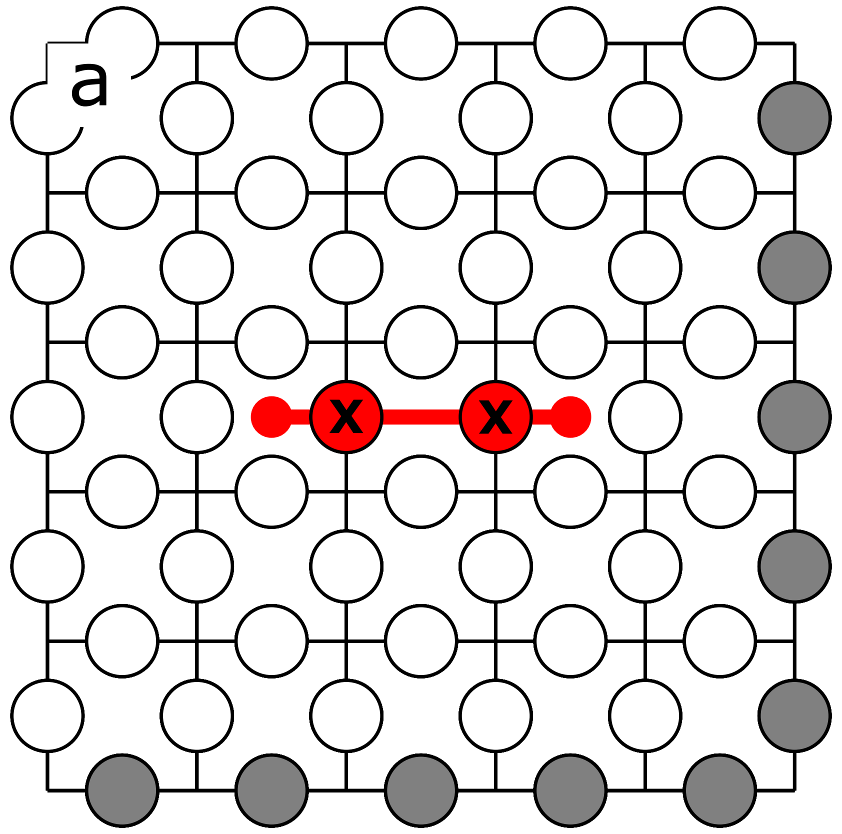
Natural to give reward after episode eliminating all errors: But weak signal.

Red error chain and blue error chain has same syndrome





If red is suggested recovery chain. This can give both positive and negative reward

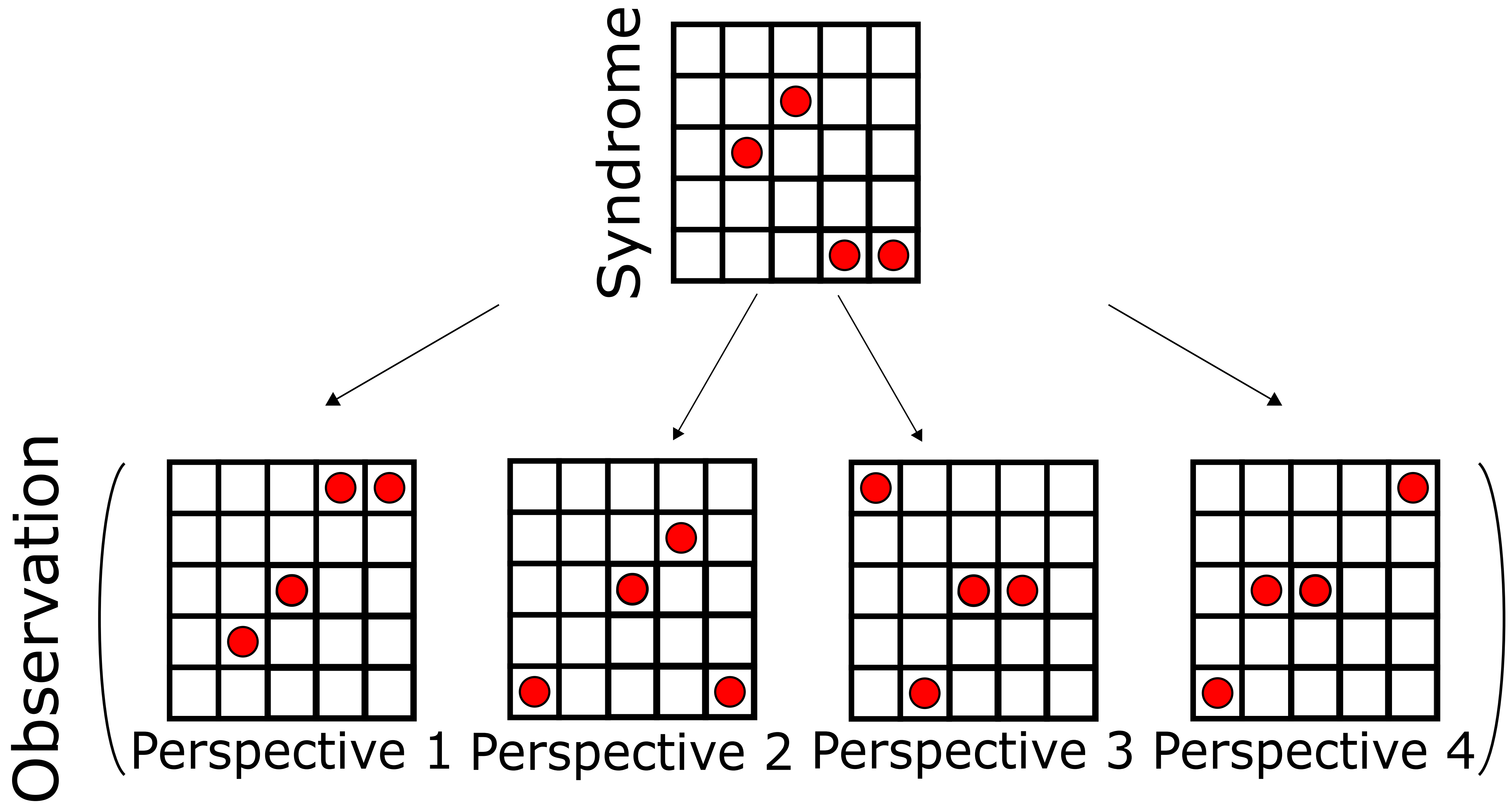


Instead, try to learn minimum number of correction steps:

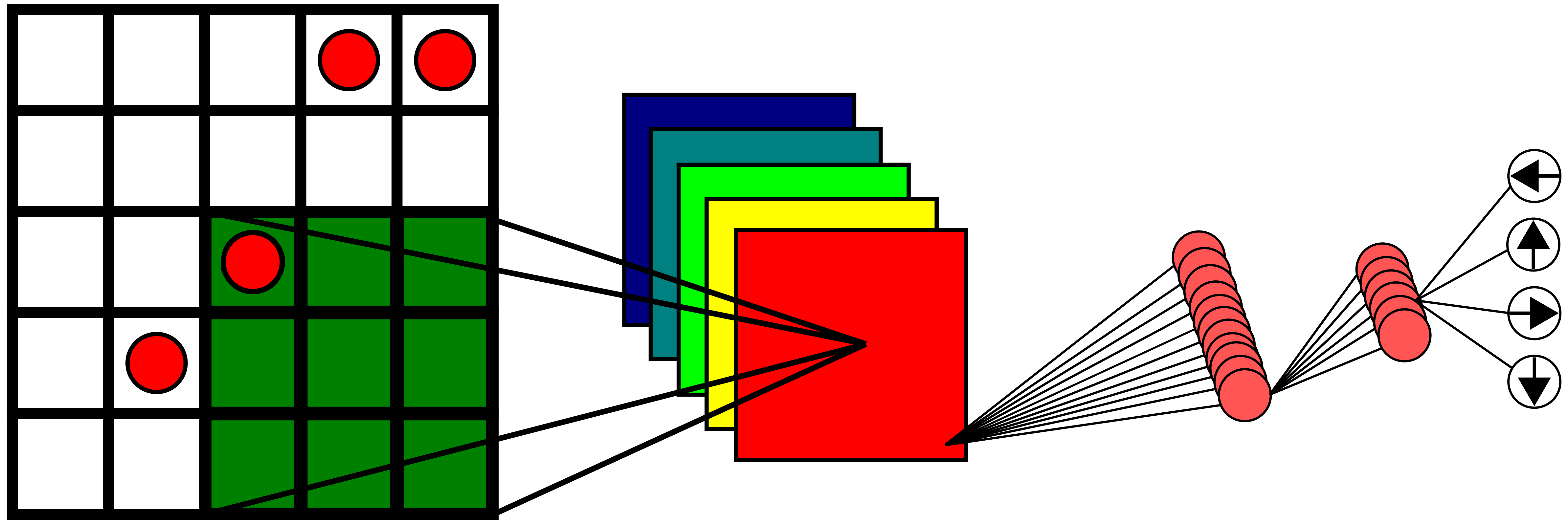
reward, r=-1 per move (i.e. we aim to learn MWPM)

Efficient implementation of Q-network

Use translational and rotational symmetry
to center each defect.



Convolutional NN



Deep Q-network

Network gives Q-values for the 4 movements of the **central** defect.
Crucial simplification, fixed number (4) actions, and doesn't have to learn about boundaries.

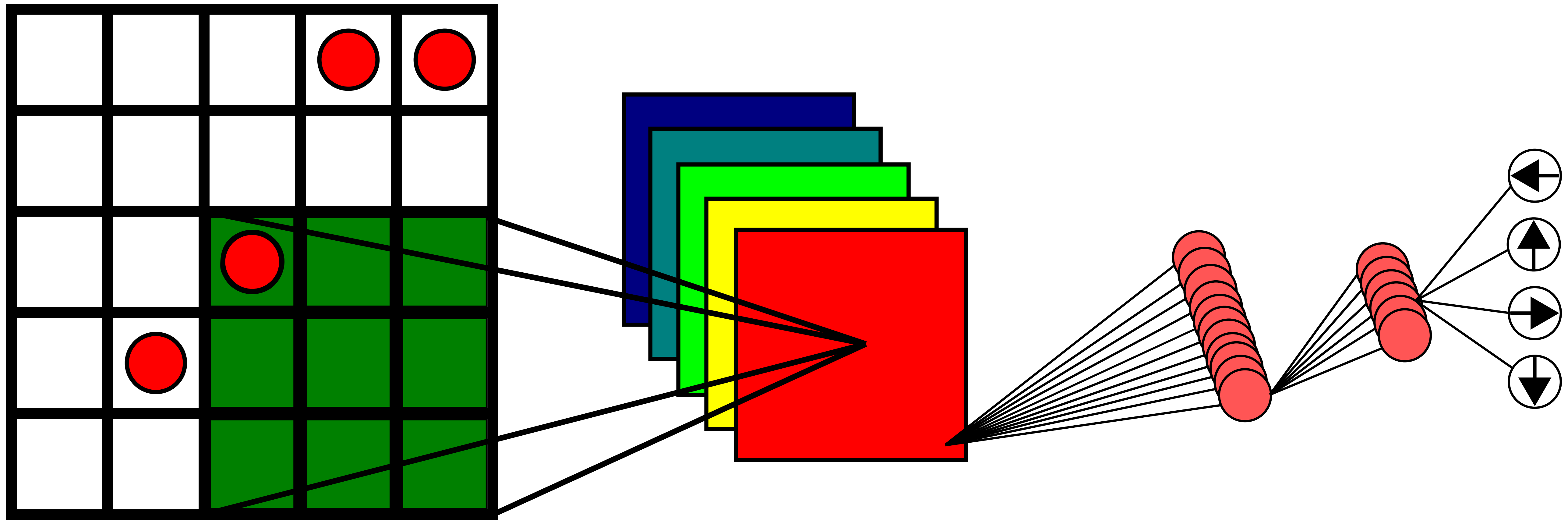


Table 2: Network architecture d=7.

#	Type	Size	# parameters
0	Input	7x7	
1	Conv.	512 filters; 3x3 size;	
		2-2 stride	5 120
2	FC	256 neurons	1 179 904
3	FC	128 neurons	32 896
4	FC	64 neurons	8 256
5	FC	32 neurons	2 080
6	FC (out)	4 neurons	132
			1 228 388

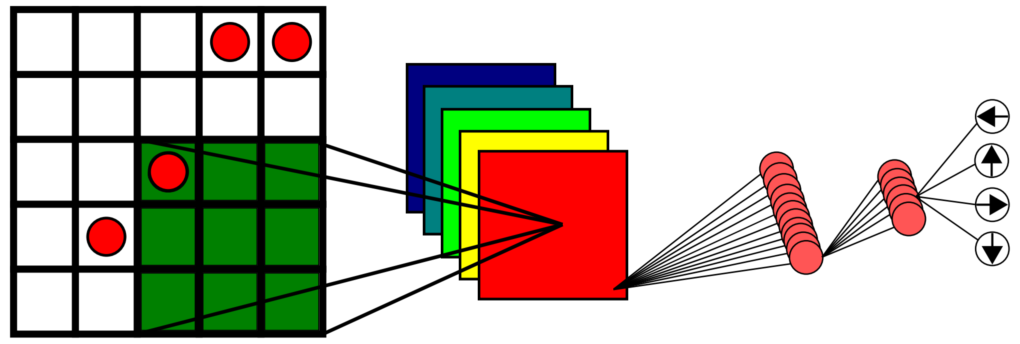
Significant reduction in number of parameters.
Size of state space for d=7, and N_S=20 defects (10% error)

$$\binom{d^2}{N_s} \approx \binom{49}{20} \sim 10^{13}$$

Training Q-network using supervised learning

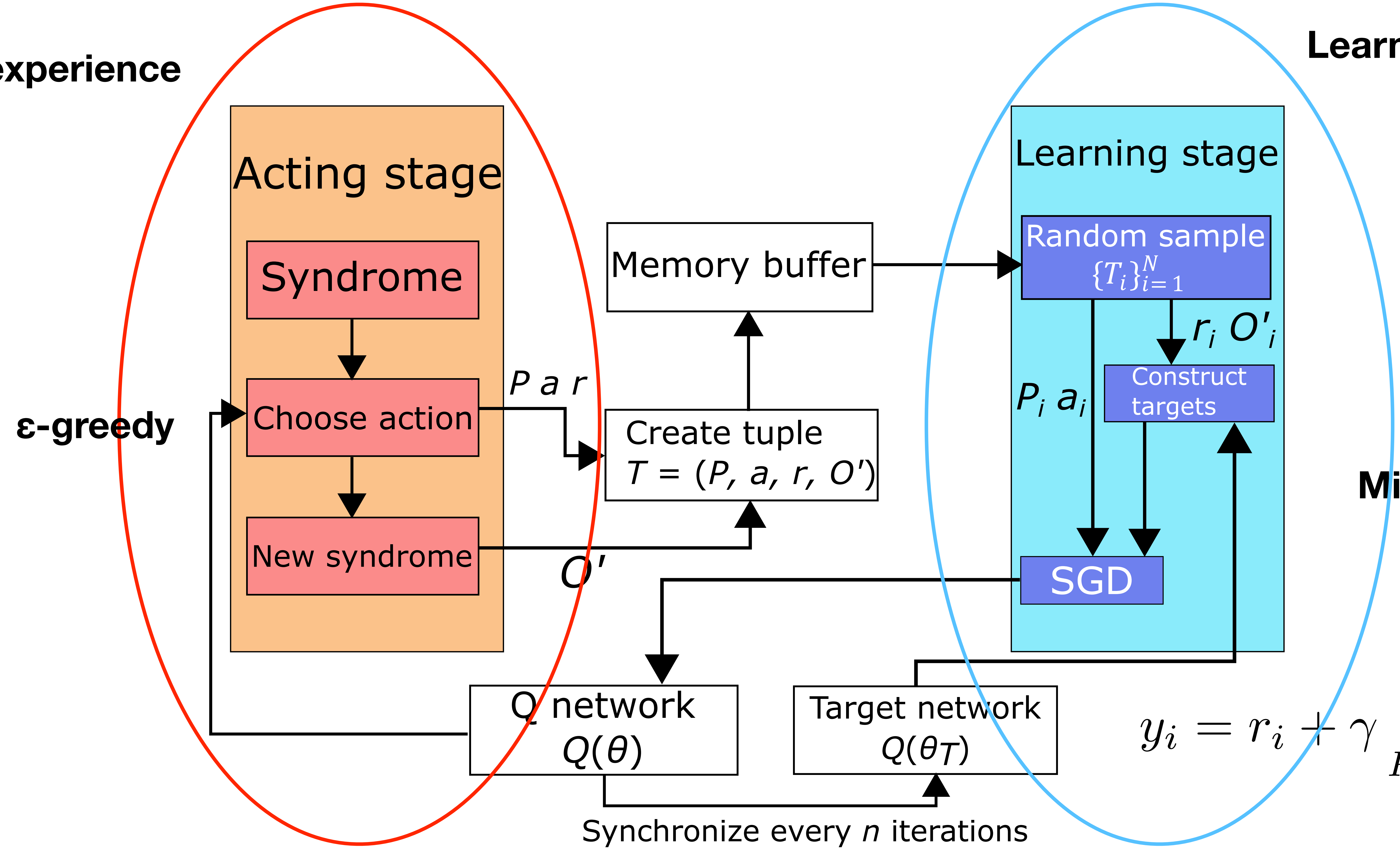
Challenging to to converge

- experience replay
- separate target and policy networks



Gain experience

Learn using experience



Minibatch training using Memory buffer

Training target

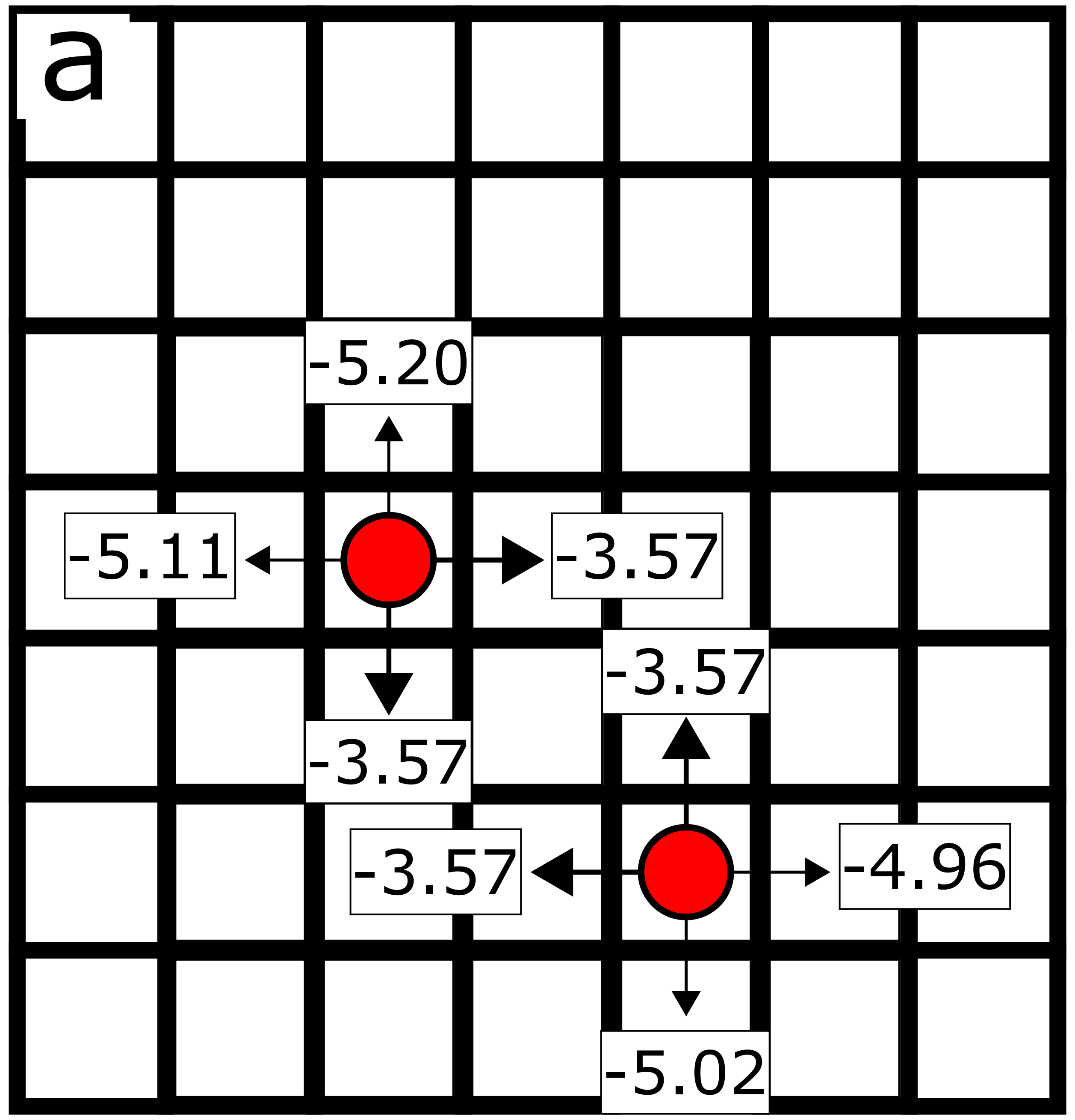
$$y_i = r_i + \gamma \max_{P' \in O'_i; a'} Q(P', a', \theta_T)$$

Results. Converged Q-network.

Examples:

Large arrow=Large Q-value for that action

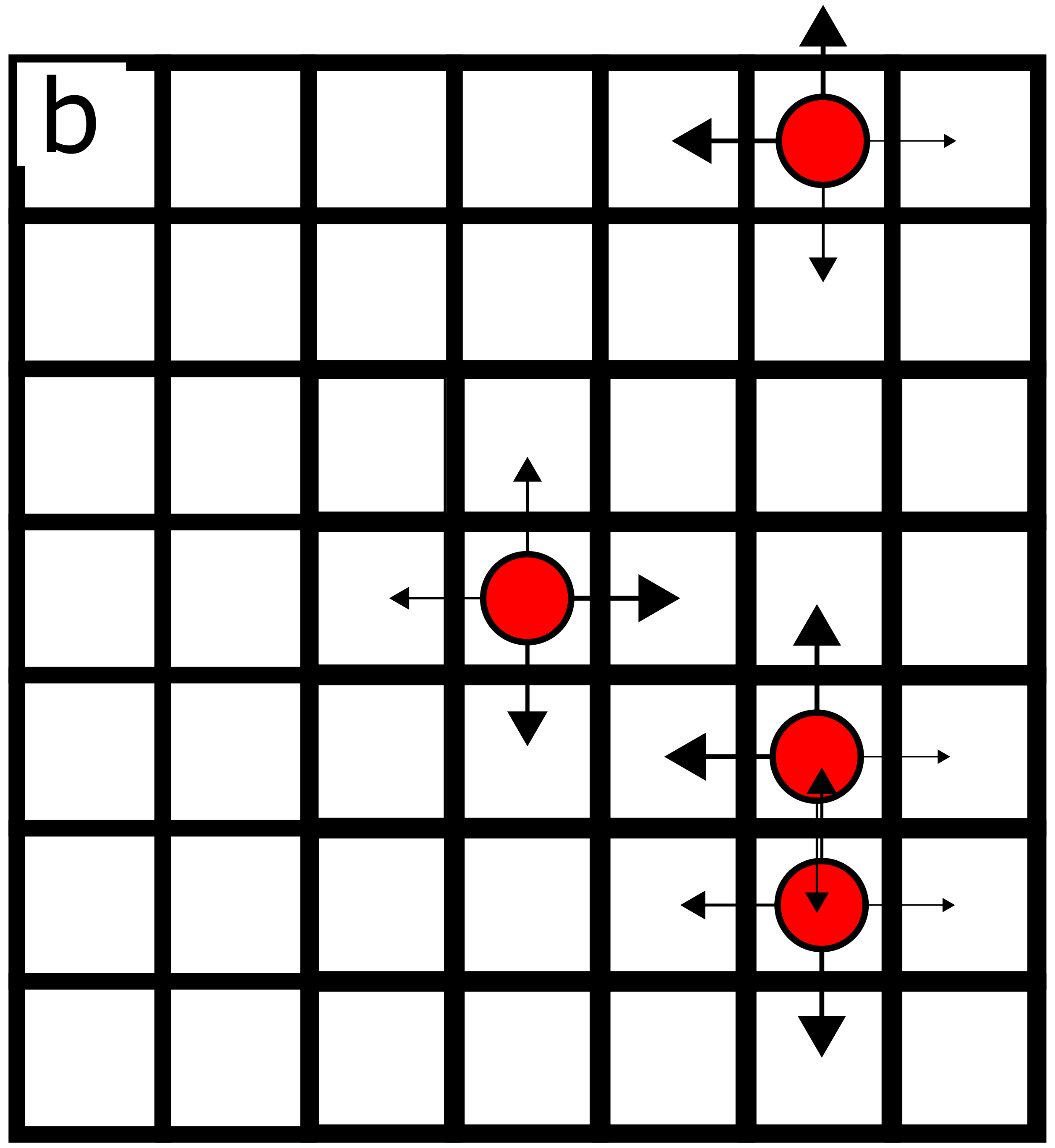
4-steps to elimination



$$R = -1 - \gamma - \gamma^2 - \gamma^3 = -3.62$$

$$\gamma = 0.95$$

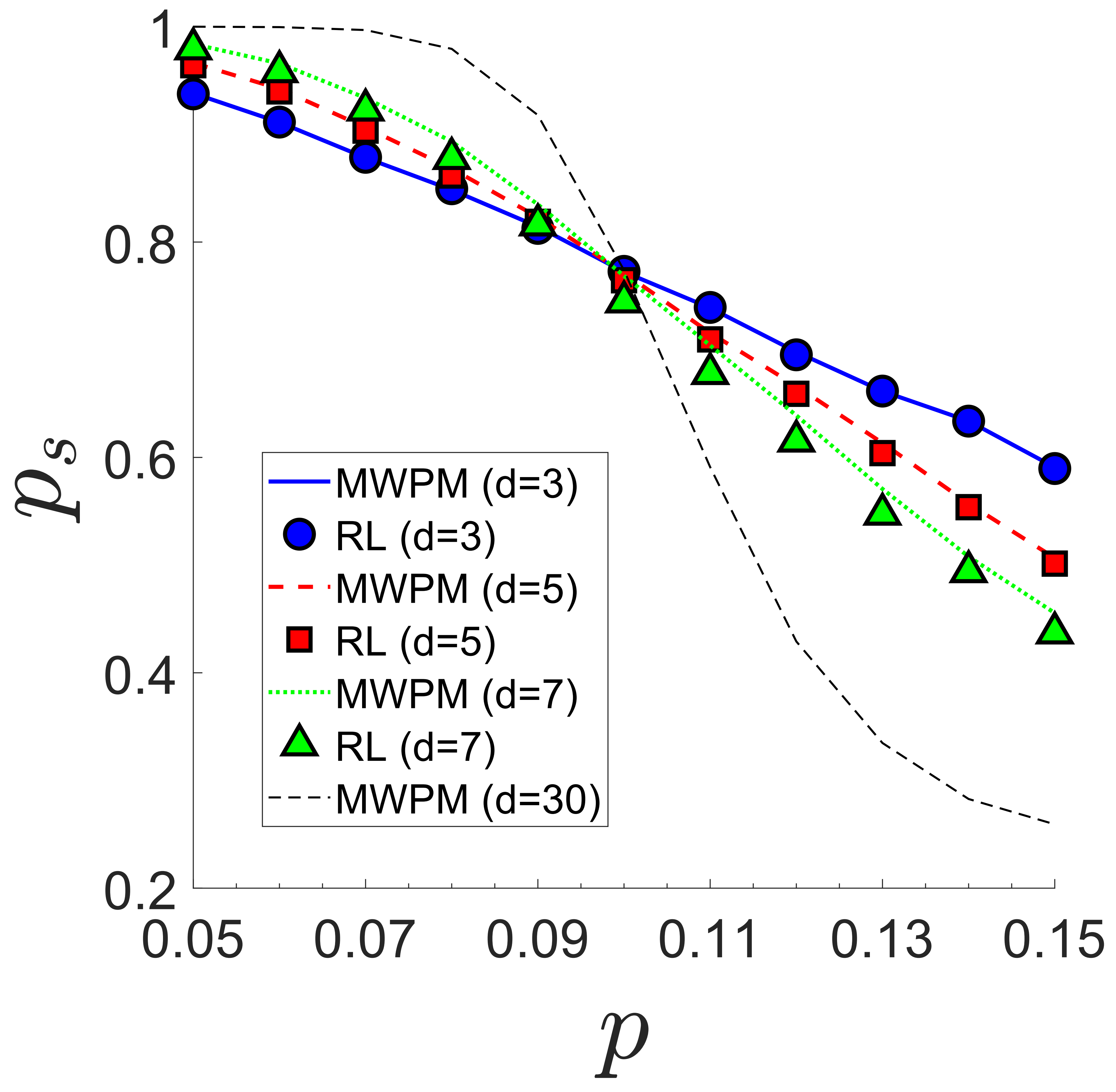
Shortest total path (MWPM)



quantitatively correct Q-values

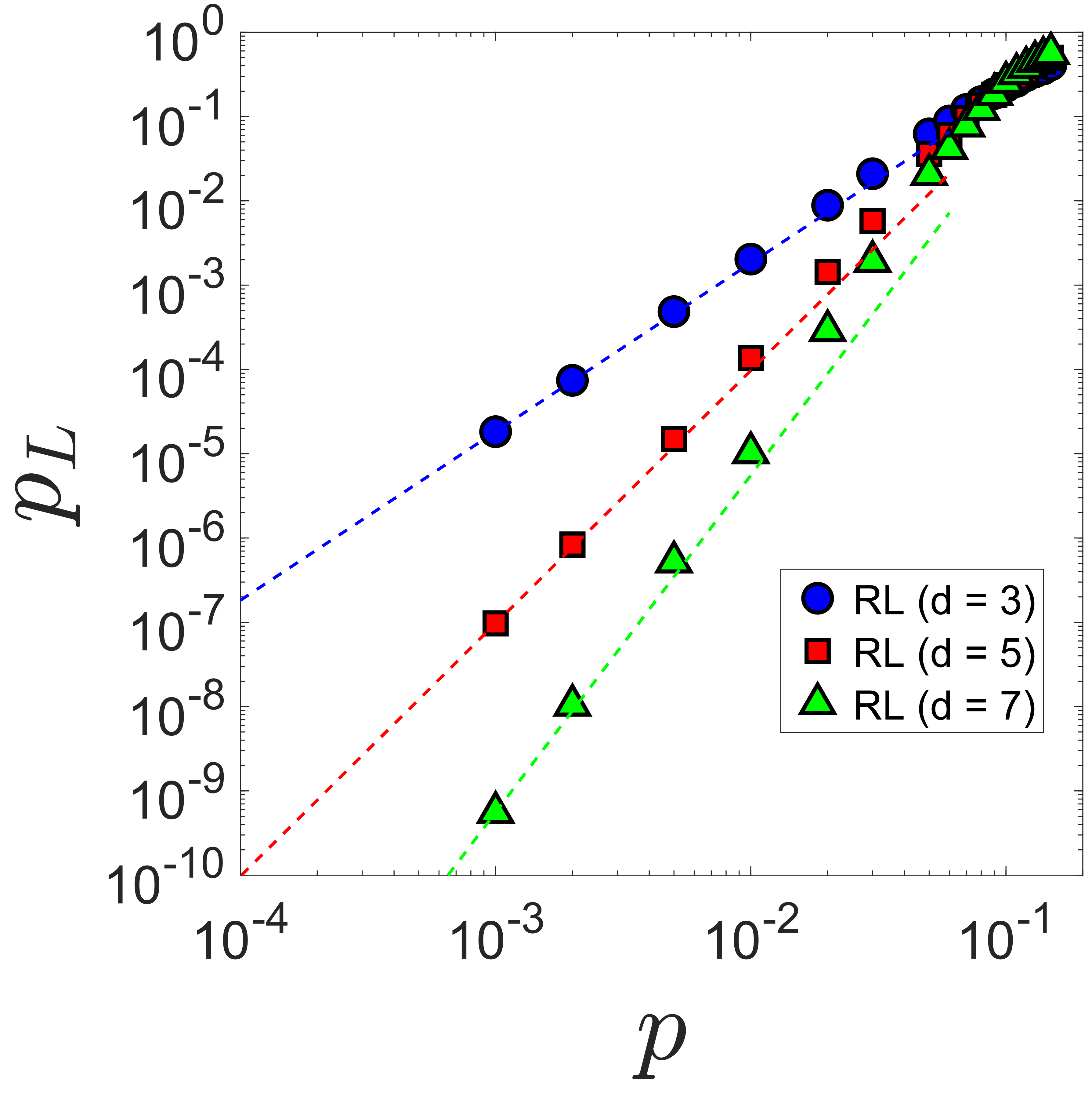
Results

Logical success-rate, large error rates close to MWPM



bit flip error rate

Logical fail-rate, small error rates identical to MWPM



Fits asymptotic form for small p:

$$p_L = 2d\binom{d}{\lceil d/2 \rceil} p^{\lceil d/2 \rceil}$$

Bottom line

We do the “simplest” error correction problem for a topological code

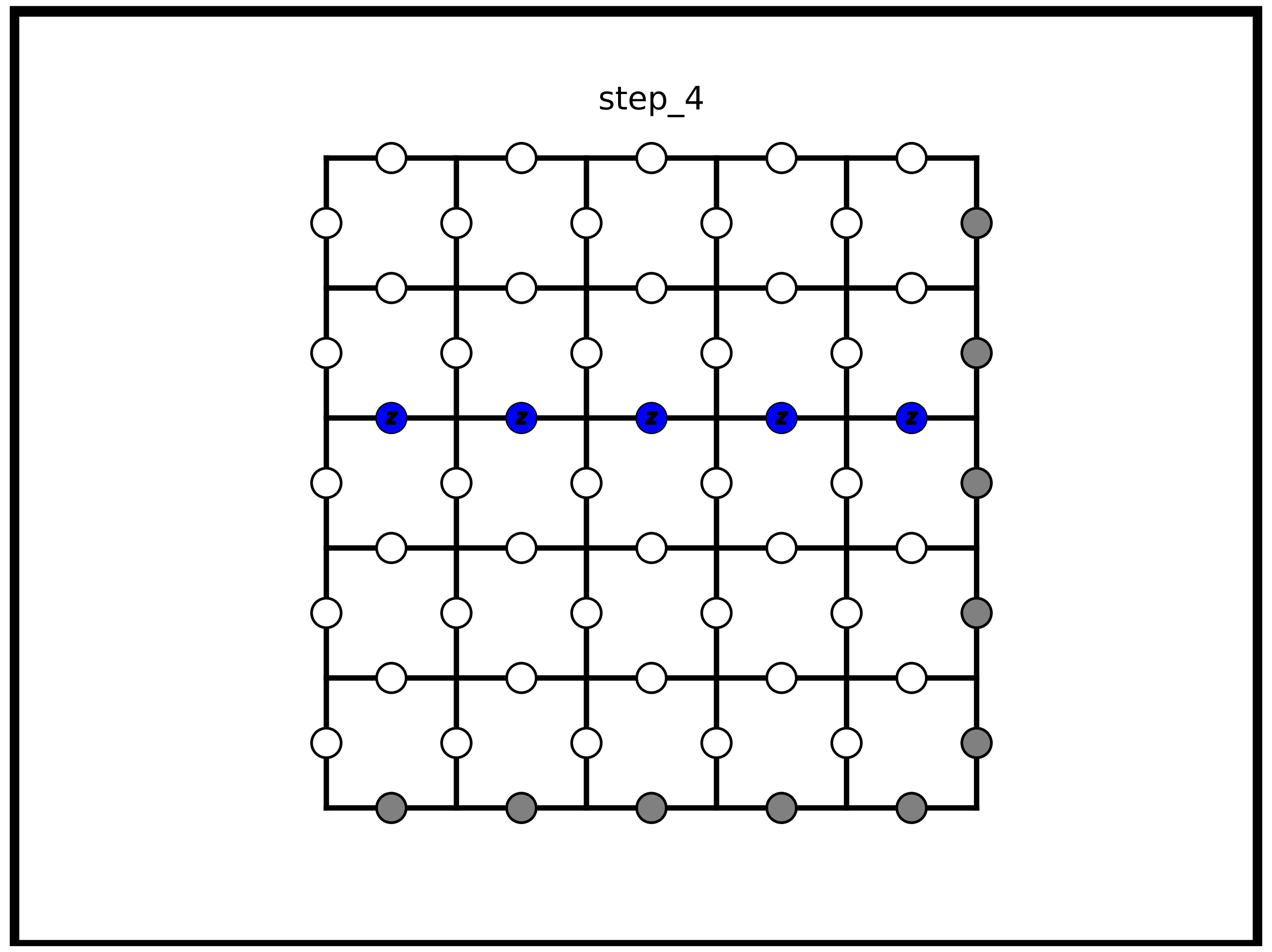
- Periodic boundary conditions
- No measurement noise/perfect syndrome
- only bit flip noise

Still challenging for reinforcement learning: deep Q-networks needed
Allows for easy benchmark

Depolarizing noise, work in progress

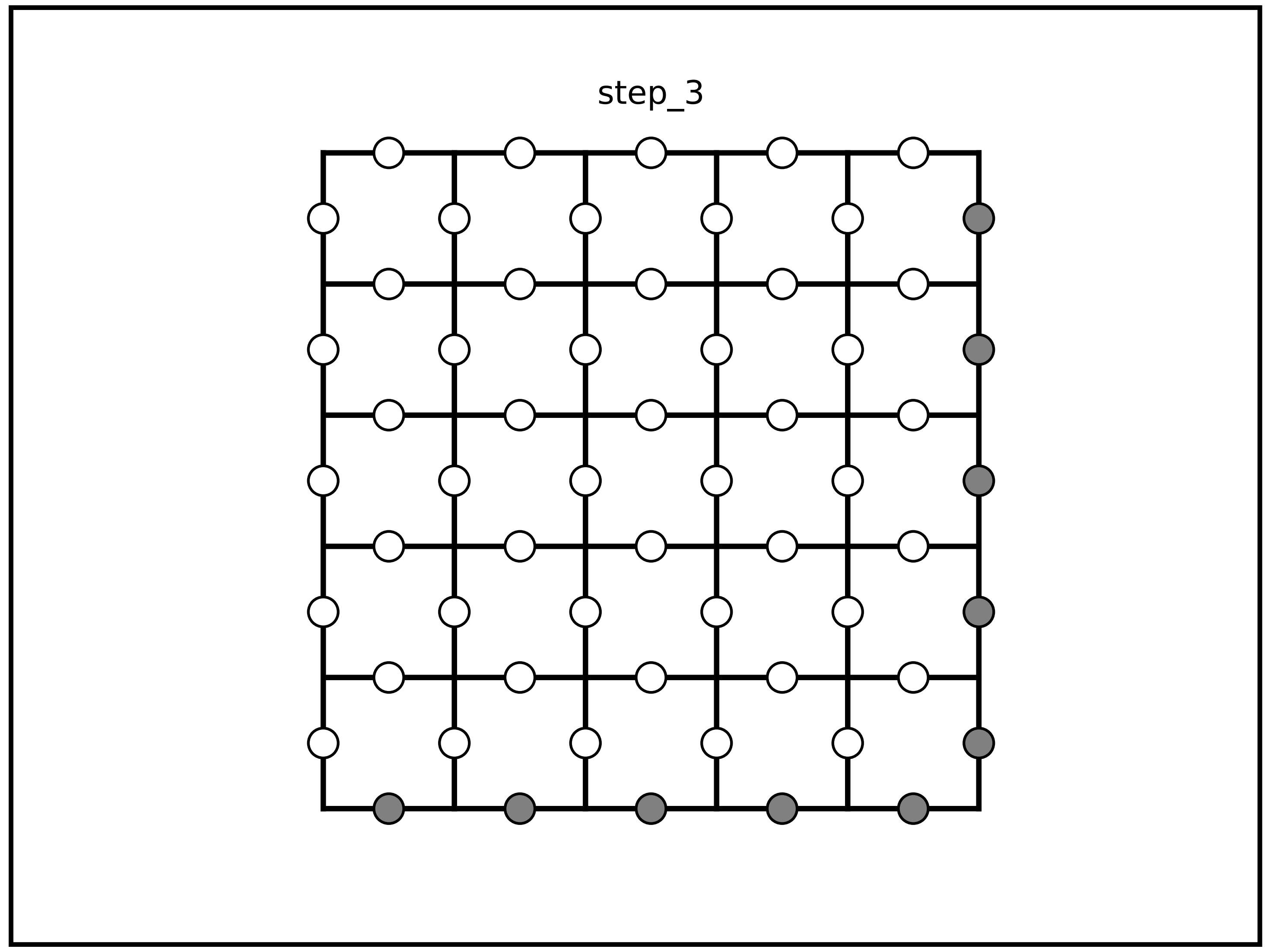
Example syndrome

MWPM



logical phase-flip

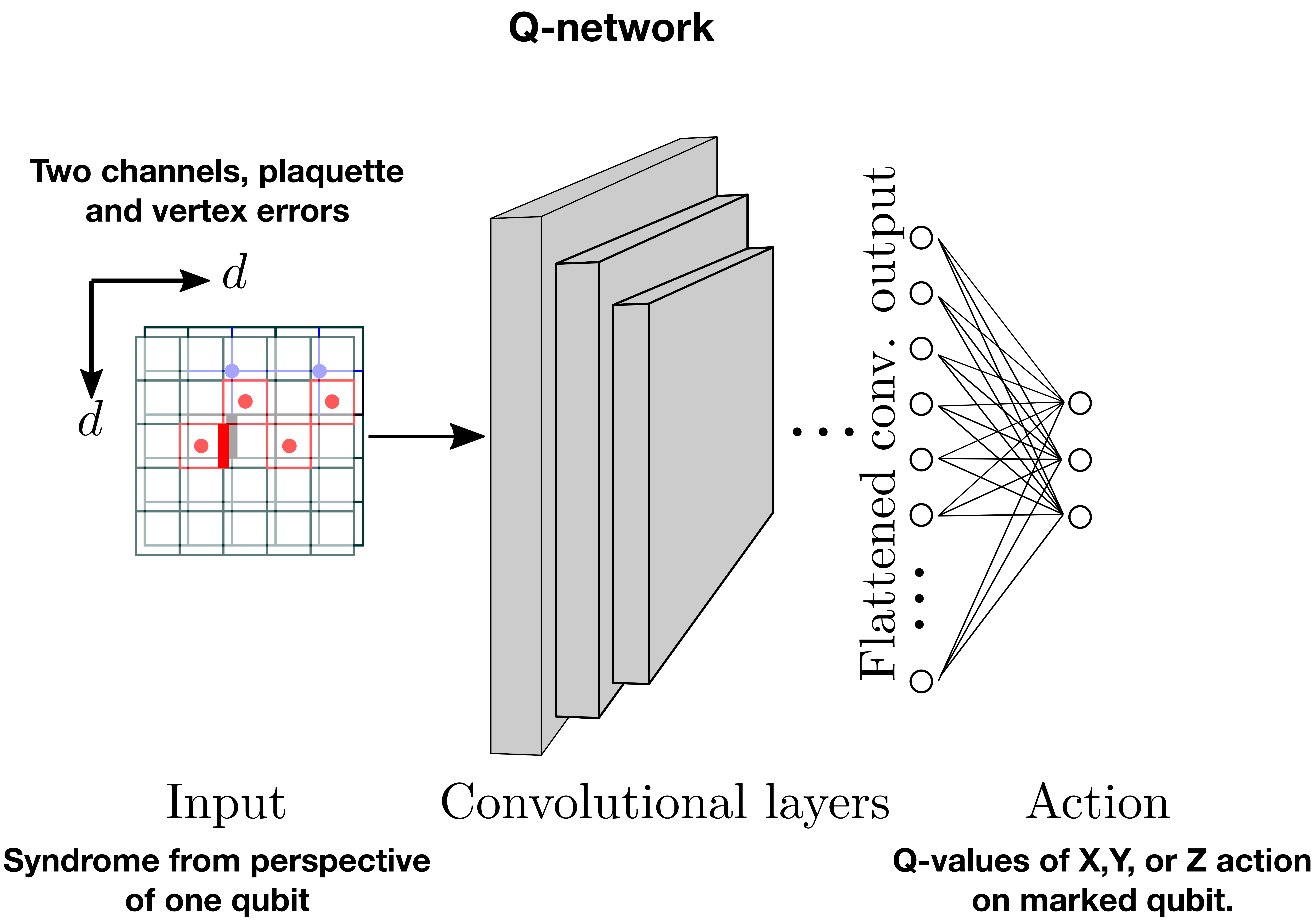
Reinforcement trained solver
reward=annihilation of complete syndrome + small intermediate reward



No logical operation

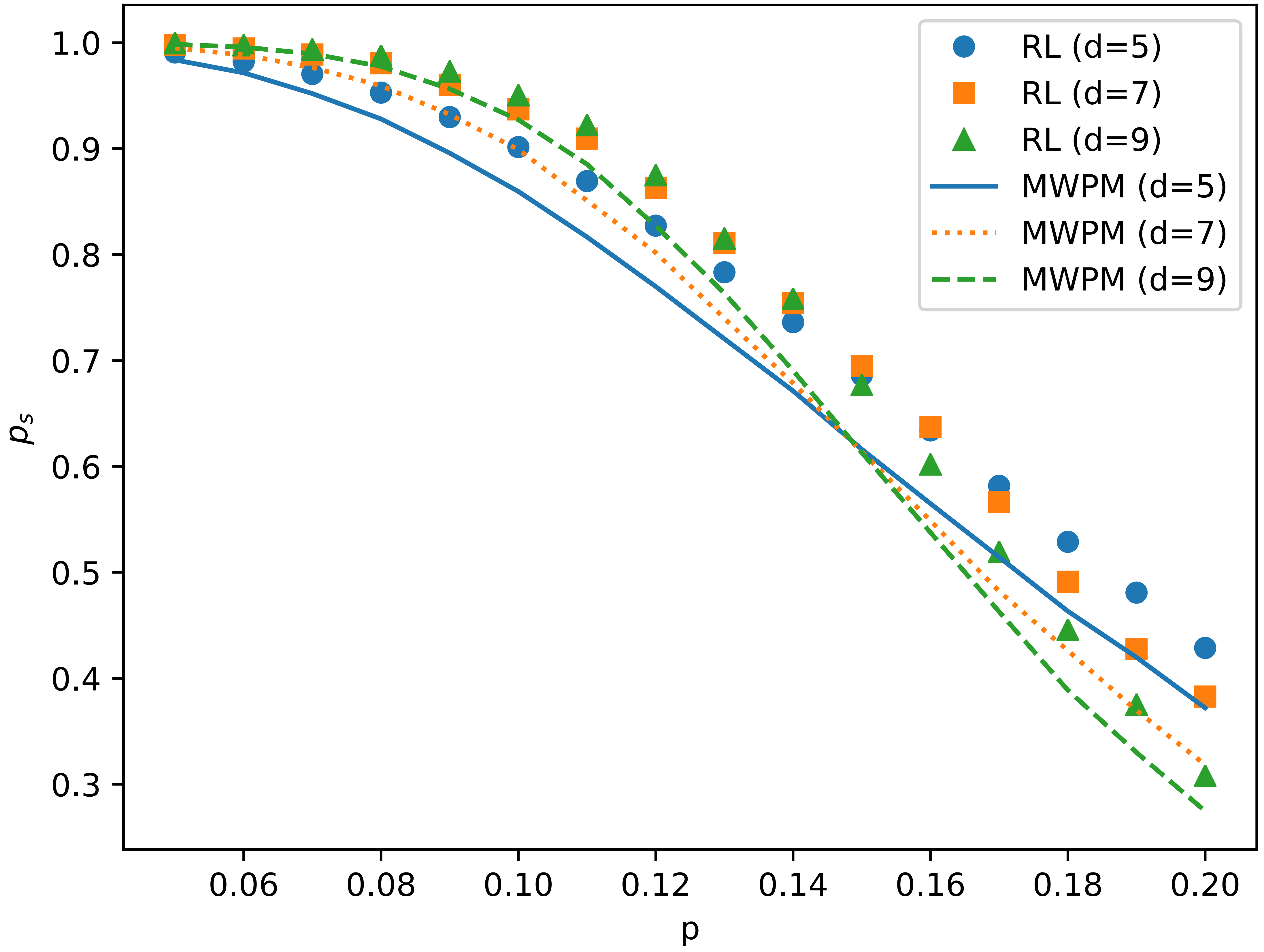
The agent can use Y to take advantage of correlations between bit-flip and phase-flip errors

Mattias Eliasson, David Fitzek, MG, in progress

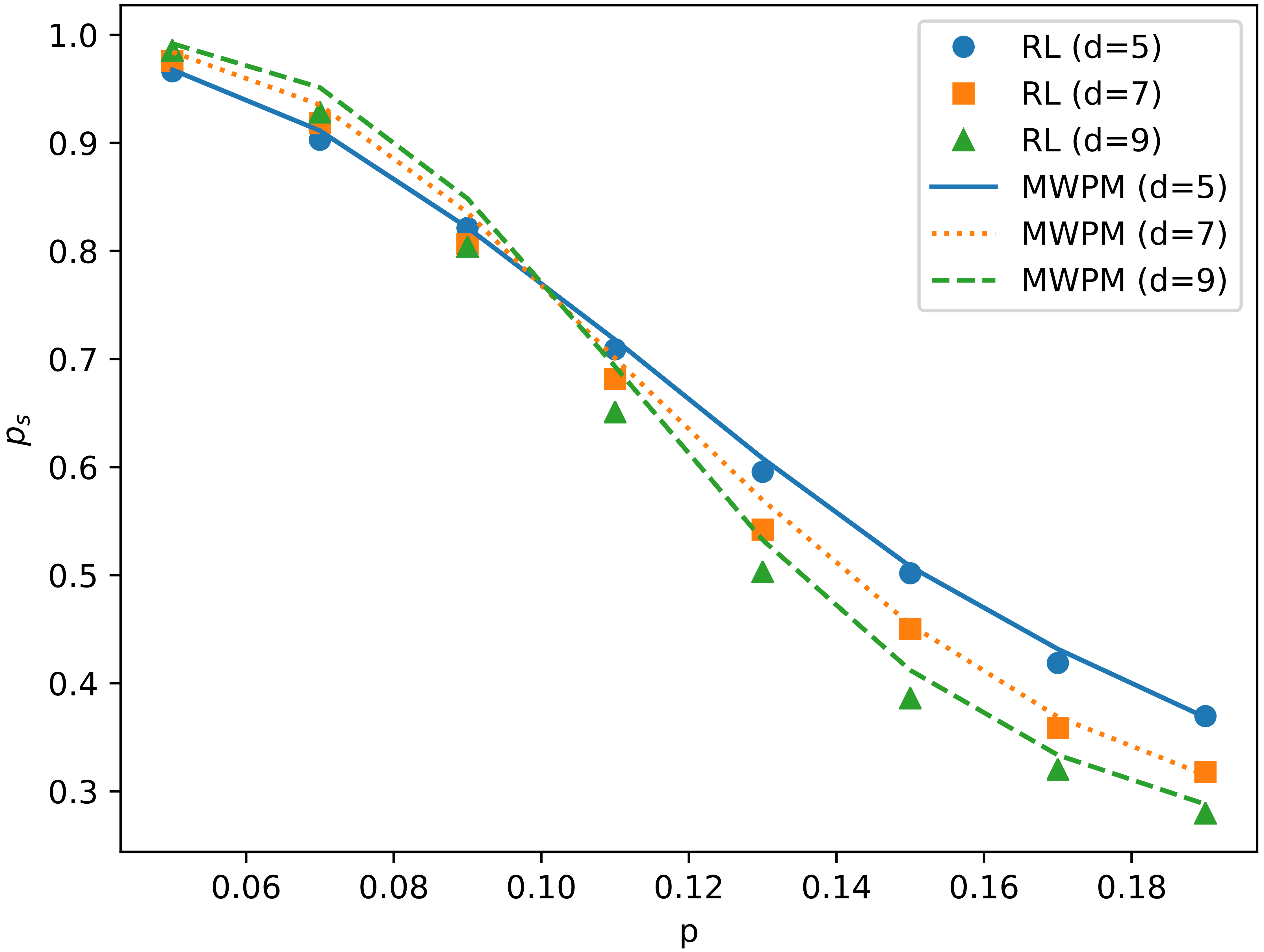

Q-network
Two channels, plaquette and vertex errors
d
d
Flattened conv. output
Input
Convolutional layers
Action
Syndrome from perspective of one qubit
Q-values of X,Y, or Z action on marked qubit.

Preliminary performance of RL solver trained on depolarizing noise

Depolarizing noise



Bit flip noise



Outperforms MWPM

	theoretical	experimental
d = 5	1.51e-3	1.45e-3
d = 7	2.12e-5	2.07e-5
d = 9	2.50e-7	4.30e-7

d=9 not fully converged!

Deep Q-networks

distance 5 code

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 128, 5, 5]	2,432
Conv2d-2	[-1, 128, 5, 5]	147,584
Conv2d-3	[-1, 120, 5, 5]	138,360
Conv2d-4	[-1, 111, 5, 5]	119,991
Conv2d-5	[-1, 104, 5, 5]	104,000
Conv2d-6	[-1, 103, 5, 5]	96,511
Conv2d-7	[-1, 90, 5, 5]	83,520
Conv2d-8	[-1, 80, 5, 5]	64,880
Conv2d-9	[-1, 73, 5, 5]	52,633
Conv2d-10	[-1, 71, 5, 5]	46,718
Conv2d-11	[-1, 64, 3, 3]	40,960
Linear-12	[-1, 3]	1,731

Total params: 899,320

trained on desktop GPU for 5 hours (using PyTorch)

distance 7 code

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 200, 7, 7]	3,800
Conv2d-2	[-1, 190, 7, 7]	342,190
Conv2d-3	[-1, 189, 7, 7]	323,379
Conv2d-4	[-1, 160, 7, 7]	272,320
Conv2d-5	[-1, 150, 7, 7]	216,150
Conv2d-6	[-1, 132, 7, 7]	178,332
Conv2d-7	[-1, 128, 7, 7]	152,192
Conv2d-8	[-1, 120, 7, 7]	138,360
Conv2d-9	[-1, 111, 7, 7]	119,991
Conv2d-10	[-1, 104, 7, 7]	104,000
Conv2d-11	[-1, 103, 7, 7]	96,511
Conv2d-12	[-1, 90, 7, 7]	83,520
Conv2d-13	[-1, 80, 7, 7]	64,880
Conv2d-14	[-1, 73, 7, 7]	52,633
Conv2d-15	[-1, 71, 7, 7]	46,718
Conv2d-16	[-1, 64, 5, 5]	40,960
Linear-17	[-1, 3]	4,803

Total params: 2,240,739

trained on desktop GPU for 12 hours

Conclusions

Deep Q-learning works well for error correction on *toric* code.
Can match or even outperform MWPM (for moderate code distance)

But, does require quite deep Q-networks

Periodic boundaries important for our approach.

Future challenges:

- **Larger code distances**
- **Improve reward scheme, use actual success or failure of error correction**
- **Include syndrome measurement error.**
- **More realistic surface code with boundaries. (Tougher due to lack of translational invariance)**

Philip Andreasson, Joel Johansson, Simon Liljestrand, Mats Granath, arXiv:1811.12338, accepted to Quantum

Mattias Eliasson, David Fitzek, MG, in progress