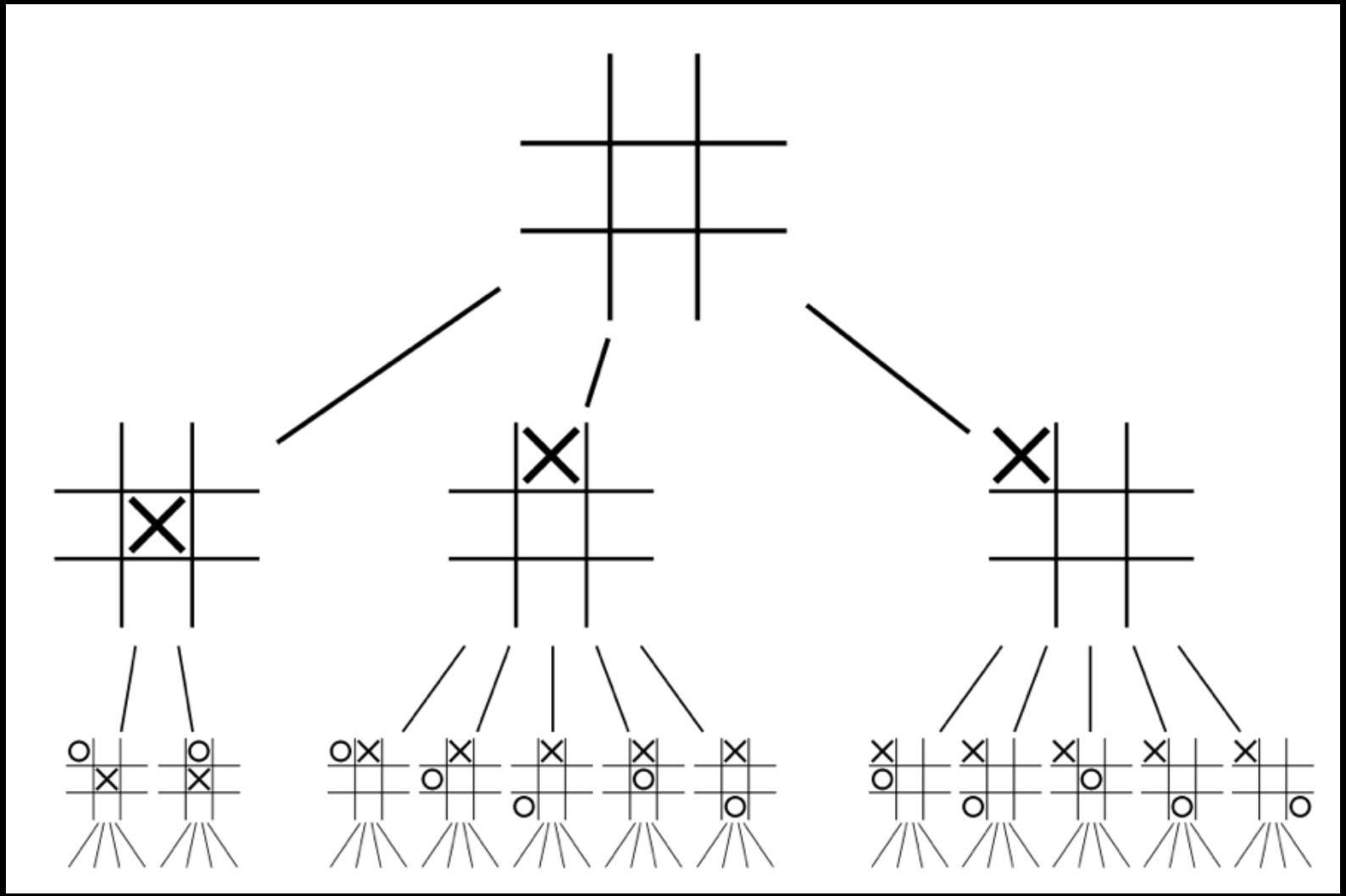
Playing Quantum Tic-Tac-Toe using Machine Learning

Presented by Peter Smit

This presentation

- Quantum Tic-Tac-Toe
- Why Machine Learning
- Techniques used
- Results

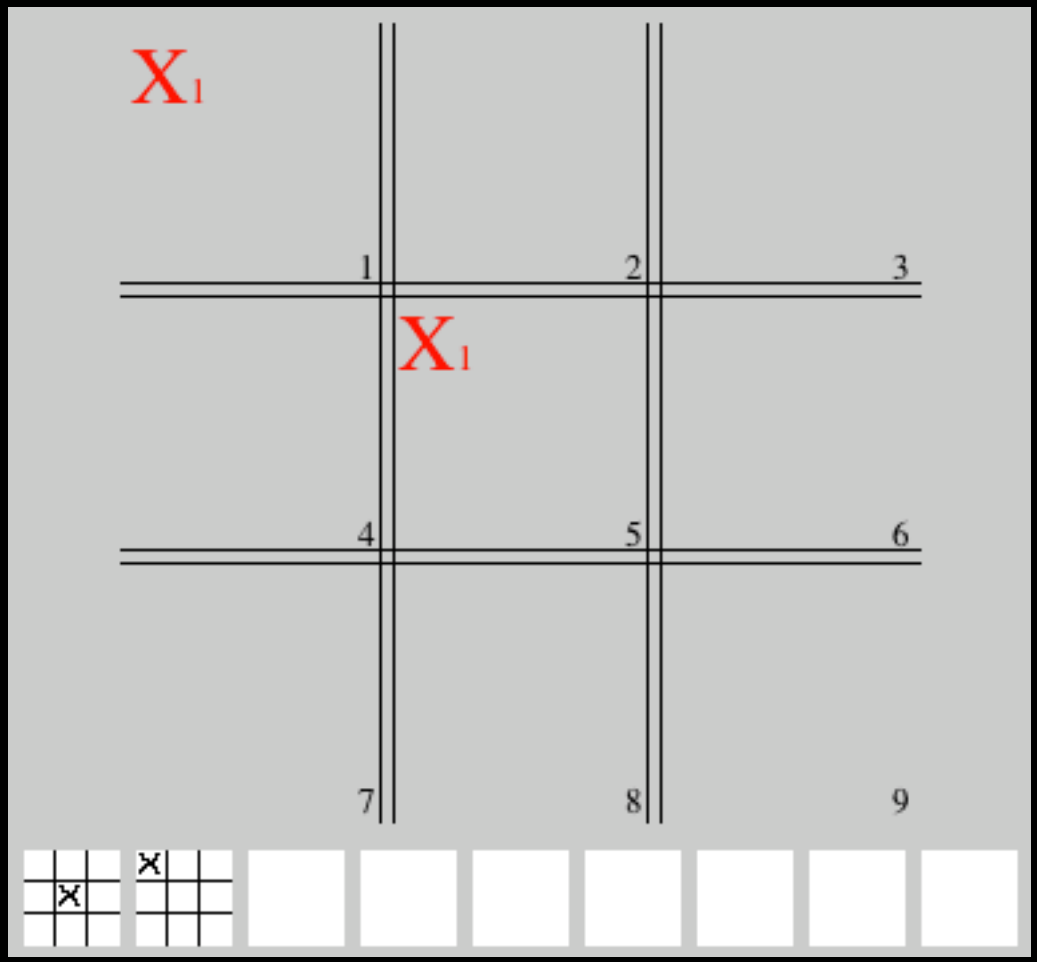
Tic-Tac-Toe

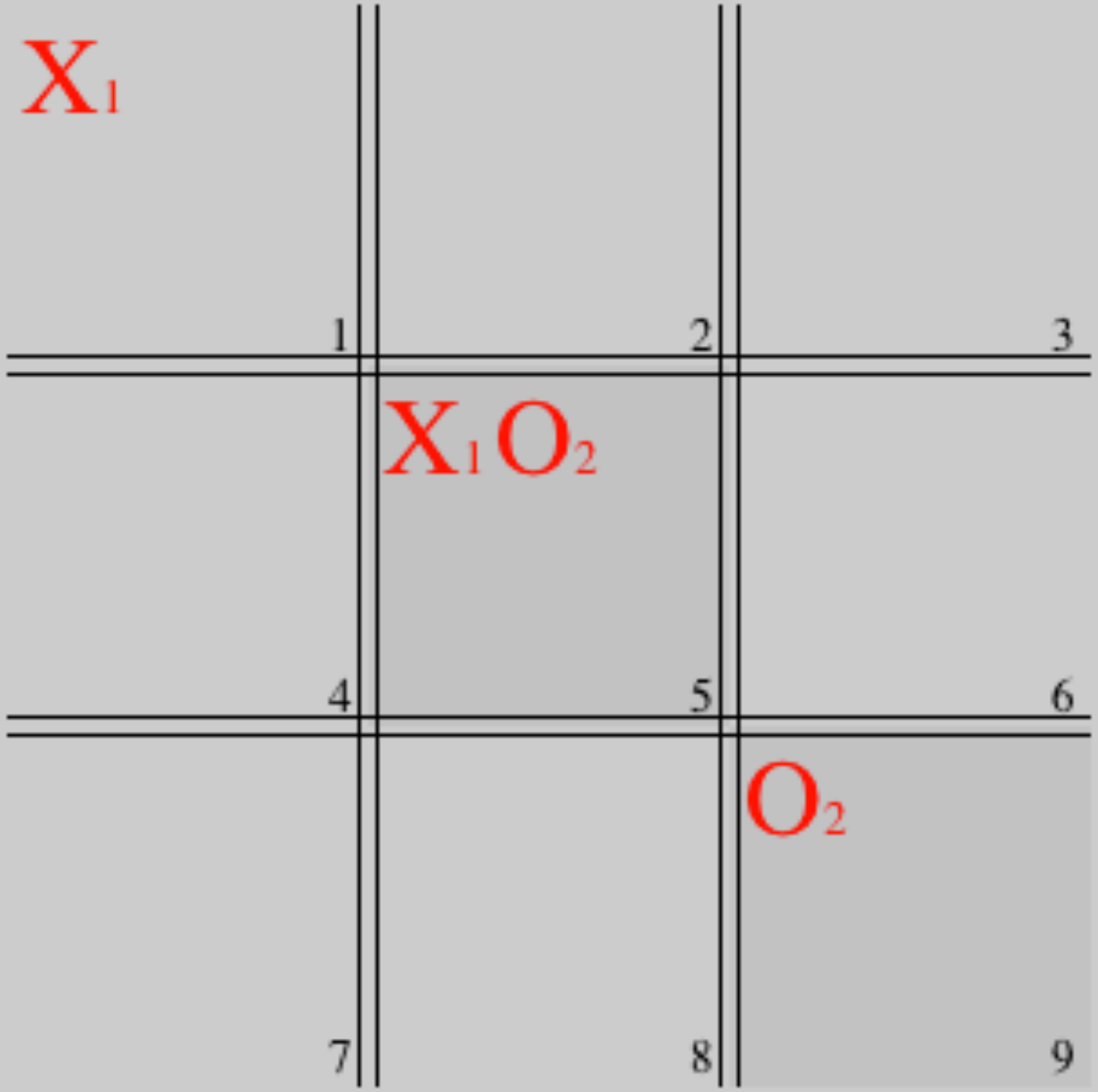


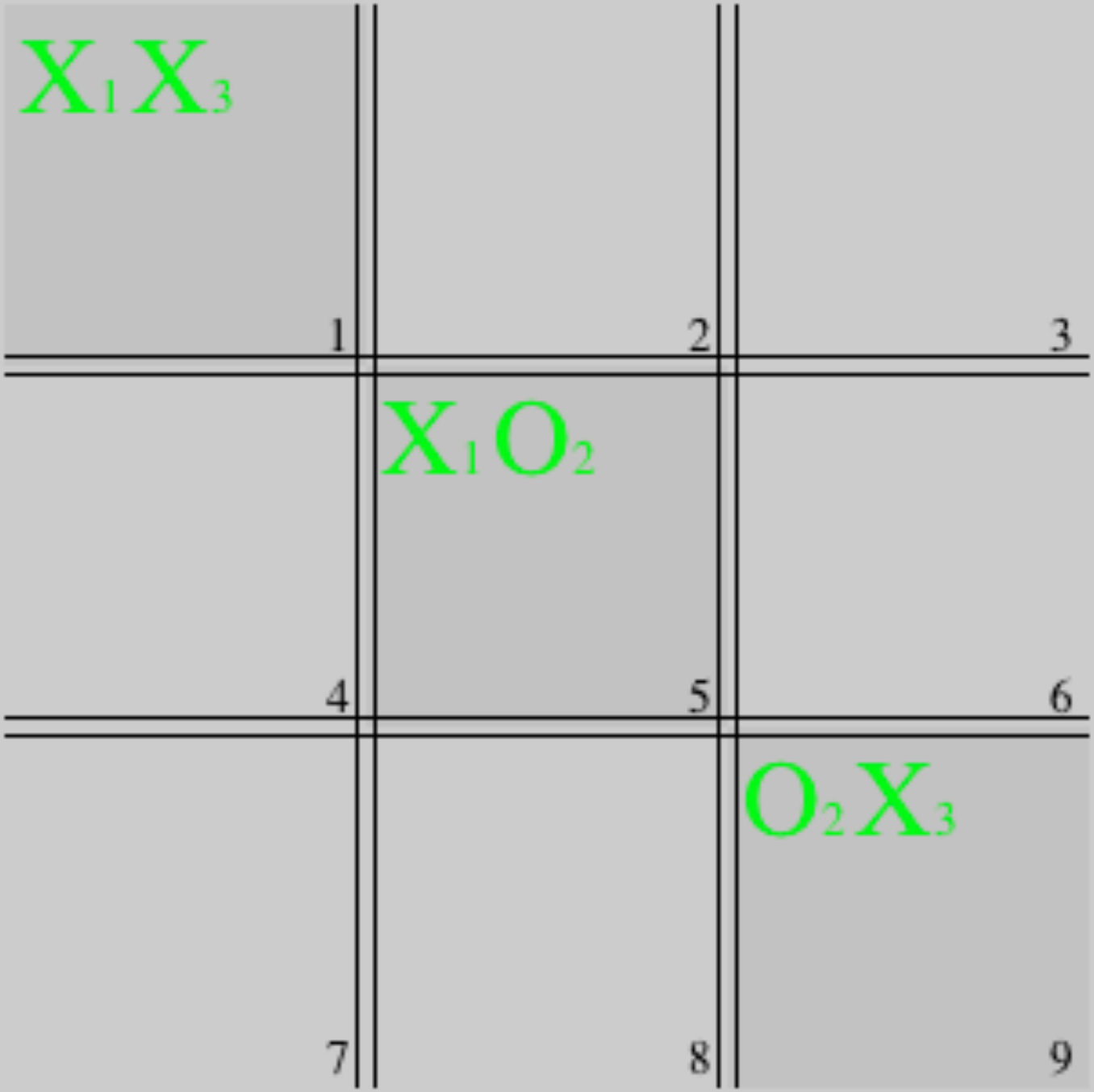
Quantum states

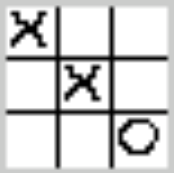
- Uncertain state of particle
- Debate in physics for many years

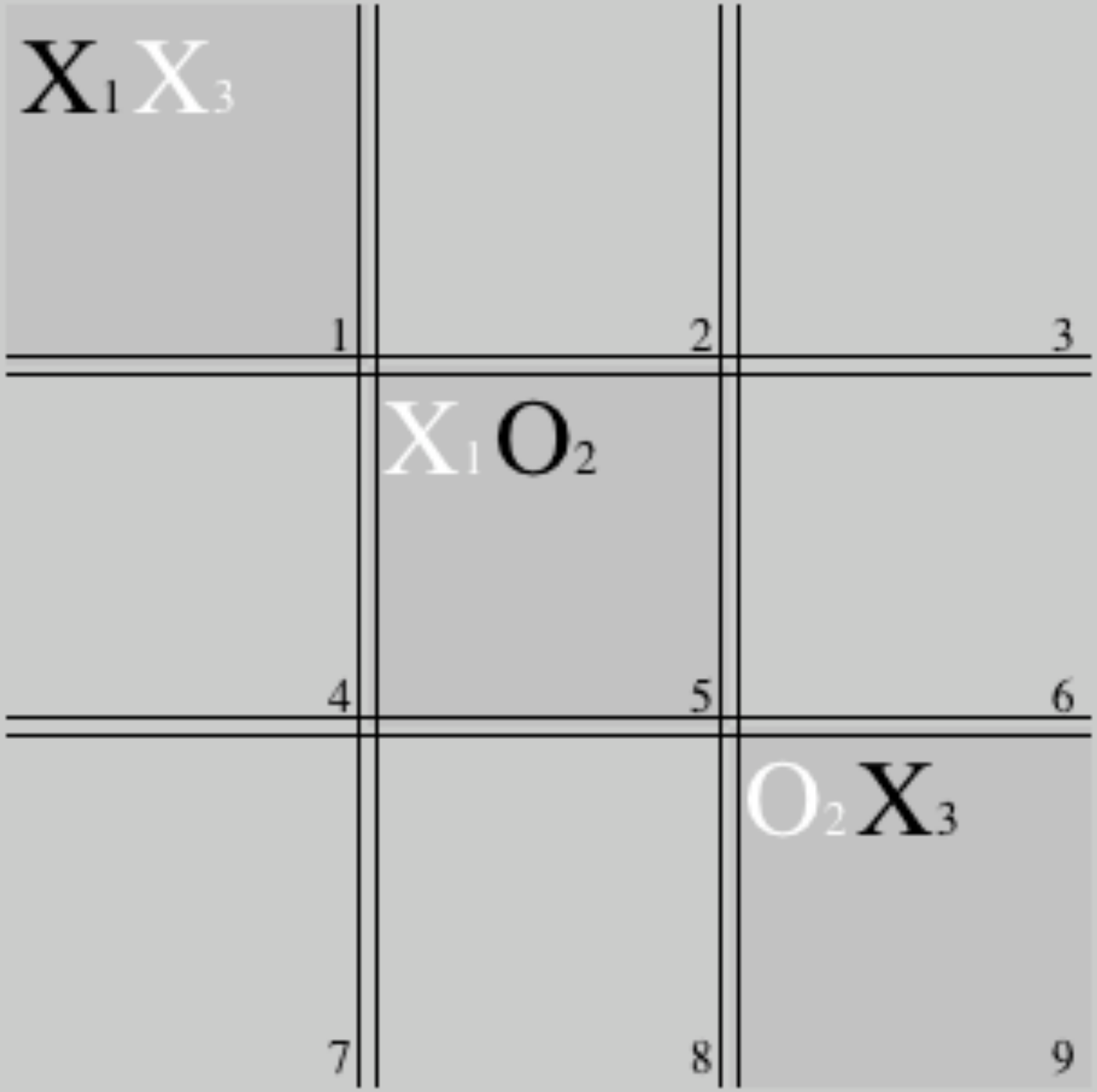
Quantum Tic-Tac-Toe

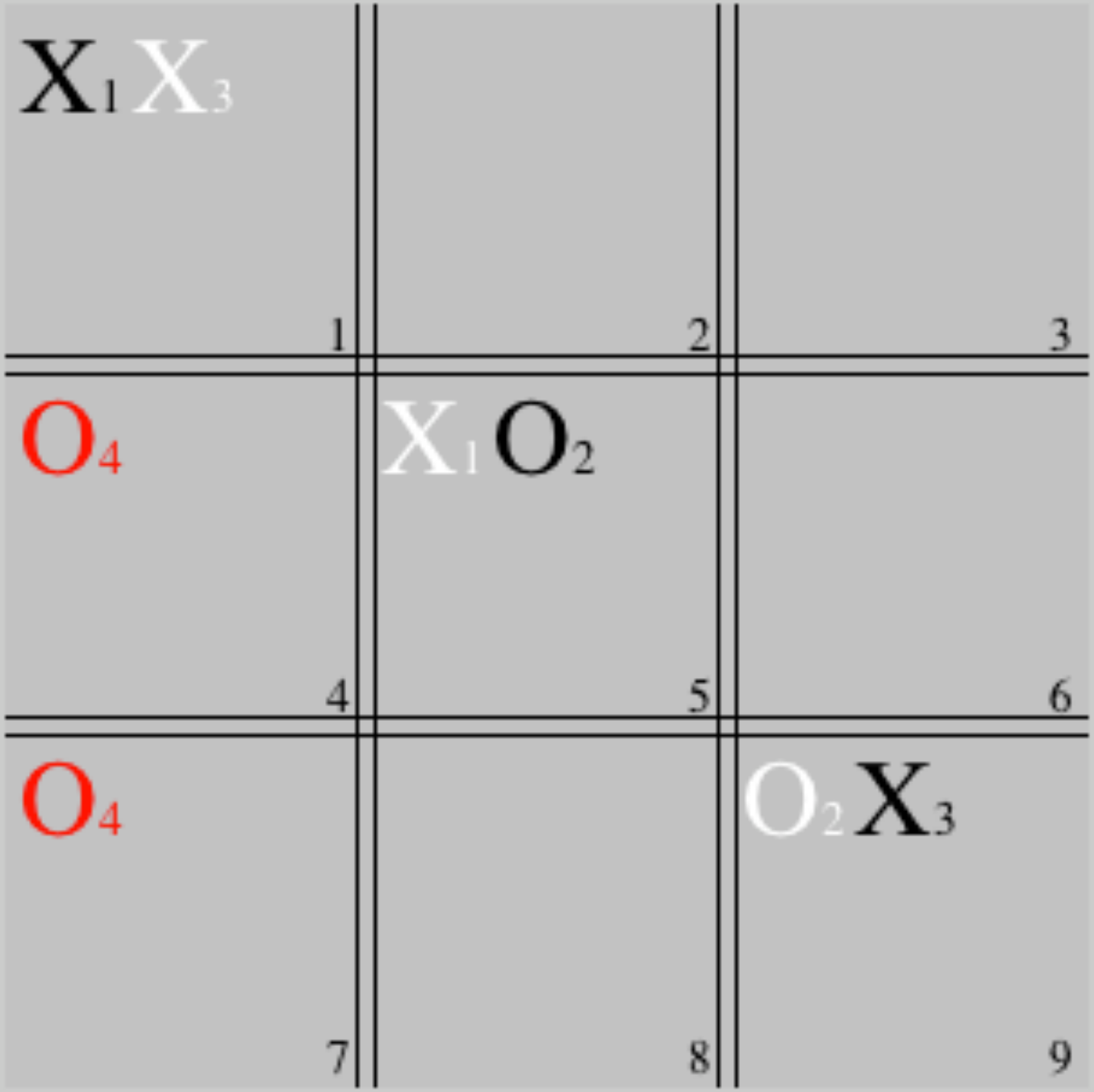


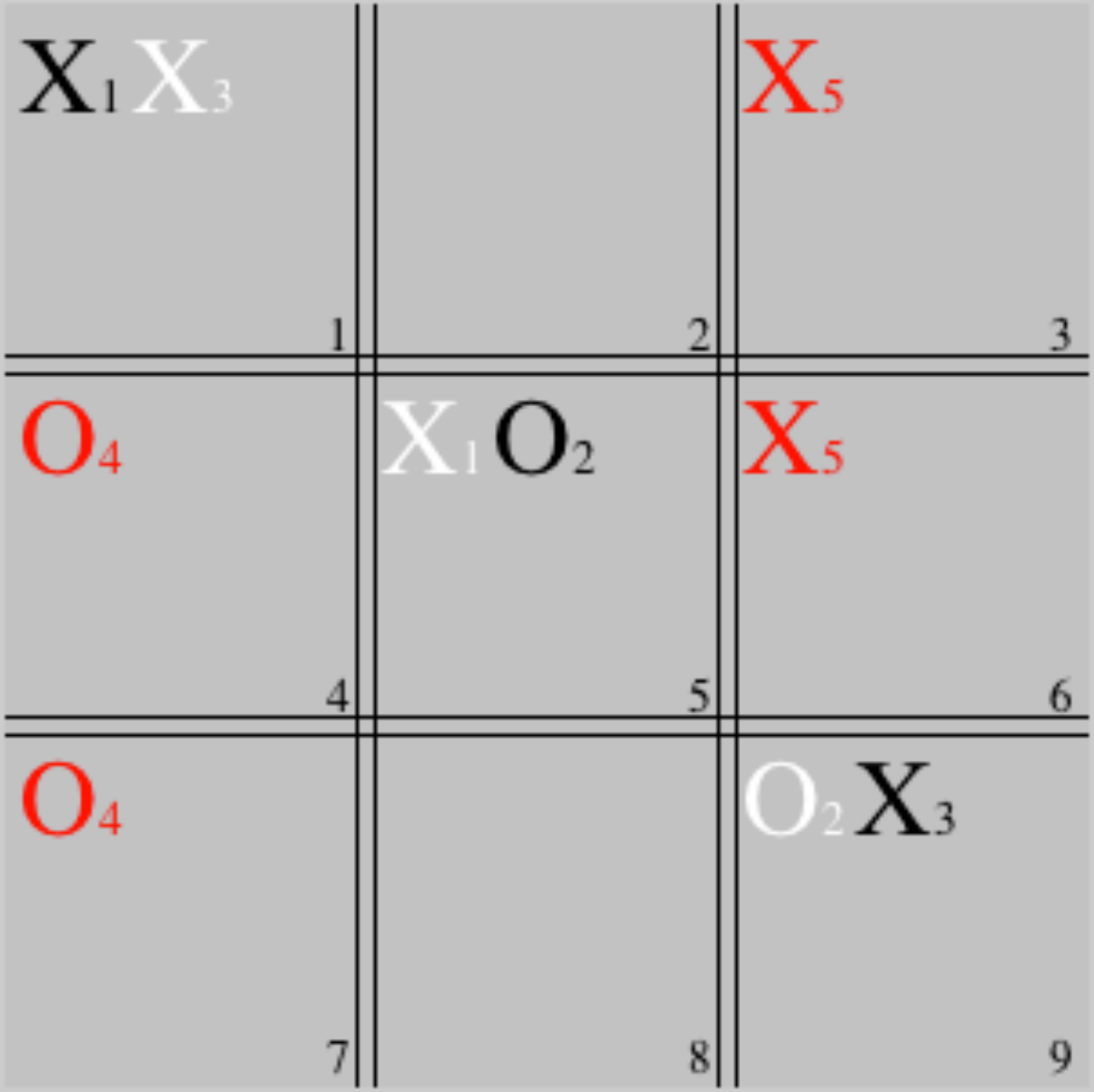

X1
1
2
3
X1O2
4
5
6
O2
7
8
9


X1X3
1
2
3
X1O2
4
5
6
O2X3
7
8
9

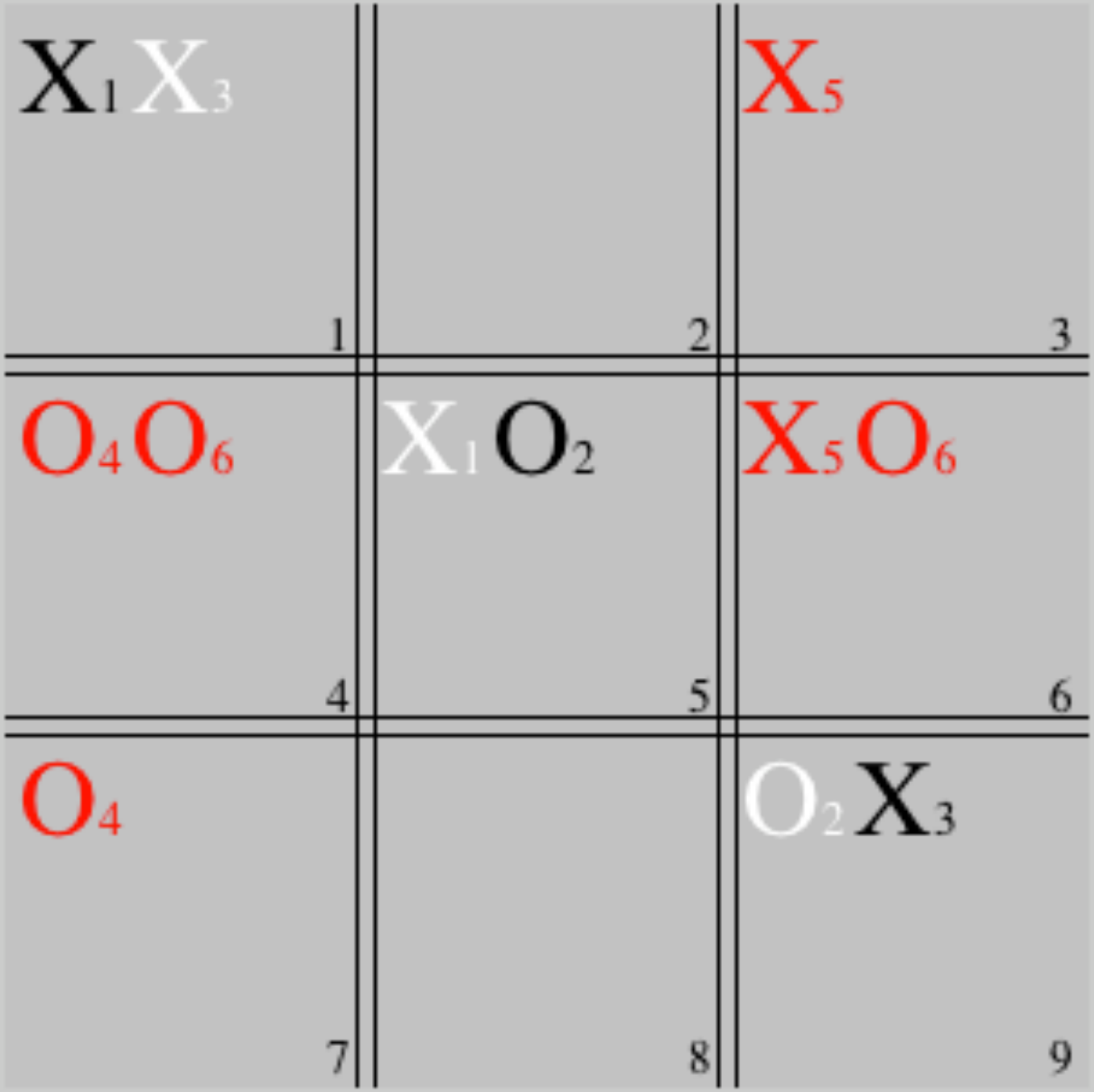



X1 X3
1
2
3
X1 O2
4
5
6
O2 X3
7
8
9

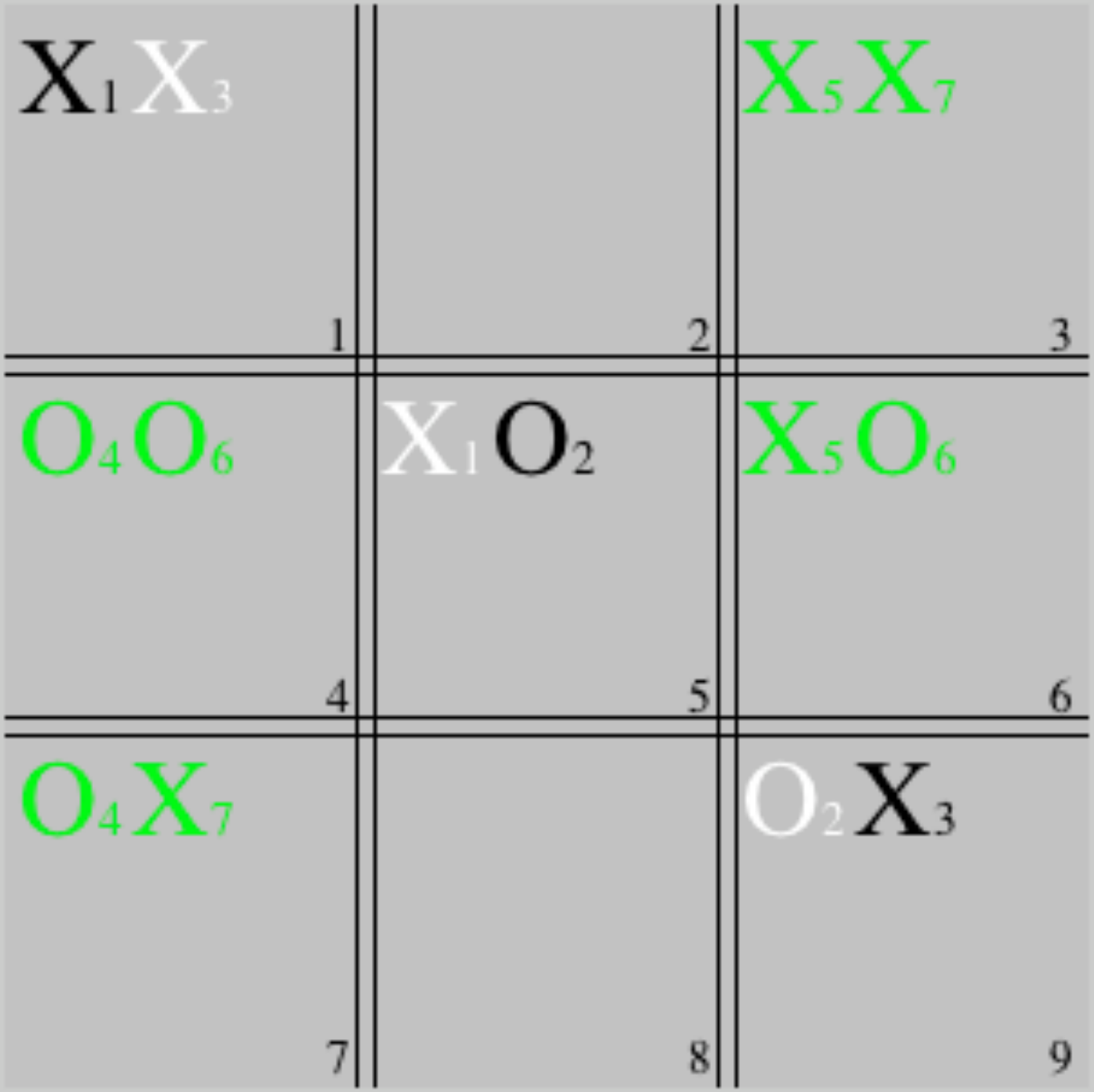

X1 X3
1
2
3
O4
X1 O2
4
5
6
O4
O2 X3
7
8
9

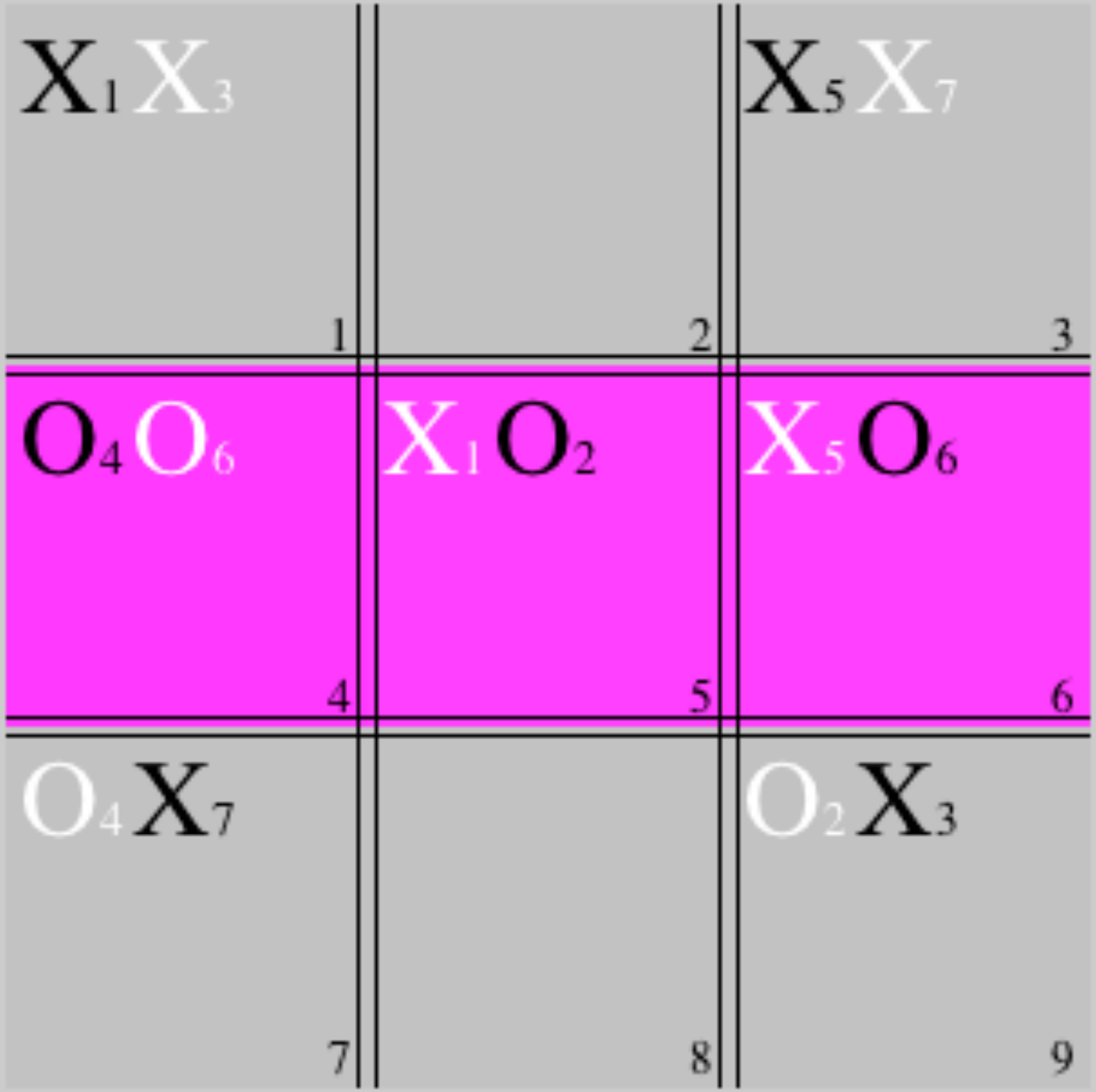

X1 X3
X5
1
2
3
O4
X1 O2
X5
4
5
6
O4
O2 X3
7
8
9





X1 X3
X5
1
2
3
O4 O6
X1 O2
X5 O6
4
5
6
O4
O2 X3
7
8
9





X1 X3
1
2
X5 X7
3
O4 O6
X1 O2
4
5
X5 O6
6
O4 X7
7
8
O2 X3
9


X1 X3
X5 X7
1
2
3
O4 O6
X1 O2
X5 O6
4
5
6
O4 X7
O2 X3
7
8
9

Why use ML?

- Explosion of the amount of states

Memory required

- Normal Tic-Tac-Toe:
- Roughly 9! = 3.6 * 10^5 States

Memory required

- Quantum Tic-Tac-Toe (3x3):
- Roughly (9!)^2 = 1.3 * 10^11 States

Memory required

- Quantum Tic-Tac-Toe (4x4):
- Roughly (16!)^2 = 4.4 * 10^26 States

Our solution

- Serialized NN 2x 400 kb

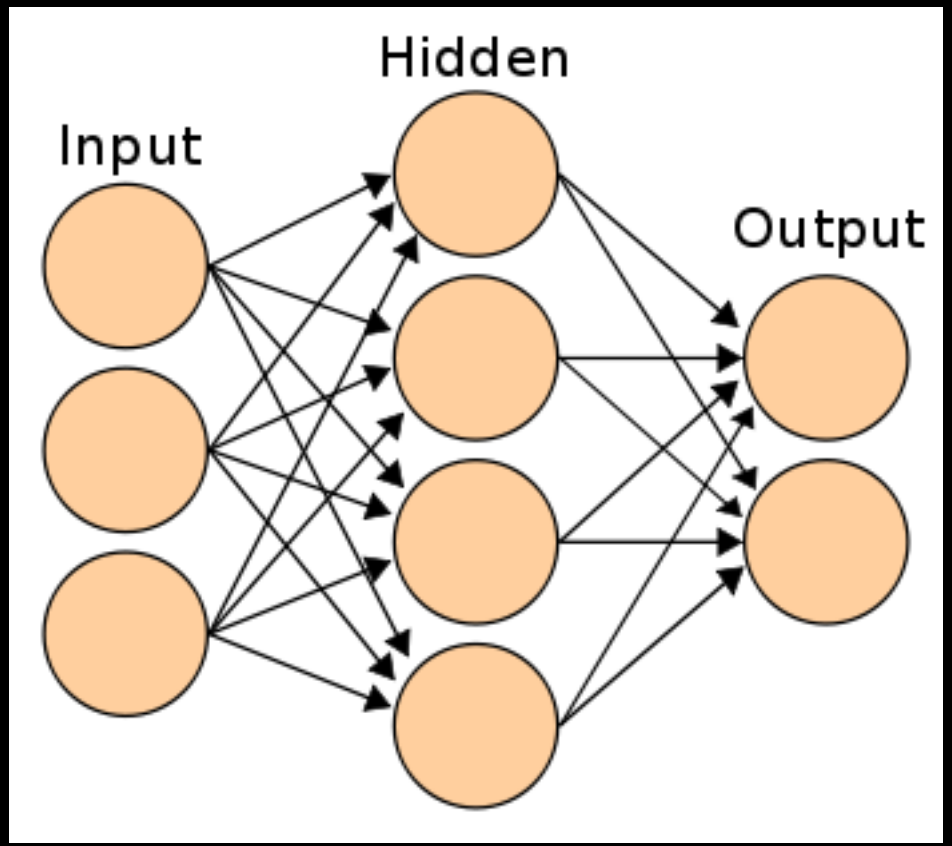
Show of hands

- Supervised Learning
- ...
- Unsupervised Learning

Temporal Difference

- Basically counting the wins and losses

Feed Forward Neural Network



Multi-Armed Bandit

So far for theory

- Board state valued by win/lose ratio
- Use NN to predict ratio
- Make random moves sometimes

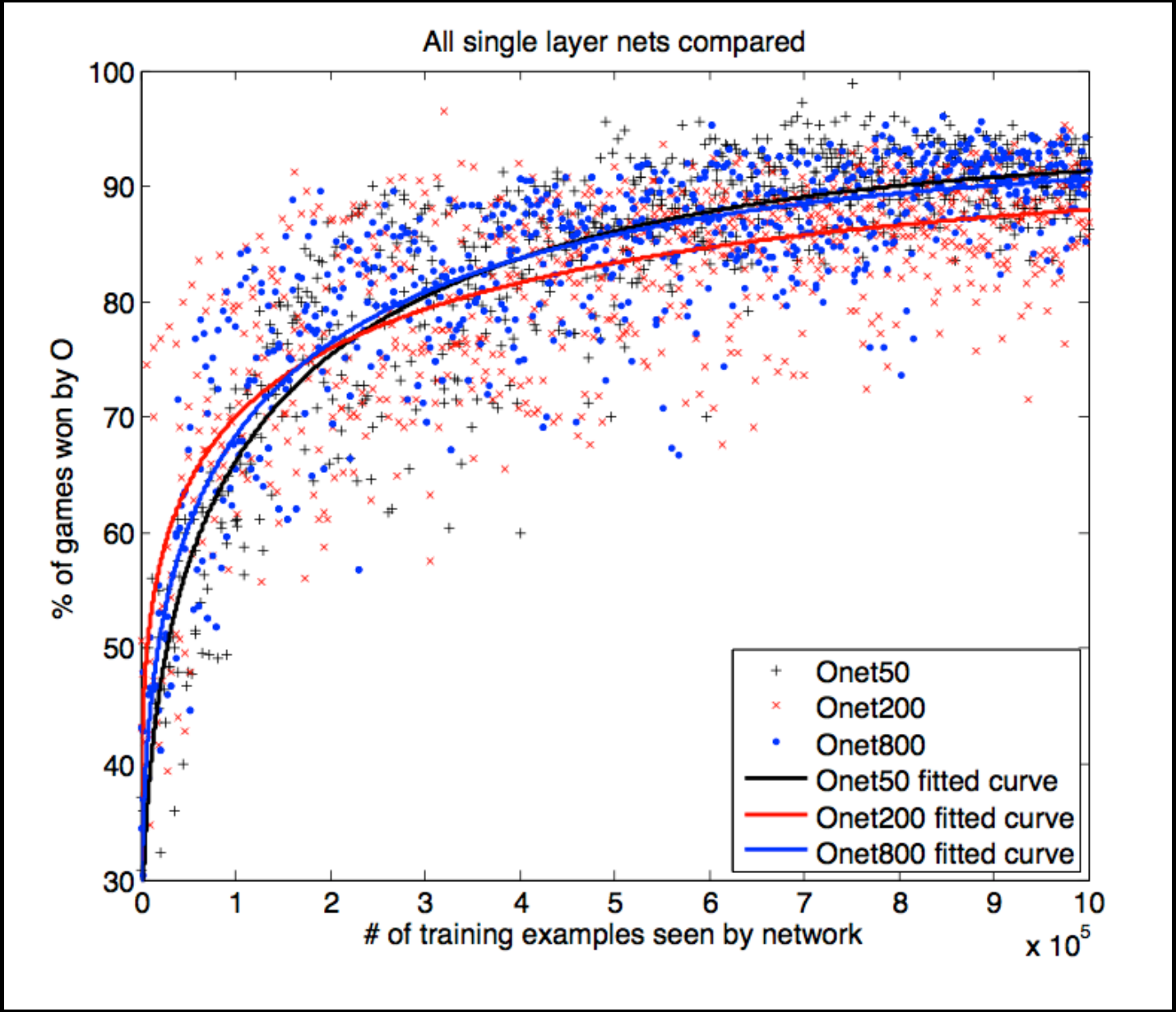
Experimental setup

Player PCs

Test PC

NN Train PC

Results


All single layer nets compared
% of games won by O
# of training examples seen by network
x 10^5
Onet50
Onet200
Onet800
Onet50 fitted curve
Onet200 fitted curve
Onet800 fitted curve

Possible Improvements

- Actually submitting to CodeCup
- Better opponents

Conclusion

- Machine learning is awesome