Intelligent Non-Player Character with Deep Learning

Meng Zhixiang, Zhang Haoze

Supervised by Prof. Michael Lyu

CUHK CSE FYP 2016 - 2017

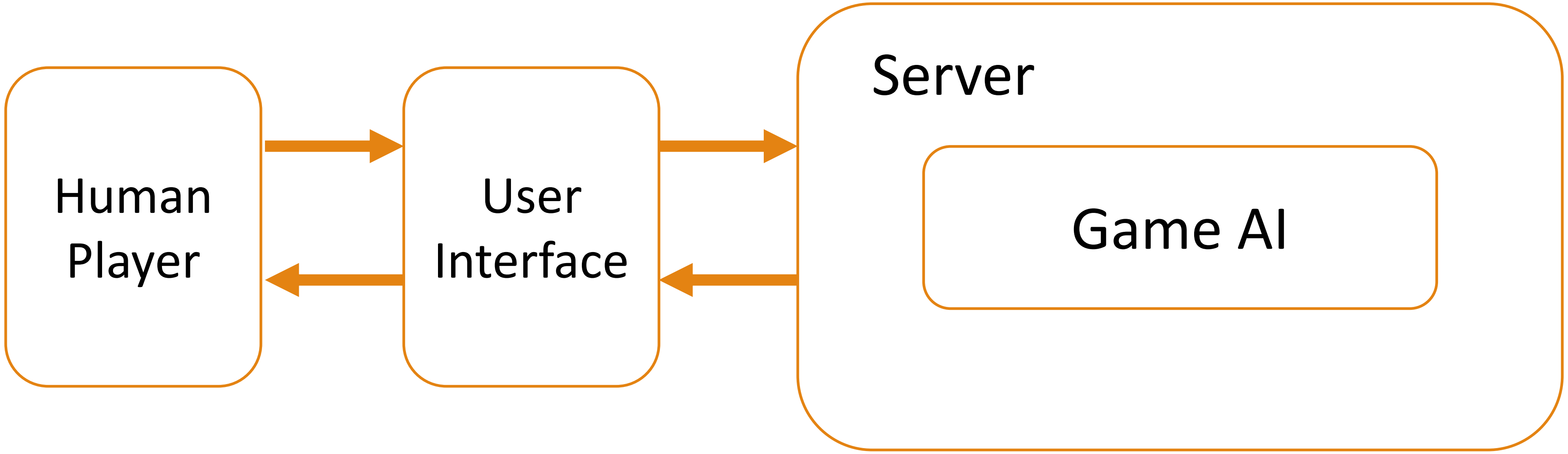
Agenda

- Review
- Objective
- Policy Network – Reinforcement Learning
- Evaluation Network – Supervised Learning
- Results
- Discussion & Conclusion

Previously on our project...

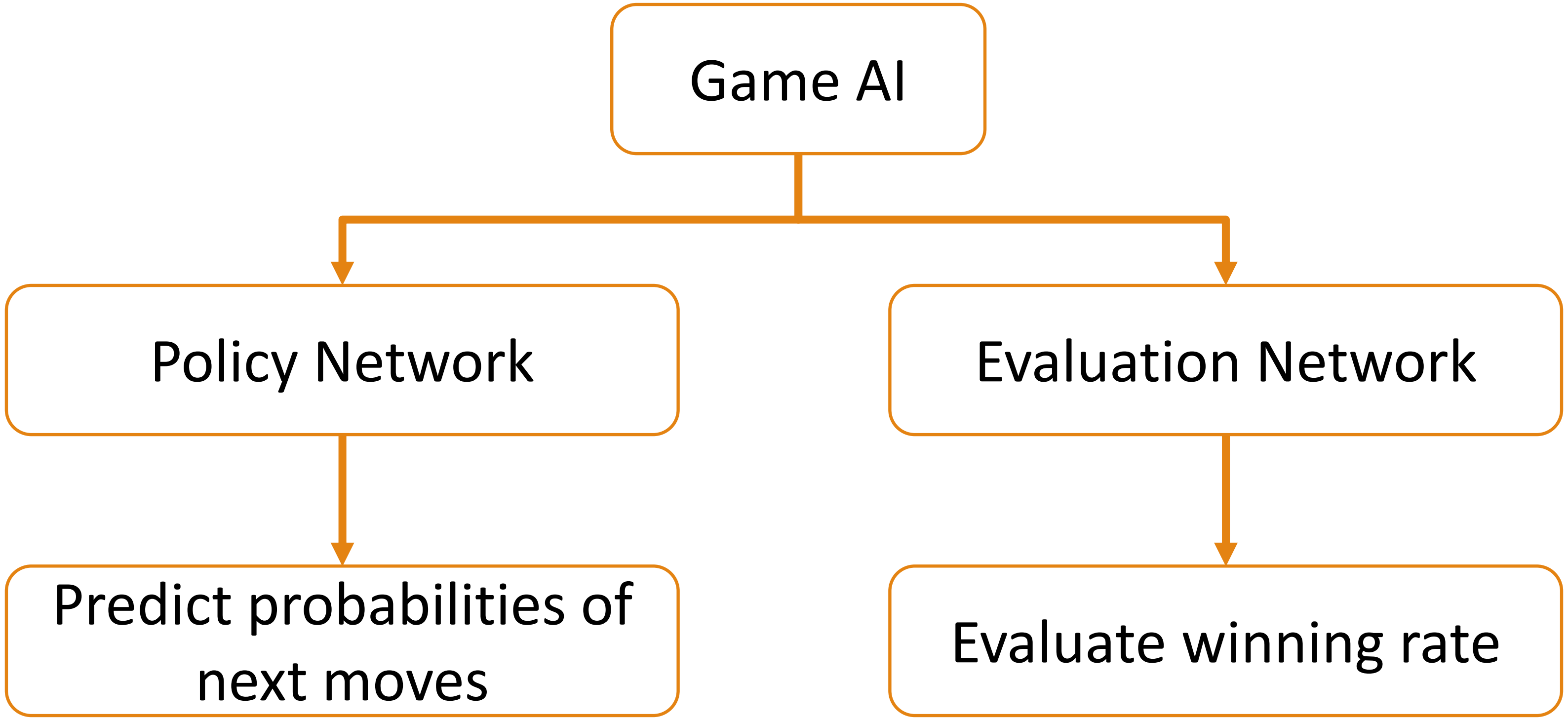
- NPC:

  character not controlled by a player but by computer through artificial intelligence



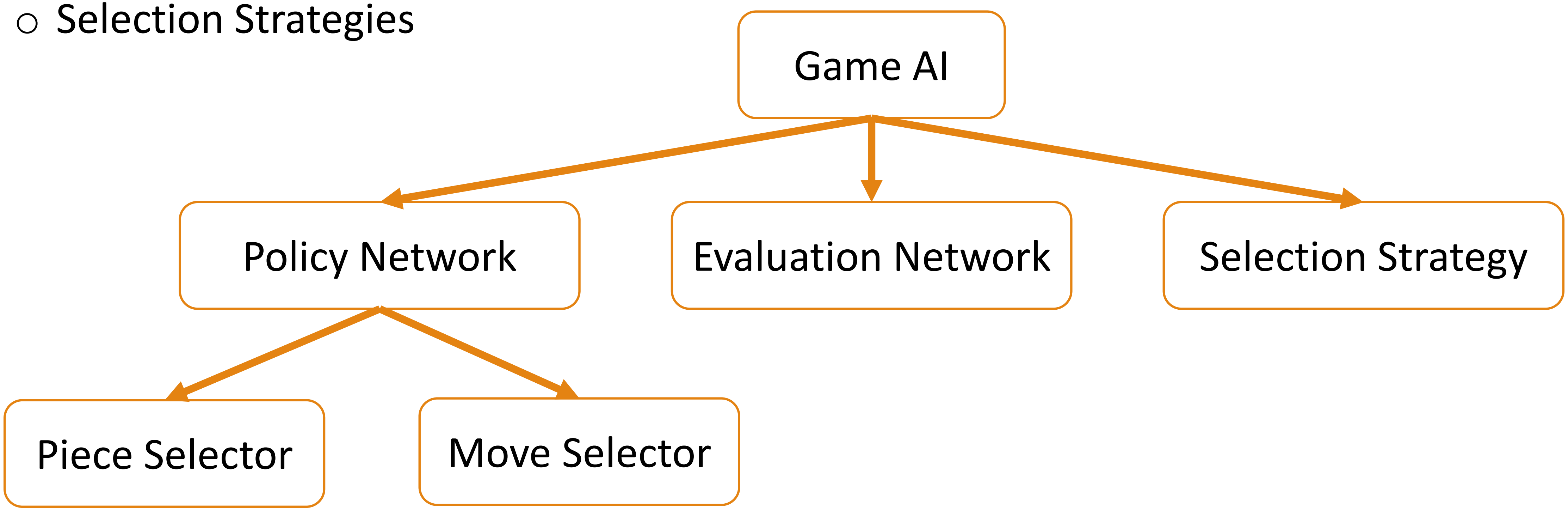
Previously on our project...

- Policy Network – Supervised Learning

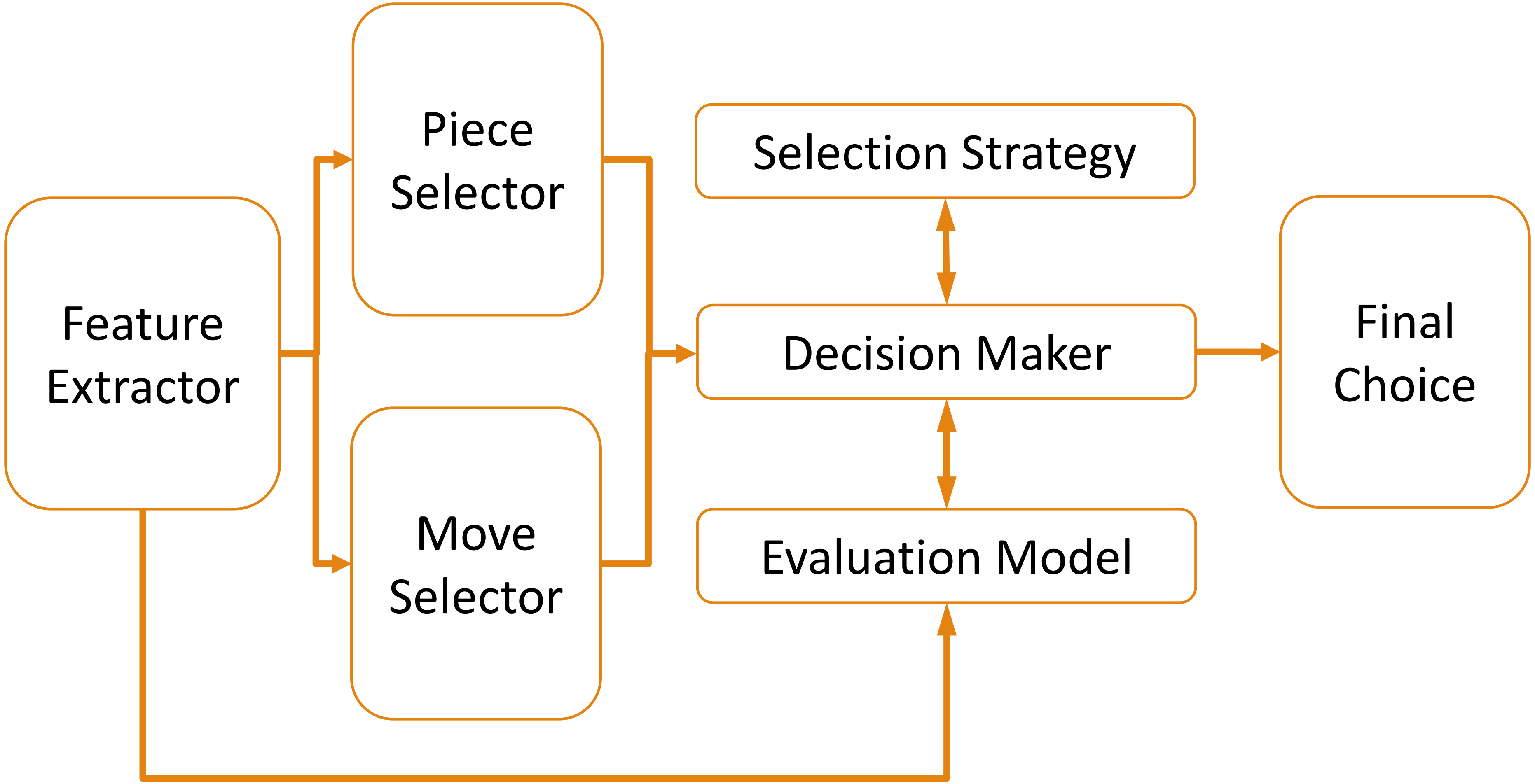


Objective

- Policy Network – Reinforcement Learning
- Evaluation Network – Supervised Learning
- Selection Strategies



General Structure



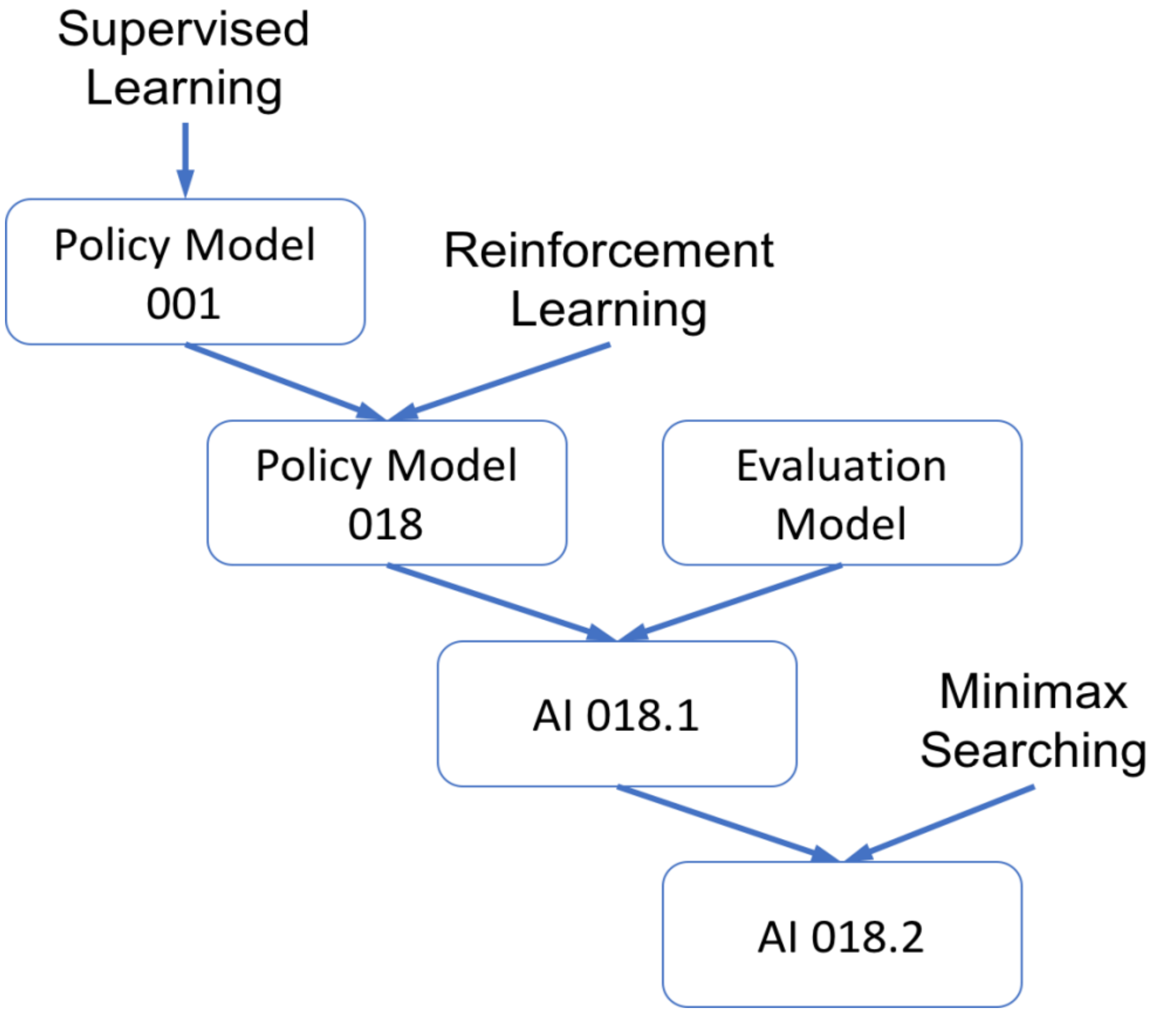
Version Iteration

Version 001: only trained by Supervised Learning

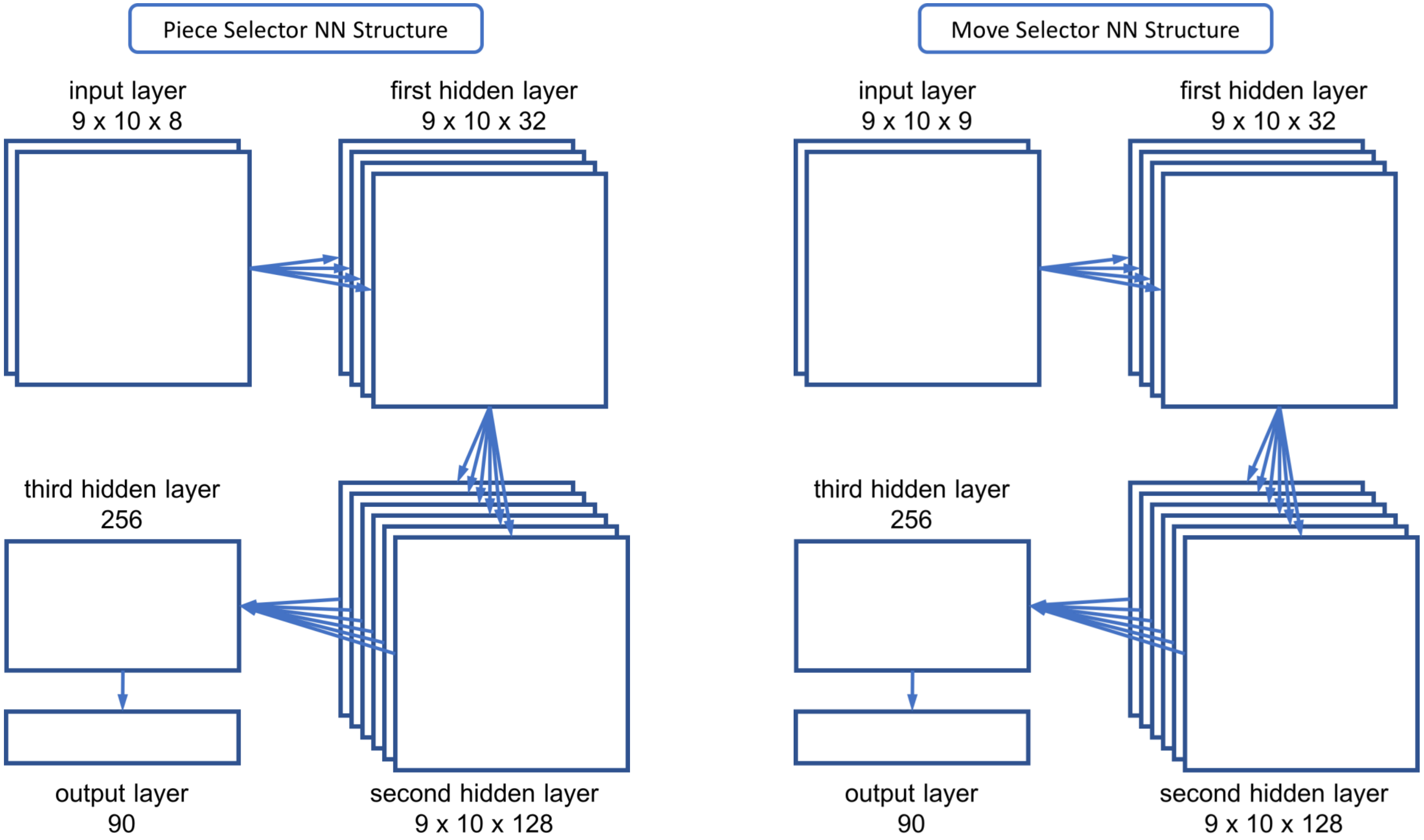
Version 018: Trained by Reinforcement Learning

Version 018.1: Add Evaluation Model

Version 018.2: Add Minimax Search



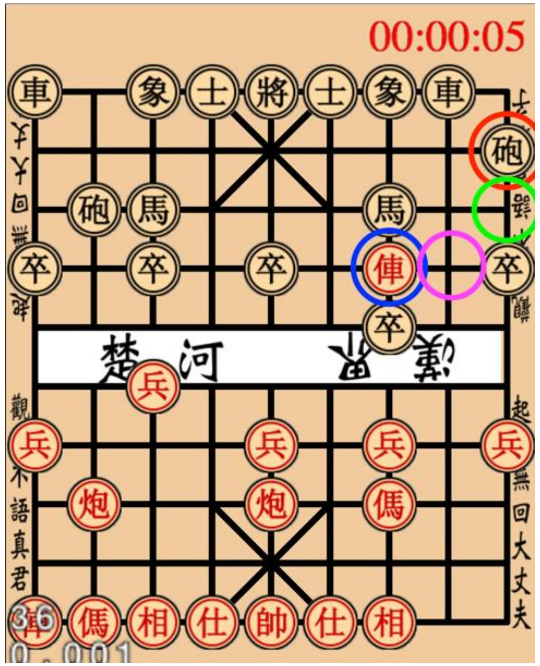
Piece Selector & Move Selector



Feature Channels

Feature Channel 1	Pieces belonging to different sides
Feature Channel 2	Pieces of Advisor type
Feature Channel 3	Pieces of Bishop type
Feature Channel 4	Pieces of Cannon type
Feature Channel 5	Pieces of King type
Feature Channel 6	Pieces of Knight type
Feature Channel 7	Pieces of Pawn type
Feature Channel 8	Pieces of Rock type
Feature Channel 9 (only for Move Selector)	Valid moves for the selected piece

Output Sample



Current chessboard
(black cannon move
from green to red)

0.1%	0.0%	0.3%	1.3%	0.0%	0.0%	0.0%	1.7%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.3%	0.0%	0.0%	0.0%	0.0%	0.4%	0.0%	95.7%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

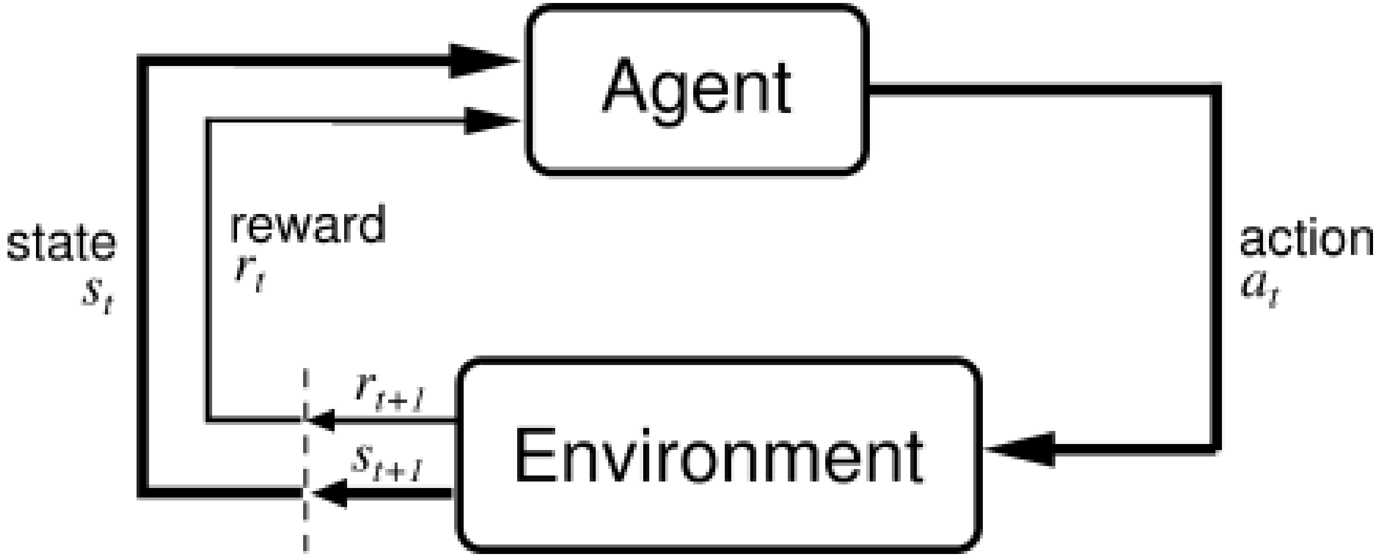
Output from piece selector
(green circle on chessboard)

0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.2%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	97.5%
0.0%	0.0%	0.0%	0.0%	0.0%	0.1%	0.0%	1.0%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Output from move selector
(red circle on chessboard)

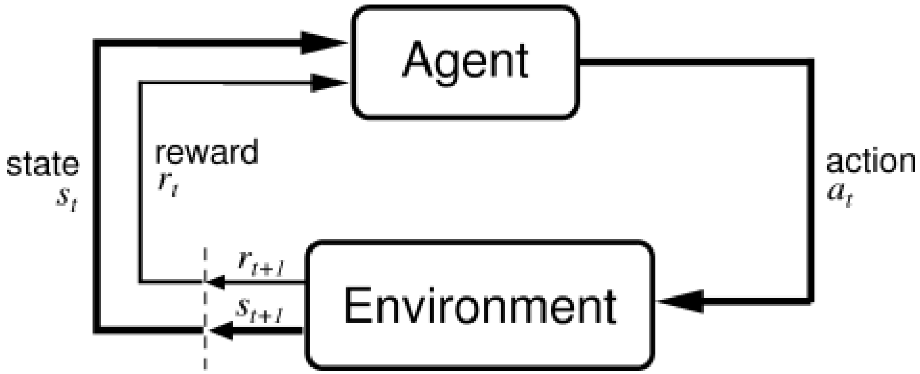
Reinforcement Learning

- inspired by behaviour psychology
- exploration vs exploitation
- how to take actions
- to maximize the reward



Reinforcement Learning

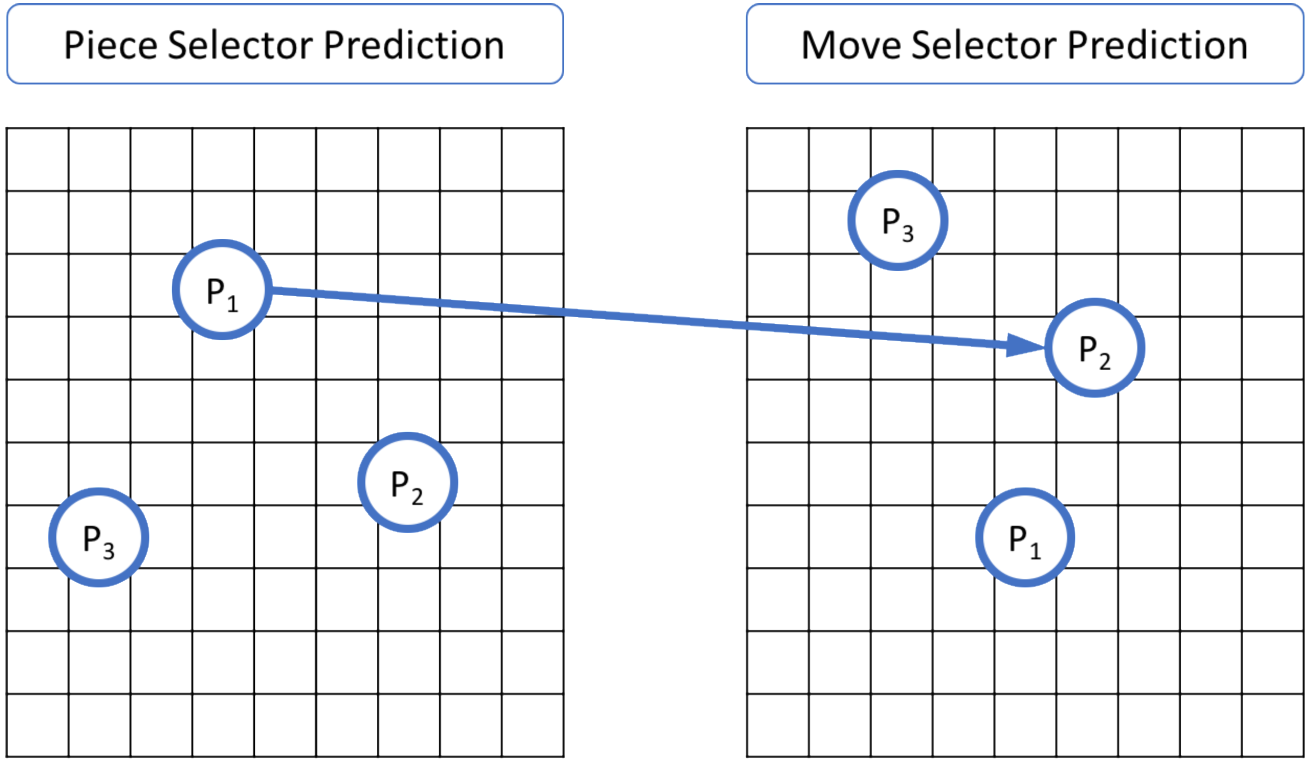
- Assigning Rewards
  - Positive Reward: 1 for moves from winning side
  - Negative Reward: not used



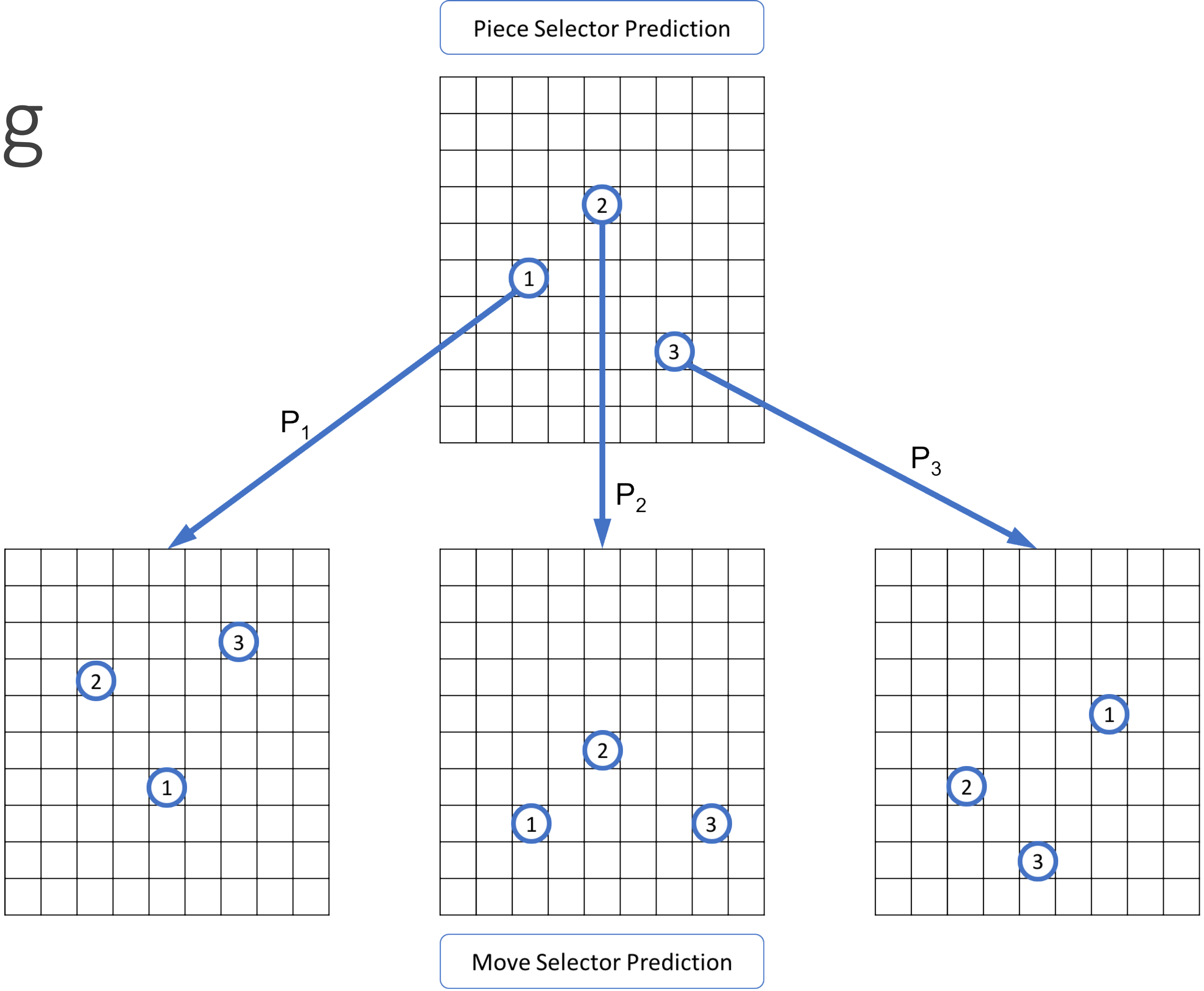
- Compete with different middle version models
  - to avoid overfitting

Training

- change the opposite model roughly every 4,000 games
- in total around 40,000 games, over 2,000,000 moves

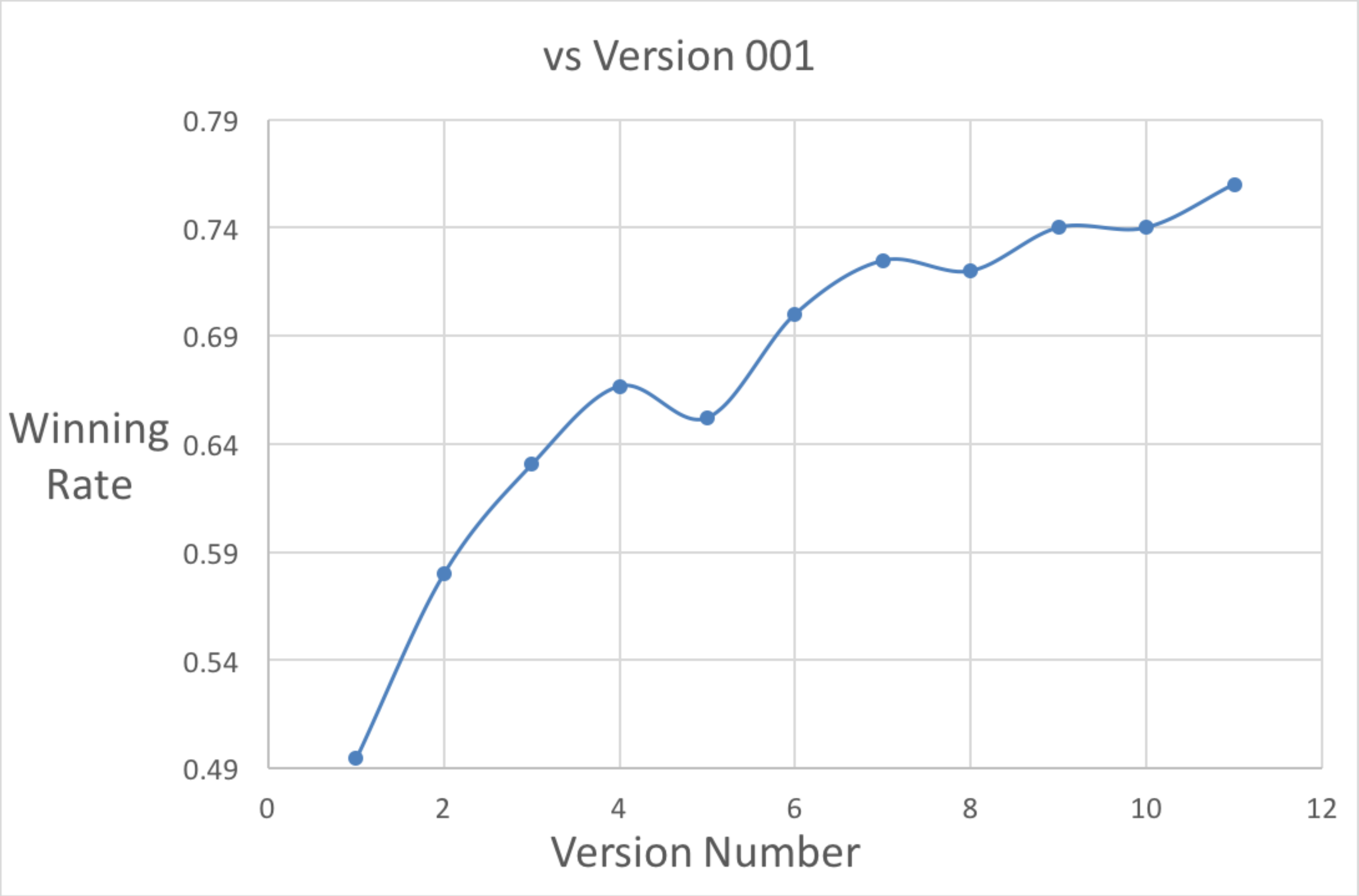


Testing



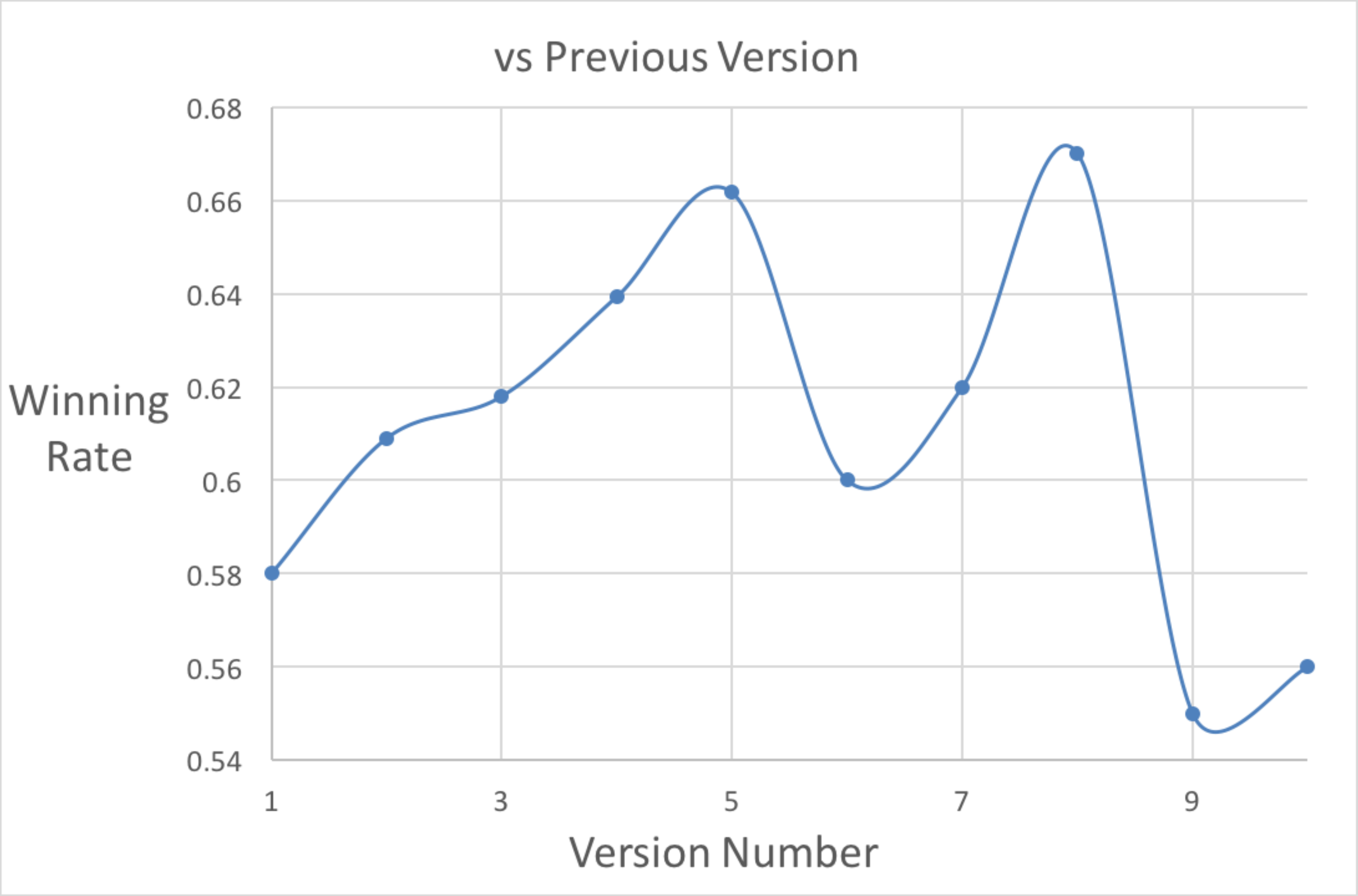
Piece Selector Prediction
2
1
3
P1
P2
P3
3
2
1
2
1
3
1
2
3
Move Selector Prediction

Results



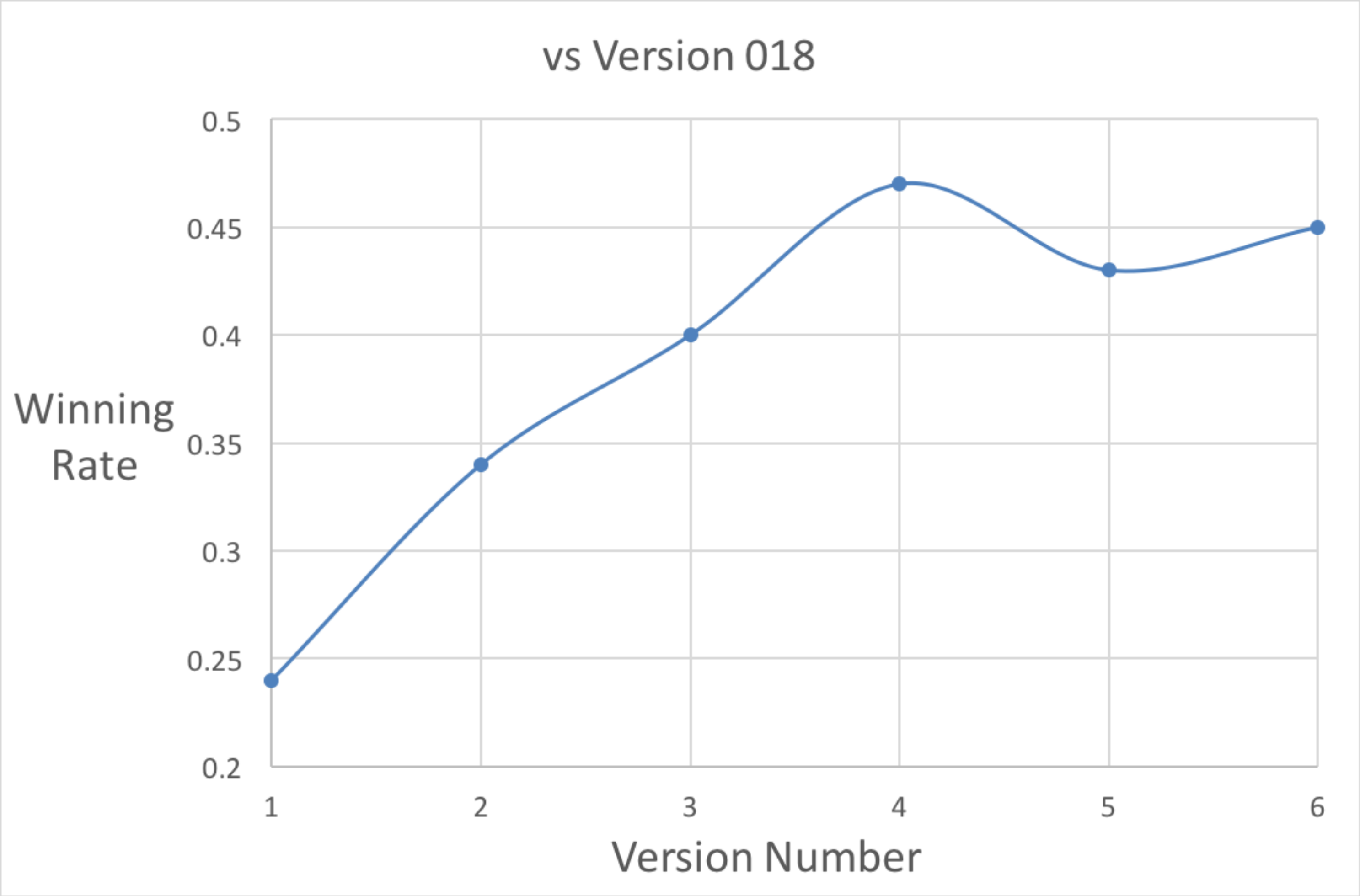
vs Version 001
0.79
0.74
0.69
0.64
0.59
0.54
0.49
Winning Rate
0
2
4
6
8
10
12
Version Number

Results



vs Previous Version
Winning Rate
0.68
0.66
0.64
0.62
0.6
0.58
0.56
0.54
1
3
5
7
9
Version Number

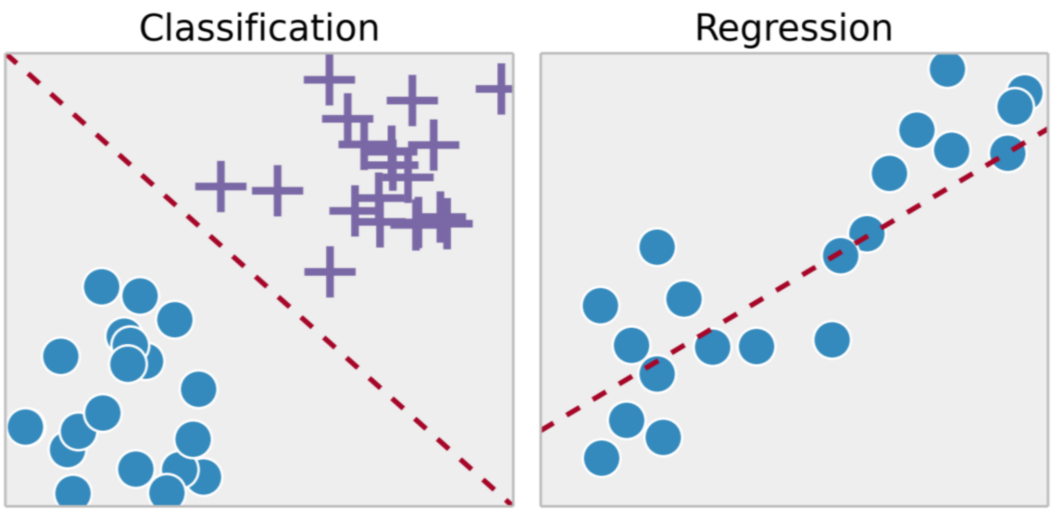
Results



vs Version 018
Winning Rate
0.5
0.45
0.4
0.35
0.3
0.25
0.2
1
2
3
4
5
6
Version Number

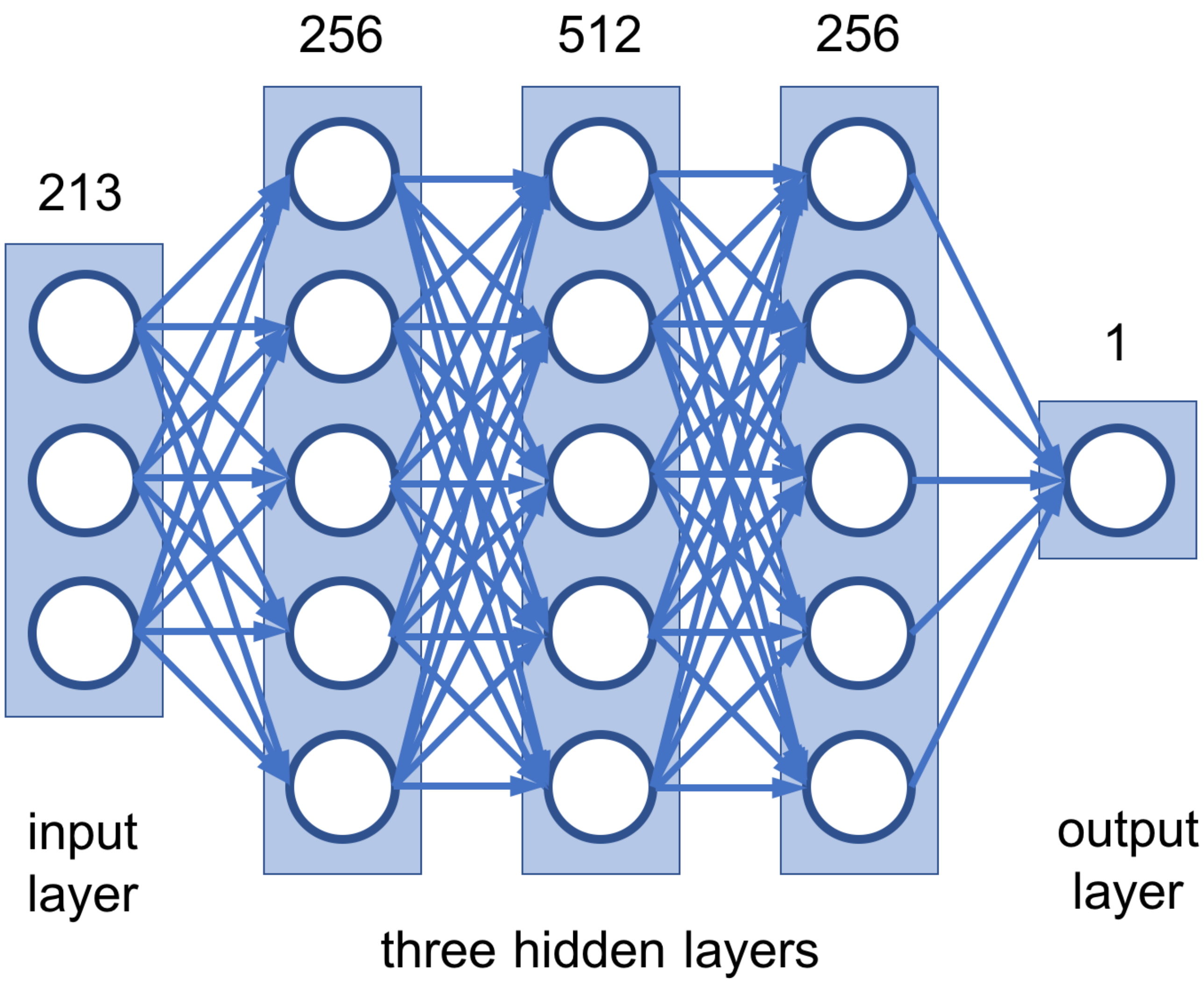
Evaluation Network

- Why?
  - Only Policy Network is not enough
  - Need to evaluate the winning rate of a chessboard status



- Supervised Learning
  - Regression Problem

Evaluation Network



Network Input

Feature	Length
Player Side	1
The Number of Pieces of Each Type	14
Pieces List (alive or not, xy-coordinates)	32 * 3
The number of valid moves for Rock, Cannon and Knight	12
Attack and Defend Map	90

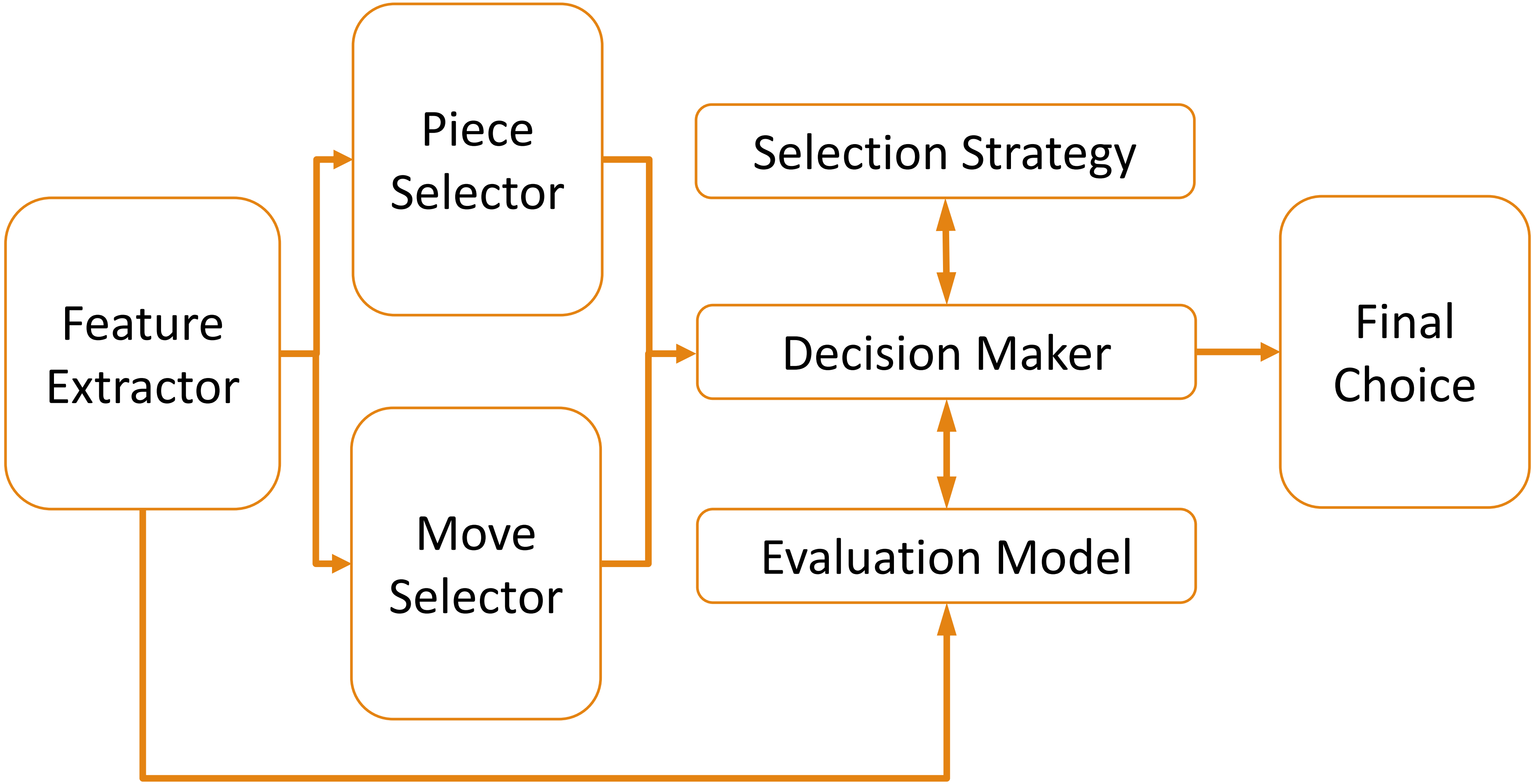
Training

- How to get the target values?
  - one evaluation function from an open-source API
  - do some mapping, shrink the range

0 ~ ±100	0 ~ ±100
±101 ~ ±700	±101 ~ ±170
over ±9000	±200

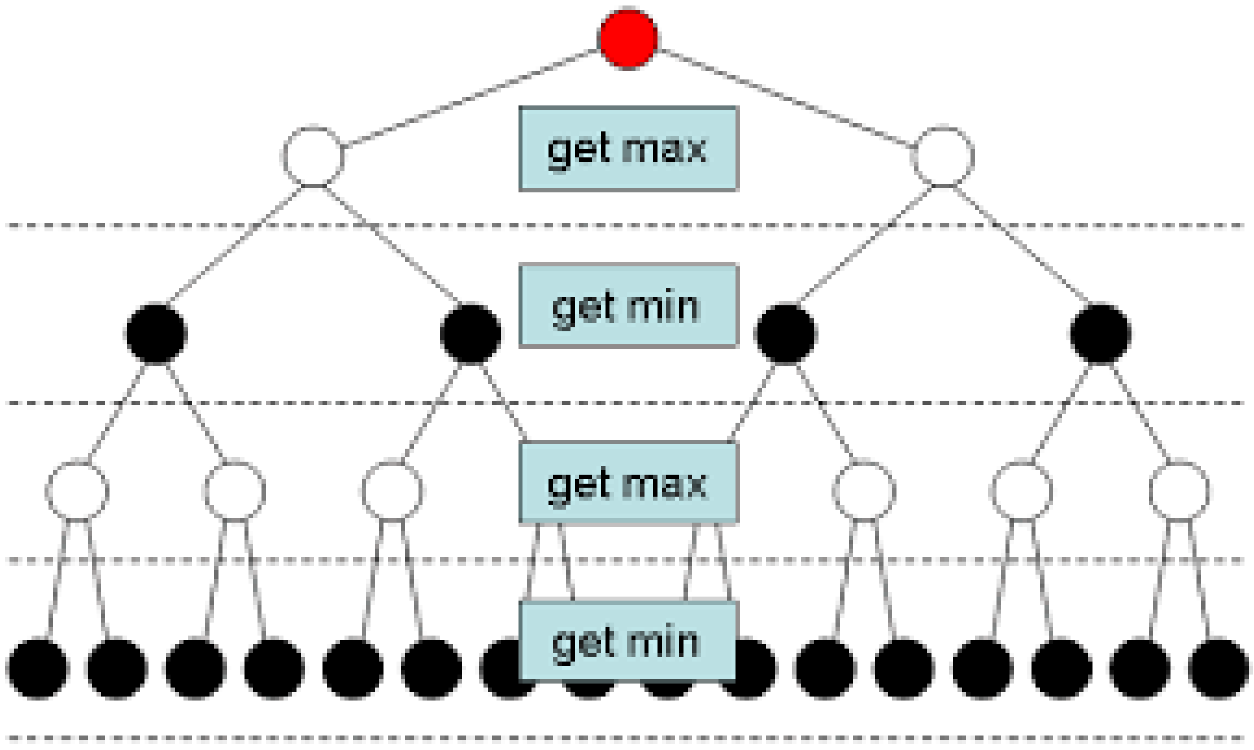
- Trained over 1,900,000 chessboard statuses
- Testing Results: loss -> ~ 20

How to use the models?

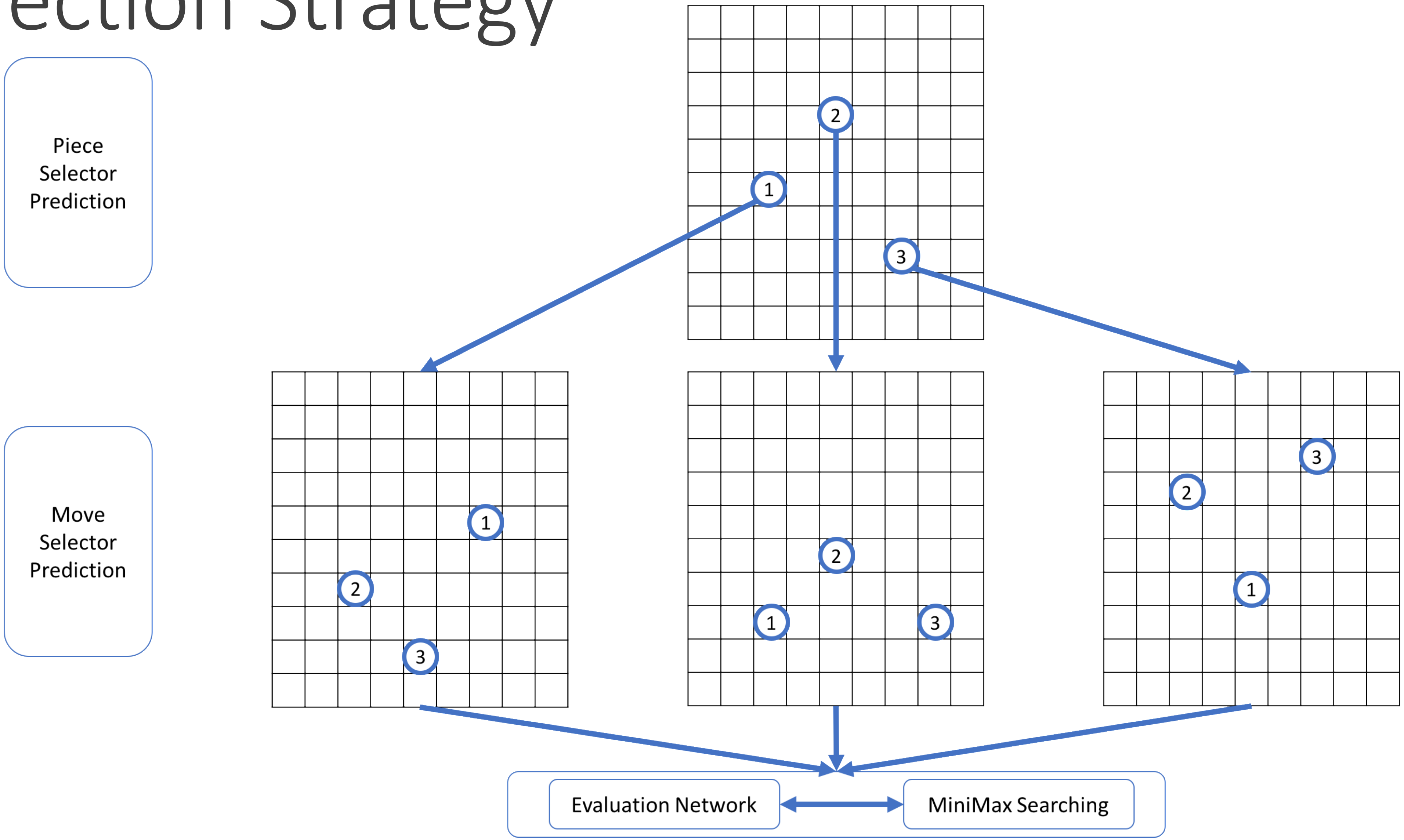


Minimax Searching

- Select minimum and maximum value in turn



Selection Strategy



Selection Strategy Enhancement

- Eliminate moves with probabilities below a certain threshold at first
- Different max breadth for different layers
  - 24 -> 12 -> 12
- Different quota for different piece types

Piece type	Quota 1	Quota 2
King	4	4
Advisor	2	2
Bishop	2	2
Rock	5	3
Cannon	5	3
Knight	4	2
Pawn	2	1

Testing

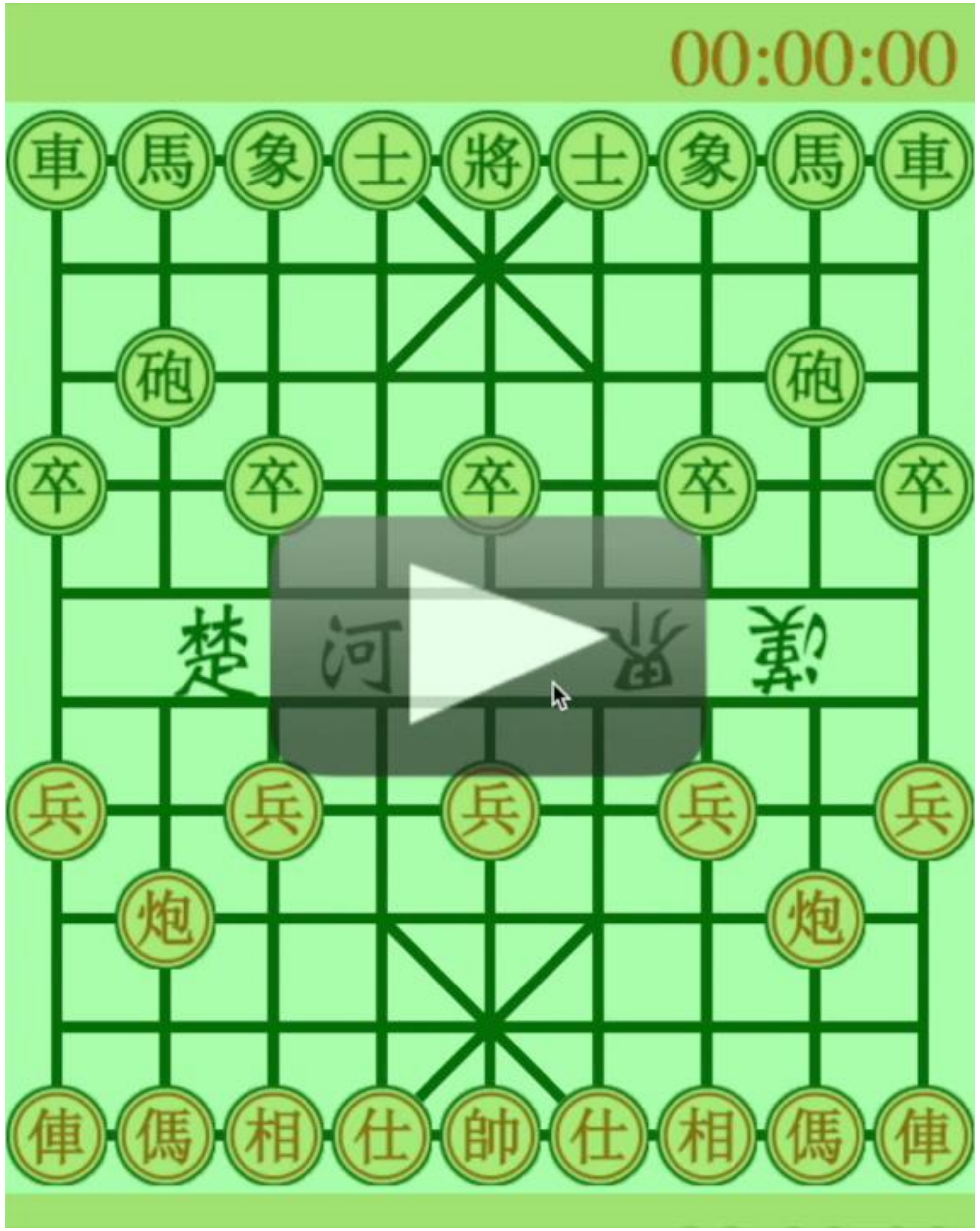
- Aliyun Server
- Socket.io
- Multiple login

Results

- The winning rate is 76%, won 19 out of 25 games
- On average, it takes 26.3 moves to win

	Number of Games	**Average Number of Moves**
Win	19	26.3
Lose	6	37.5

Demo



Discussion & Conclusion

- Policy Network and Evaluation Network
- Supervised Learning and Reinforcement Learning
- performed much better than the model in Term 1
- can compete with ordinary people now

- need further improvement:
  - negative reward is not working in Reinforcement Learning
  - continue to train the model
  - try different model structures

Q&A