Suphx: Mastering Mahjong with Deep Reinforcement Learnig

Junjie Li, Qiwei Ye, Li Zhao, Tao Qin, Tie-Yan Liu, Hsiao-Wuen Hon Microsoft Research Asia
Sotetsu Koyamada Kyoto University
Guoqing Liu University of Science and Technology of China
Chao Wang Tsinghua University
Ruihan Yang Nankai University

Introduction

Mahjong

complex rules and abundance of hidden information

↓

very challenging game for AI research.

Introduction

Suphx (short for Super Phoenix),

An AI system for 4-player Japanese Mahjong (Riichi Mahjong),

Suphx adopts a deep convolutional neural network as its model.

Introduction

1, It trained through supervised learning from the logs of human professional players.

2, It boosted through self-play reinforcement learning (RL), with the networks as the policy.

This network introduces several techniques to address challenges.

Introduction

- Global reward prediction

It trains a predictor to predict the final reward of the game based on information from the current and previous rounds.

Introduction

- Oracle guiding

It introduces the Oracle Agent, which allows players to see complete information about other players, including their private and wall tiles.

In their RL training process, they gradually drop the perfect information from the oracle agent, and finally convert it to a normal agent which only takes observable information as input.

Introduction

- parametric Monte-Carlo policy adaptation (pMCPA)

As rounds progress, policies trained offline are gradually modified and adapted to specific rounds during the online play phase to improve agent run-time performance.

Introduction

They evaluated Suphx on the most popular and competitive Mahjong platform, Tenhou, which has more than 350,000 active users.

Overview

Model	Functionality
Discard model	Decide which tile to discard in normal situations
Riichi model	Decide whether to declare Riichi
Chow model	Decide whether/what to make a Chow
Pong model	Decide whether to make a Pong
Kong model	Decide whether to make a Kong

Table 1: Five models in Suphx

Suphx learns five models to handle different situations: the discard model, the Riichi model, the Chow model, the Pong model, and the Kong model, as summarized in Table 1.

Overview

There are two kinds of situations that a Mahjong player needs to take actions.

- The Draw situation
- The Other-Discard situation

Overview

- The Draw situation

Suphx draws a tile from the wall. If its private tiles can form a winning hand with the drawn tile, the winning model decides whether to declare winning. If yes, it declares and the round is over. Otherwise, the round proceeds in the following order

1. Kong step
2. Riichi step
3. Discard step

Check each action in turn to see if it can be done.

Overview

- The Other-Discard situation

Other players discard a tile. If Suphx can form a winning hand with this tile, the winning model decides whether to declare winning. If yes, it declares and the round is over. Otherwise, it checks whether a Chow, Pong, or Kong can be made with the discarded tile. If not, it is the other players' turn to take actions; otherwise, the Chow, Pong, or Kong model decides what action to take:

Overview

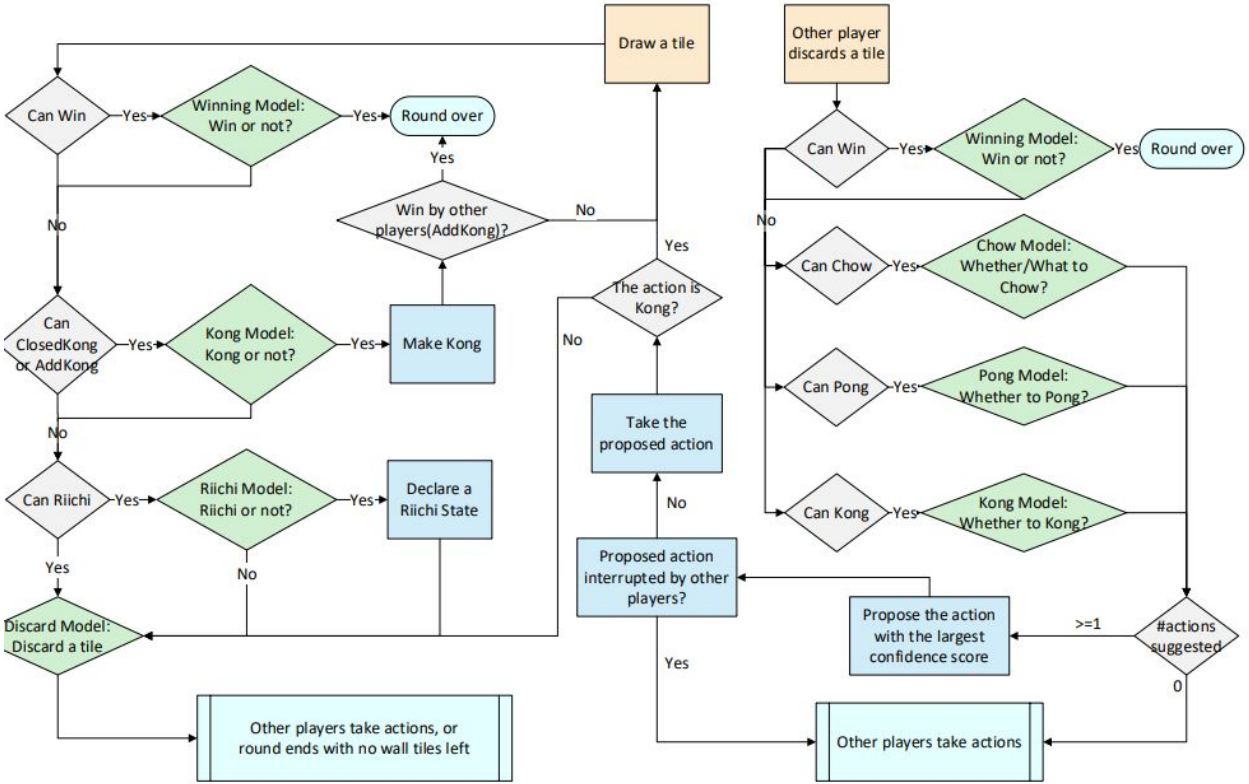


Figure 1: Decision flow of Suphx

Overview

Deep convolutional neural networks (CNNs) have shown powerful representational capabilities.

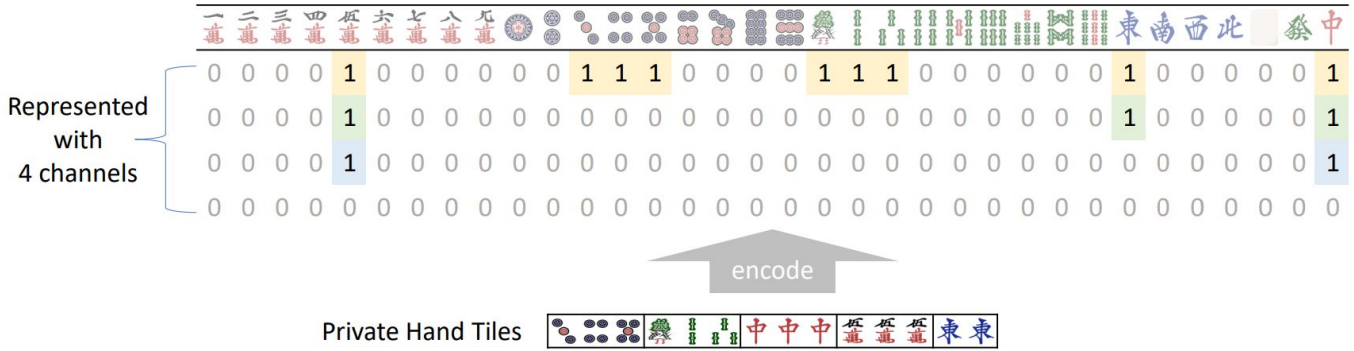
Suphx has adopted it as the model architecture for its policies.

They design a set of features to encode the observed information into channels that can be digested by CNNs.

Overview

Figure 2: Example of state

Overview



As there are 34 unique tiles in Japanese Mahong, they use multiple 34 × 1 channels to represent a state.

Overview

	Discard	Riichi	Chow	Pong	Kong
Input	34×838		34×958		
Output	34	2			

Table 2: Input/out dimensions of different models

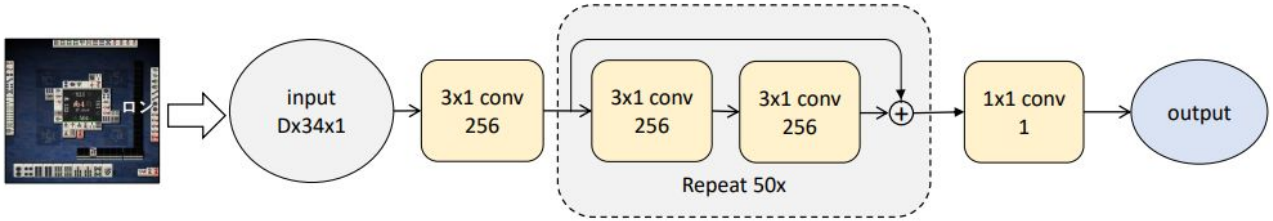


Figure 4: Structure of the discard model

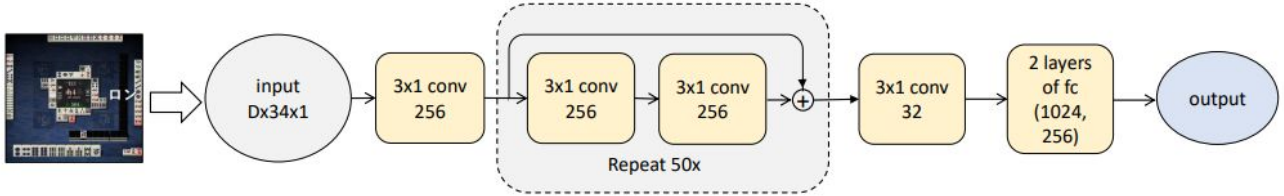


Figure 5: Structures of the Riichi, Chow, Pong, and Kong models

Learning Algorithm

The learning of Suphx contains three major steps.

1, They train the five models of Suphx by supervised learning, using (state, action) pairs of top human players collected from the Tenhou platform.

2, They improve the supervised models through self-play reinforcement learning (RL), with the models as policy.

They adopt the popular policy gradient algorithm and introduce global reward prediction and oracle guiding to handle the unique challenges of Mahjong.

3, During online playing, they employ run-time policy adaptation to leverage the new observations on the current round in order to perform even better.

Learning Algorithm

- Distributed Reinforcement Learning

The training of Suphx is based on distributed reinforcement learning.

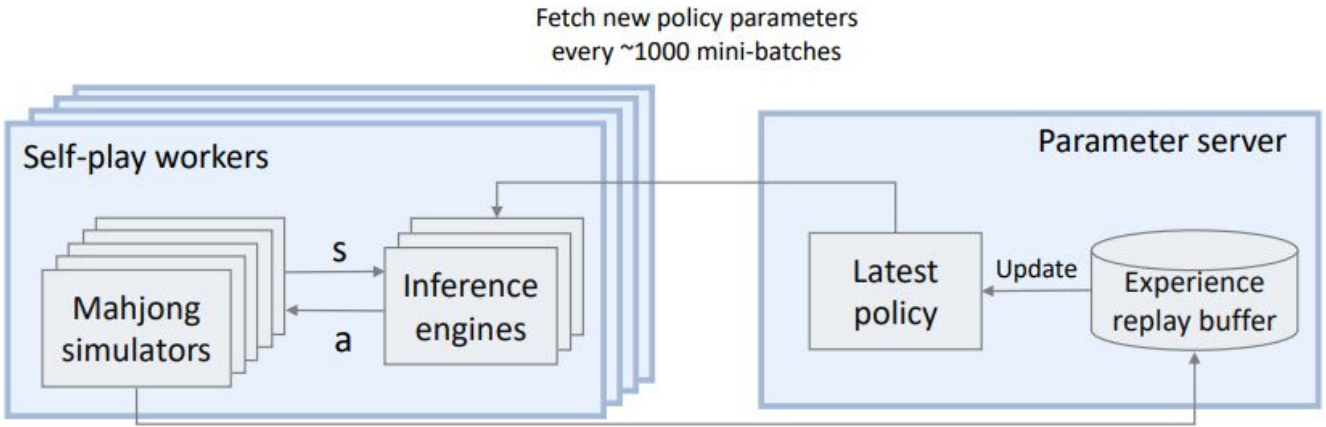


Figure 6: Distributed RL system in Suphx

Learning Algorithm

- Global Reward Prediction

To provide effective signal for RL training, they need to appropriately attribute the final game reward (a global reward) to each round of the game.

For this purpose, they introduce a global reward predictor Φ, which predicts the final game reward given the information of the current round and all previous rounds of this game.

Learning Algorithm

- Global Reward Prediction

The training data for this reward predictor Φ come from the logs of top human players in Tenhou, and Φ is trained by minimizing the following mean square error:

$$\min \frac{1}{N} \sum_{i=1}^{N} \frac{1}{K_i} \sum_{j=1}^{K_i} (\Phi(x_i^1, \cdots, x_i^j) - R_i)^2,$$

N: the number of games in the training data, Ri: the final game reward of the i-th game, Ki: the number of rounds in the i-th game,

x k i: the feature vector of the k-th round in the i-th game, including the score of this round, the current accumulated round score, the dealer position, the counters of repeat dealer and Riichi bets.

Learning Algorithm

- Oracle Guiding

Facilitate learning with oracle agents that have access to complete information.

First, they train the oracle agent through reinforcement learning, using all the features including the perfect ones. Then they gradually drop out the perfect features so that the oracle agent will eventually transit to a normal agent:

Learning Algorithm

- Parametric Monte-Carlo Policy Adaptation

The mahjong playing order is not fixed, making it difficult to construct a regular game tree.

Therefore, MCTS cannot be applied directly to mahjong.

In this study, they devise a new method, named parametric Monte Carlo policy adaptation (pMCPA).

Learning Algorithm

- Parametric Monte-Carlo Policy Adaptation

1. Simulations: Randomly sample private tiles for the three opponents and wall tiles from the pool of tiles excluding their own private tiles, and then use the offline-trained policy to roll out and finish the whole trajectory.

2. Adaptation: Perform gradient updates using the rollout trajectories to finetune the offline policy.

3. Inference: Use the fine tuned policy to play against other players in this round.

Offline Evaluation

- Supervised Learning

Model	Training Data Size	Test Accuracy	Previous Accuracy (*6*)
Discard model	15M	76.7%	68.8 %
Riichi model	5M	85.7 %	-
Chow model	10M	95.0%	90.4%
Pong model	10M	91.9 %	88.2%
Kong model	4M	94.0 %	-

Table 3: Results for supervised learning

Offline Evaluation

・Reinforcement Learning

To demonstrate the value of each RL component in Suphx, they trained several Mahjong agents:

• SL: the supervised learning agent.

• SL-weak: an under-trained version of the SL agent.

• RL-basic: the basic version of the reinforcement learning agent. In RLbasic, the discard model was initialized with the SL discard model and then boosted through the policy gradient method with round scores as reward and entropy regularization. The Riichi, Chow, Pong, and Kong models remain the same as those of the SL agent.6

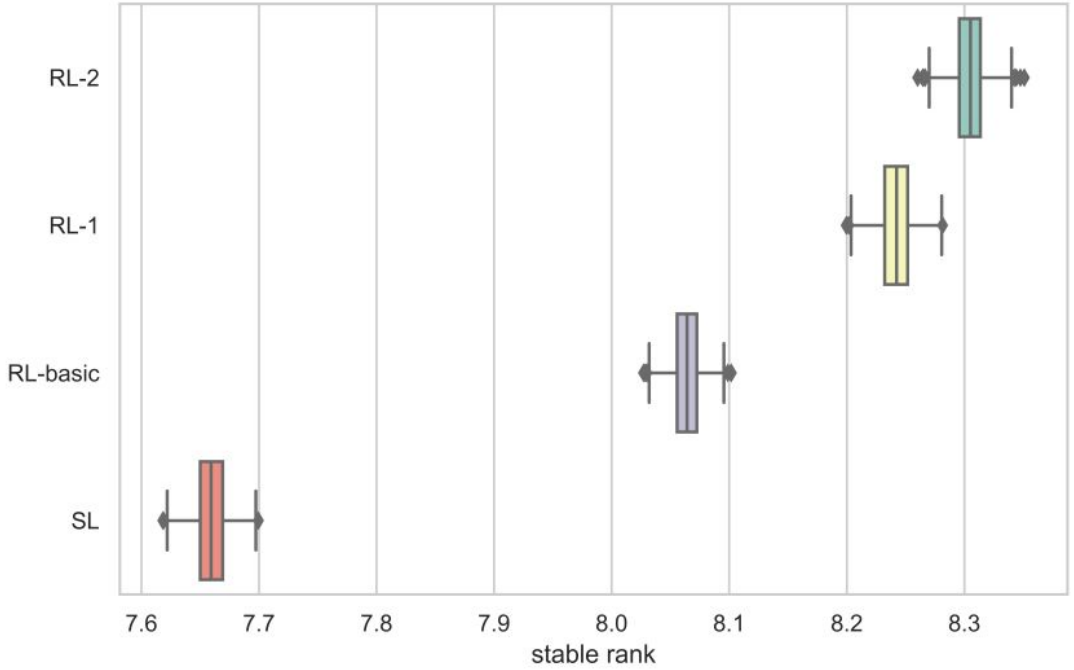
• RL-1: the RL agent that enhances RL-basic with global reward prediction.

• RL-2: the RL agent that further enhances RL-1 with oracle guiding.

In both RL-1 and RL-2, they also only trained the discard model using RL, and left the other four models the same as those of the SL agent.

Offline Evaluation

Each agent plays against 3 SL-weak agents on these games. For the evaluation metric, they computed the stable rank of an agent following the rules of Tenhou.



Online Evaluation

To evaluate the real performance of Suphx, they let it play on Tenhou in the expert room.

Suphx played 5000+ games in the expert room and achieved 10 dan in terms of record rank9 and 8.74 dan in terms of stable rank.10 It is the first and only AI in Tenhou that achieves 10 dan in terms of record rank.

Online Evaluation

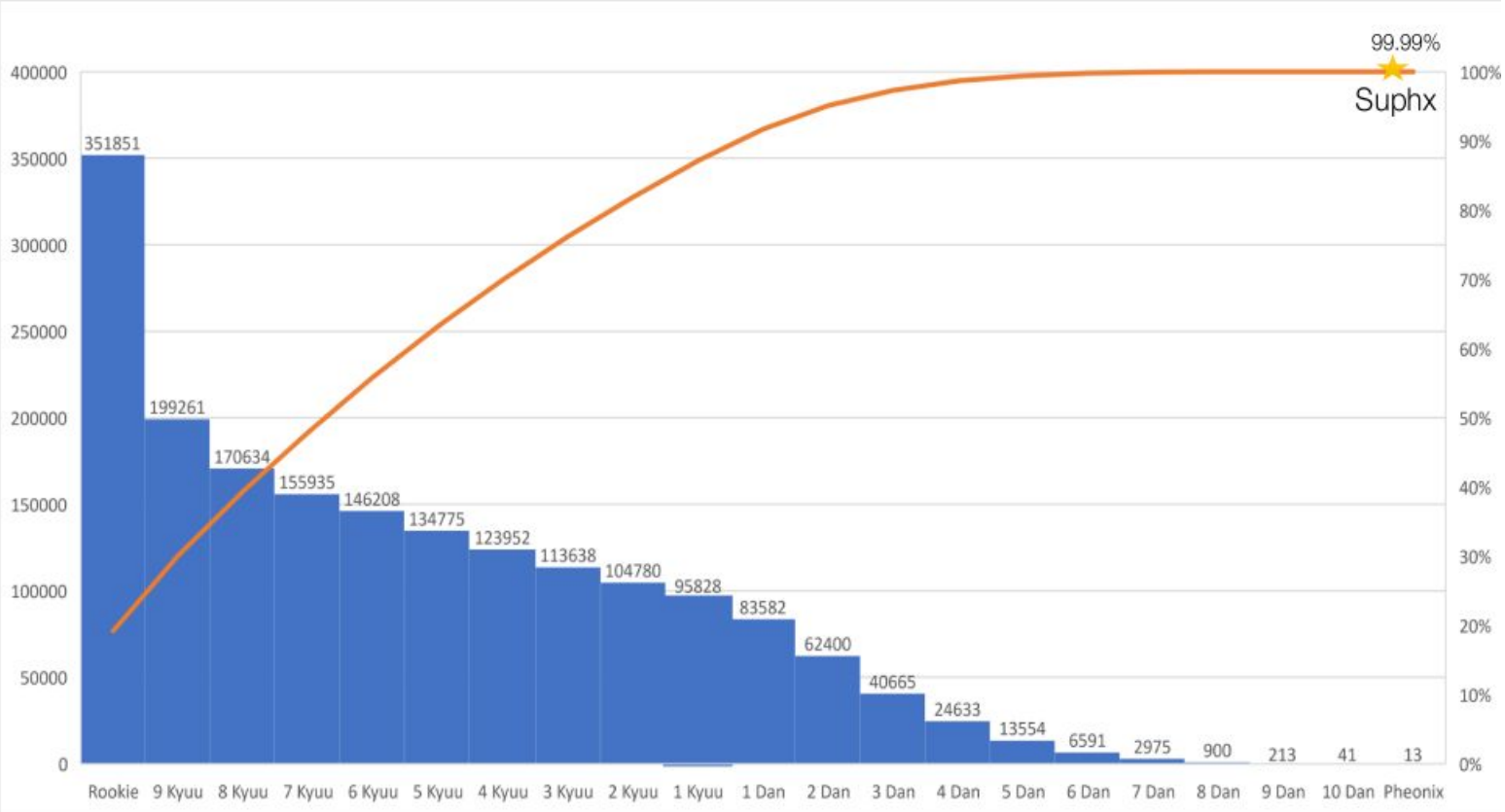
	#Game	record rank	Stable Rank
Bakuuchi	30,516	9 dan	6.59
NAGA	9,649	8 dan	6.64
Top human	8,031	10 dan	7.46
Suphx	5,760	10 dan	**8.74**

Table 4: Comparison with other AIs and top human players.

Stable Rank

$$\frac{5 \times n_1 + 2 \times n_2}{n_4} - 2.$$

Online Evaluation



Online Evaluation

	1st Rank	2nd Rank	3rd Rank	4th Rank	Win Rate	Deal-in Rate
Bakuuchi	28.0%	26.2%	23.2%	22.4%	23.07%	12.16%
NAGA	25.6%	27.2%	25.9%	21.1%	22.69%	11.42%
Top human	28.0%	26.8%	24.7%	20.5%	-	-
Suphx	29.3%	27.5%	24.4%	**18.7%**	22.83%	**10.06%**

Table 5: More statistics: rank distribution and win/deal-in rate

Conclusion and Discussion

- Global Reward Prediction

In this system, the reward predictor takes limited information as its input. Clearly, more information will lead to better reward signal.

They are investing how to leverage prefect information (e.g., by comparing the private initial hands of different players) to measure the difficulty of a round/game and then boost the reward predictor.

Conclusion and Discussion

- Oracle Guiding

They instantiated this concept using the gradual transition from an oracle agent to a normal agent by means of perfect feature dropout.

Conclusion and Discussion

- Run-time Policy

They did not adapt the policy only for the first hand, but continued to adapt the policy as the game progressed and more information became observable.

Doing so should further improve the performance of the policy.

Furthermore, since the policy is adapted gradually, it is even possible to use policy adaptation in online play with affordable computational resources.

Thank you for your attention!