ICLR

Strength Estimation and Human-Like Strength Adjustment in Games

Chun-Jung Chen [1,2], Chung-Chin Shih[1], Ti-Rong Wu[1]

[1] Academia Sinica
[2] National Taiwan University

Motivation

- AI has achieved superhuman performance in board games
- How to learn from these AI?
  - AI is too strong, directly playing with it results in frustration

- **Questions**: Can AI provide suitable strength for human players to learn?
- **Contribution**: Propose a **strength system** which can estimate player's strength and adjust strengths accordingly to offer a better human-AI learning experience

Challenges in Strength Estimation with AI

- Playing multiple games to estimate strength:
  - Can **accurately** predict strength after around 20 games (Liu, 2019)
  - Issue: Requires a lot of playtime ➔ **inefficient**
- Predicting by history game records:
  - Does not require playing, making it **efficient** (Moudřík & Neruda, 2016)
  - Issue: Previous state-of-the-art (SOTA) achieved only 49% accuracy even with 100 game records ➔ highly **inaccurate**

➔ How to **efficiently and accurately** estimate player strength?

Challenges in Strength Adjustment with AI

- Existing methods can adjust strength, but
  - AI plays differently from humans ➔ impacts learning and experience (Wu et al., 2019)
  - Requires fine-tuning to align with real-world strength

➔ How to **adjust playing strengths** while simultaneously offer **human-like behavior**?

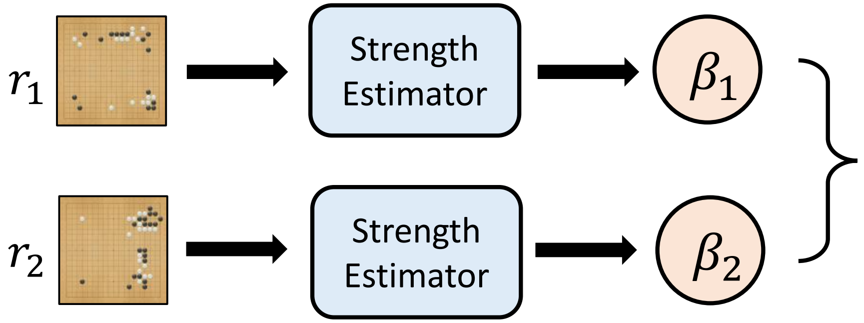
Strength Estimator (SE)

- We proposed Strength Estimator (SE) network, $\text{SE}_\theta(p) = \beta$, which predicts a strength score (β) for a given state-action pair $p = (s, a)$

Bradley-Terry model

$P(i \succ j) = \frac{\gamma_i}{\gamma_i + \gamma_j} = \frac{e^{\beta_i}}{e^{\beta_i} + e^{\beta_j}}$ represents the probability that individuals i defeats j

- Higher γ_i indicates a stronger individual



$$P(r_1 \succ r_2) = \frac{e^{\beta_1}}{e^{\beta_1} + e^{\beta_2}}$$

➔ $P(r_1 \succ r_2) \rightarrow 1$ if r_1 is stronger than r_2

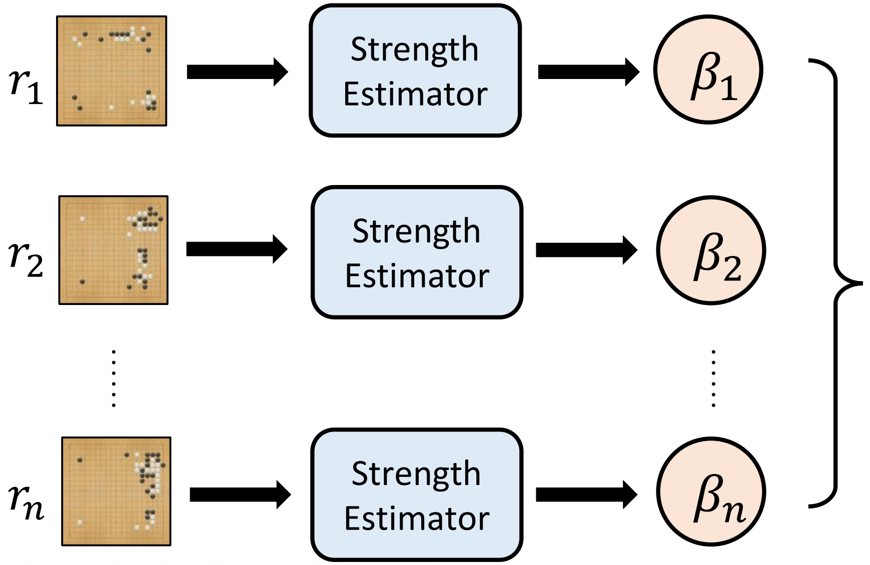
Strength Estimator (SE)

- We proposed Strength Estimator (SE) network, $\text{SE}_\theta(p) = \beta$, which predicts a strength score (β) for a given state-action pair $p = (s, a)$

Generalized Bradley-Terry Model

$$P(i \succ \{j, k\}) = \frac{\gamma_i}{\gamma_i + \gamma_j + \gamma_k} = \frac{e^{\beta_i}}{e^{\beta_i} + e^{\beta_j} + e^{\beta_k}}$$

represents the probability that individuals i defeats j and k



$$P(r_1 \succ \{r_2, r_3, \dots, r_n\}) = \frac{e^{\beta_1}}{e^{\beta_1} + e^{\beta_2} + e^{\beta_3} + \dots + e^{\beta_n}}$$

➔ $P(r_1 \succ \{r_2, r_3, \dots, r_n\}) \rightarrow 1$ if r_1 is the strongest in $\{r_1 \sim r_n\}$

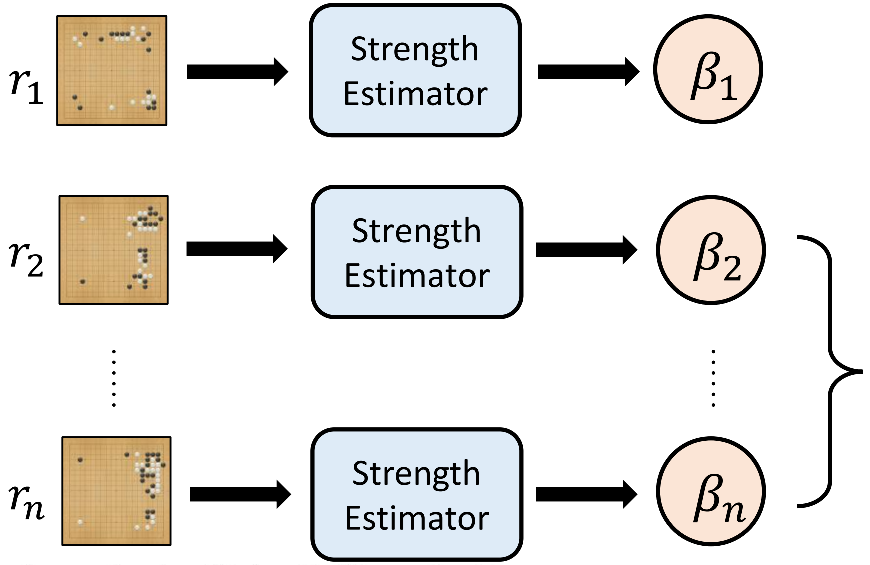
Strength Estimator (SE)

- We proposed Strength Estimator (SE) network, $\mathrm{SE}_\theta(p) = \beta$, which predicts a strength score (β) for a given state-action pair $p = (s, a)$

Generalized Bradley-Terry Model

$$P(i \succ \{j, k\}) = \frac{\gamma_i}{\gamma_i + \gamma_j + \gamma_k} = \frac{e^{\beta_i}}{e^{\beta_i} + e^{\beta_j} + e^{\beta_k}}$$

represents the probability that individuals i defeats j and k



$$P(r_2 \succ \{r_3, \dots, r_n\}) = \frac{e^{\beta_2}}{e^{\beta_2} + e^{\beta_3} + \dots + e^{\beta_n}}$$

➔ $P(r_2 \succ \{r_3, \dots, r_n\}) \to 1$ if r_2 is the strongest in $\{r_2 \sim r_n\}$

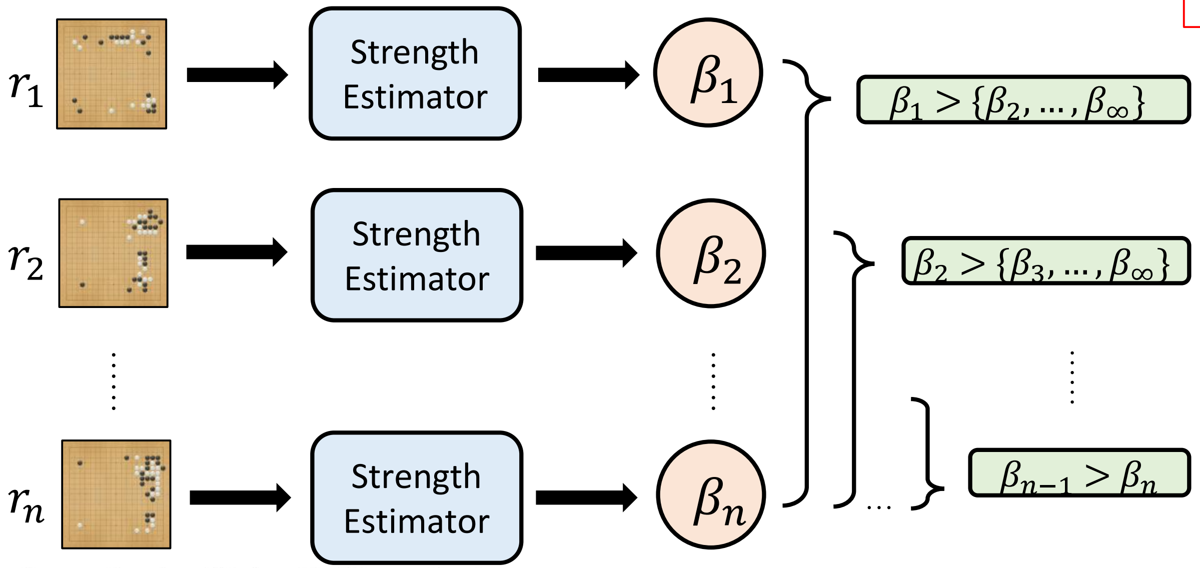
Strength Estimator (SE)

- We proposed Strength Estimator (SE) network, $\mathrm{SE}_\theta(p) = \beta$, which predicts a strength score (β) for a given state-action pair $p = (s, a)$

Generalized Bradley-Terry Model

$$P(i \succ \{j, k\}) = \frac{\gamma_i}{\gamma_i + \gamma_j + \gamma_k} = \frac{e^{\beta_i}}{e^{\beta_i} + e^{\beta_j} + e^{\beta_k}}$$

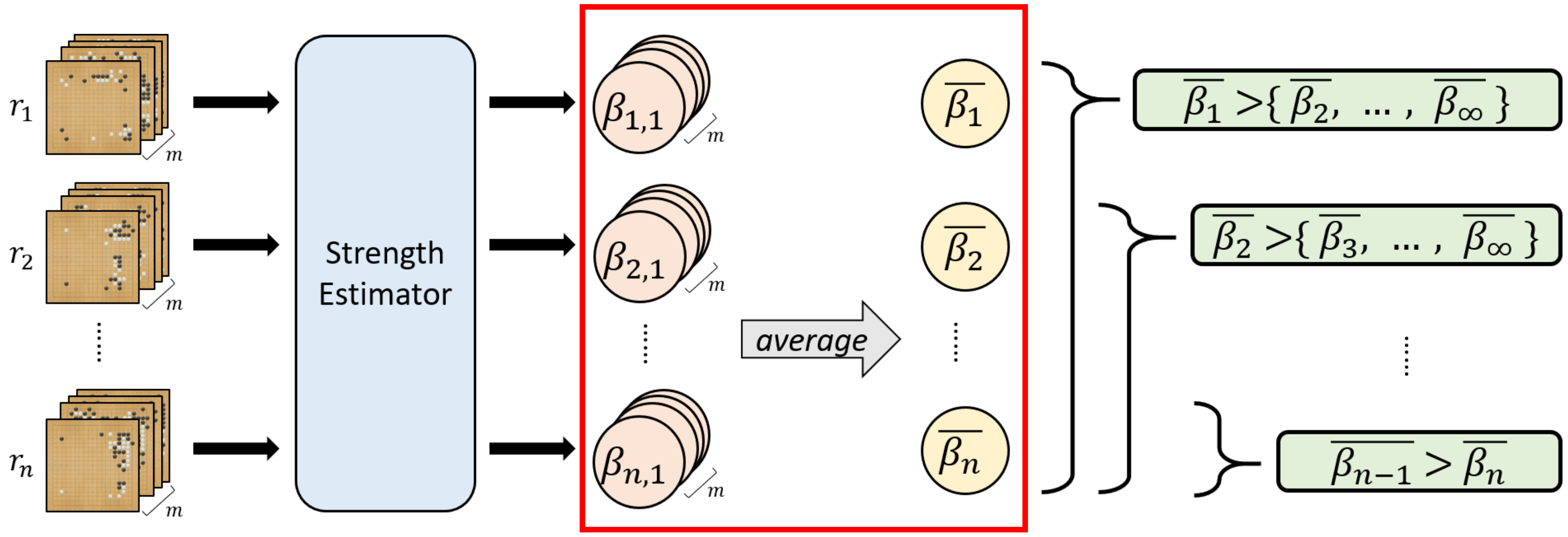
represents the probability that individuals i defeats j and k



$$\rightarrow \prod_{i=1}^{n-1} P(r_i \succ \{r_{i+1}, r_{i+2}, \ldots, r_n\}) \rightarrow 1$$

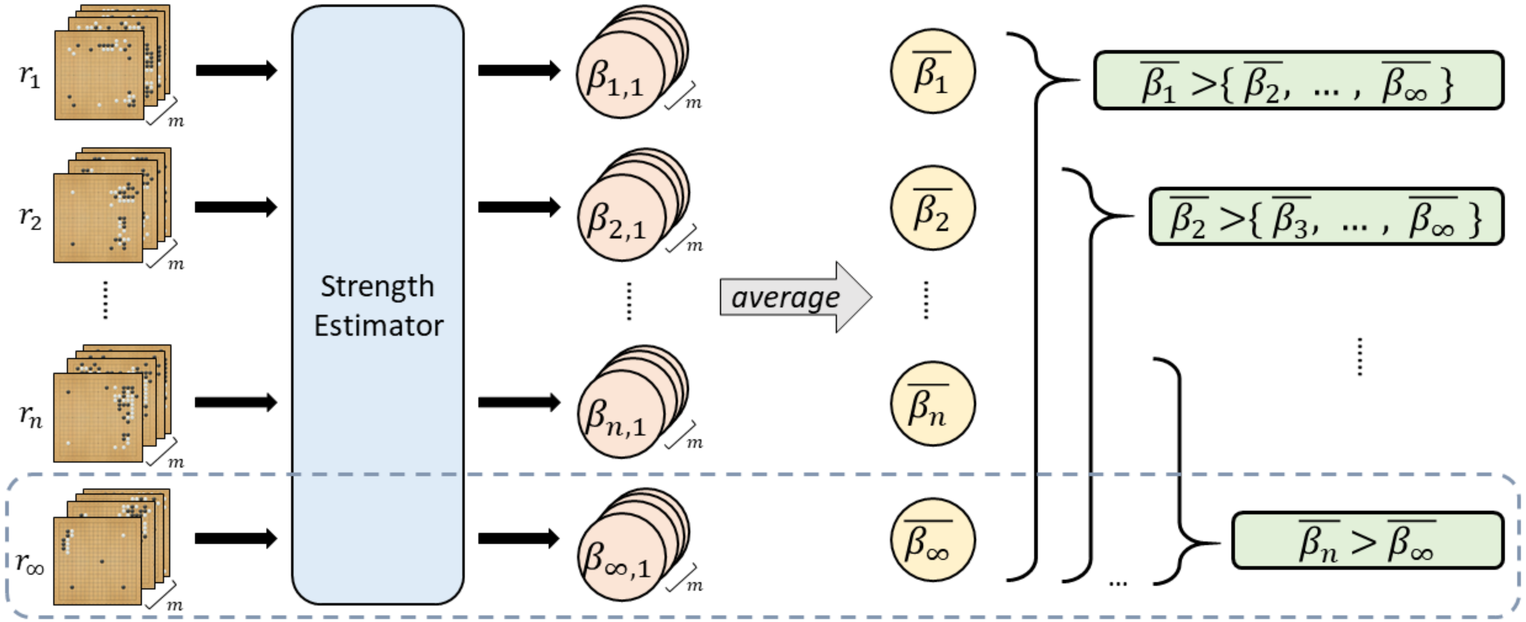
Strength Estimator (SE)

- To stabilize the prediction, we propose using **aggregated $\boldsymbol{\beta}$ values** to estimate overall strength
  - For each rank r_i, m state-action pairs is used



Strength Estimator (SE)

- Handling out-of-distribution actions:
  - Introduce a weakest rank, r_∞, among all ranks
  - Disturb the state-action pair by replacing the action to a random action



SE-MCTS

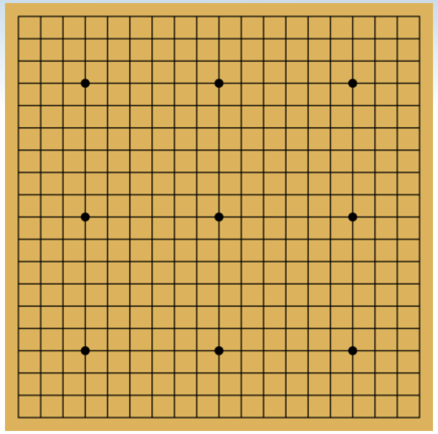
- **Human-like strength adjustment**
  - To adjust to a specific rank
    - Obtain the strength score (β_{target}) which is evaluated by the strength estimator
    - Prioritize the move in MCTS search based on how close to the target strength score through PUCT formula

$$a^* = \underset{a}{argmax}\left\{Q(s,a) + c \cdot \left(P(s,a) - c_1 \cdot \hat{\delta}(s,a)\right) \cdot \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}\right\},$$

$$\text{where}\begin{cases} \hat{\delta}(s,a) = \text{normalized } \delta(s,a) & \text{// normalized difference in [0,1]} \\ \delta(s,a) = |\beta(s,a) - \beta_{target}| & \text{// difference between action and target strength} \end{cases}$$

Dataset: Go and Chess

	Go	Chess
Dataset	FoxWeiqi	LiChess
# Ranks	11	8
Ranks	r_1: 9 Dan r_2: 8 Dan r_3: 7 Dan r_4: 6 Dan r_5: 5 Dan r_6: 4 Dan r_7: 3 Dan r_8: 2 Dan r_9: 1 Dan r_{10}: 1-2 Kyu r_{11}: 3-5 Kyu	r_1: Elo 1000-1199 r_2: Elo 1200-1399 r_3: Elo 1400-1599 r_4: Elo 1600-1799 r_5: Elo 1800-1999 r_6: Elo 2000-2199 r_7: Elo 2200-2399 r_8: Elo 2400-2599

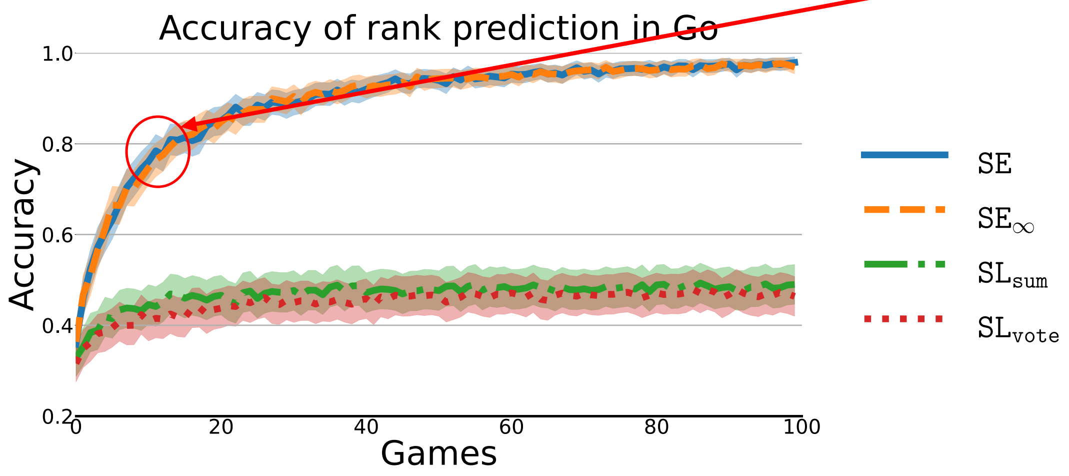

Go

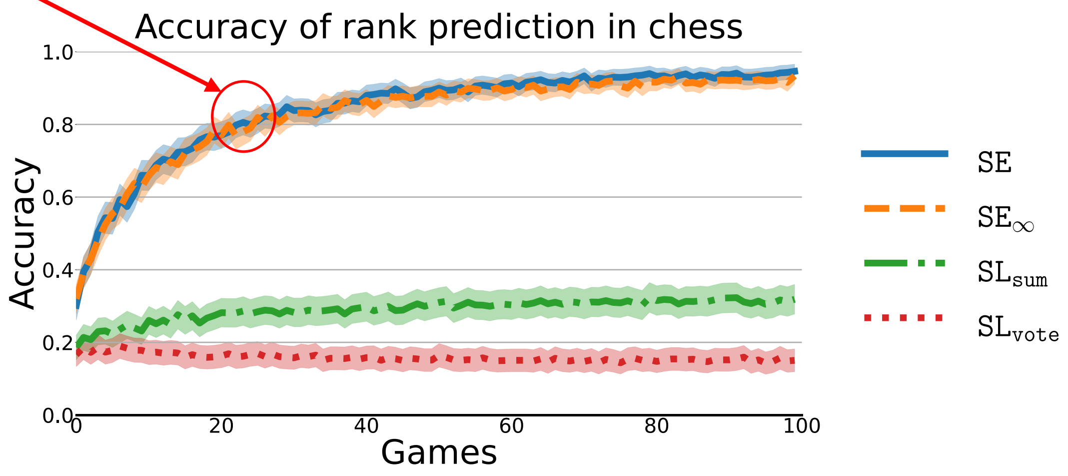

Chess

Strength Estimation in Go and Chess

- We compare SE and two Supervised-learning (SL) methods (Moudřík & Neruda, 2016) in both Go and chess

SE achieves over 80% accuracy within just 15 and 26 games in Go and chess





Strength Adjustment in Go

- MCTS: High move accuracy but cannot adjust strength
- SA-MCTS: Can adjust strengths but lowest move accuracy
- SE_∞-MCTS: Perform high move accuracy and also can adjusts strength

	MCTS	SA-MCTS	SE_∞-MCTS
r_1 (9 dan)	53.05% ± 0.95%	47.00% ± 0.95%	**53.73%** ± 0.95%
r_2 (8 dan)	53.79% ± 0.97%	45.83% ± 0.97%	**54.30%** ± 0.97%
r_3 (7 dan)	52.70% ± 0.98%	46.70% ± 0.98%	**53.88% ± 0.98%**
r_4 (6 dan)	52.50% ± 0.92%	45.86% ± 0.92%	**53.58% ± 0.92%**
r_5 (5 dan)	49.48% ± 0.93%	42.29% ± 0.92%	**50.35% ± 0.93%**
r_6 (4 dan)	49.44% ± 0.91%	42.72% ± 0.90%	**50.87% ± 0.91%**
r_7 (3 dan)	50.75% ± 0.89%	42.68% ± 0.88%	**51.40% ± 0.89%**
r_8 (2 dan)	50.17% ± 0.93%	40.94% ± 0.92%	**50.99% ± 0.93%**
r_9 (1 dan)	48.10% ± 0.89%	40.94% ± 0.88%	**49.44% ± 0.89%**
r_{10} (1-2 kyu)	46.95% ± 0.91%	36.58% ± 0.88%	**47.84% ± 0.91%**
r_{11} (3-5 kyu)	46.87% ± 0.89%	36.64% ± 0.86%	**48.23% ± 0.89%**
average	50.35% ± 0.28%	42.56% ± 0.28%	**51.33% ± 0.28%**

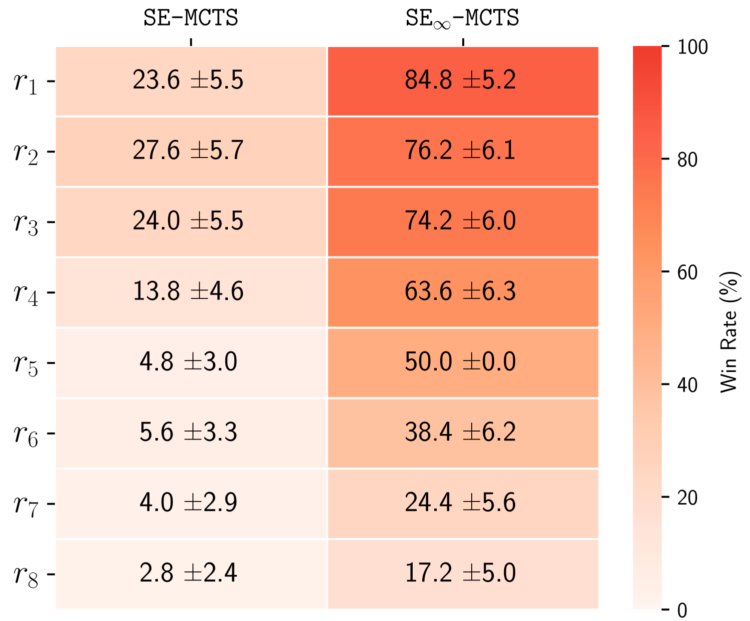
Win Rate (%)

	MCTS	SA-MCTS	SE-MCTS	SE_∞-MCTS
r_1	92.0 ±3.4	75.6 ±5.3	4.4 ±2.6	73.6 ±5.5
r_2	92.0 ±3.4	62.0 ±6.0	6.0 ±3.0	73.2 ±5.5
r_3	92.0 ±3.4	50.8 ±6.2	1.2 ±1.4	62.4 ±6.0
r_4	92.0 ±3.4	55.6 ±6.2	2.0 ±1.7	51.6 ±6.2
r_5	92.0 ±3.4	48.4 ±6.2	0.0 ±0.0	50.0 ±0.0
r_6	92.0 ±3.4	49.2 ±6.2	1.2 ±1.4	43.6 ±6.2
r_7	92.0 ±3.4	42.0 ±6.1	1.2 ±1.4	43.2 ±6.2
r_8	92.0 ±3.4	31.2 ±6.1	0.0 ±0.0	22.4 ±5.2
r_9	92.0 ±3.4	29.6 ±5.7	0.0 ±0.0	20.8 ±5.0
r_{10}	92.0 ±3.4	19.2 ±4.9	0.0 ±0.0	5.6 ±2.9
r_{11}	92.0 ±3.4	8.0 ±3.4	0.0 ±0.0	4.0 ±2.4

Strength Adjustment in Chess

- MCTS: High move accuracy but cannot adjust strength
- SA-MCTS: Can adjust strengths but lowest move accuracy
- SE_{∞}-MCTS: Perform high move accuracy and also can adjusts strength

$rank(Elo)$	MCTS	SA-MCTS	SE_{∞}-MCTS
$r_1(2400-2599)$	**51.97**% ±0.69%	50.17% ±0.69%	51.51% ±0.69%
$r_2(2200-2399)$	**51.58**% ±0.69%	47.49% ±0.69%	51.14% ±0.69%
$r_3(2000-2199)$	**49.23**% ±0.69%	45.01% ±0.69%	49.19% ±0.69%
$r_4(1800-1999)$	46.52% ±0.69%	41.78% ±0.68%	**47.26**% ±0.69%
$r_5(1600-1799)$	45.45% ±0.69%	38.18% ±0.67%	**46.62**% ±0.69%
$r_6(1400-1599)$	44.33% ±0.69%	36.84% ±0.67%	**46.12**% ±0.69%
$r_7(1200-1399)$	41.54% ±0.68%	31.49% ±0.64%	**43.04**% ±0.68%
$r_8(1000-1199)$	41.89% ±0.68%	30.72% ±0.64%	**43.08**% ±0.68%
average	46.56% ±0.24%	40.21% ±0.24%	**47.25**% ±0.24%



Summary

- We propose a comprehensive strength system that
  - Strength estimation: efficient and accurate (SOTA)
  - Strength adjustment: human-like, utilized the predicted strength
  - **Enhances human-AI interaction** by providing adaptive and realistic gameplay experiences

Thank You for Your Attention

Our code and model are available at
https://rlg.iis.sinica.edu.tw/papers/strength-estimator/