Reinforcement Learning on Mancala

Andrew Ortega, Mythri Kulkarni, Angelina Cottone, Alyssa Ann Toledo, Harris Habib, Nick Gomez, Shriya Rudrashetty, Aatish Lobo

What is Mancala?

How to play:

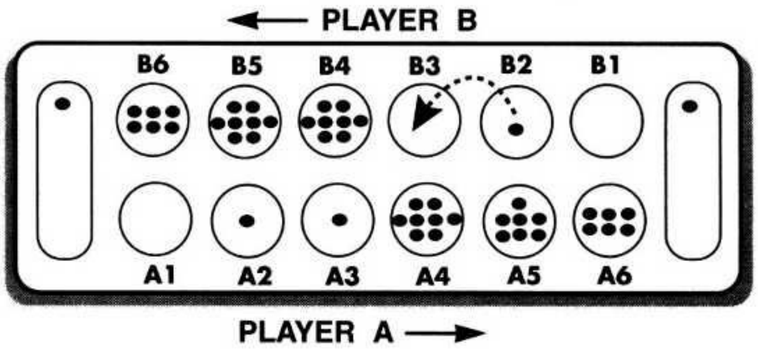
- Grab set of stones from a pocket on your side
  and drop one stone in each pocket you pass.
- When looping around, don't add a stone to opponent
  base.

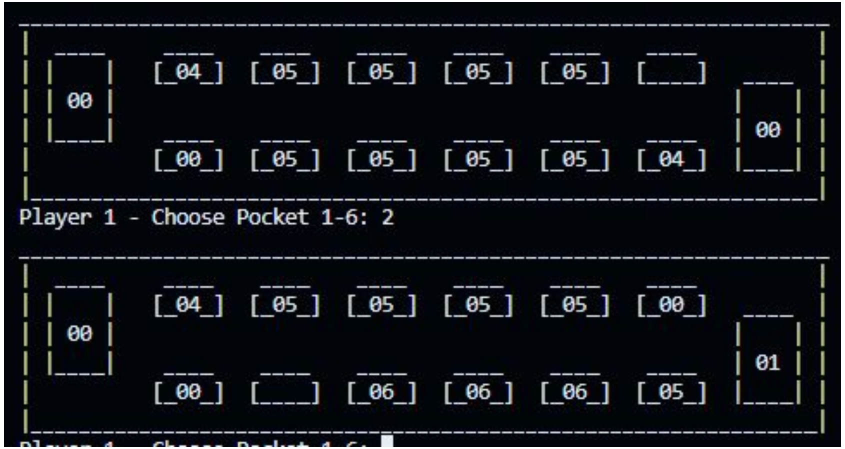
Important Rules:

- Capture
- Additional Turns (Free turn)

Scaffolding

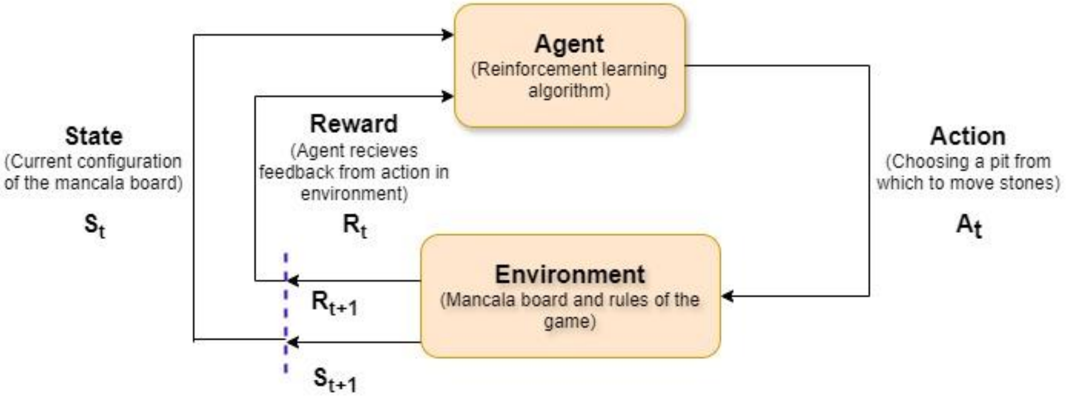
- Code from mkgray/mancala-reinforcement-learning repository on Github
- Terminal version of Mancala game that could have 0, 1, or 2 human players
- Trained model with basic Q-learning algorithm by having it play against a random # picker for 100000 games





Q-Learning Recap

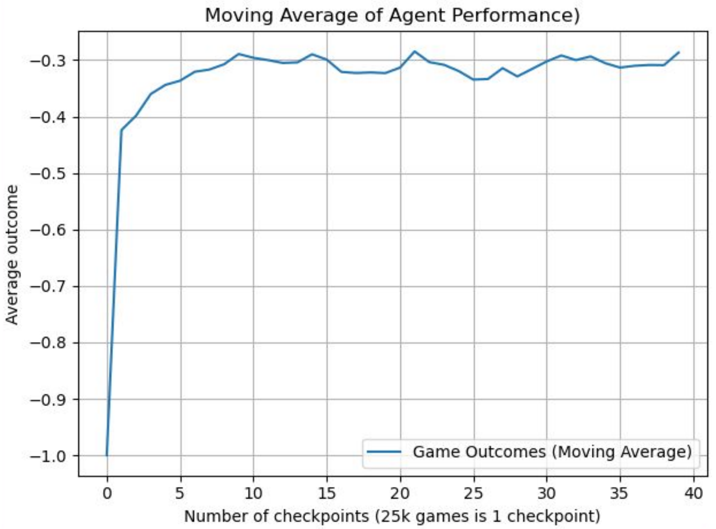
$$\underbrace{Q'(s,a)}_{\text{New Q value}} = \overbrace{\underbrace{Q(s,a)}_{\text{Current Q value}}}^{\text{Immediate reward}} + \underbrace{\alpha}_{\text{Learning rate}} \overbrace{[\underbrace{R(s,a)}_{\text{Reward for current action}} + \underbrace{\gamma}_{\text{Discount rate}} \underbrace{maxQ'(s',a')}_{\text{The highest value of Q among the next possible actions}} - \underbrace{Q(s,a)}_{\text{Current Q value}}]}^{\text{Future reward}}$$



Base RL Model

What we started with:

- An average 30% win-rate
- Poor Reward System (poor training)
- Moving Average where:
  1= win, 0 = Draw, -1 = Loss.



Reward Adjustment to Base Model

New Reward System:

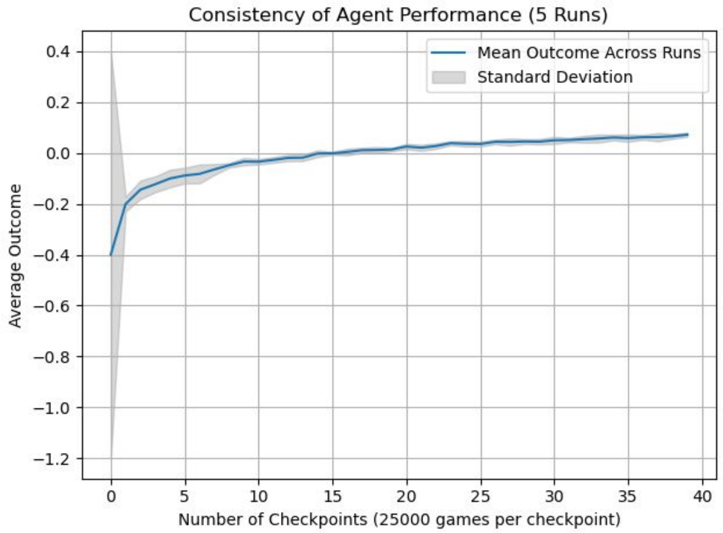
- + 1000 for winning
- - 1000 for losing
- + 5 per marble captured
- + 2 for each marble gained at end of turn
- + total marbles ended with

Previous Reward System:

- + total marbles ended with

Final (Baseline) Result:

- Average win rate of 47.52%



Now we're starting to get somewhere...

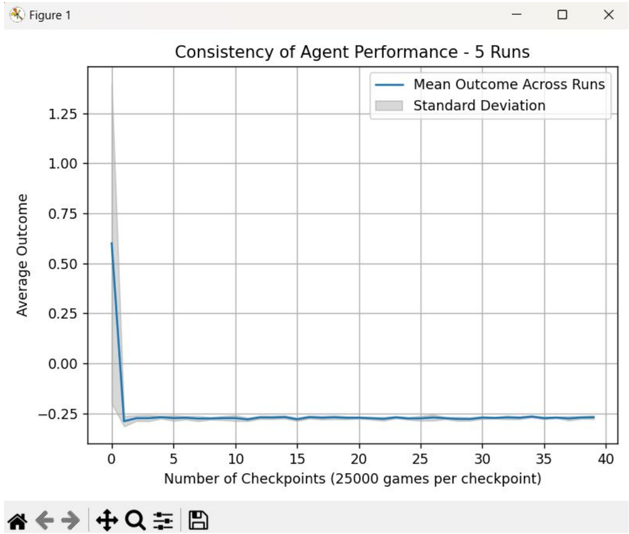
Potential-Based Reward Shaping

- Goal of reward shaping → include additional rewards for positive state transitions
- Ex) $F(s, s') > 0$ → positive reward for the transition between states s to s′
  - Provides heuristic knowledge

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + F(s,s') + \gamma \max_{a'} Q_i(s',a') - Q(s,a)]$$

$$F(s,s') = \gamma\Phi(s') - \Phi(s)$$

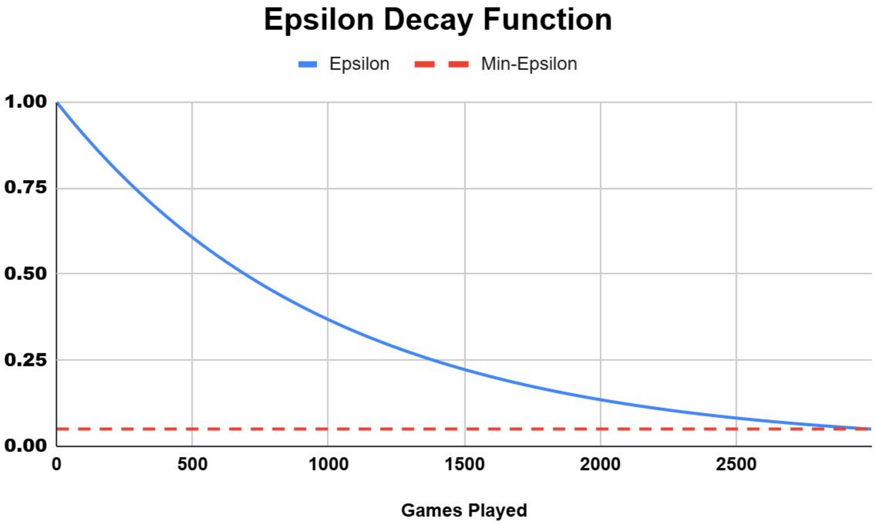
- New rewards include...
  - Positive difference in agent/opponent stores
  - Adding stones to agent store
  - Taking stones from opponent pockets
  - Obtaining extra turns for the agent
  - Leaving empty pockets on the opponent side
- Win rate for Run 1: 33.28%
- Average win rate: 35.04%



Epsilon-Decay

Goal:

- Start with lots of exploration (100%) and decrease over time.
- Caps at 5% exploration



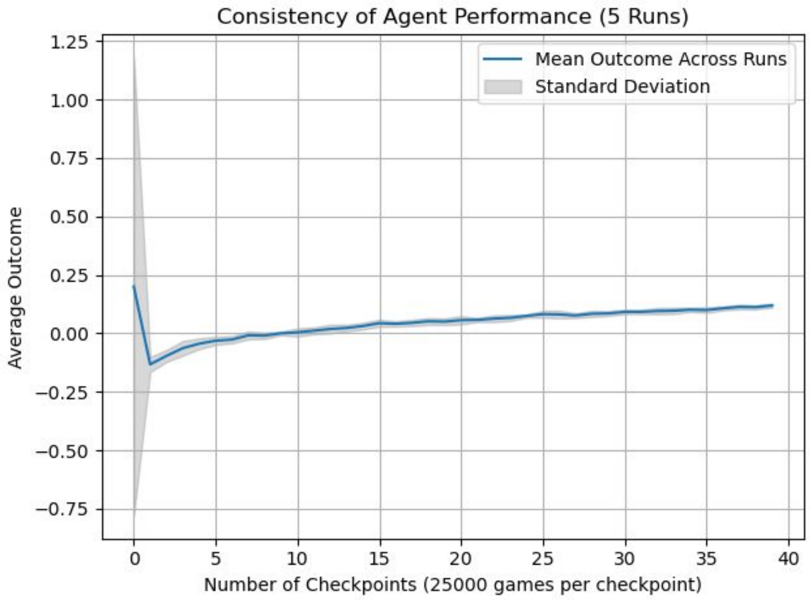
Epsilon-Decay

Results we noticed:

- Increase in performance
- Average Win Rate of 49.66%

What does this mean:

- We are learning and applying Q-values correctly. Led to a good policy
- We avoided premature suboptimal policies

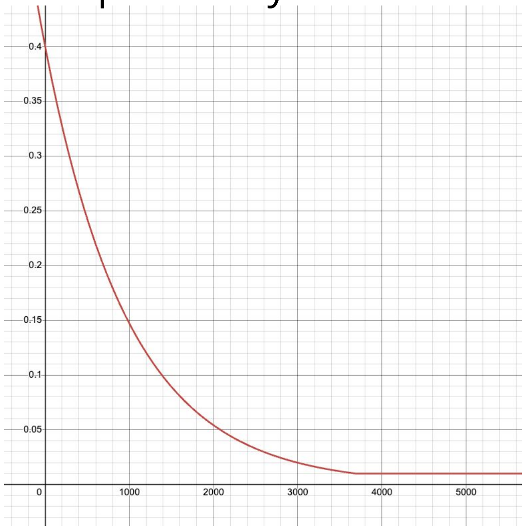


Alpha Decay

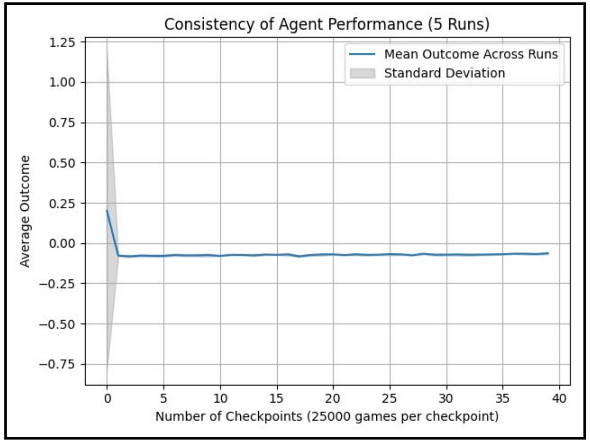
- Alpha: how much new information gets incorporated into Q-table
  - 1: no convergence over time (stochastic)
  - 0: no new info
- Decay factor close to 1
- Min alpha value: 0.01
- Starting alpha: 0.4
- Win rate: 43.3%

Alpha decay function

α



Games played



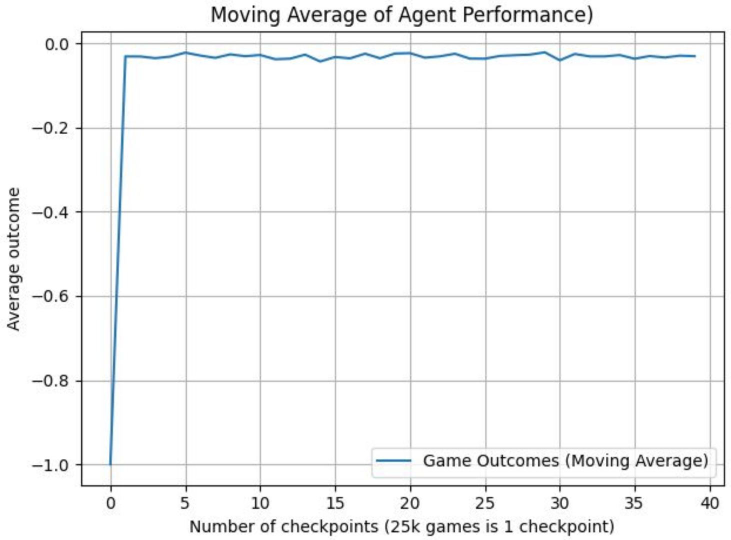
Softmax

Original Model: Epsilon-Greedy

- Simple w/ fixed probabilities, but suboptimal performance
- Continues random exploration even after identifying best actions

Alternative: Softmax

- Probabilistic action section based on Q-values
- Temperature parameter
    - High temperature → more exploration
    - Temperature decay → gradual shift to exploitation
- More sophisticated exploration, but more complex



$$\mathrm{softmax}(x)_i = \frac{e^{\frac{y_i}{T}}}{\Sigma_j^N e^{\frac{y_j}{T}}}$$

Softmax + Epsilon-Greedy

Challenges of Softmax:

- Computationally expensive
- Causes long training times

Solution: Hybrid Approach (Softmax and Epsilon-Greedy)

- Generate random number:
  - If number < epsilon → select random action w/ fixed probability
  - If number ≥ epsilon → select action based on softmax probabilities
- Helped reduce computational cost by limiting frequent softmax calculations
- Did not improve win rate (45.22%)

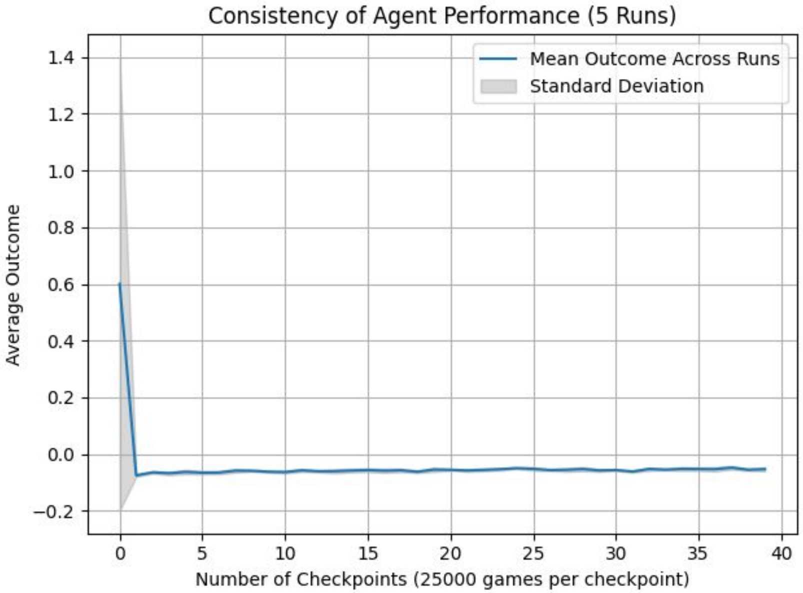
Double Q-Learning

Added a second Q-table to original model

- Used two estimators to reduce overestimations/bias
- Random value decides which table to update

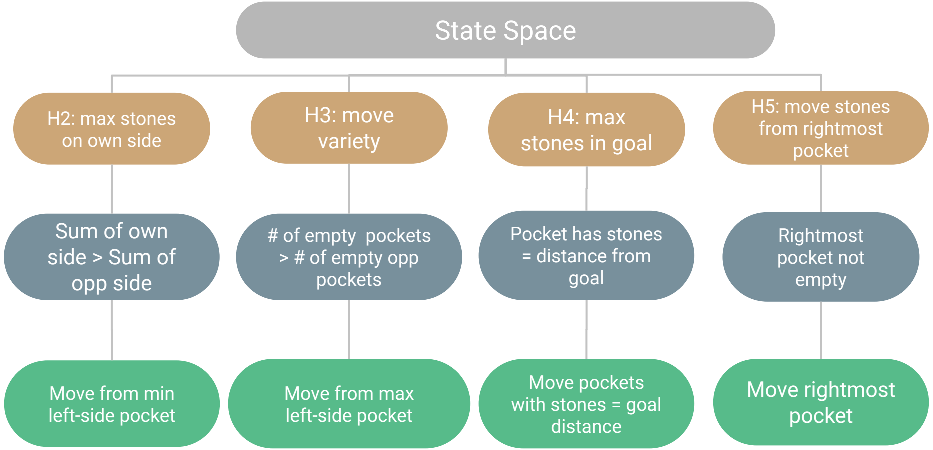
Outcomes:

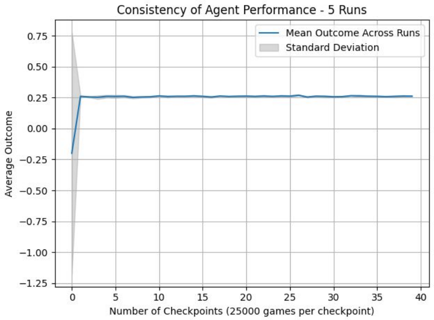
- 46.29% win rate
- Initializing state values to 0 may have unintentionally limited exploration in early stages of training.



State Aggregation

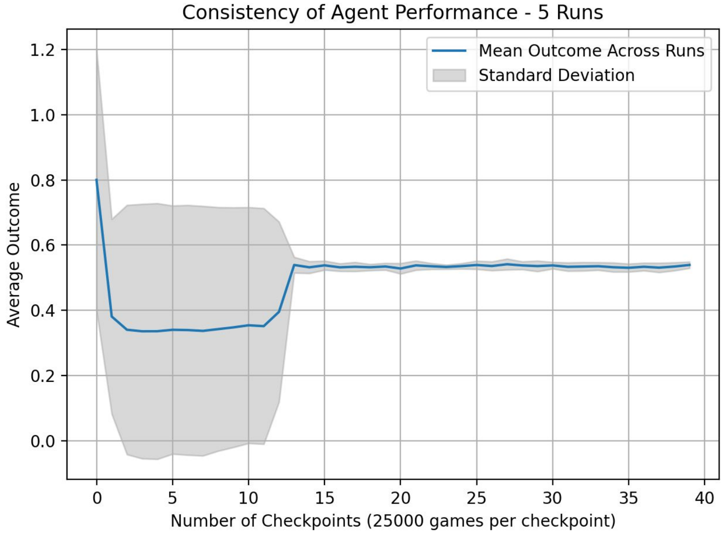
- Meant to facilitate faster training by reducing the observed state space
  - Addresses issue of state complexity
- Categorized logged states into groups of meta-states
  - Meta-states defined according to mancala training heuristics
  - Grouping priority in cases of overlap were based on heuristic strength
    - 5 -> 4 -> 2 -> 3
  - Divilly, Colin & O' Riordan, Colm & Hill, Seamus. (2013).
- State-action associations
- Win Rate:
  - (left) State aggregation only -> avg 60%
  - (right) With softmax adjustments -> avg 72%





Reward Clipping

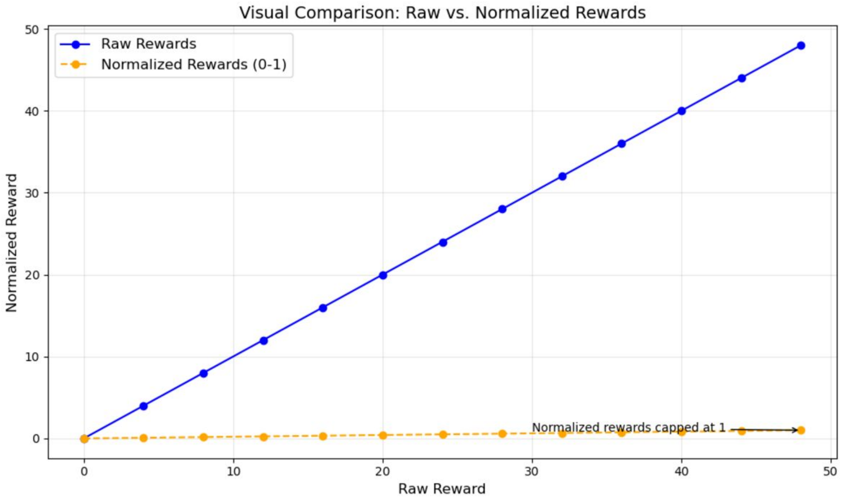
- Restricts rewards to a fixed range
- **Purpose:**
  - Prevents extreme rewards (outliers) from dominating the learning process
  - Stabilizes Q-value updates, ensuring agent does overreact to small or large rewards
- **Issues:**
  - The agent couldn't differentiate between small or large gains, leading to suboptimal performance
  - Agent chooses to make less strategic moves because all rewards are clipped to similar values making it harder to identify valuable actions
  - Had a 4% decrease in win rate after clipping rewards to around [-1,1]



$$Q'(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma max Q'(s',a') - Q(s,a)]$$

Normalization

- Reward normalization scales rewards proportionally, dividing by a maximum value to keep them consistent across actions
- **Purpose:**
  - The agent can focus on the importance of rewards than on its raw magnitude.
  - Makes rewards more consistent and stabilize Q-value updates
- **Issues:**
  - Normalized rewards like 0.02 (1/48) caused minimal updates to Q-values, slowing convergence.
  - Actions with significant advantages like capturing 10 stones vs 1 became less distinguishable.



Scenario	Raw Reward	Normalized Reward	Impact
5 Stones	+5	0.10	Low signal for important move
20 Stones	+20	0.42	Reduced importance of big gains
Winning	+48	1.00	Treated similar to smaller moves

Final Model

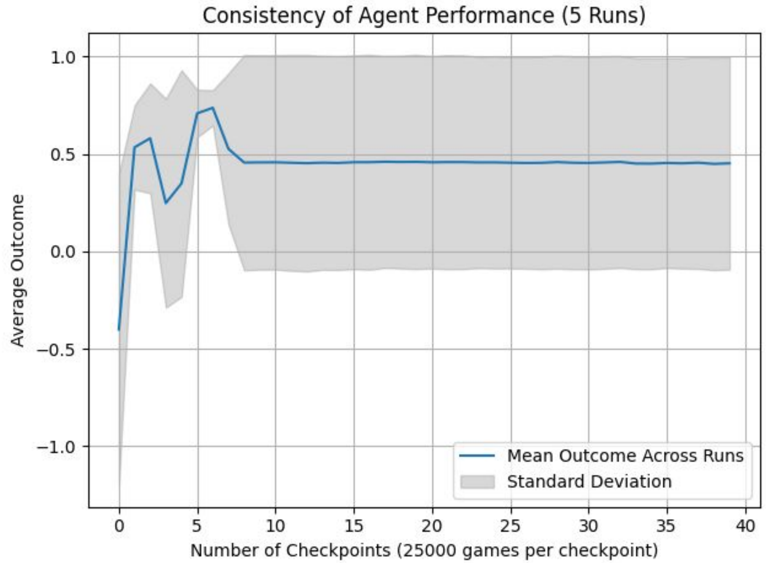
Hybrid Softmax with Epsilon & Alpha-Decay

Parameters:

	Initial	Decay	Minimum
Temperature	1.0	0.999	0.1
Epsilon	1.0	0.999	0.05
Alpha	0.4	0.999	0.01

Results:

- Overall average win rate: 72.59%
    - 4 out of 5 runs: average win rate of 83.42%
    - 1 run: win rate of 29.26%
- Large standard deviation, indicating inconsistent performance



Issues/Limitations

- ★ Model should not play against a random agent
  - Humans do not pick random pockets
  - AlphaGo learned by playing against one of a few agents
- ★ Model still chooses random values 5% of the time when playing humans
- ★ Computational resources: results may be improved with more games
- ★ Testing more alpha, epsilon values could yield higher accuracy
- ★ User interface - hard to follow

Thank You!
Questions?