A Tutorial on Google AlphaGo

Bo An

Slides based on:

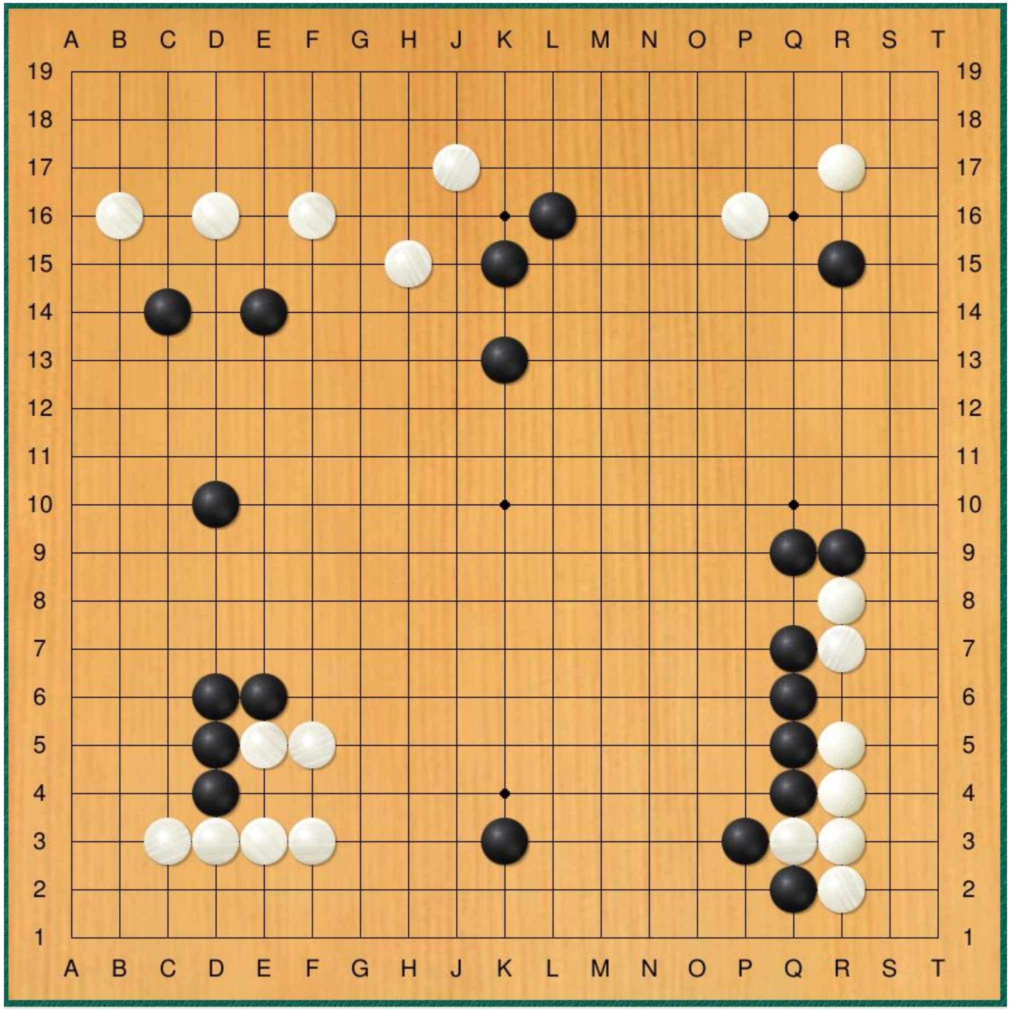
- **Silver, David, et al. "Mastering the game of Go with deep neutral networks and tree search ." Nature 529.7587 (2016): 484-489.**
- **Shane Moon's presentation**

Games in AI

- Ideal test bed for AI research
  - *Clear results*
  - *Clear motivation*
  - *Good challenge*
- Success in search-based approach
  - *chess (1997, Deep Blue)*
  - *and others*
- Not successful in the game of Go
  - *Go is to Chess as Poetry is to Double-entry accounting*
  - *It goes to the core of artificial intelligence, which involves the study of learning and decision-making, strategic thinking, knowledge representation, pattern recognition and, perhaps most intriguingly, intuition*

The game of Go

- A 4,000 years old board game from China
- Standard size 19×19
- Two players, Black and White, place the stones in turns
- Stones can not be moved, but can be captured and taken off
- Larger territory wins



AlphaGo vs European Champion (Fan Hui 2-Dan*)$^{\text{rank}}$

October 5– 9, 2015

- Time limit: 1 hour
- **AlphaGo Wins (5:0)**

AlphaGo vs World Champions

March 9 – 15, 2016 (Lee Sedol)

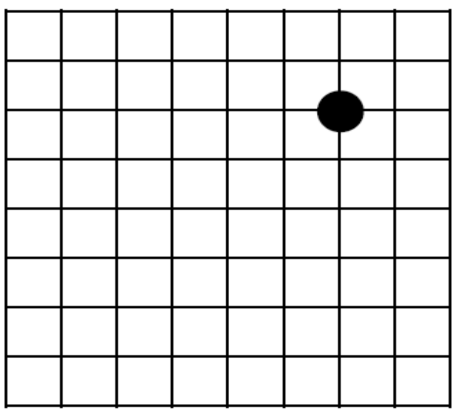
- Time limit: 2 hours
- Venue: Seoul, Four Seasons Hotel
- AlphaGo Wins (4:1)

May 23 – 27, 2017 (Ke Jie)

- Ve
- Al

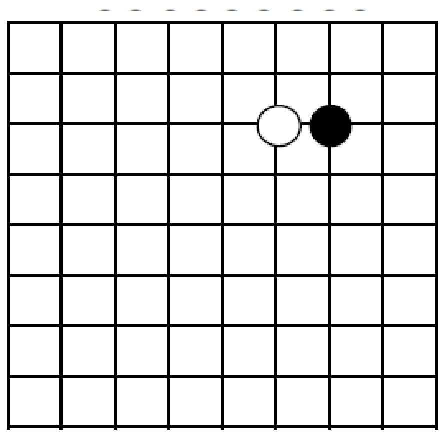
Computer Go AI - Definition

d = 1



=

d = 2



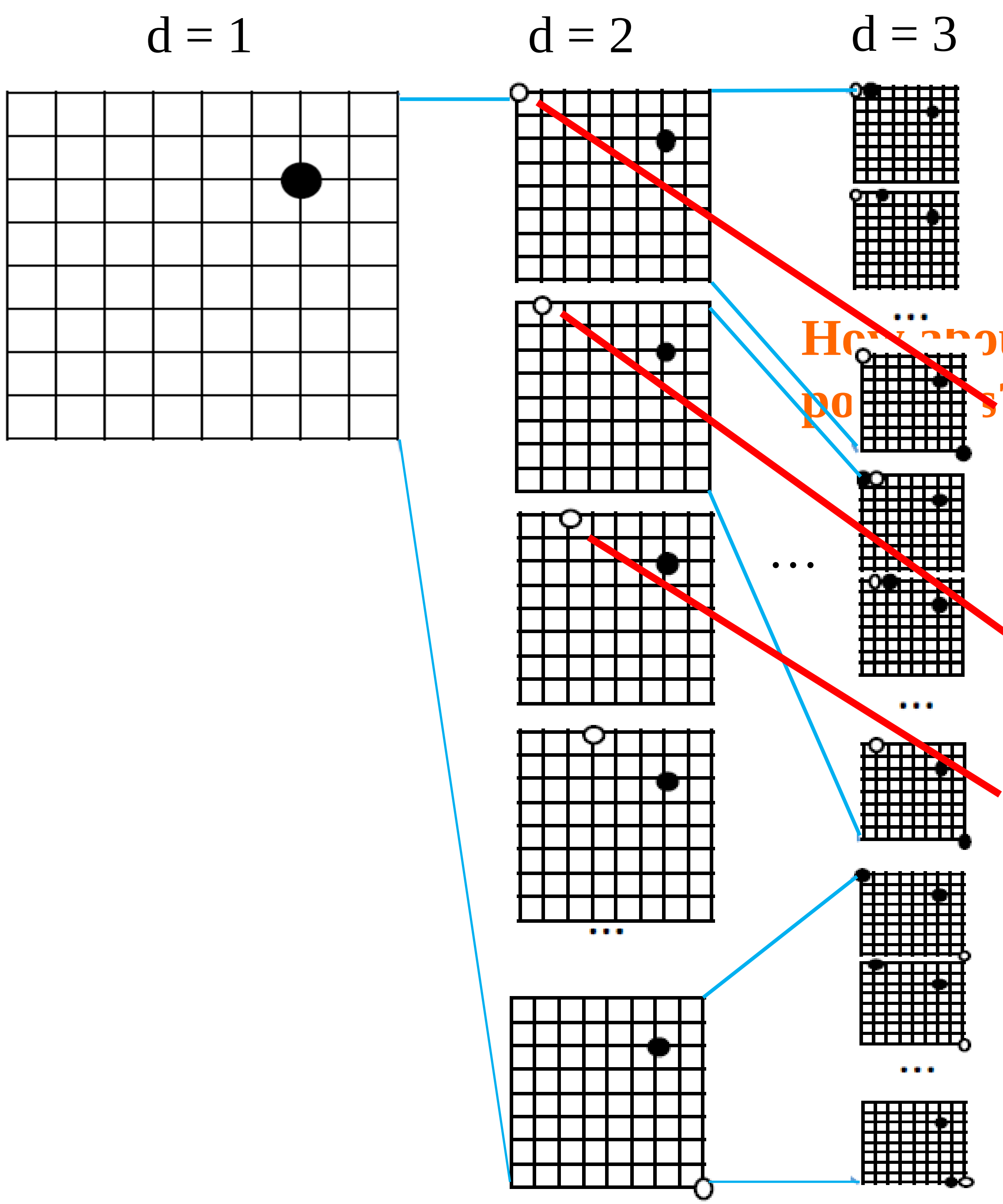
s (state)

e.g. we can represent the board into a matrix-like form

a (action)

Given ***s***, pick the best ***a***

Computer Go AI – An Implementation Idea?



… d = *max D* ***~ 19x19=361***

Process the simulation until the game ends, then report win / lose results

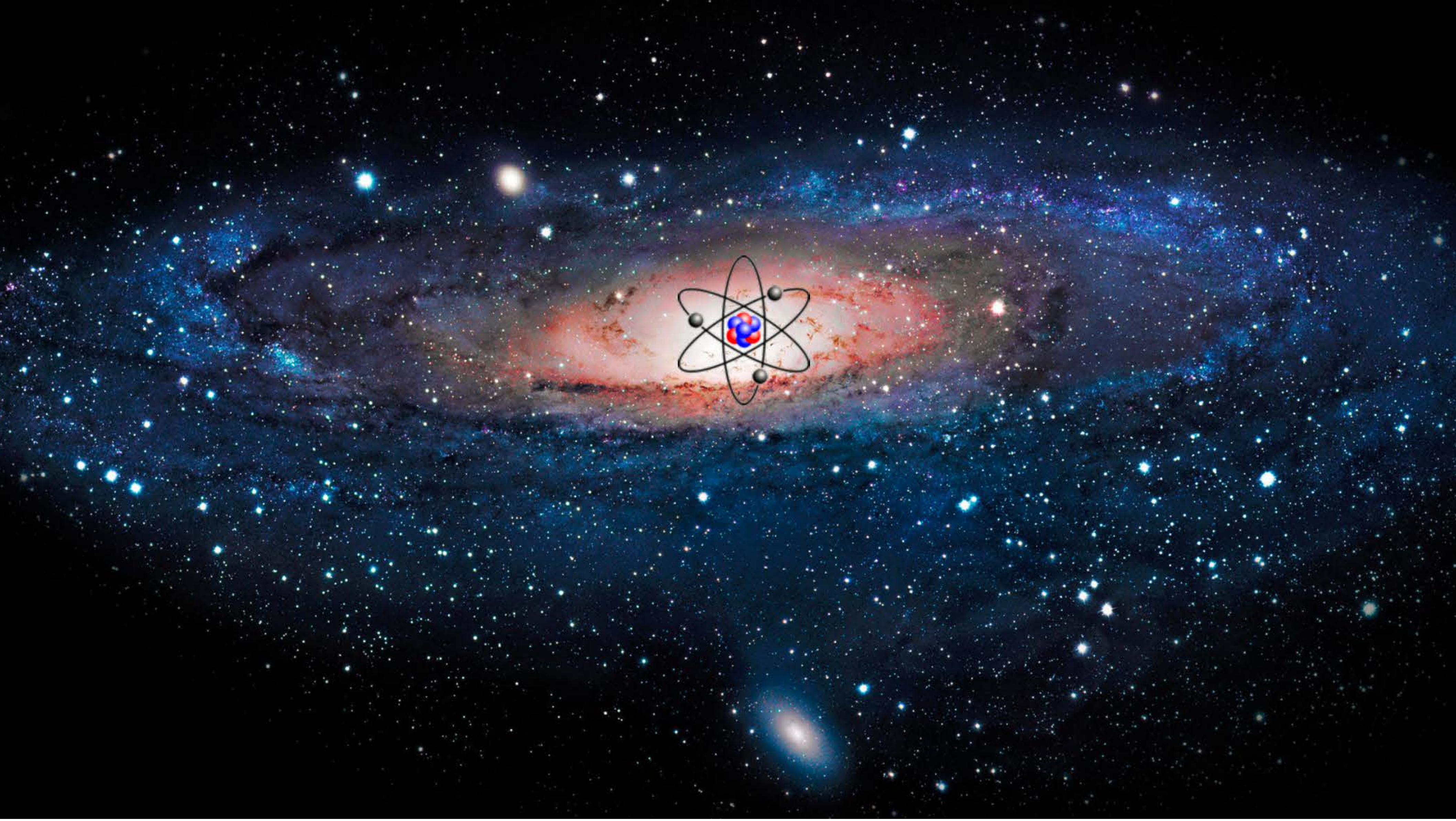
How about simulating all possible board positions?

e.g. it wins 13 times if the next stone gets placed here

Choose the "next action / stone" that has the most win-counts in the full-scale simulation

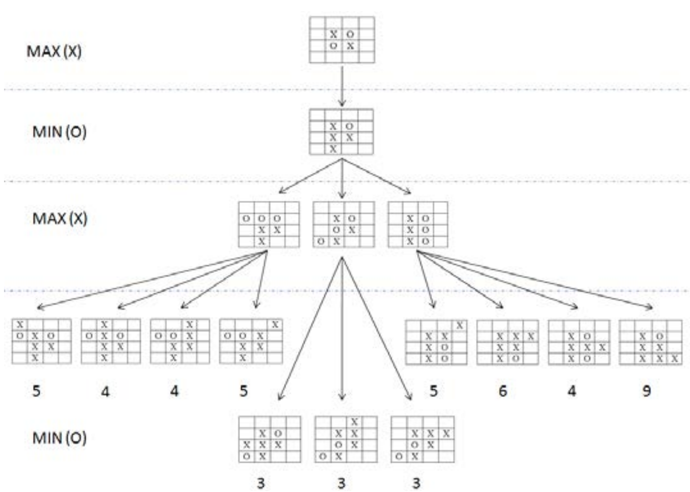
Can also apply Minmax search learned in CZ3005

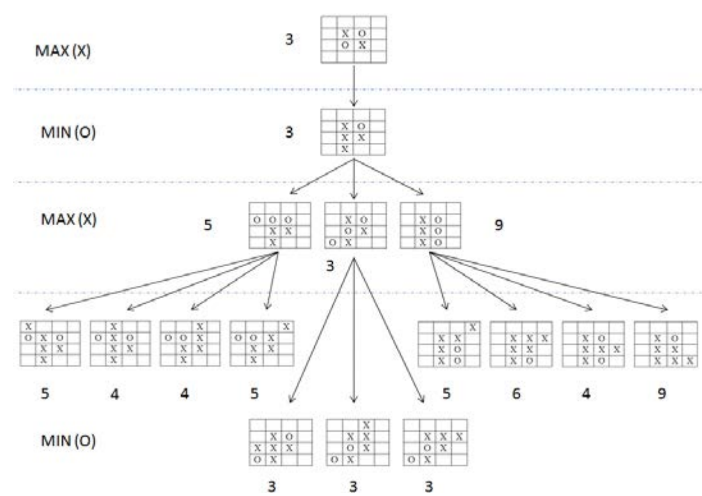
This is NOT possible; it is said the possible configurations of the board exceeds the number of atoms in the universe



AlphaGo: Key Ideas

- Objective: Reduce search space without sacrificing quality
- Key Idea 1: Take advantage of human top players’ data
  - *Deep learning*
- Key Idea 2: Self-play
  - *Reinforcement learning*
- Key Idea 3: Looking ahead
  - *Monte Carlo tree search*
  - *We learned Minimax search with evaluation functions*

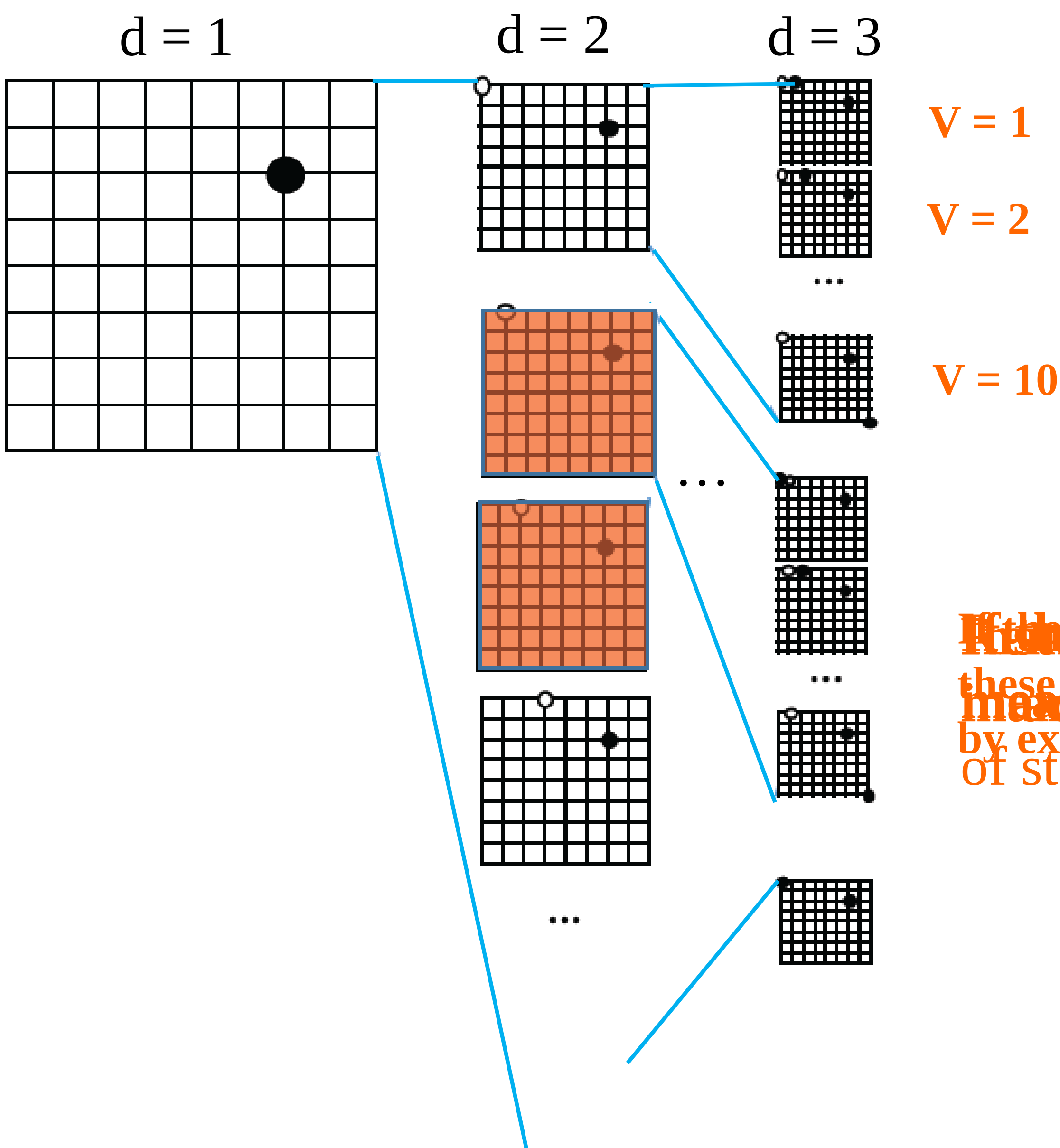




Reducing Searching Space

1. Reducing "action candidates" (Breadth Reduction)

2. Position evaluation ahead of time (Depth Reduction)



… d = *maxD*

Win?

Lose?

If there is a model that can tell you that these moves are not common / probable (e.g. by experts, etc.)…

Remove these from search candidates in advance (breadth reduction)

If there is a function that can measure V(s): "board evaluation of state *s*"

Instead of simulating until the maximum depth (depth reduction)

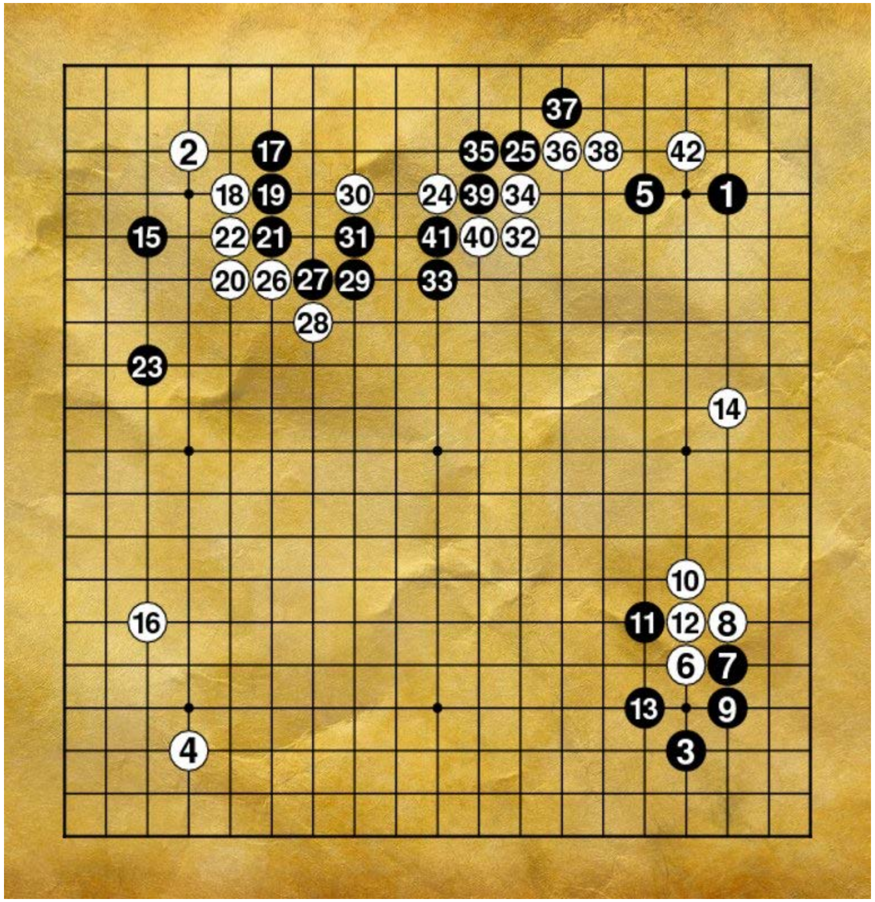
1. Reducing “action candidates”

Learning: P (next action | current state)

= P (a | s)

1. Reducing “action candidates”

(1)Imitating expert moves (supervised learning)



<u>Current State</u> <u>Next State</u>

s1 → Prediction Model → s2

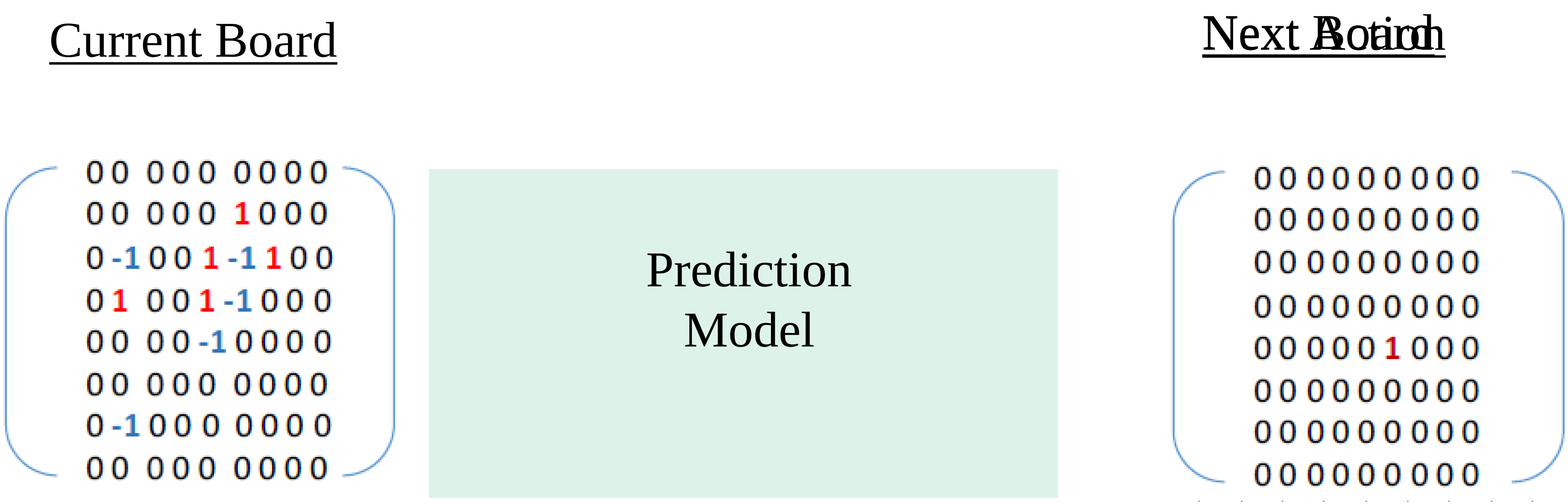
s2 → Prediction Model → s3

s3 → Prediction Model → s4

Data: Online Go Experts (5 ~ 9 dan)
160K games, 30M board positions

1. Reducing “action candidates”

(1)Imitating expert moves (supervised learning)

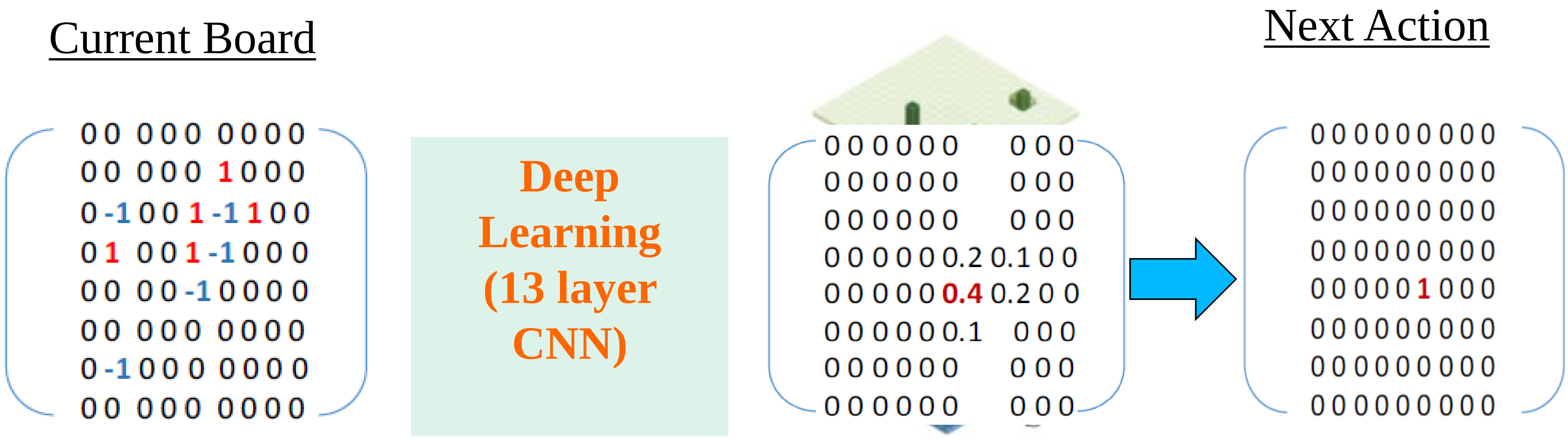


s $\qquad f: s \longrightarrow a$ $\qquad a$

There are 19×19 = 361 possible actions (with different possibilities)

1. Reducing “action candidates”

(1)Imitating expert moves (supervised learning)

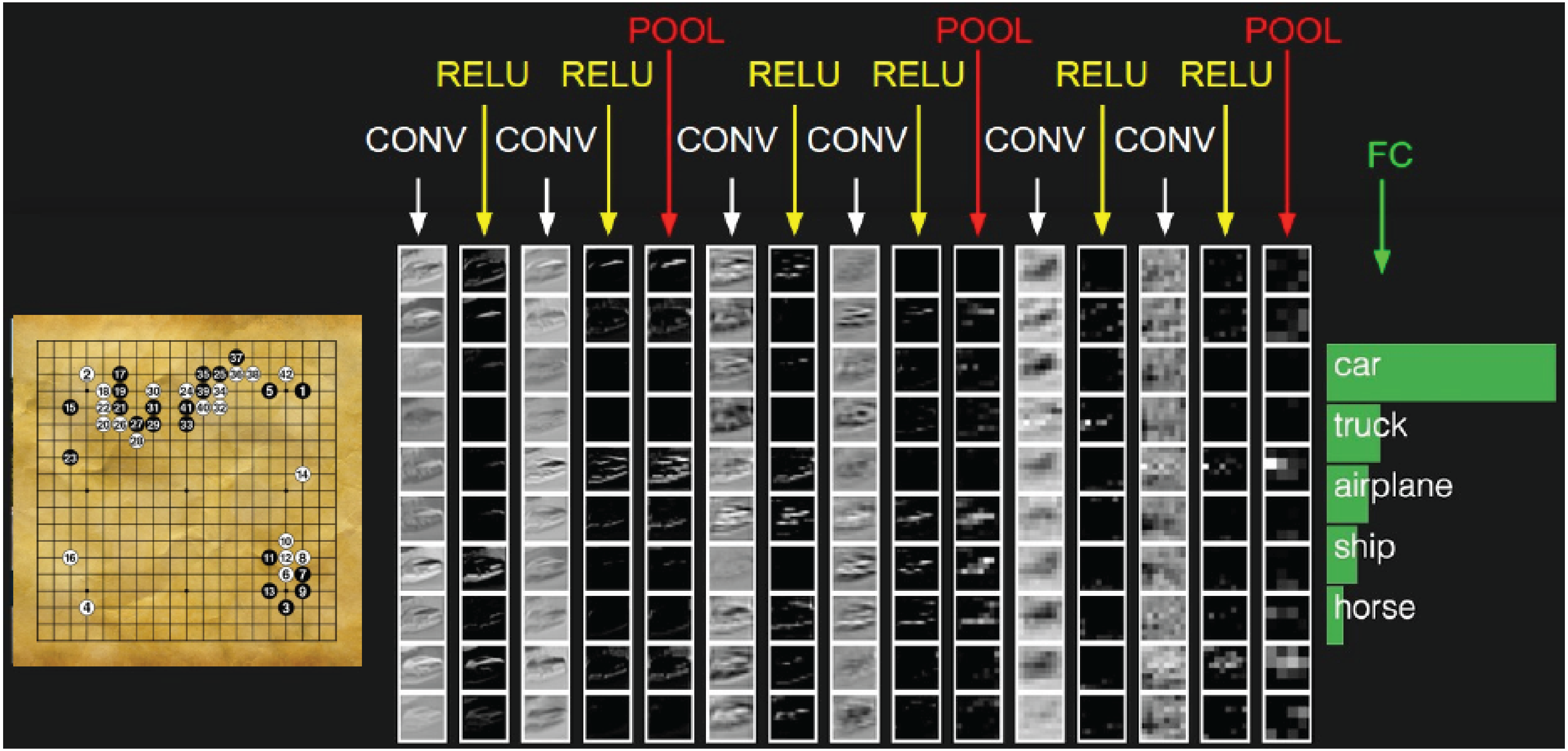


s $\quad$ $g: s \longrightarrow \mathbf{p}(a|s)$ $\quad$ $\mathbf{p}(a|s)$ $\quad$ **argmax** $\quad$ a

Convolutional Neural Network (CNN)

Go: abstraction is the key to win

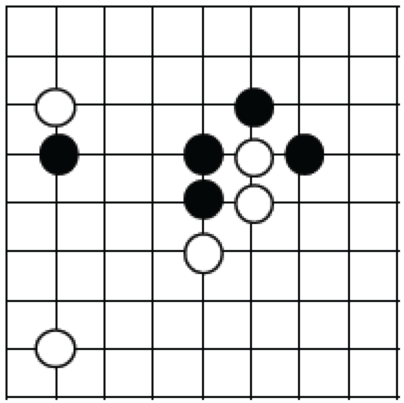
CNN: abstraction is its *forte*



1. Reducing "action candidates"

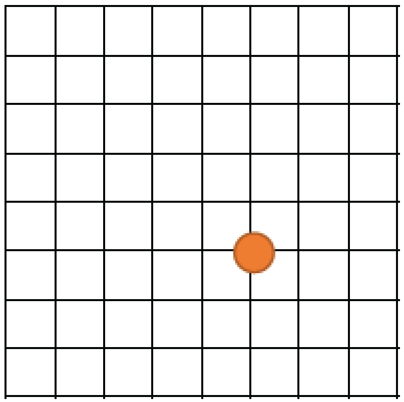
(1)Imitating expert moves (supervised learning)

Current Board



Expert Moves Imitator Model
(w/CNN)

Next Action



Training: $$\Delta\sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial \sigma}$$

1. Reducing “action candidates”

(2)Improving through self-plays (reinforcement learning)

improving by playing against itself

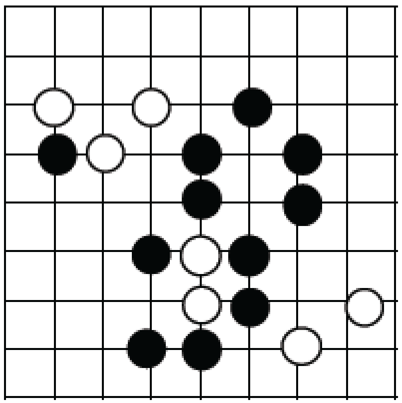
Expert Moves Imitator Model (w/CNN)	VS	Expert Moves Imitator Model (w/CNN)

Return: board positions, win/lose info

1. Reducing "action candidates"

(2)Improving through self-plays (reinforcement learning)

Board position



Expert Moves Imitator Model
(w/CNN)

win/loss

Win
Loss
z = +1
z = -1

Training: $\Delta\rho \propto \frac{\partial \log p_\rho(a_t|s_t)}{\partial \rho} z_t \,.$

1. Reducing “action candidates”

(2)Improving through self-plays (reinforcement learning)

older models vs. newer models

Expert Moves Imitator Model

VS

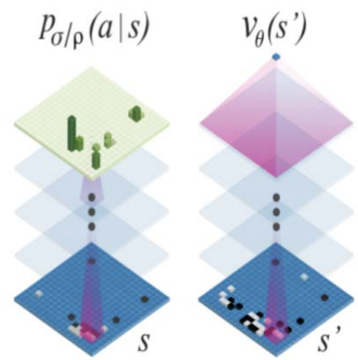
Updated Model ver 1,000,000

It uses the same topology as the expert moves imitator model, and just uses the updated parameters

Return: board positions, win/lose info

The final model wins 80% of the time when playing against the first model

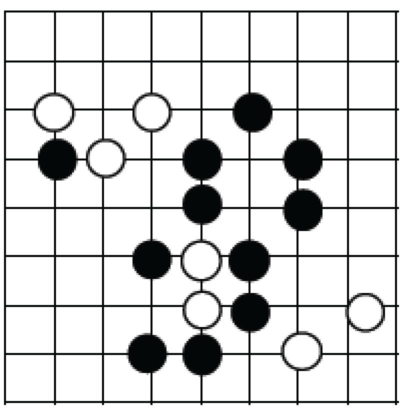
2. Board Evaluation



Adds a regression layer to the model
Predicts values between 0~1
Close to 1: a good board position
Close to 0: a bad board position

Board position

Win / Lose



Updated Model ver 1,000,000

Value Prediction Model (Regression)

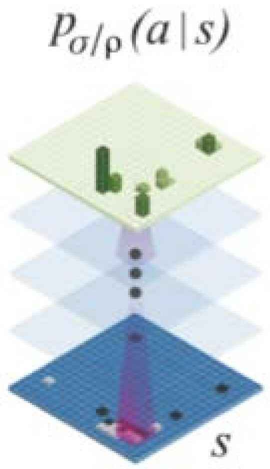
Win (0 / 1)

Training: $$\Delta\theta \propto \frac{\partial v_\theta(s)}{\partial\theta}(z - v_\theta(s))$$

Reducing Search Space

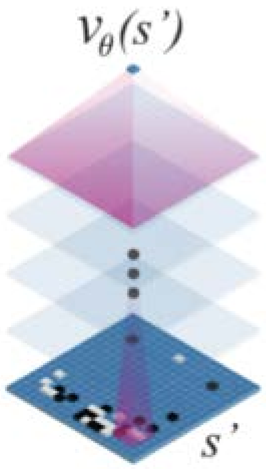
1. Reducing “action candidates” (Breadth Reduction)

Policy Network



2. Board Evaluation (Depth Reduction)

Value Network



Looking Ahead (Monte Carlo Search Tree)

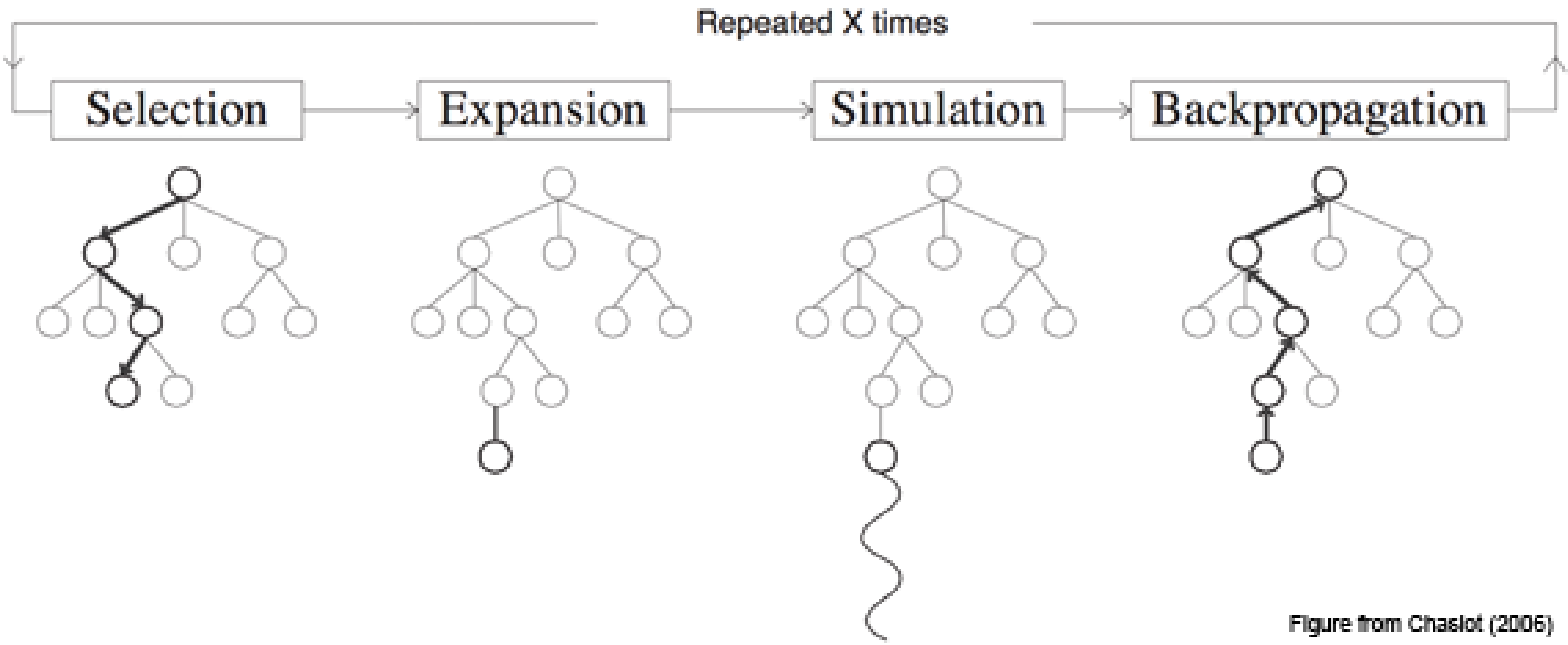
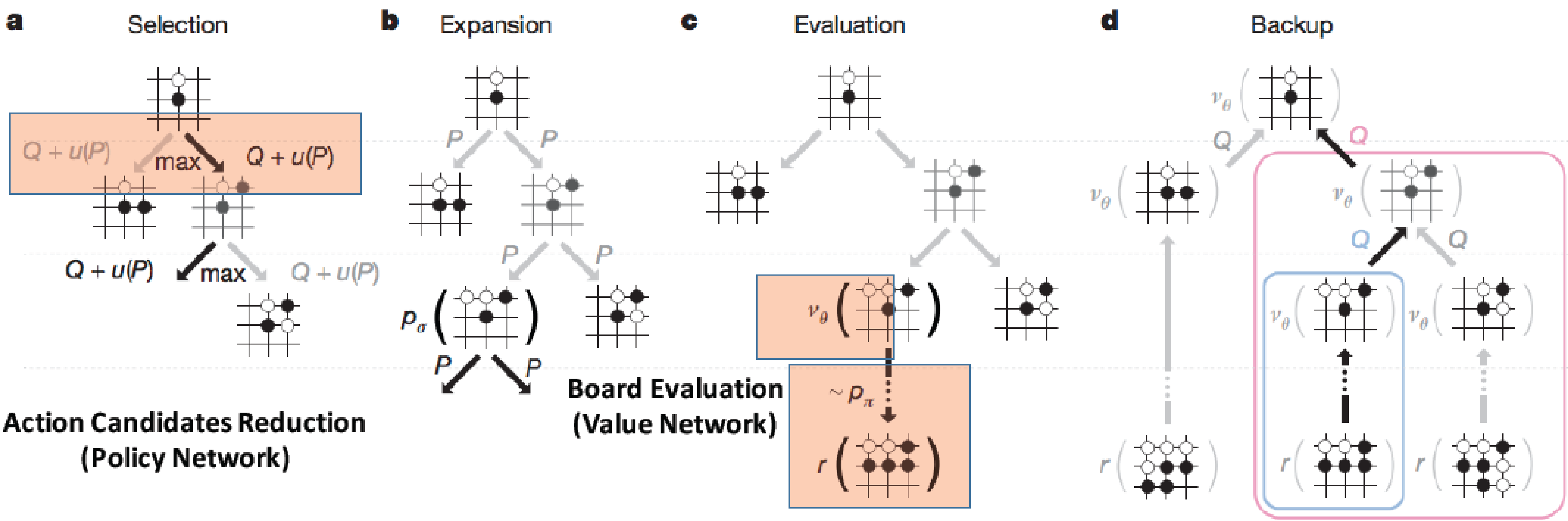


Figure from Chaslot (2006)

Looking Ahead (Monte Carlo Search Tree)

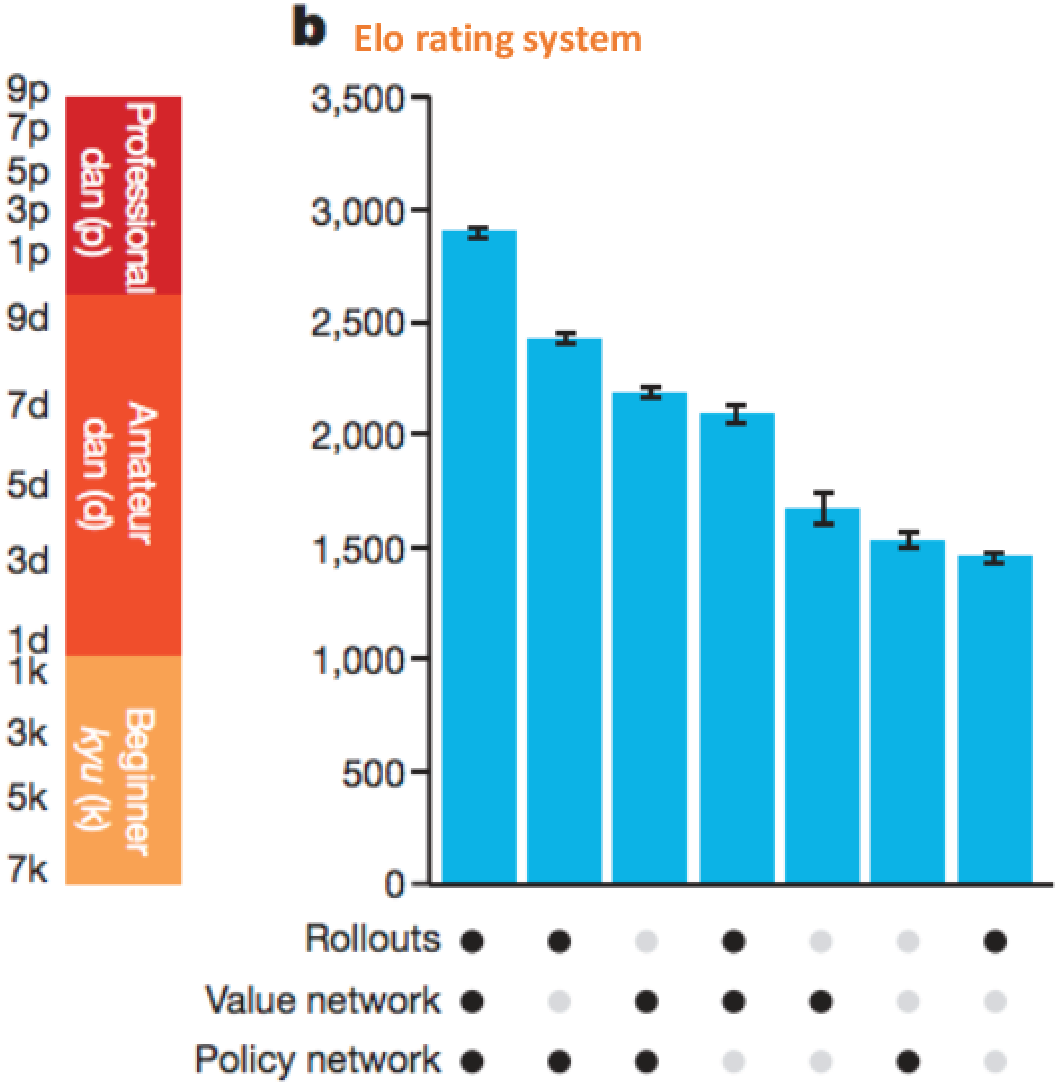


Action Candidates Reduction (Policy Network)

Board Evaluation (Value Network)

(Rollout): Faster version of estimating p(a|s)
→ uses shallow networks (3 ms → 2µs)

Results



Performance with different combinations of AlphaGo components

AlphaGo

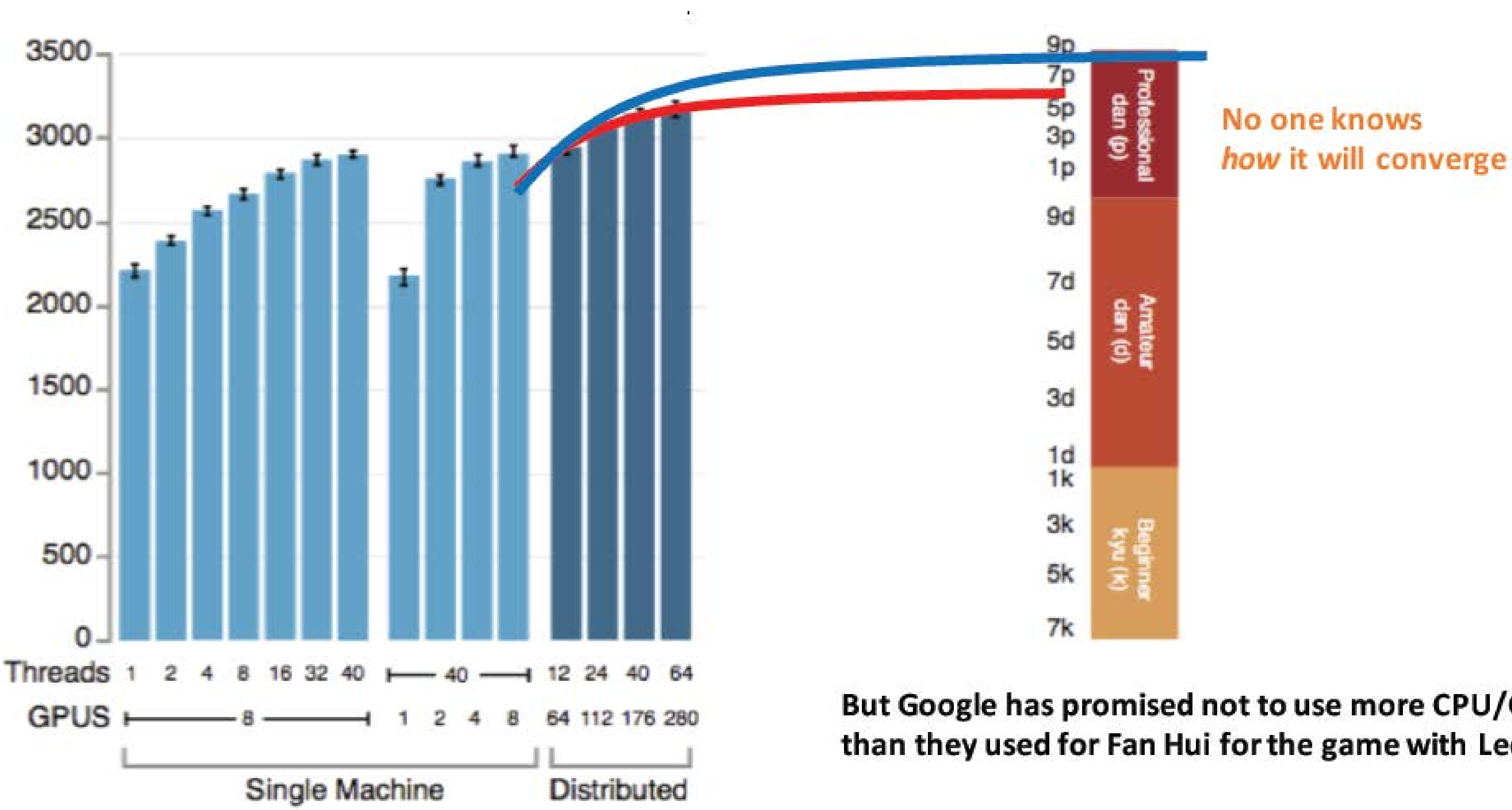
Lee Sedol 9-dan vs AlphaGo Energy Consumption

Lee Sedol	AlphaGo
- Recommended calories for a man per day : ~2,500 kCal - Assumption: Lee consumes the entire amount of per-day calories in this one game 2,500 kCal * 4,184 J/kCal ~= 10M [J]	- Assumption: CPU: ~100 W, GPU: ~300 W - **1,202** CPUs, **176** GPUs 170,000 J/sec * 5 hr * 3,600 sec/hr ~= 3,000M [J]

A very, very tough calculation;)

AlphaGo

Taking CPU / GPU resources to virtually infinity



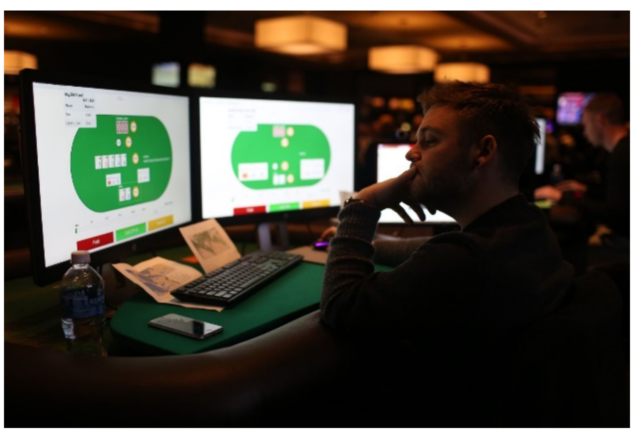
But Google has promised not to use more CPU/GPUs than they used for Fan Hui for the game with Lee

Libratus vs World Champions

The first AI to defeat top human poker players

January 11 to 31, 2017

- Venue: Pittsburgh
- 120,000 hands



Architecture of Libratus

Abstraction
(offline)

- action abstraction
- card abstraction
- took the game size from 10^{161} to 10^{12}

Equilibrium Finding
(offline)

- CFR
- CFR^{+}
- Monte Carlo CFR

Decomposition and Subgame Refinement
(offline)

- endgame solving
- subgame re-solving
- max-margin subgame refinement