

# AlphaGo

How it works

Presenter: Shane (Seungwhan) Moon

PhD student

Language Technologies Institute, School of Computer Science  
Carnegie Mellon University

# AlphaGo vs European Champion (Fan Hui 2-Dan<sup>\*</sup>)

rank



**October 5 – 9, 2015**

**<Official match>**

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

# AlphaGo vs World Champion (Lee Sedol 9-Dan)



**March 9 – 15, 2016**

**<Official match>**

- Time limit: 2 hours

**Venue: Seoul, Four Seasons Hotel**



# Lee Sedol

[wiki](#)

# Computer Go AI?



CONSERVATION

SONGBIRDS  
À LA CARTE

*Illegal harvest of millions  
of Mediterranean birds*

PAGE 452

RESEARCH ETHICS

SAFEGUARD  
TRANSPARENCY

*Don't let openness backfire  
on individuals*

PAGE 459

POPULAR SCIENCE

WHEN GENES  
GOT 'SELFISH'

*Dawkins's calling  
card 40 years on*

PAGE 462

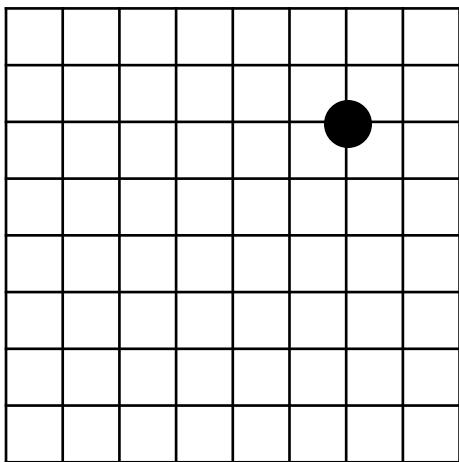
NATURE ASIA.COM

28 January 2016

Vol. 529, No. 7587

# Computer Go AI – Definition

$d = 1$



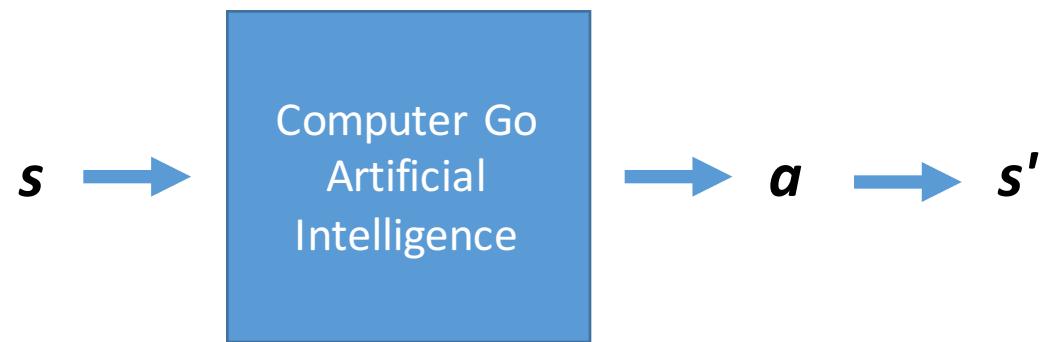
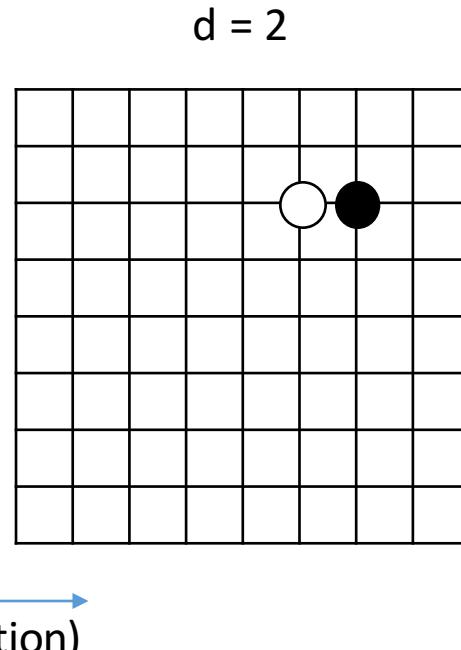
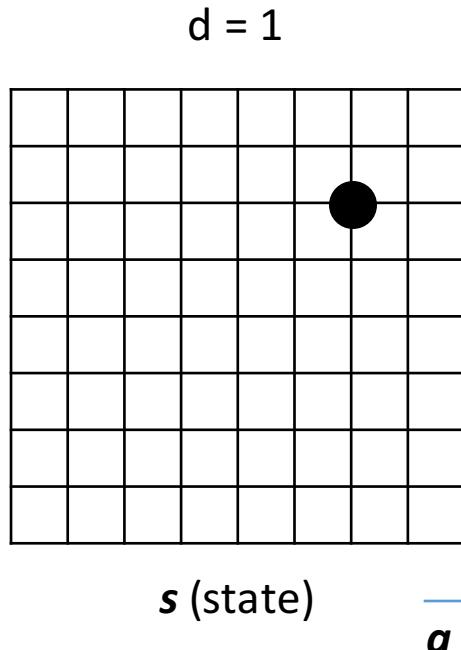
$s$  (state)

$$= \begin{array}{c} 000000000 \\ 000000000 \\ 000000\textcolor{red}{1}00 \\ 000000000 \\ 000000000 \\ 000000000 \\ 000000000 \\ 000000000 \end{array}$$

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

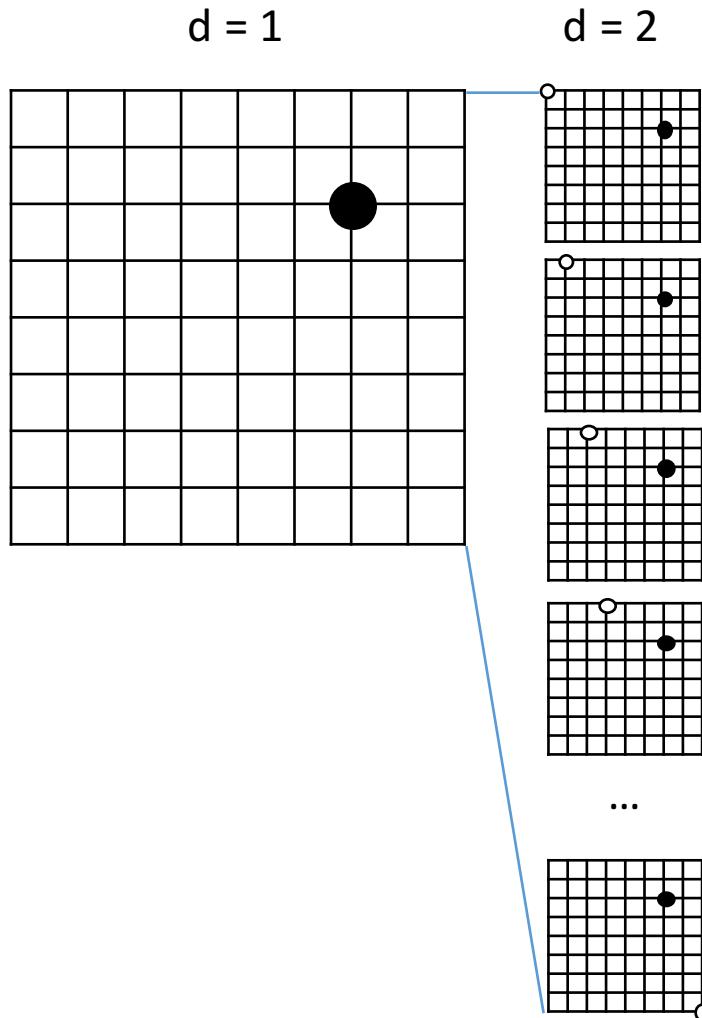
\* The actual model uses other features than board positions as well

# Computer Go AI – Definition



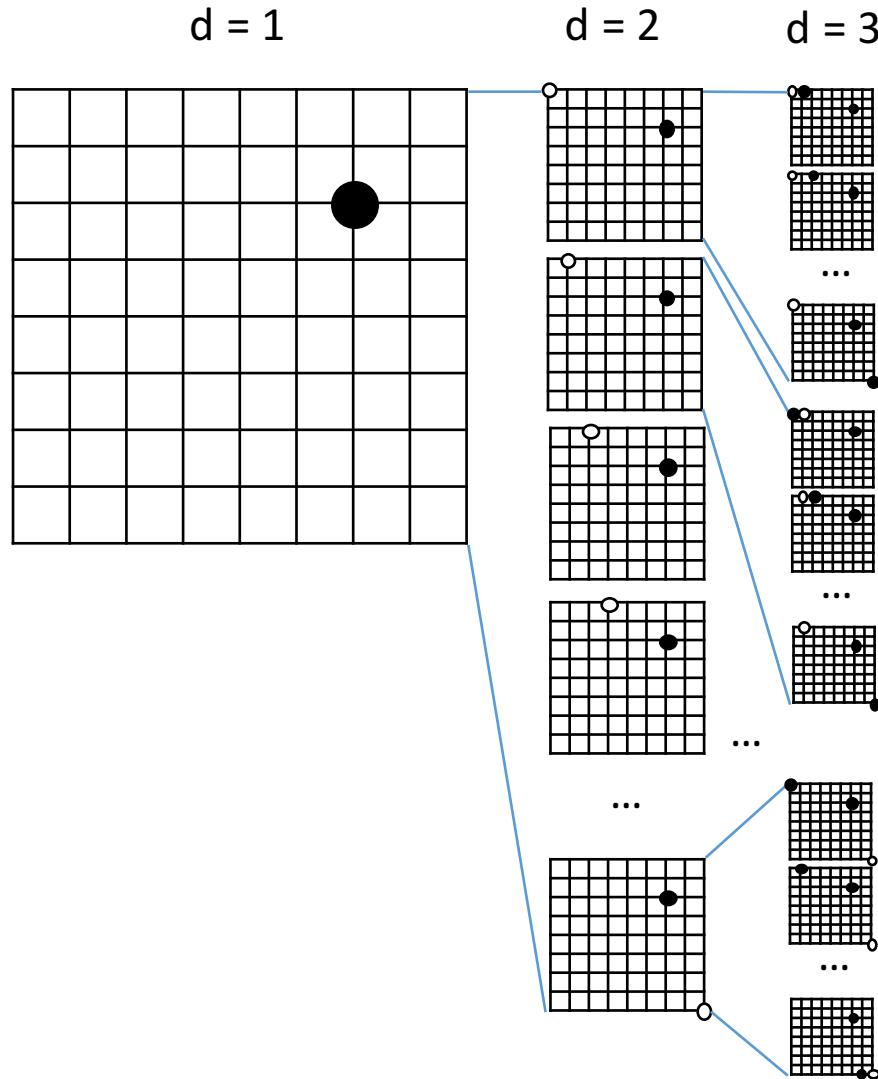
Given  $s$ , pick the best  $a$

# Computer Go AI – An Implementation Idea?

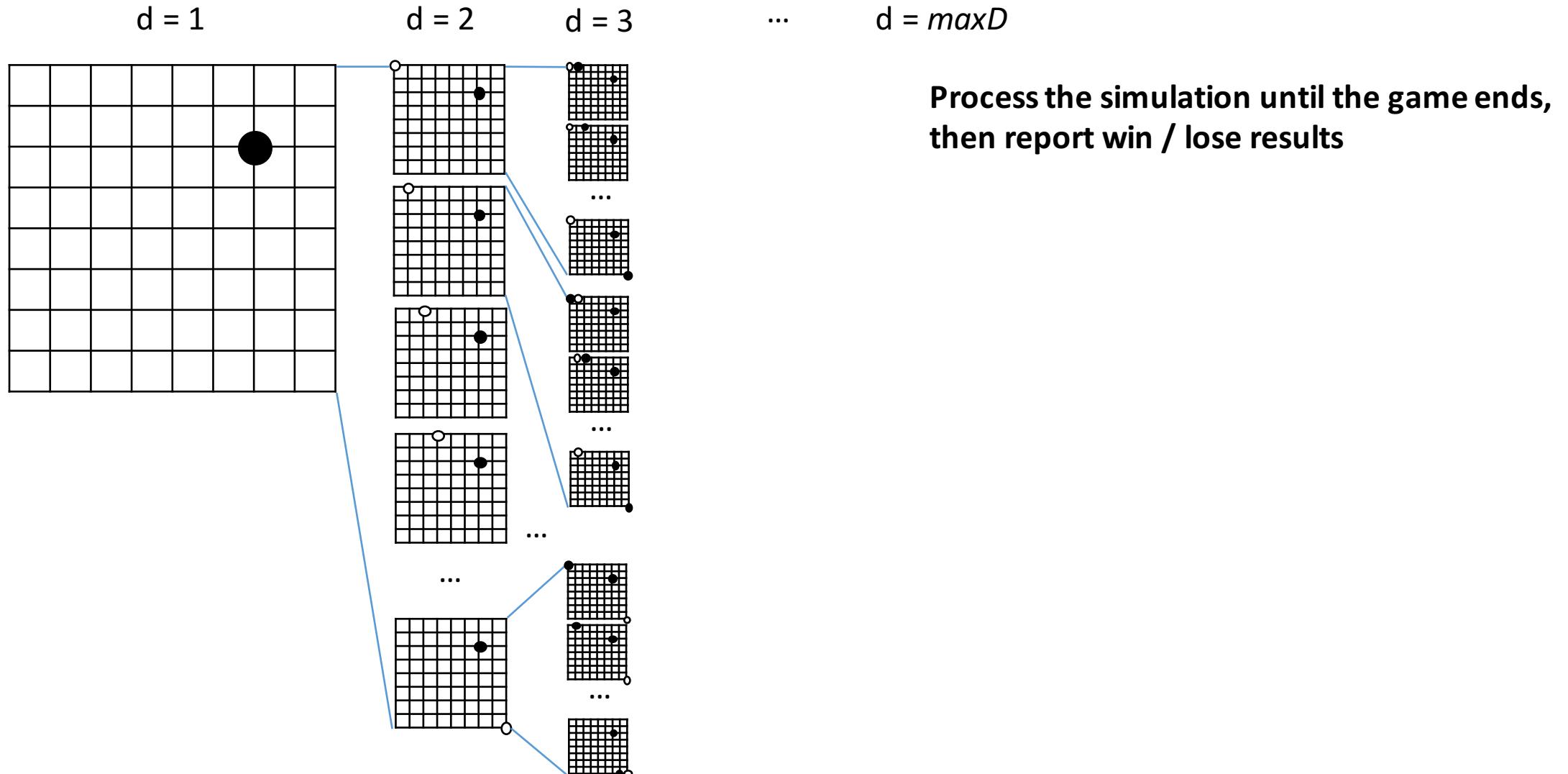


How about simulating all possible board positions?

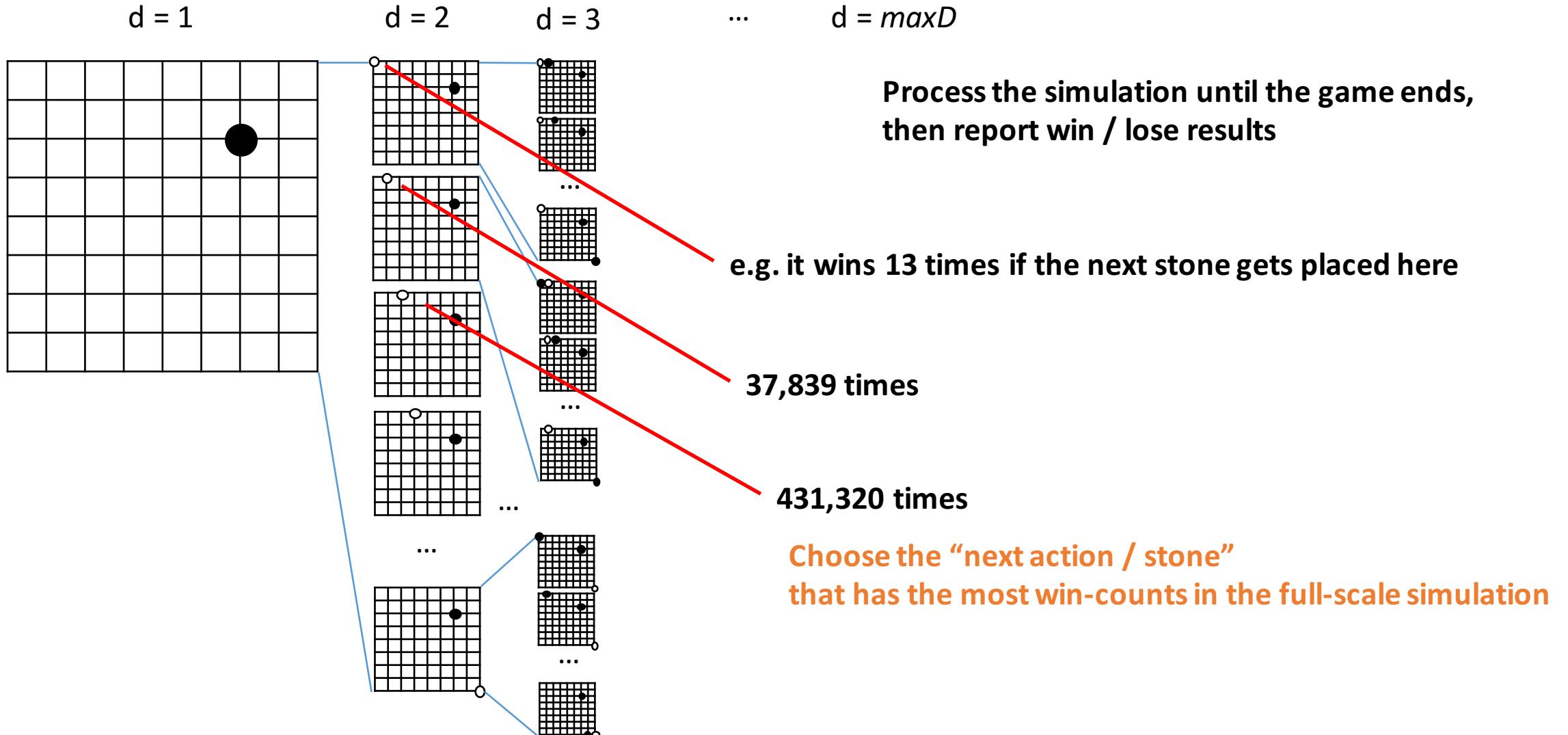
# Computer Go AI – An Implementation Idea?



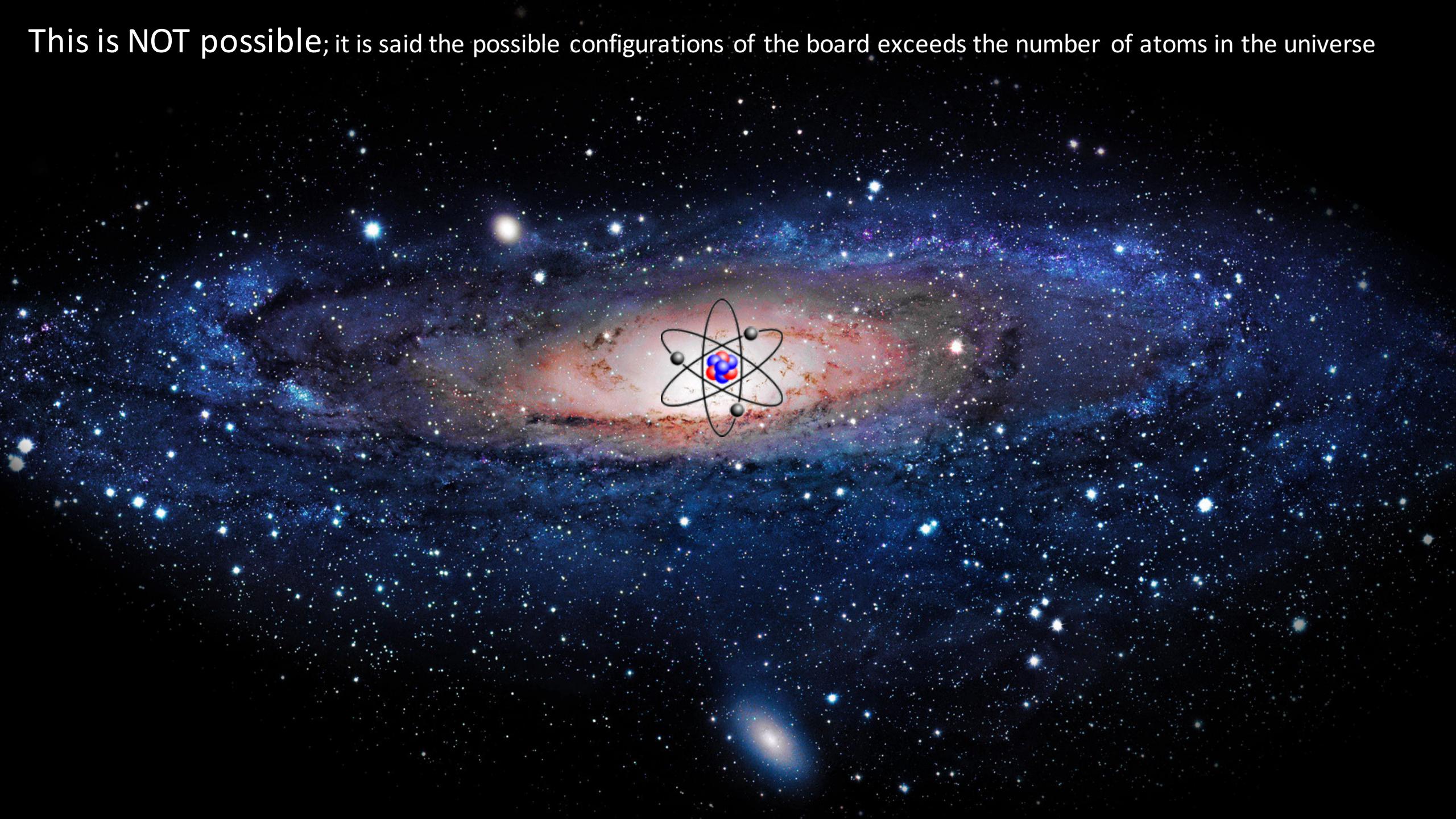
# Computer Go AI – An Implementation Idea?



# Computer Go AI – An Implementation Idea?



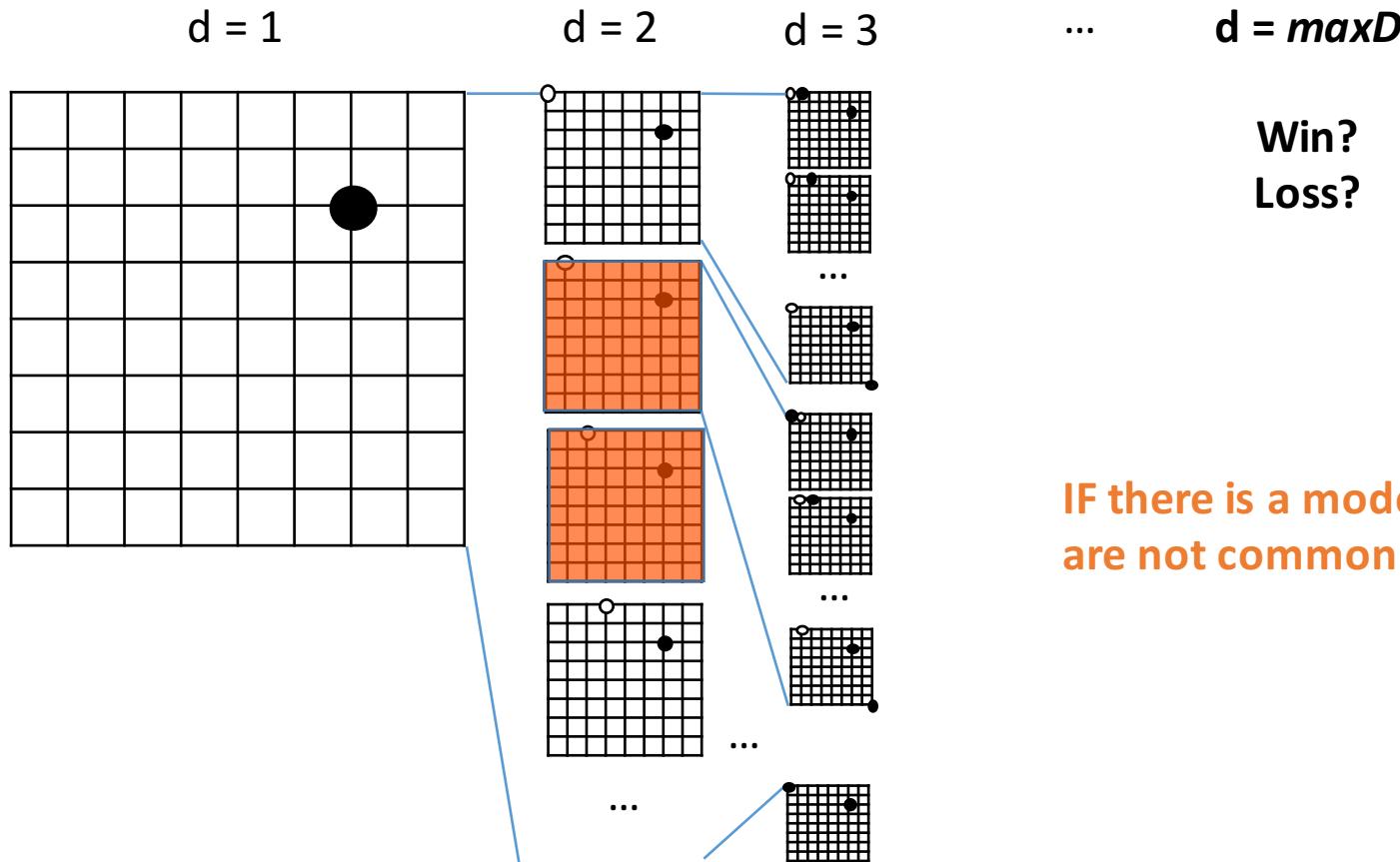
This is NOT possible; it is said the possible configurations of the board exceeds the number of atoms in the universe



# Key: To Reduce Search Space

# Reducing Search Space

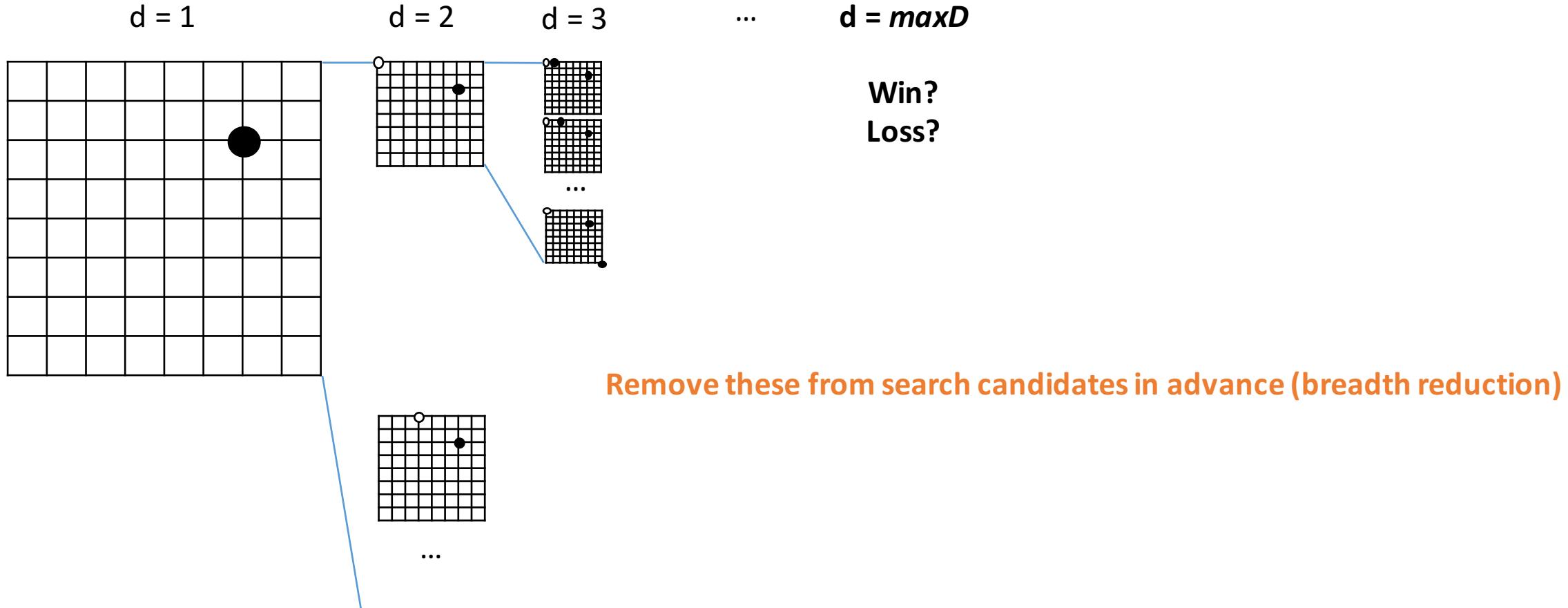
## 1. Reducing “action candidates” (Breadth Reduction)



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

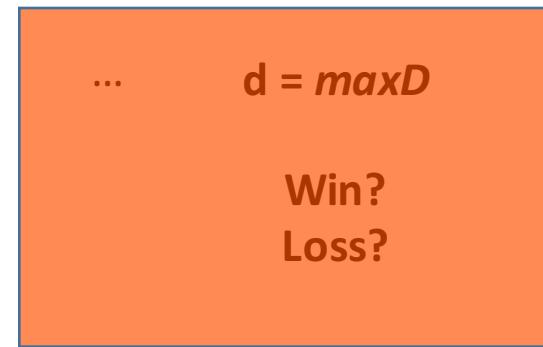
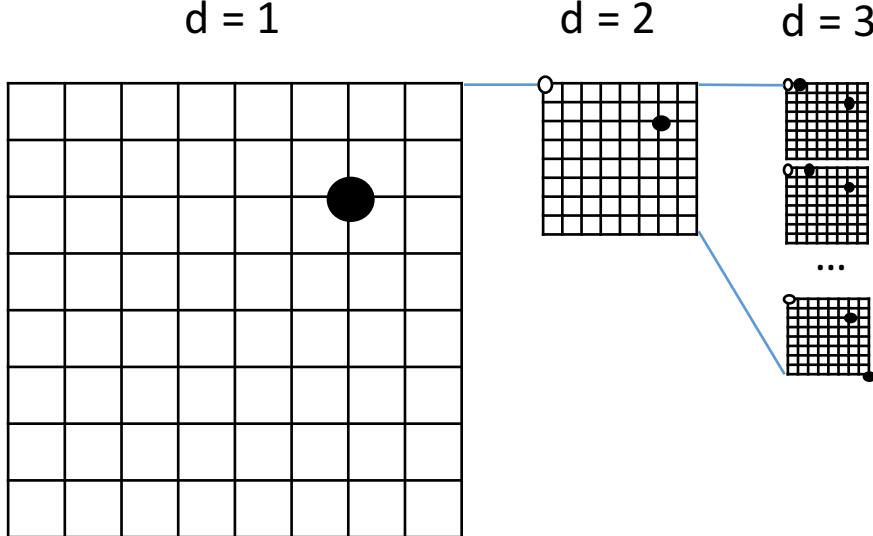
# Reducing Search Space

## 1. Reducing “action candidates” (Breadth Reduction)



# Reducing Search Space

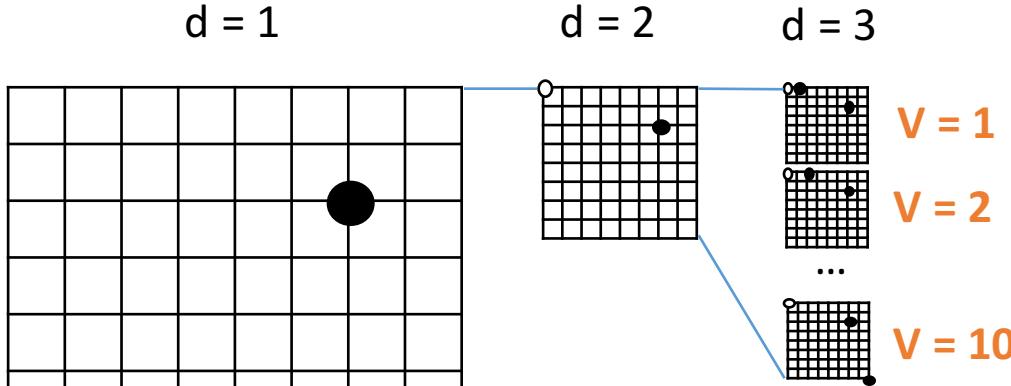
## 2. Position evaluation ahead of time (Depth Reduction)



Instead of simulating until the maximum depth ..

# Reducing Search Space

## 2. Position evaluation ahead of time (Depth Reduction)



IF there is a function that can measure:  
 $V(s)$ : “board evaluation of state  $s$ ”

# Reducing Search Space

- 1. Reducing “action candidates” (Breadth Reduction)**
  
  
  
  
  
  
  
  
- 2. Position evaluation ahead of time (Depth Reduction)**

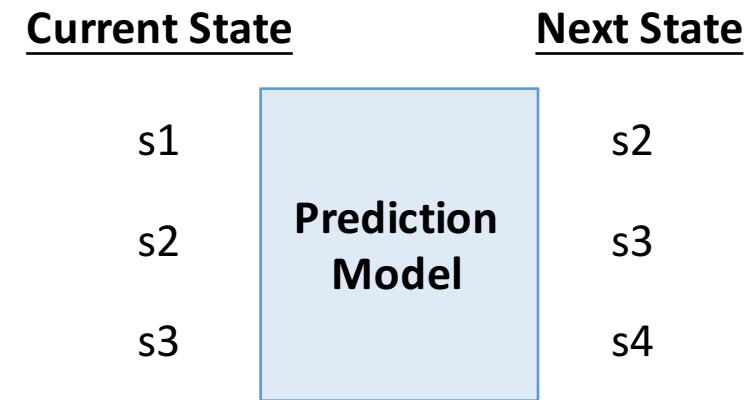
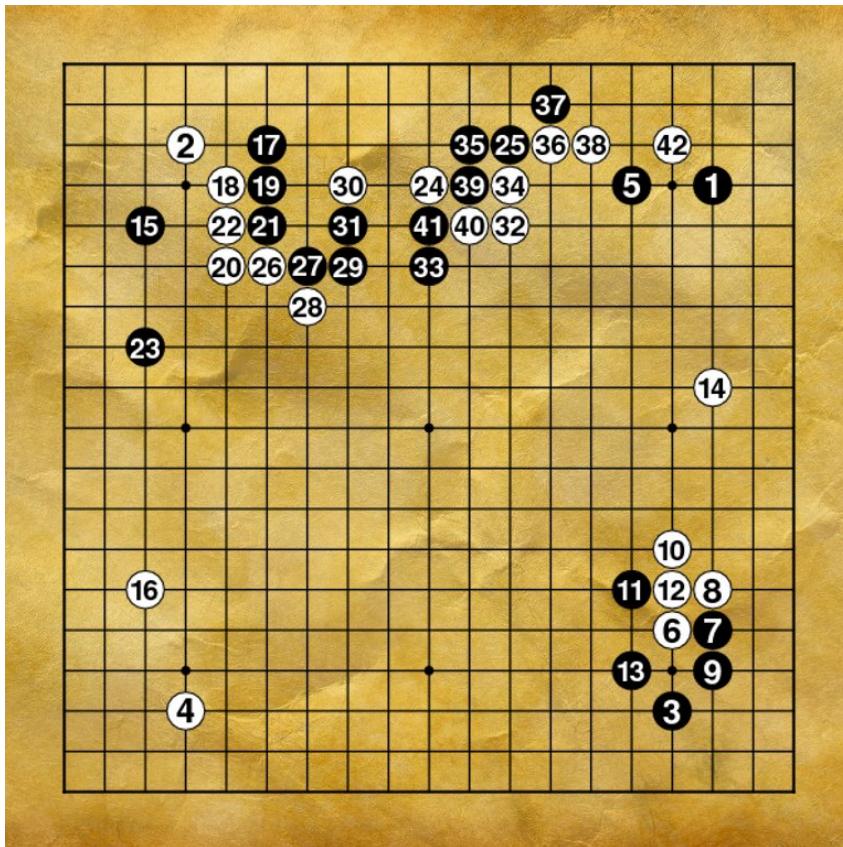
# 1. Reducing “action candidates”

Learning:  $P(\text{next action} \mid \text{current state})$

$$= P(a \mid s)$$

# 1. Reducing “action candidates”

## (1) Imitating expert moves (supervised learning)

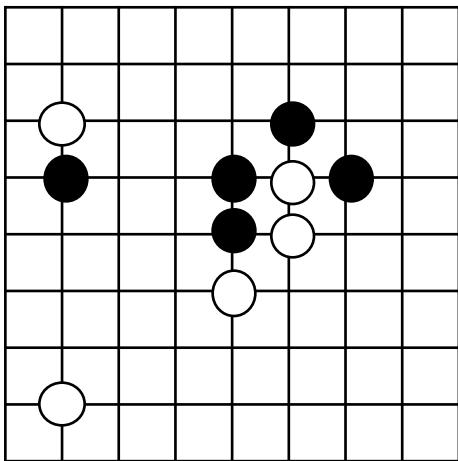


**Data:** Online Go experts (5~9 dan)  
160K games, 30M board positions

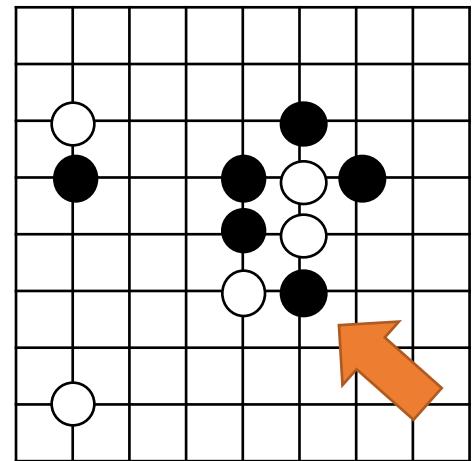
# 1. Reducing “action candidates”

(1) Imitating expert moves (supervised learning)

Current Board



Next Board

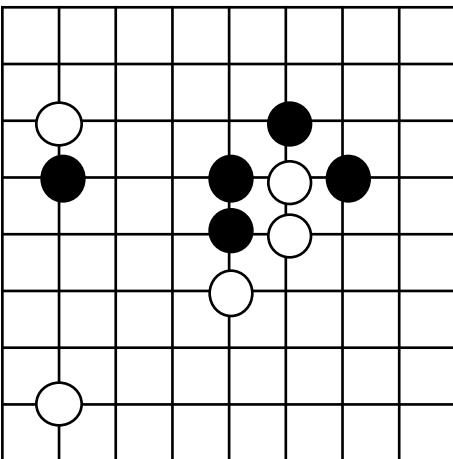


**Prediction Model**

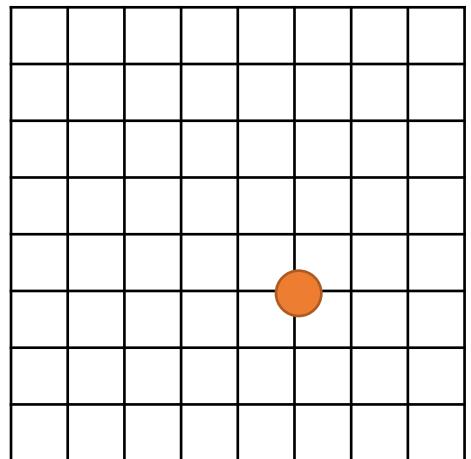
# 1. Reducing “action candidates”

(1) Imitating expert moves (supervised learning)

Current Board



Next Action



**Prediction Model**

There are  $19 \times 19 = 361$   
possible actions  
(with different probabilities)

# 1. Reducing “action candidates”

## (1) Imitating expert moves (supervised learning)

Current Board

00 000 0000
00 000 <b>1</b> 000
0 <b>-1</b> 00 <b>1</b> - <b>1</b> <b>1</b> 00
0 <b>1</b> 00 <b>1</b> - <b>1</b> 000
00 00 <b>-1</b> 0000
00 000 0000
0 <b>-1</b> 000 0000
00 000 0000

Prediction Model

Next Action

0000000000
0000000000
0000000000
0000000000
0000000000
00000 <b>1</b> 000
0000000000
0000000000
0000000000

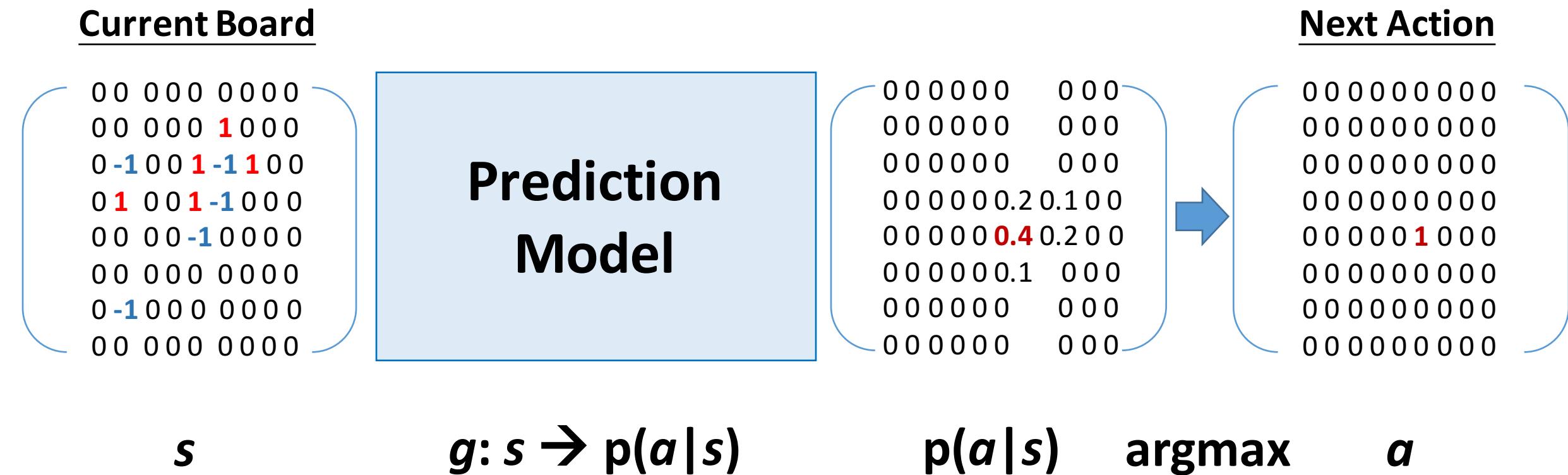
*s*

$f: s \rightarrow a$

*a*

# 1. Reducing “action candidates”

## (1) Imitating expert moves (supervised learning)



# 1. Reducing “action candidates”

## (1) Imitating expert moves (supervised learning)

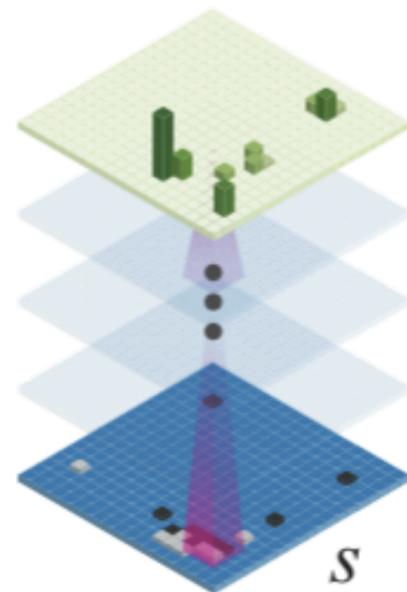
Current Board

00 000 0000  
00 000 **1**000  
0 -100 **1**-1**1**00  
0 **1** 00 **1**-1000  
00 00 -**1**0000  
00 000 0000  
0 -1000 0000  
00 000 0000

$s$

**Prediction Model**

$g: s \rightarrow p(a|s)$



$p(a|s)$

**argmax**

$a$

Next Action

000000000  
000000000  
000000000  
000000000  
00000 **1**000  
000000000  
000000000  
000000000

# 1. Reducing “action candidates”

(1) Imitating expert moves (supervised learning)

Current Board

00 000 0000  
00 000 **1**000  
0-100 **1**-**1**00  
**1** 00 **1**-**1**000  
00 00 -**1**0000  
00 000 0000  
0-1000 0000  
00 000 0000

Deep Learning  
(13 Layer CNN)

$s$

$g: s \rightarrow p(a|s)$

$p(a|s)$

$\text{argmax}$

$a$

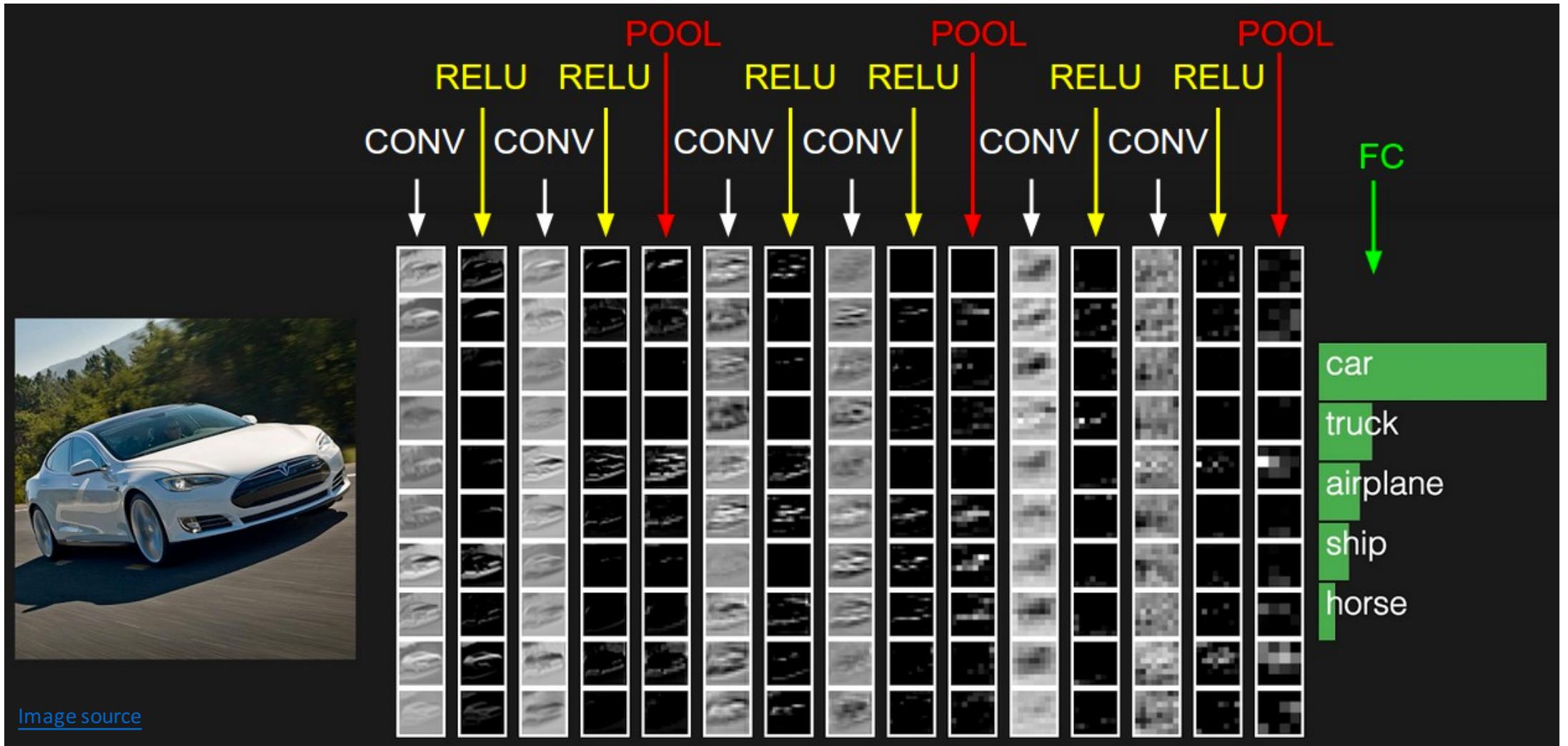
Next Action

000000 000  
000000 000  
000000 000  
000000.20.100  
00000 **0.4** 0.200  
000000.1 000  
000000 000  
000000 000



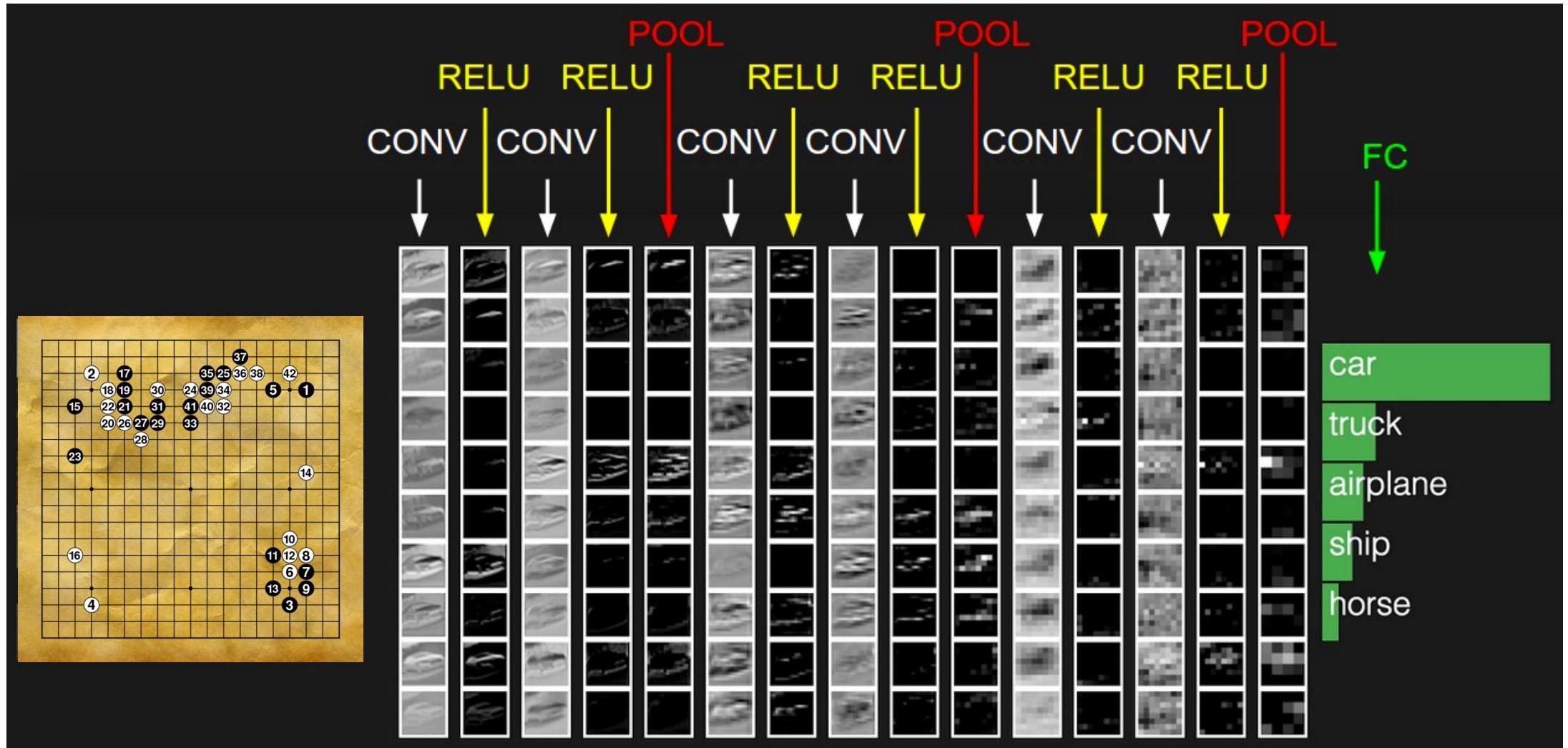
000000000  
000000000  
000000000  
000000000  
000000000  
00000 **1**000  
000000000  
000000000  
000000000

# Convolutional Neural Network (CNN)



CNN is a powerful model for image recognition tasks; it abstracts out the input image through convolution layers

# Convolutional Neural Network (CNN)



And they use this CNN model (similar architecture) to evaluate the board position; which learns “some” spatial invariance

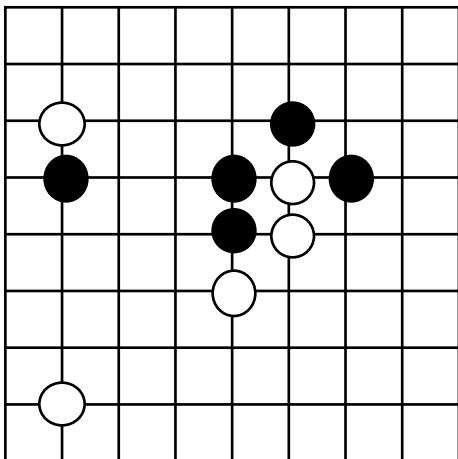
**Go:** abstraction is the key to win

**CNN:** abstraction is its *forte*

# 1. Reducing “action candidates”

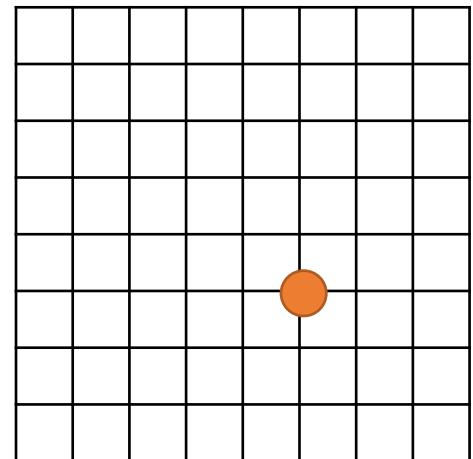
(1) Imitating expert moves (supervised learning)

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)



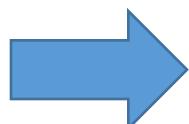
# 1. Reducing “action candidates”

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

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

vs

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

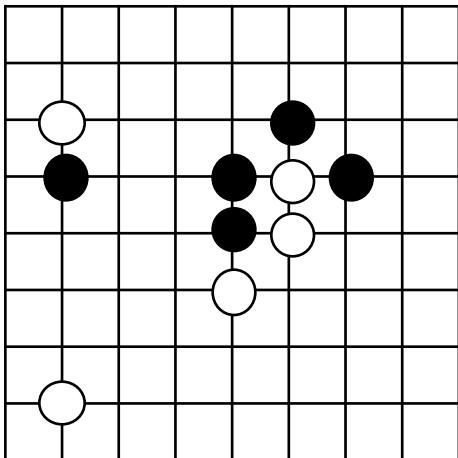


**Return:** board positions, win/lose info

# 1. Reducing “action candidates”

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

Board position



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

win/loss

**Loss**

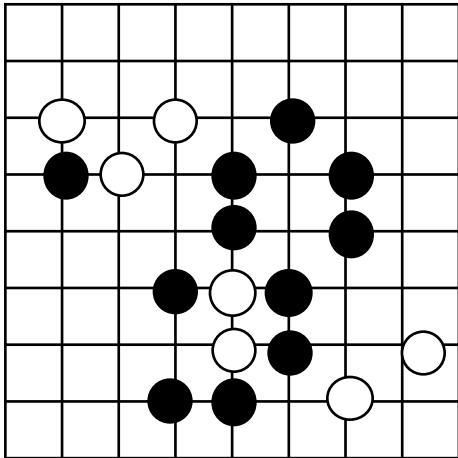
$z = -1$

**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)

Board position



win/loss

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

**Win**

$z = +1$

**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)

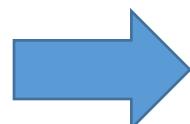
Older models vs. newer models

**Updated Model  
ver 1.1**

vs

**Updated Model  
ver 1.3**

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



**Return:** board positions, win/lose info

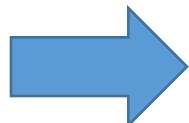
# 1. Reducing “action candidates”

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

**Updated Model  
ver 1.3**

vs

**Updated Model  
ver 1.7**



**Return:** board positions, win/lose info

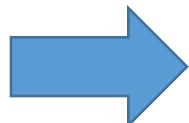
# 1. Reducing “action candidates”

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

**Updated Model  
ver 1.5**

vs

**Updated Model  
ver 2.0**



**Return:** board positions, win/lose info

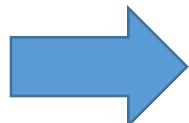
# 1. Reducing “action candidates”

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

**Updated Model  
ver 3204.1**

vs

**Updated Model  
ver 46235.2**



**Return:** board positions, win/lose info

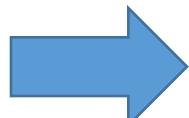
# 1. Reducing “action candidates”

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

**Expert Moves  
Imitator Model**

vs

**Updated Model  
ver 1,000,000**

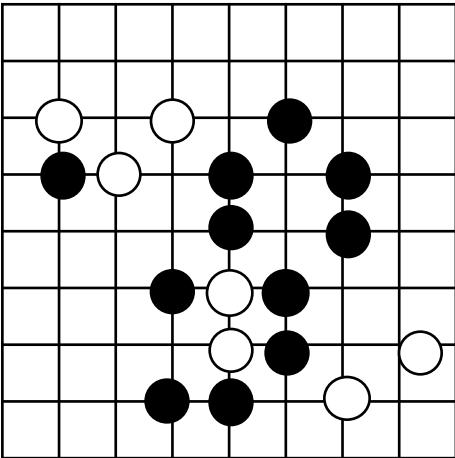


**The final model wins 80% of the time  
when playing against the first model**

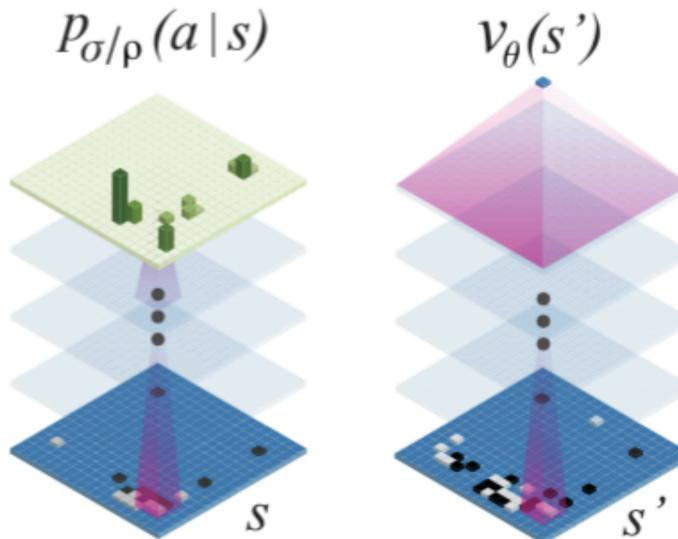
## 2. Board Evaluation

## 2. Board Evaluation

Board Position



**Updated Model  
ver 1,000,000**



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

Win / Loss

**Value  
Prediction  
Model  
(Regression)**

**Win  
(0~1)**

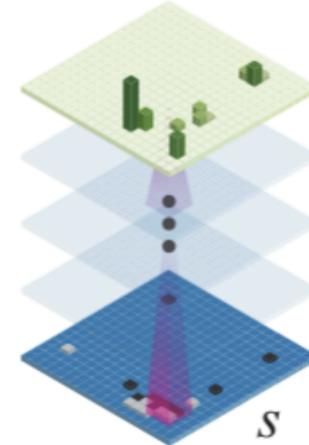
**Training:**  $\Delta\theta \propto \frac{\partial v_{\theta}(s)}{\partial \theta} (z - v_{\theta}(s))$

# Reducing Search Space

1. Reducing “action candidates”  
(Breadth Reduction)

Policy Network

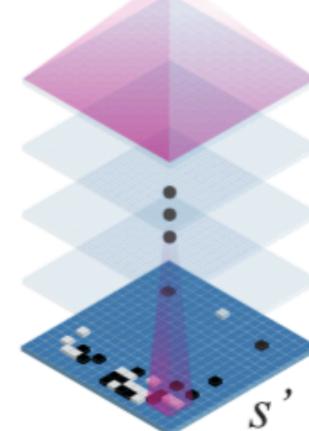
$$p_{\sigma/\rho}(a|s)$$



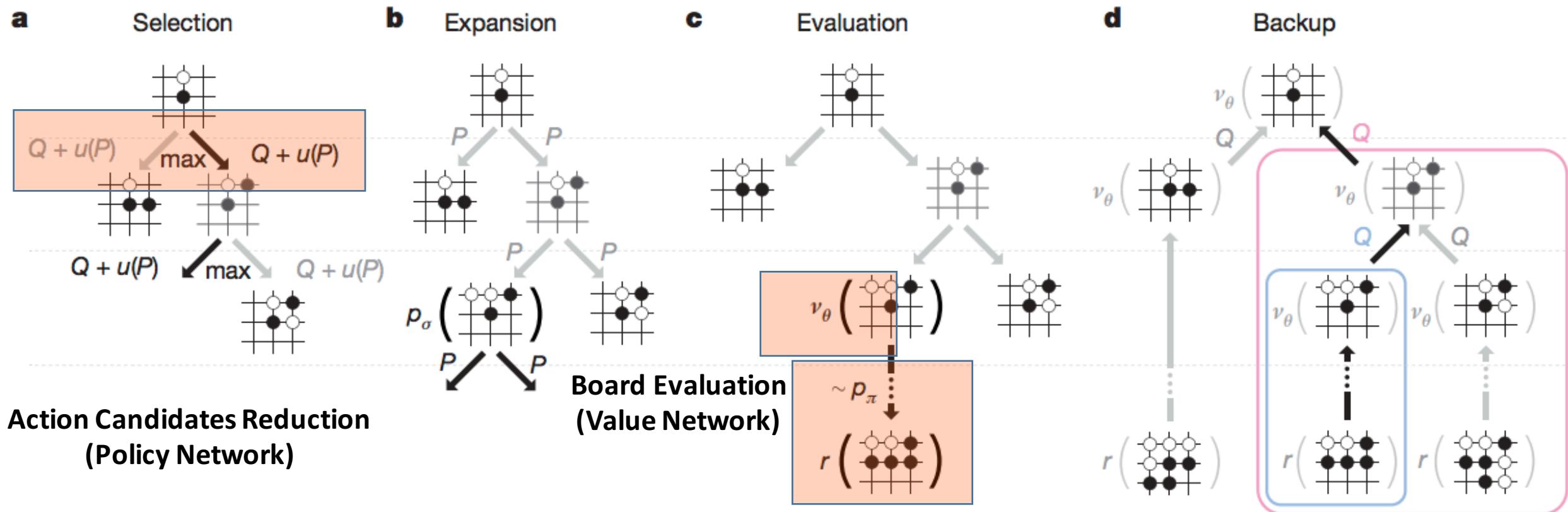
2. Board Evaluation (Depth Reduction)

Value Network

$$v_\theta(s')$$

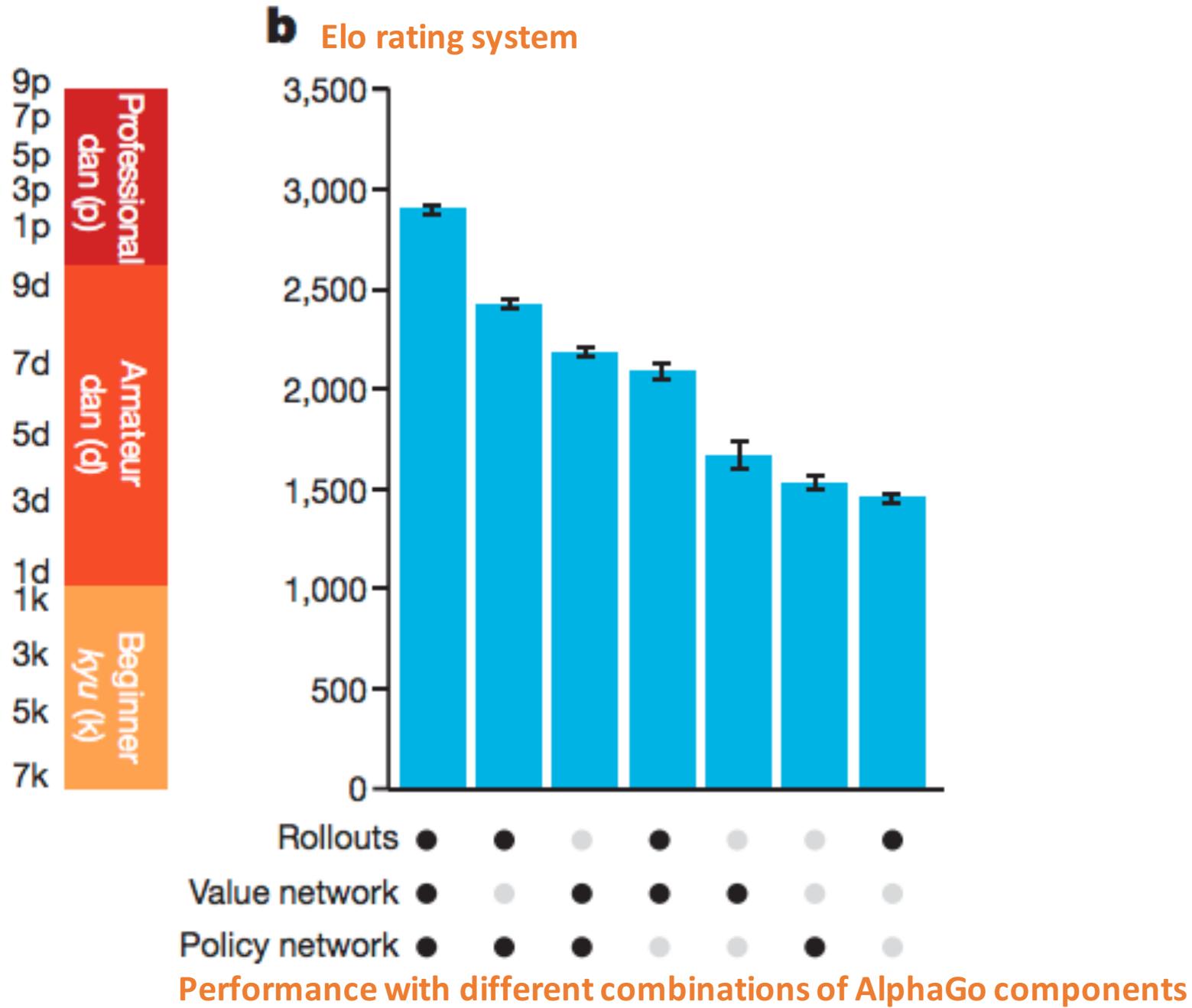


# Looking ahead (w/ Monte Carlo Search Tree)



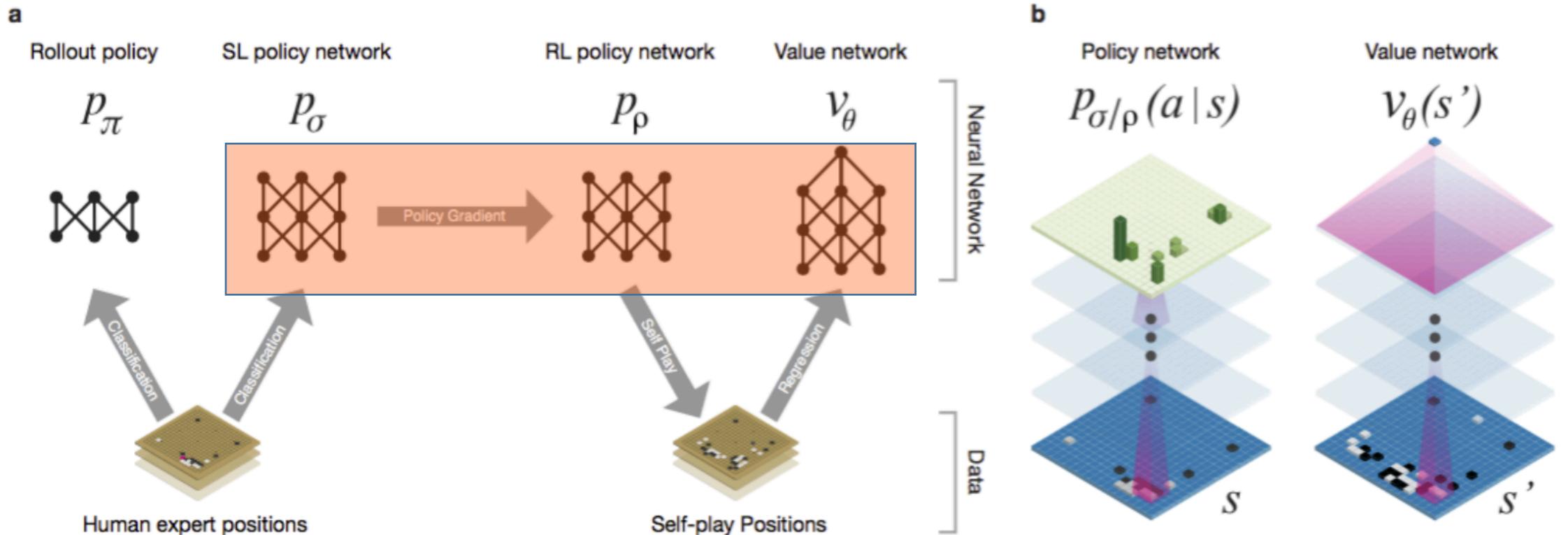
(Rollout): Faster version of estimating  $p(a|s)$   
→ uses shallow networks (3 ms → 2μs)

# Results



# Takeaways

Use the networks trained for a certain task (with different loss objectives) for several other tasks



# Lee Sedol 9-dan vs AlphaGo



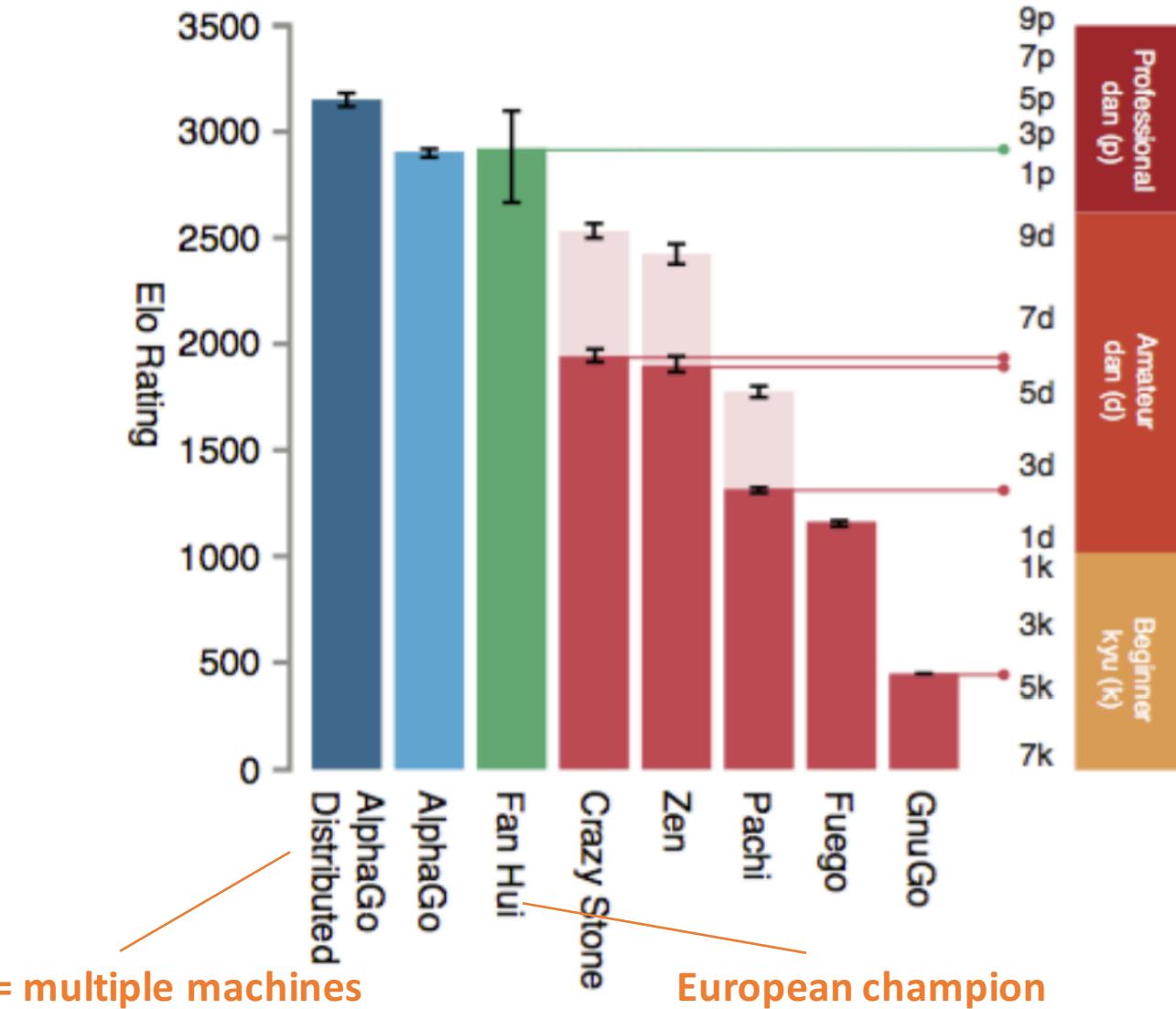
# Lee Sedol 9-dan vs AlphaGo

## Energy Consumption

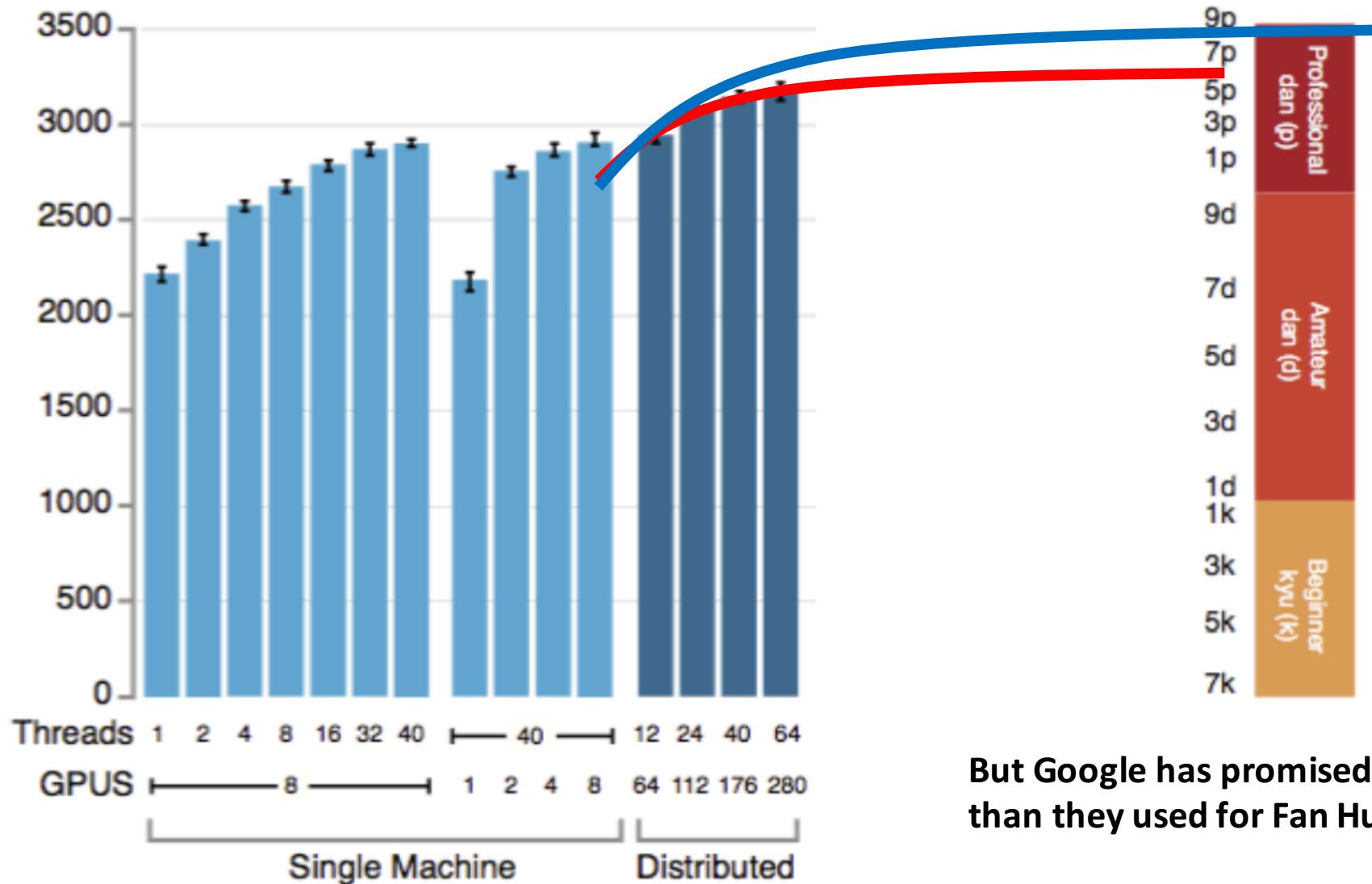
Lee Sedol	AlphaGo
<ul style="list-style-type: none"><li>- Recommended calories for a man per day : ~2,500 kCal</li><li>- Assumption: Lee consumes the entire amount of per-day calories in this one game <math>2,500 \text{ kCal} * 4,184 \text{ J/kCal}</math></li></ul> <p><b><math>\approx 10M \text{ [J]}</math></b></p>	<ul style="list-style-type: none"><li>- Assumption: CPU: ~100 W, GPU: ~300 W</li><li>- <b>1,202 CPUs, 176 GPUs</b></li></ul> $170,000 \text{ J/sec} * 5 \text{ hr} * 3,600 \text{ sec/hr}$ <p><b><math>\approx 3,000M \text{ [J]}</math></b></p>

A very, very rough calculation ;)

# AlphaGo is estimated to be around ~5-dan



# Taking CPU / GPU resources to virtually infinity?



No one knows  
how it will converge

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

# AlphaGo learns millions of Go games every day

AlphaGo will presumably converge to some point eventually.

However, in the Nature paper they don't report how AlphaGo's performance improves as a function of times AlphaGo plays against itself (self-plays).

# What if AlphaGo learns Lee's game strategy

Google said they won't use Lee's game plays as AlphaGo's training data

**Even if it does, it won't be easy to modify the model trained over millions of data points with just a few game plays with Lee**

**(prone to over-fitting, etc.)**

# AlphaGo's Weakness?

# AlphaGo – How It Works

Presenter: Shane (Seungwhan) Moon

PhD student

Language Technologies Institute, School of Computer Science

Carnegie Mellon University

[me@shanemoon.com](mailto:me@shanemoon.com)

3/2/2016

# Reference

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