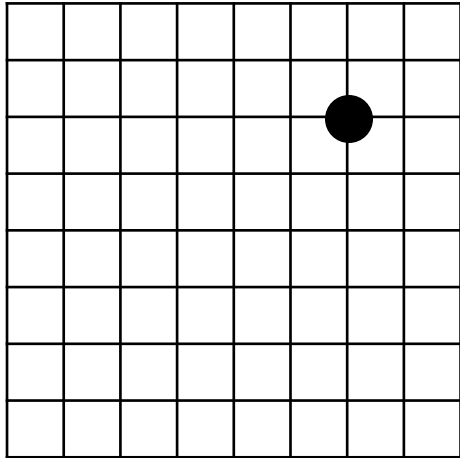


Computer Go AI – Definition

$d = 1$



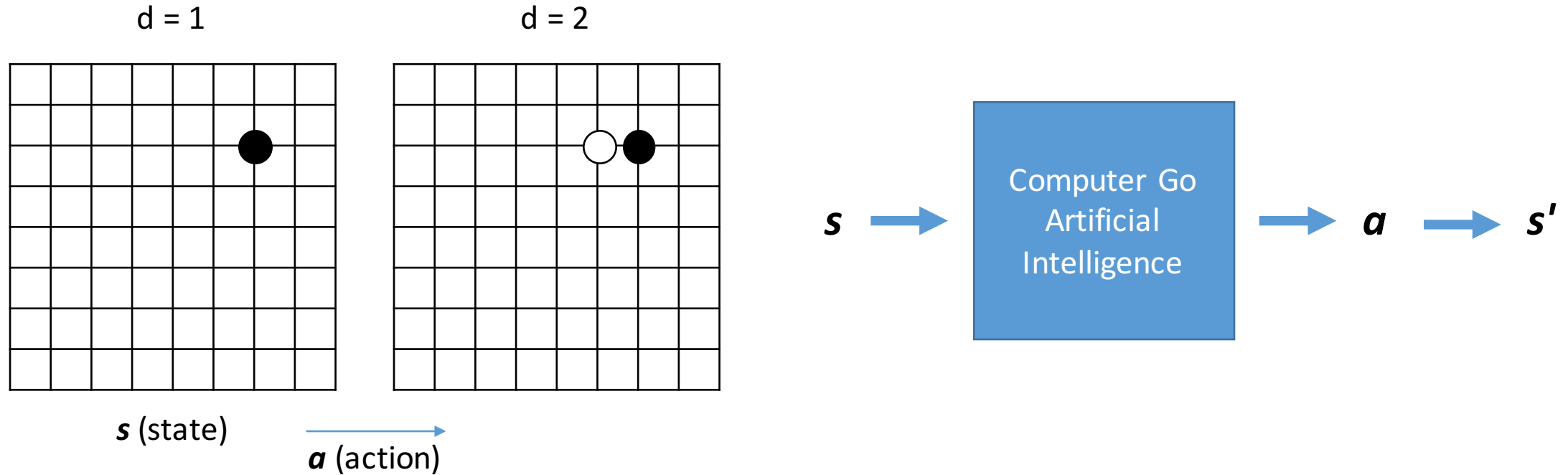
s (state)

$$= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

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

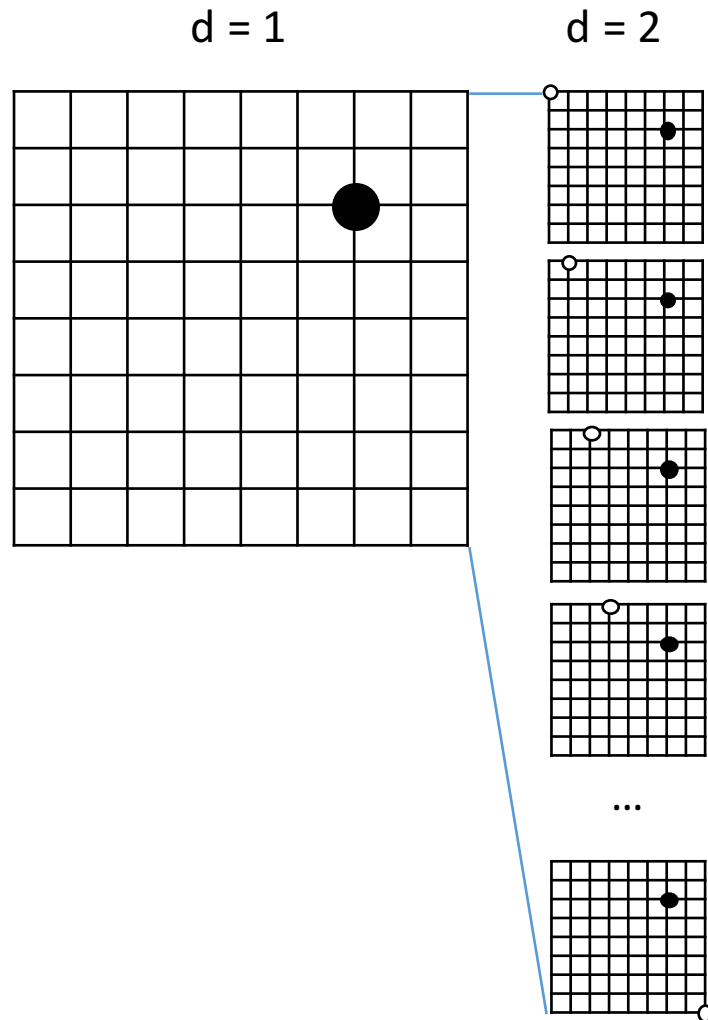
** The actual model uses other features than board positions as well*

Computer Go AI – Definition



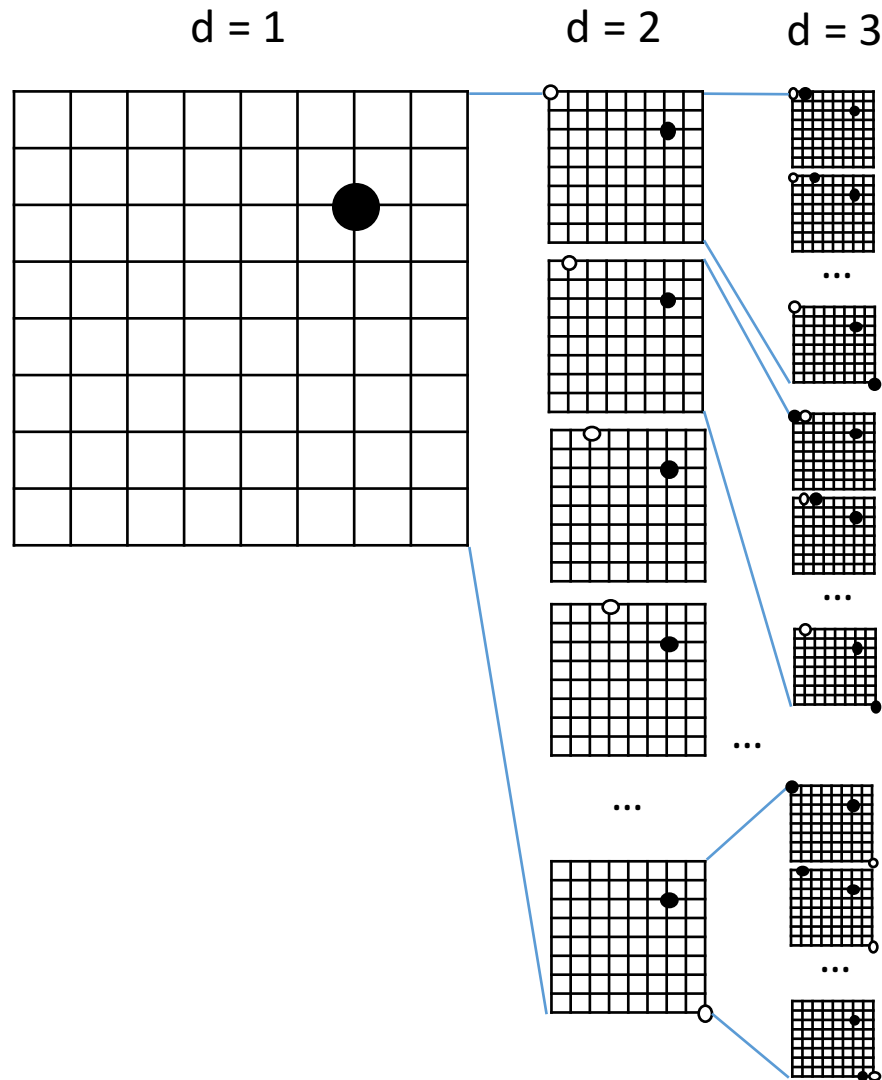
Given s , pick the best a

Computer Go AI – An Implementation Idea?

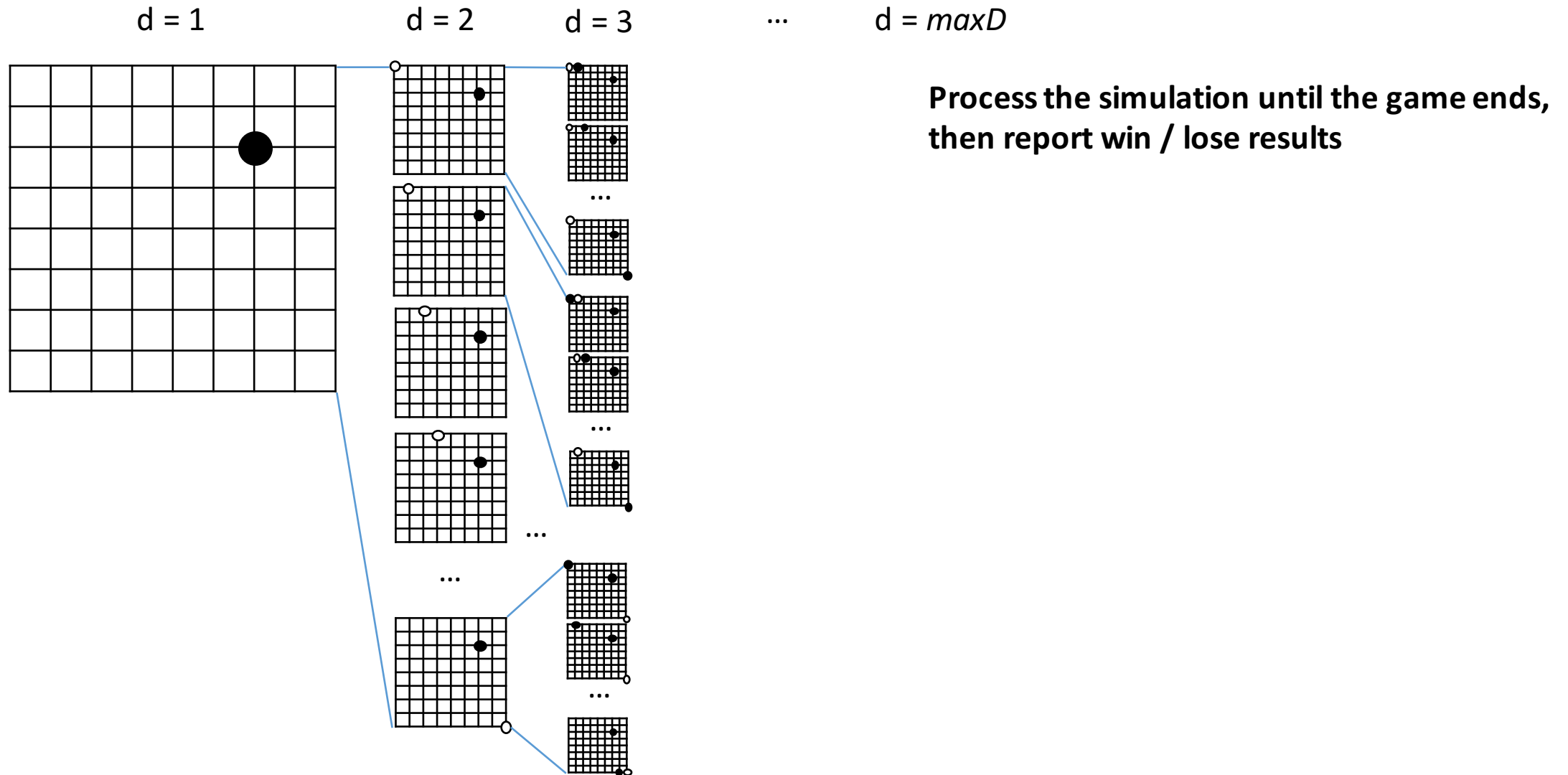


How about simulating all possible board positions?

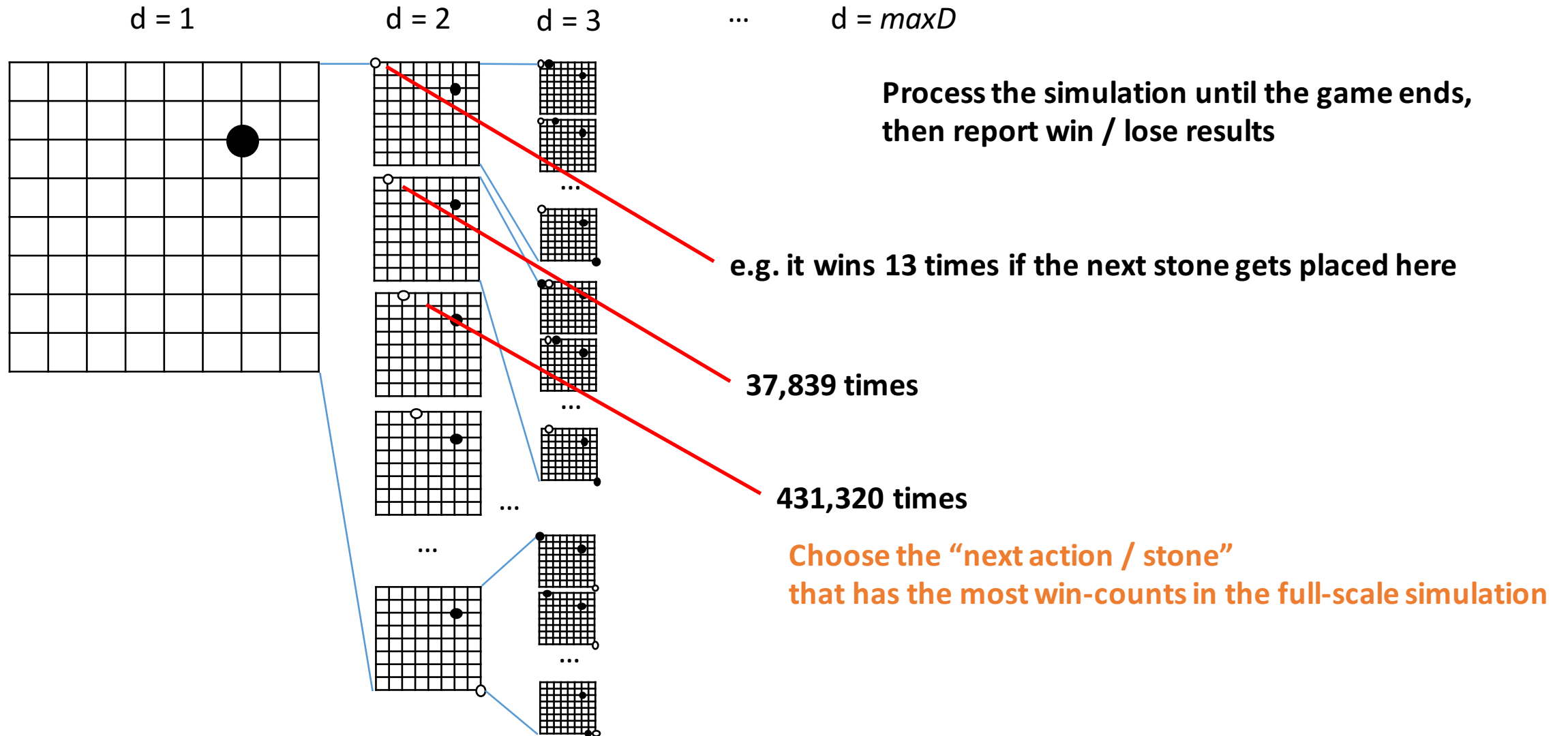
Computer Go AI – An Implementation Idea?



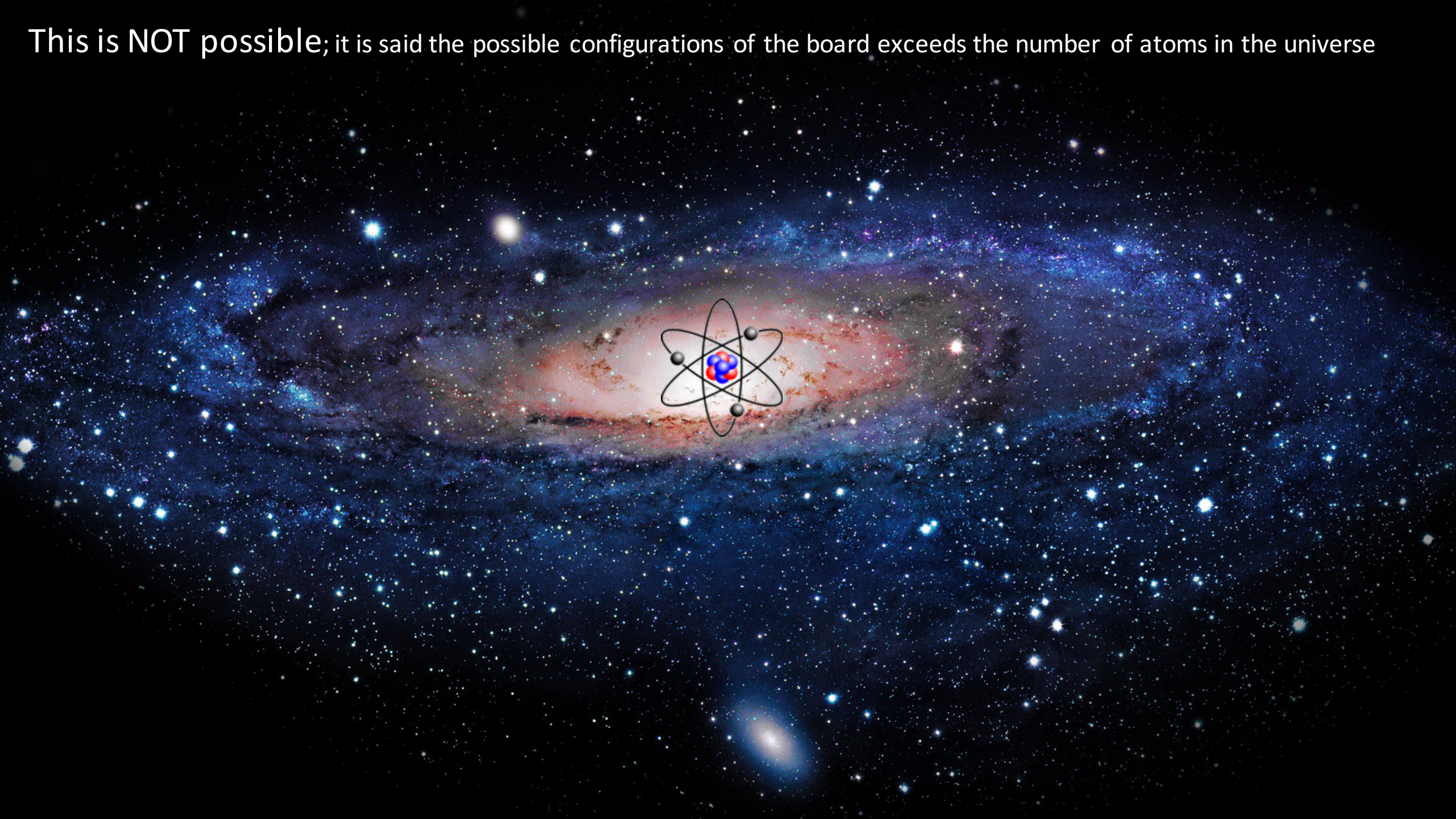
Computer Go AI – An Implementation Idea?



Computer Go AI – An Implementation Idea?



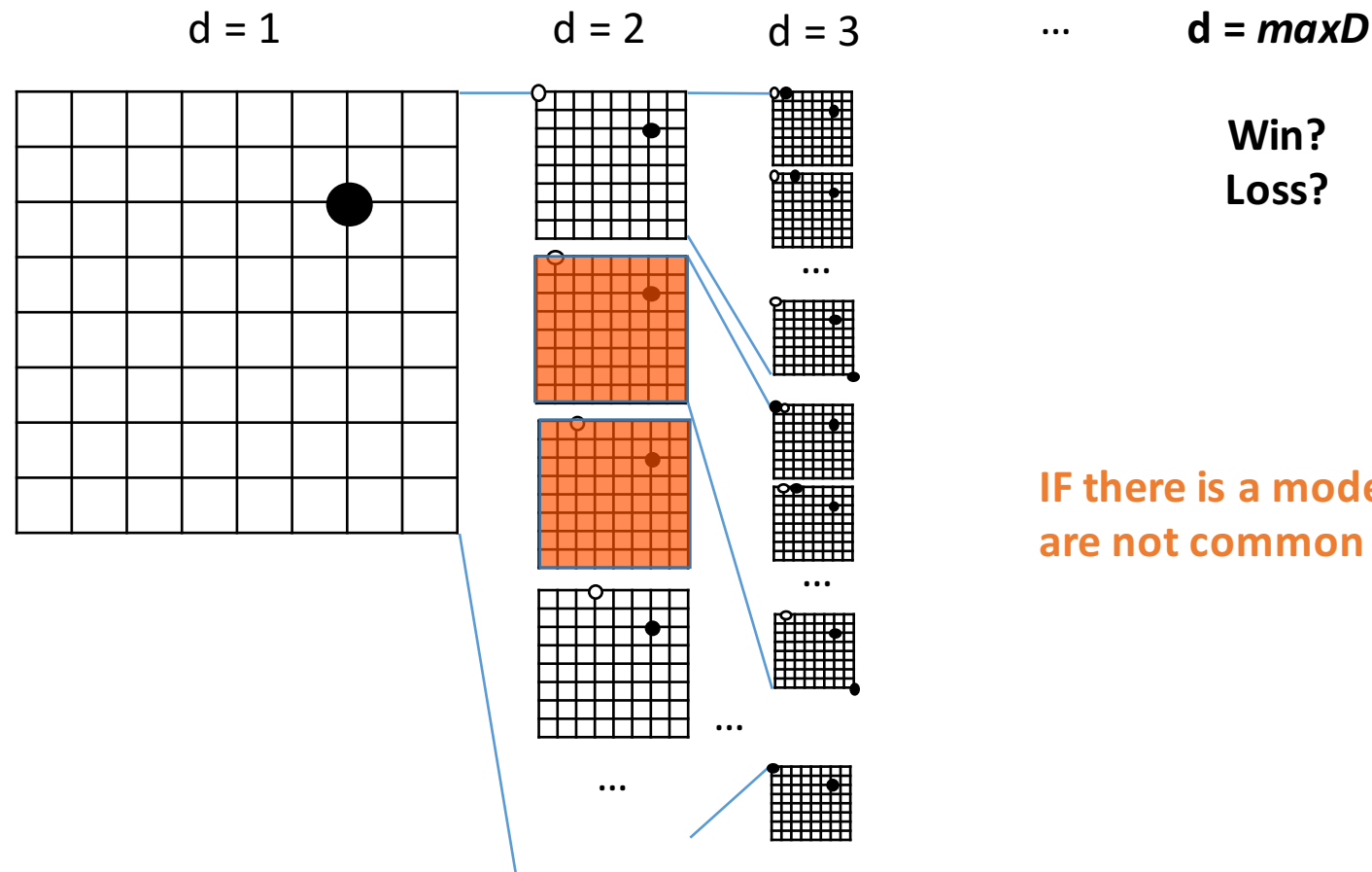
This is NOT possible; it is said the possible configurations of the board exceeds the number of atoms in the universe



Key: To Reduce Search Space

Reducing Search Space

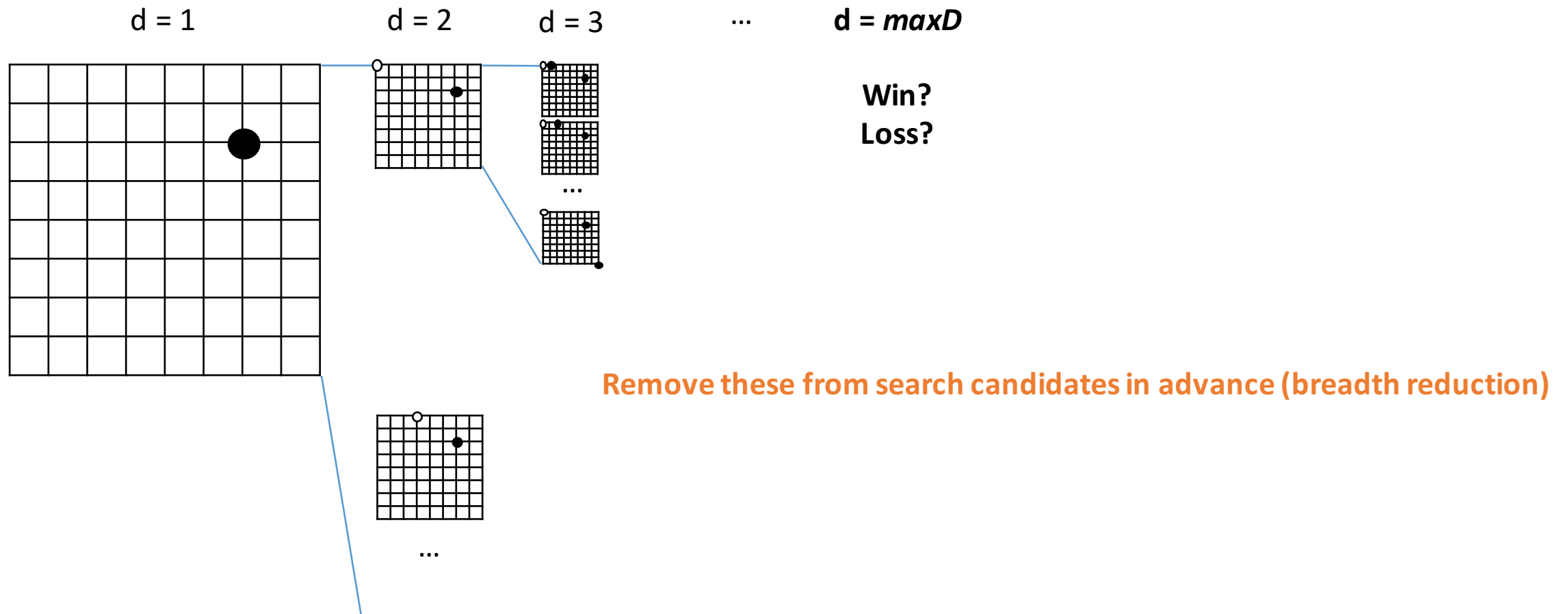
1. Reducing “action candidates” (Breadth Reduction)



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

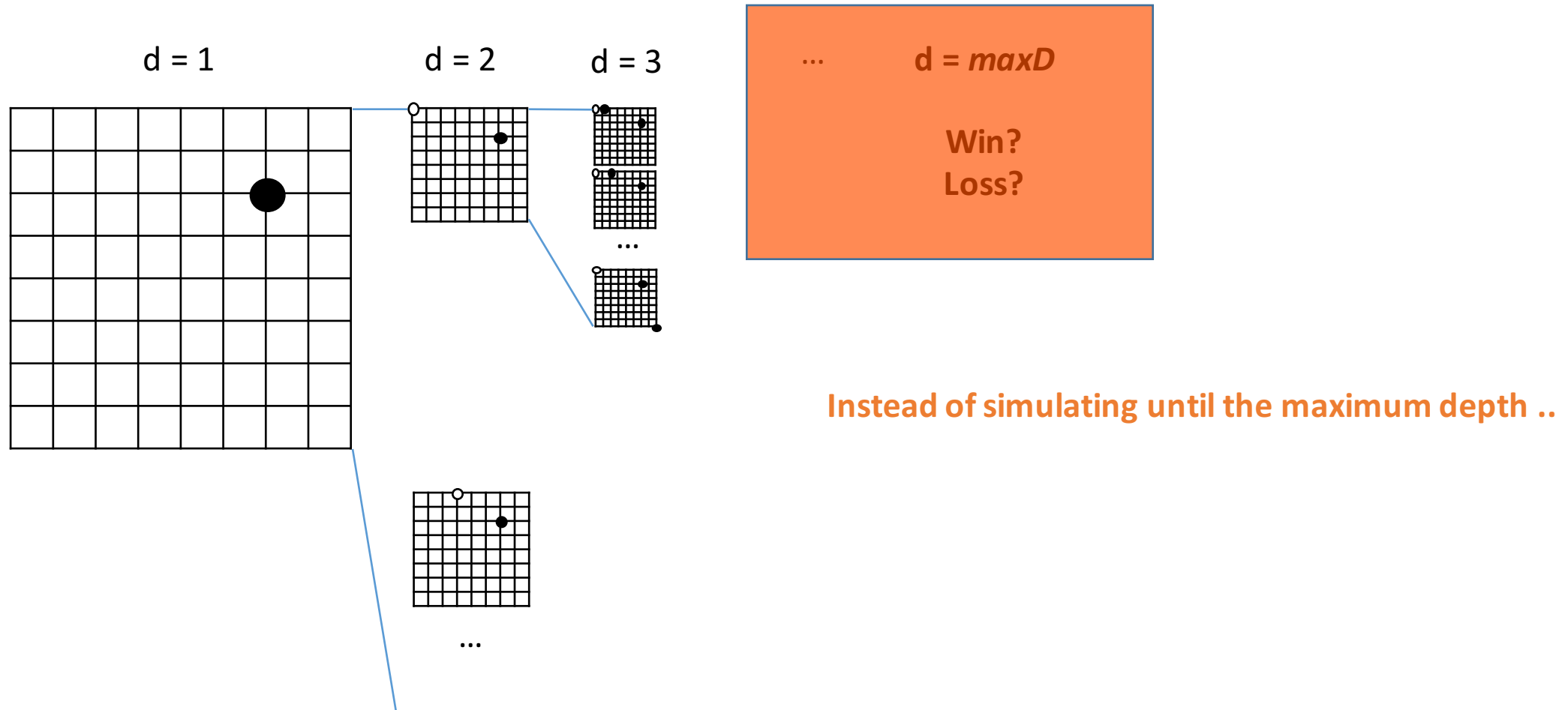
Reducing Search Space

1. Reducing “action candidates” (Breadth Reduction)



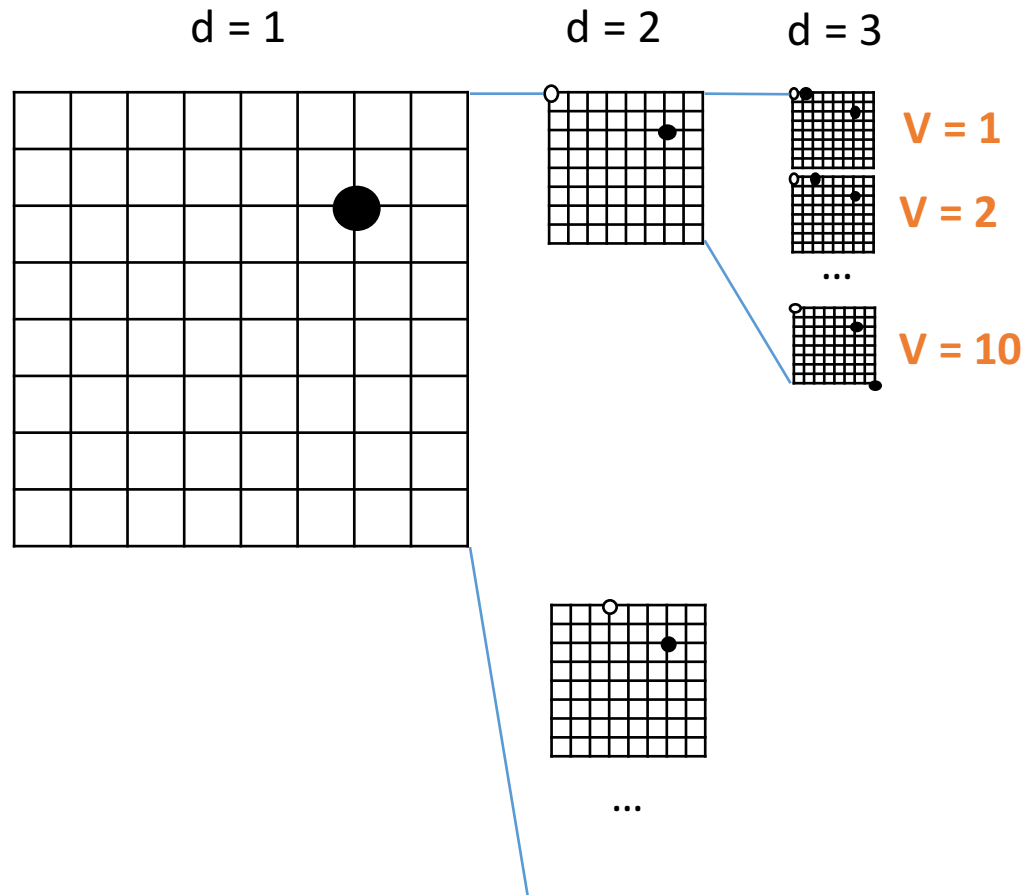
Reducing Search Space

2. Position evaluation ahead of time (Depth Reduction)



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)**

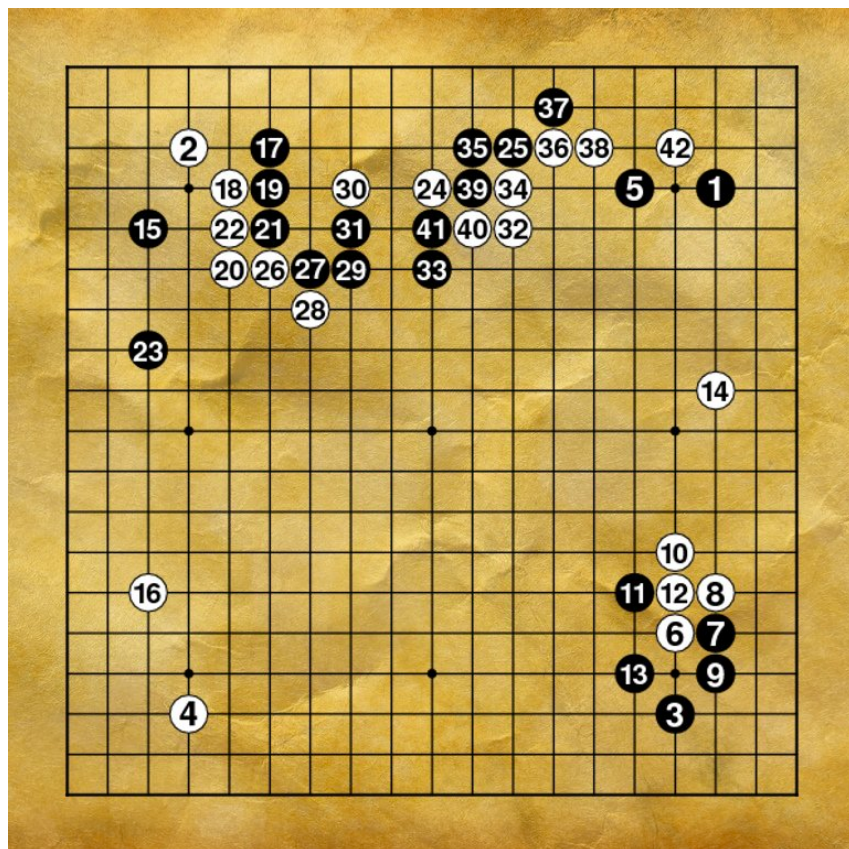
1. Reducing “action candidates”

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

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

1. Reducing “action candidates”

(1) Imitating expert moves (supervised learning)



Current State

Next State

s1

s2

s2

s3

s3

s4

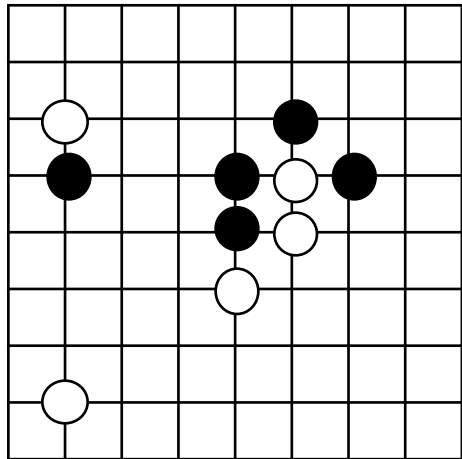
**Prediction
Model**

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

1. Reducing “action candidates”

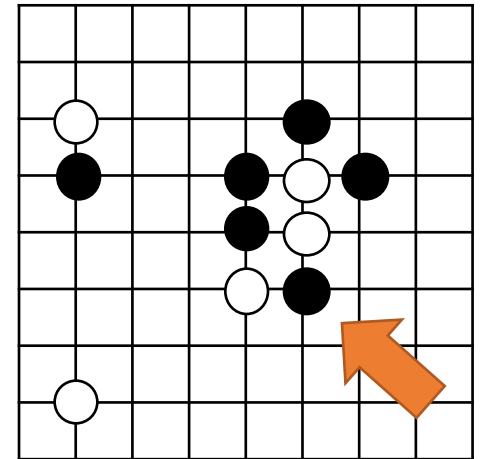
(1) Imitating expert moves (supervised learning)

Current Board



Prediction Model

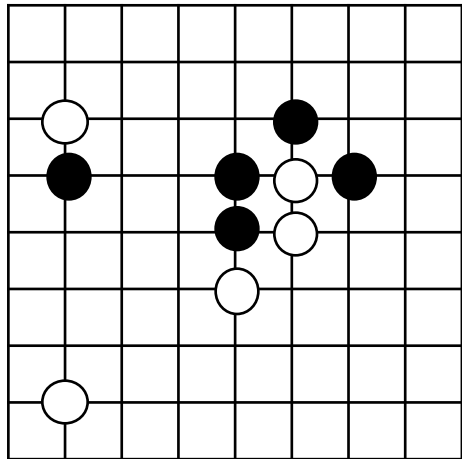
Next Board



1. Reducing “action candidates”

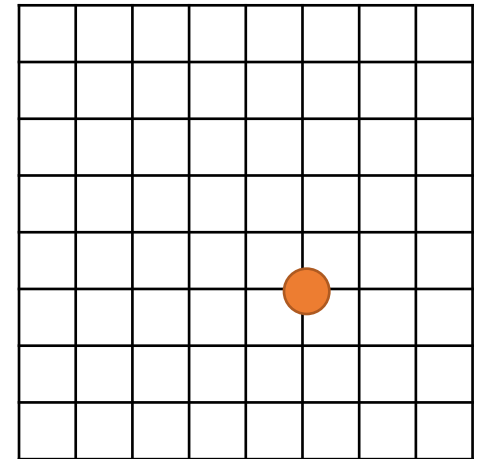
(1) Imitating expert moves (supervised learning)

Current Board



Prediction Model

Next Action



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

1. Reducing “action candidates”

(1) Imitating expert moves (supervised learning)

Current Board

0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 -1 0 0 1 -1 1 0
0 1 0 0 1 -1 0 0
0 0 0 -1 0 0 0 0
0 0 0 0 0 0 0 0
0 -1 0 0 0 0 0 0
0 0 0 0 0 0 0 0

s

Prediction Model

$f: s \rightarrow a$

Next Action

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

a

1. Reducing “action candidates”

(1) Imitating expert moves (supervised learning)

Current Board

0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 -1 0 0 1 -1 1 0
0 1 0 0 1 -1 0 0
0 0 0 -1 0 0 0 0
0 0 0 0 0 0 0 0
0 -1 0 0 0 0 0 0
0 0 0 0 0 0 0 0

s

**Prediction
Model**

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

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0.2 0.1 0
0 0 0 0 0 0.4 0.2 0
0 0 0 0 0 0.1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

$p(a|s)$

argmax

a

Next Action

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

1. Reducing “action candidates”

(1) Imitating expert moves (supervised learning)

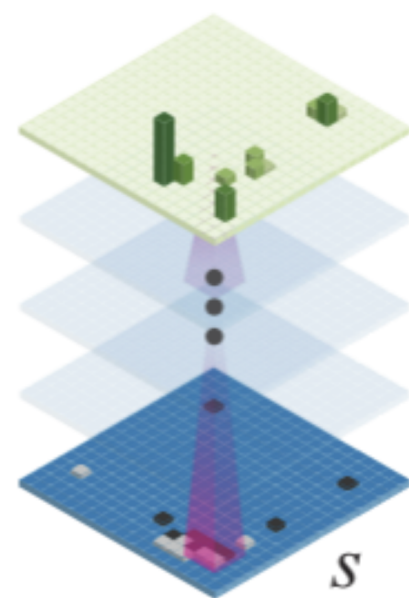
Current Board

0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0
0	-1	0	0	1	-1	1	0
0	1	0	0	1	-1	0	0
0	0	0	-1	0	0	0	0
0	0	0	0	0	0	0	0
0	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

s

**Prediction
Model**

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



$p(a|s)$

Next Action

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

argmax

a

1. Reducing “action candidates”

(1) Imitating expert moves (supervised learning)

Current Board

0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 -1 0 0 1 -1 1 0
0 1 0 0 1 -1 0 0
0 0 0 -1 0 0 0 0
0 0 0 0 0 0 0 0
0 -1 0 0 0 0 0 0
0 0 0 0 0 0 0 0

s

Deep Learning
(13 Layer CNN)

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

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0.2 0.1 0
0 0 0 0 0 0.4 0.2 0
0 0 0 0 0 0.1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

$p(a|s)$

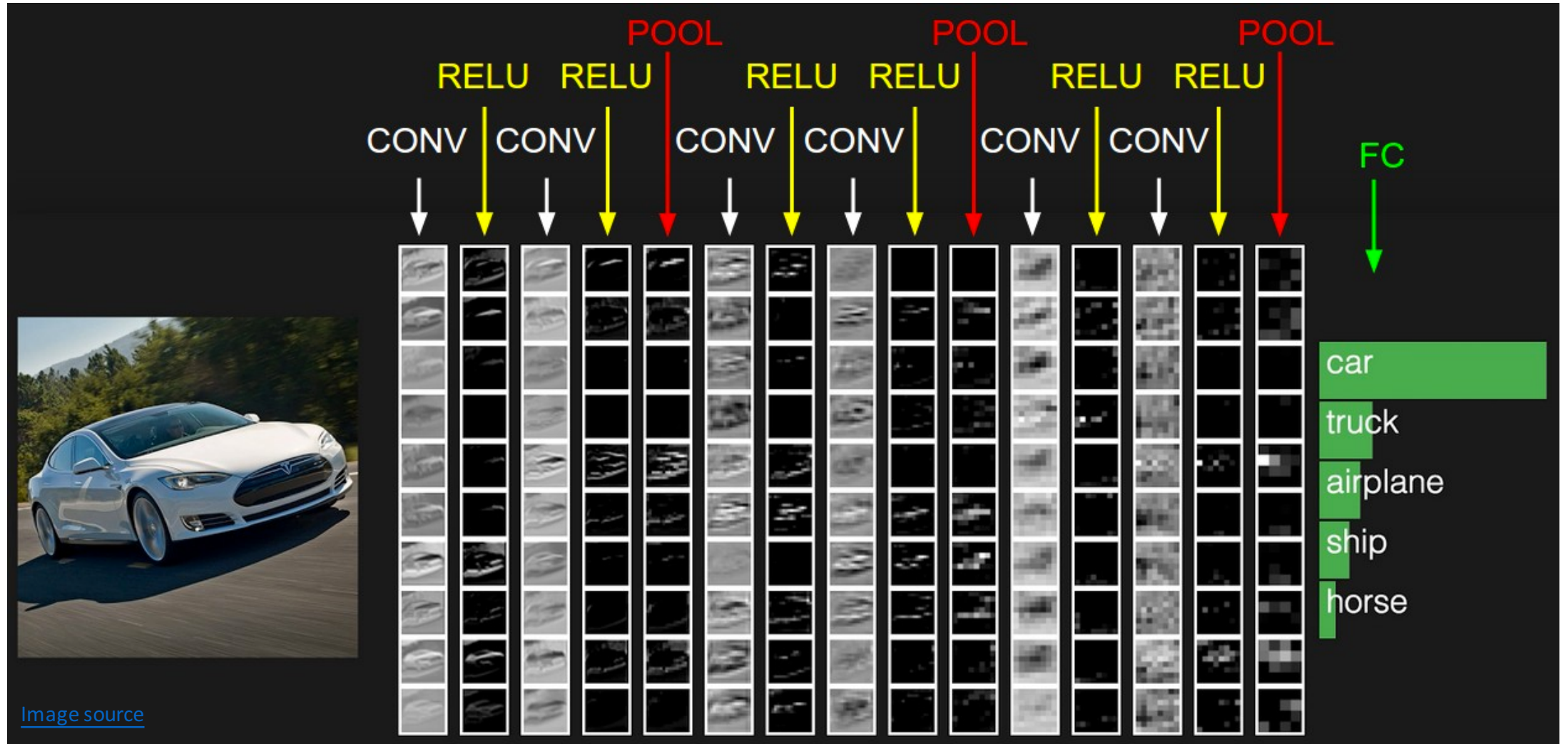
argmax

a

Next Action

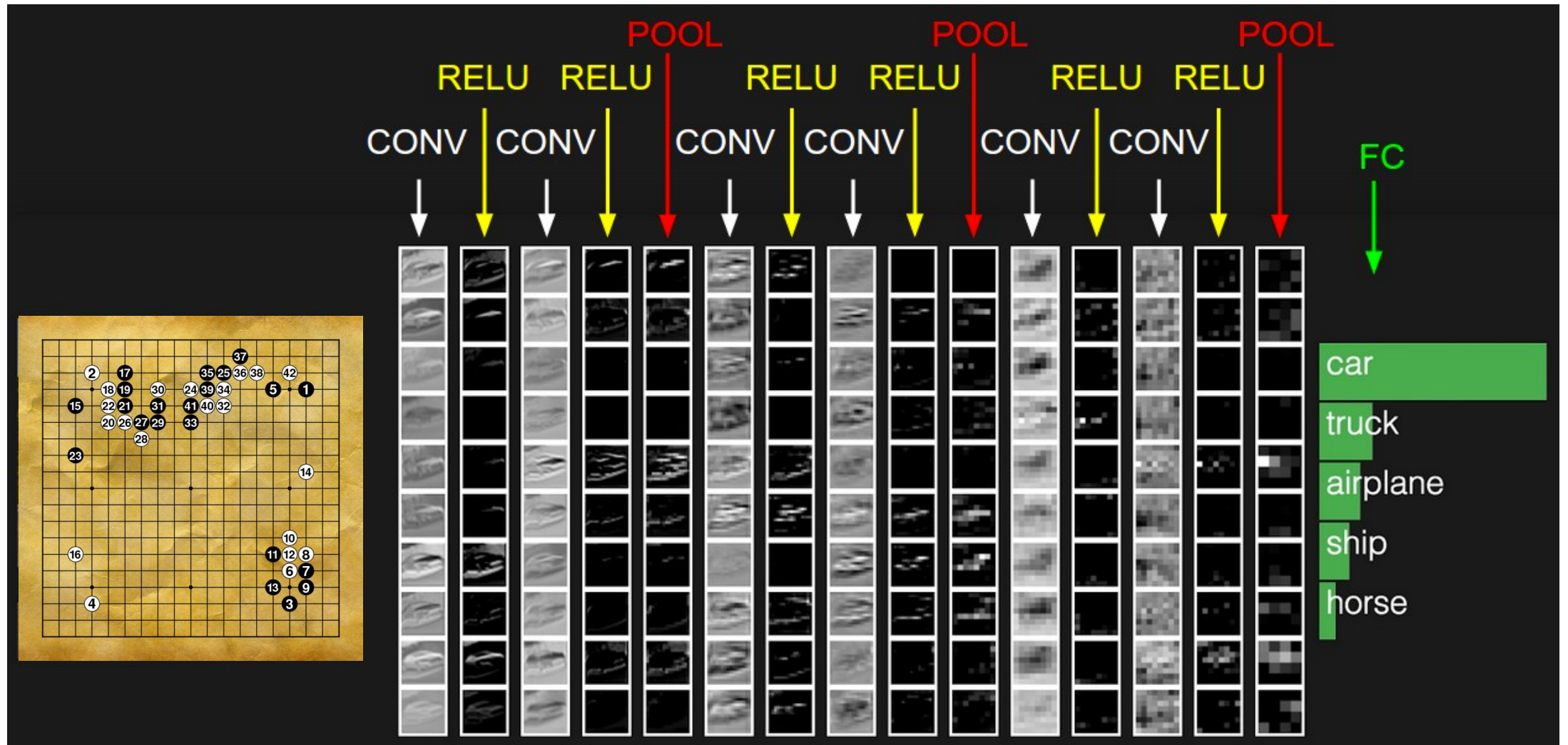
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

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)



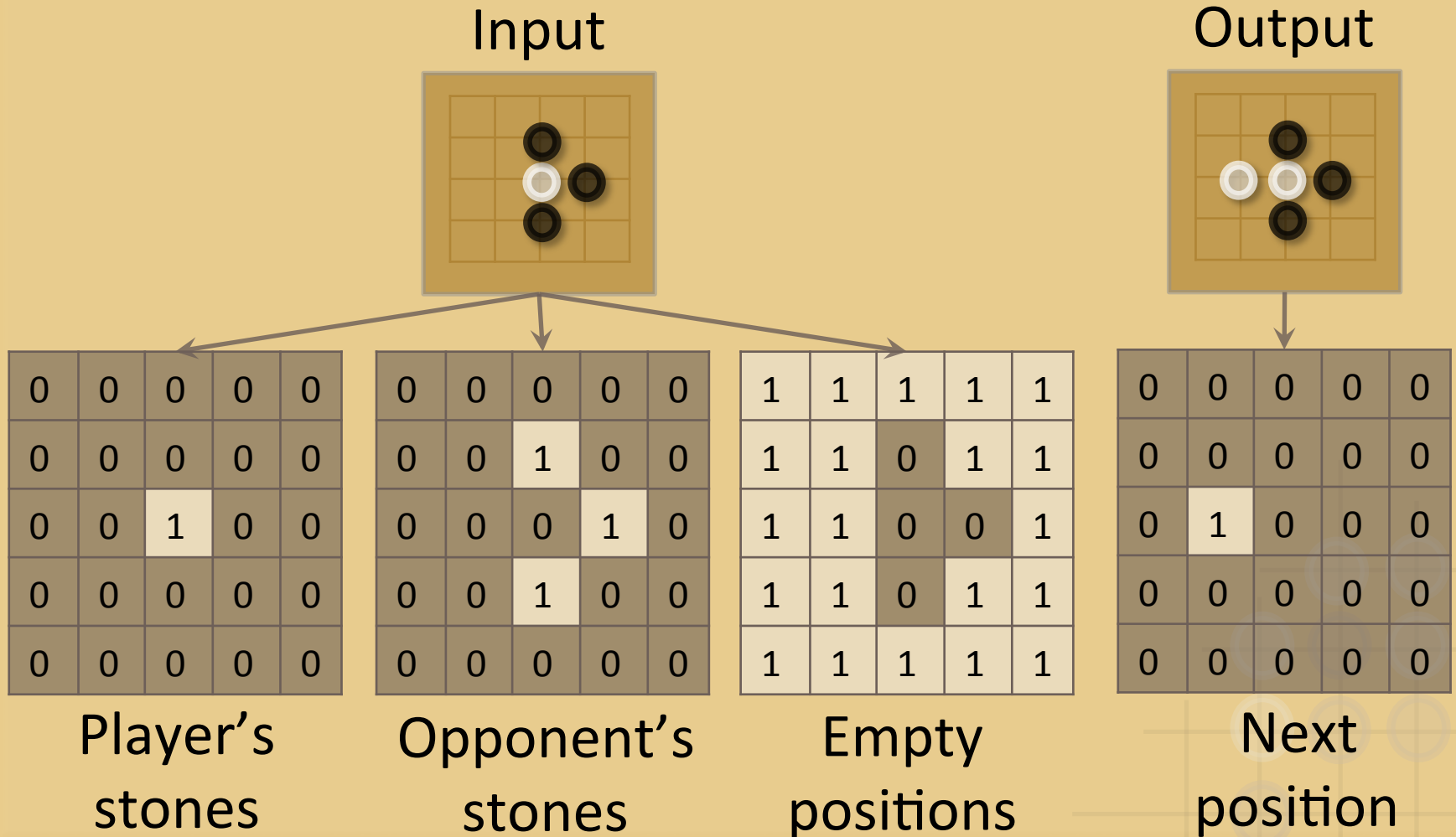
And they use this CNN model (similar architecture) to evaluate the board position; *which learns "some" spatial invariance*

Go: abstraction is the key to win

CNN: abstraction is its *forte*

Training Neural Networks

- One-Hot Encoding:



Training Neural Networks

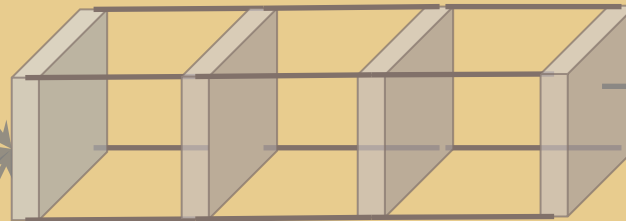
Inputs: $\mathbf{s}$

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

0	0	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	1	0	0
0	0	0	0	0

1	1	1	1	1
1	1	0	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	1	1

Input layer Convolutional layer Output layer



Forward propagation

Outputs:
 $p_w(\mathbf{a}|\mathbf{s})$

0	0	0	0	0
0	.5	0	0	0
0	.3	0	0	0
0	.2	0	0	0
0	0	0	0	0

Training Neural Networks

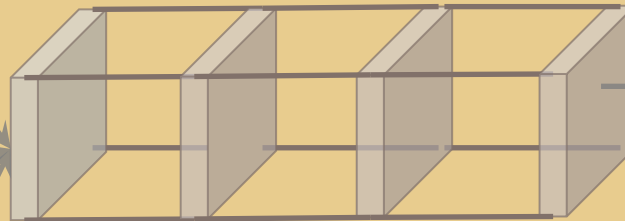
Inputs: $\mathbf{s}$

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

0	0	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	1	0	0
0	0	0	0	0

1	1	1	1	1
1	1	0	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	1	1

Input layer Convolutional layer Output layer



Outputs:
 $p_w(\mathbf{a}|\mathbf{s})$

0	0	0	0	0
0	.5	0	0	0
0	.3	0	0	0
0	.2	0	0	0
0	0	0	0	0

Cost function: $-\log(p_w(a_i|\mathbf{s}))$



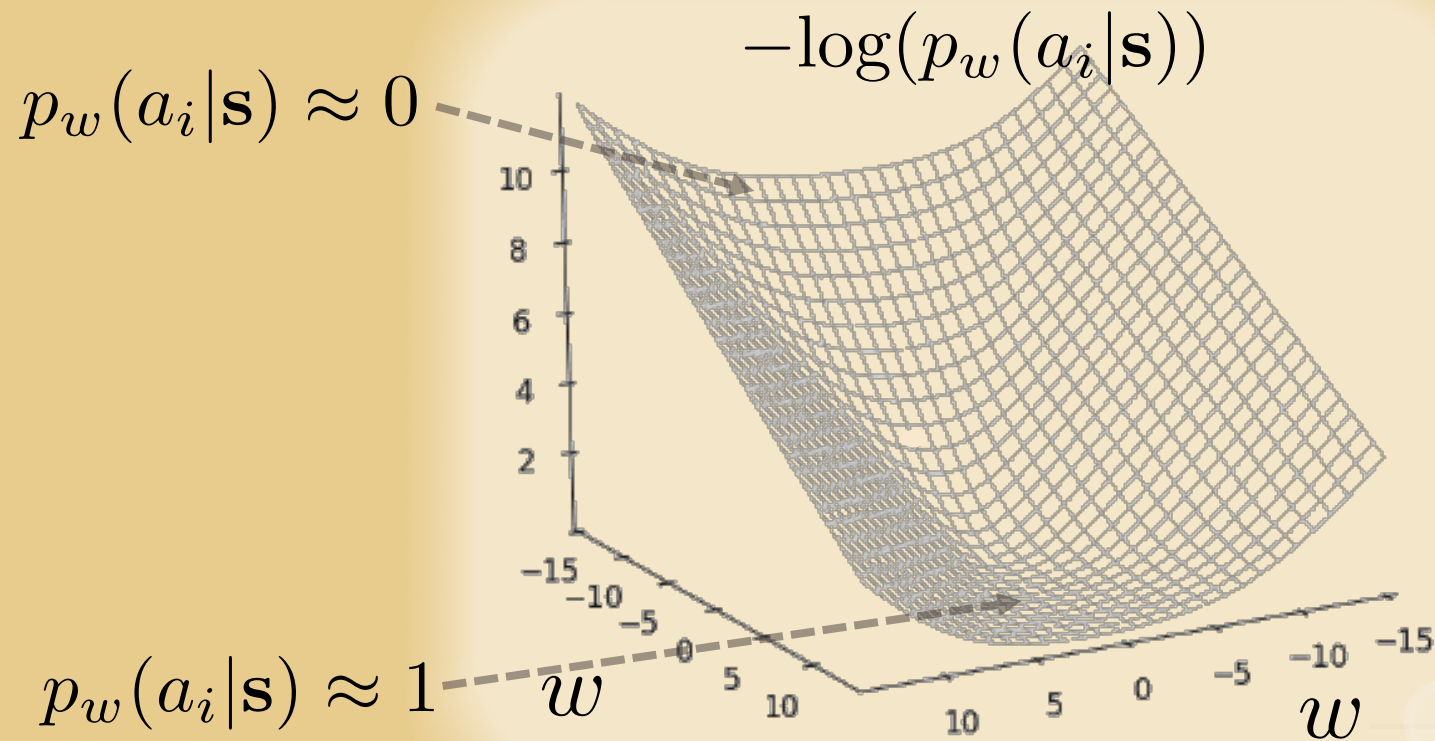
Backward propagation

$$w = w - \eta \frac{\partial -\log(p_w(a_i|\mathbf{s}))}{\partial w}$$

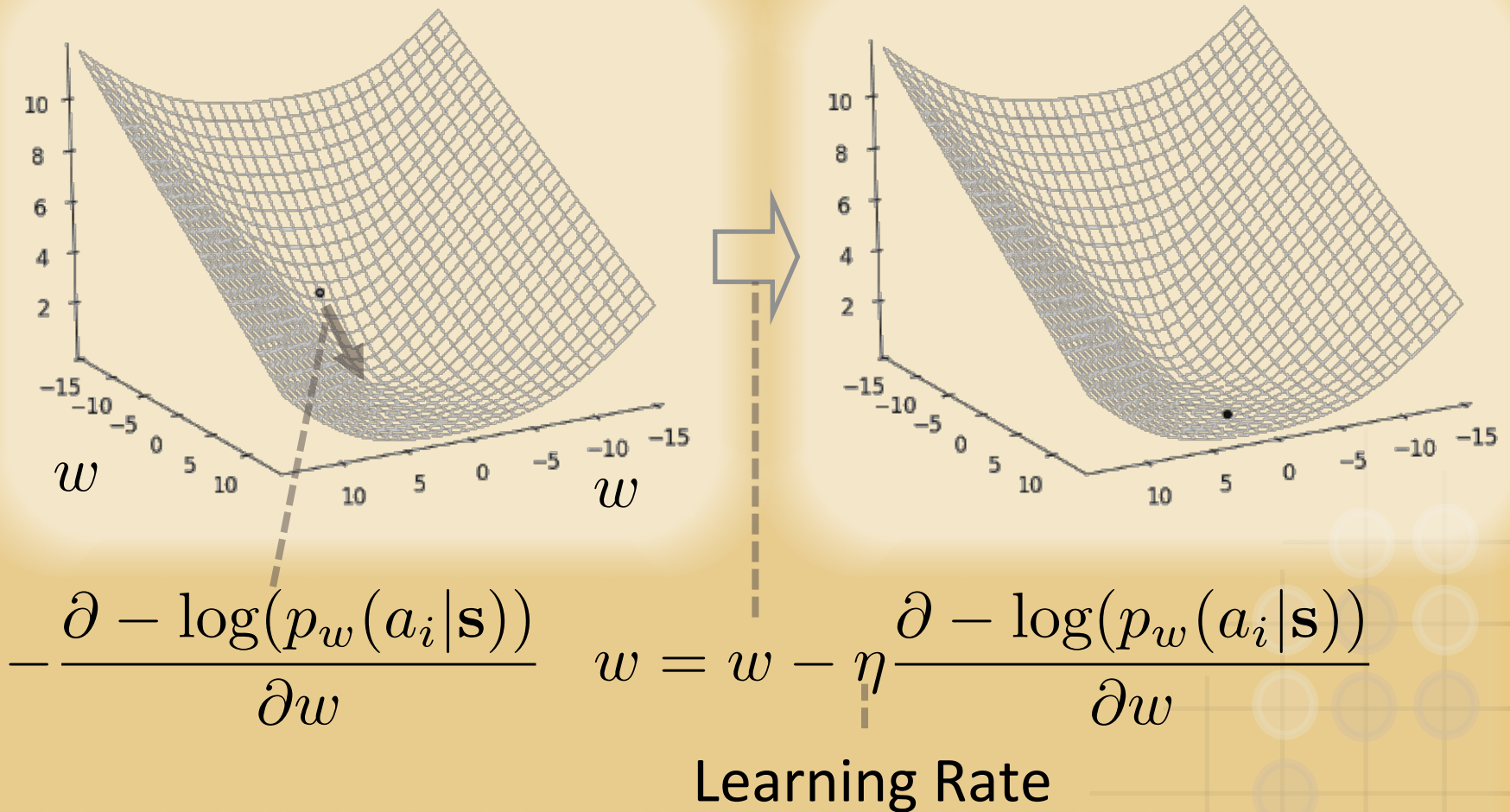
0	0	0	0	0
0	0	0	0	0
0	1	0	0	0
0	0	0	0	0
0	0	0	0	0

Golden: a_i

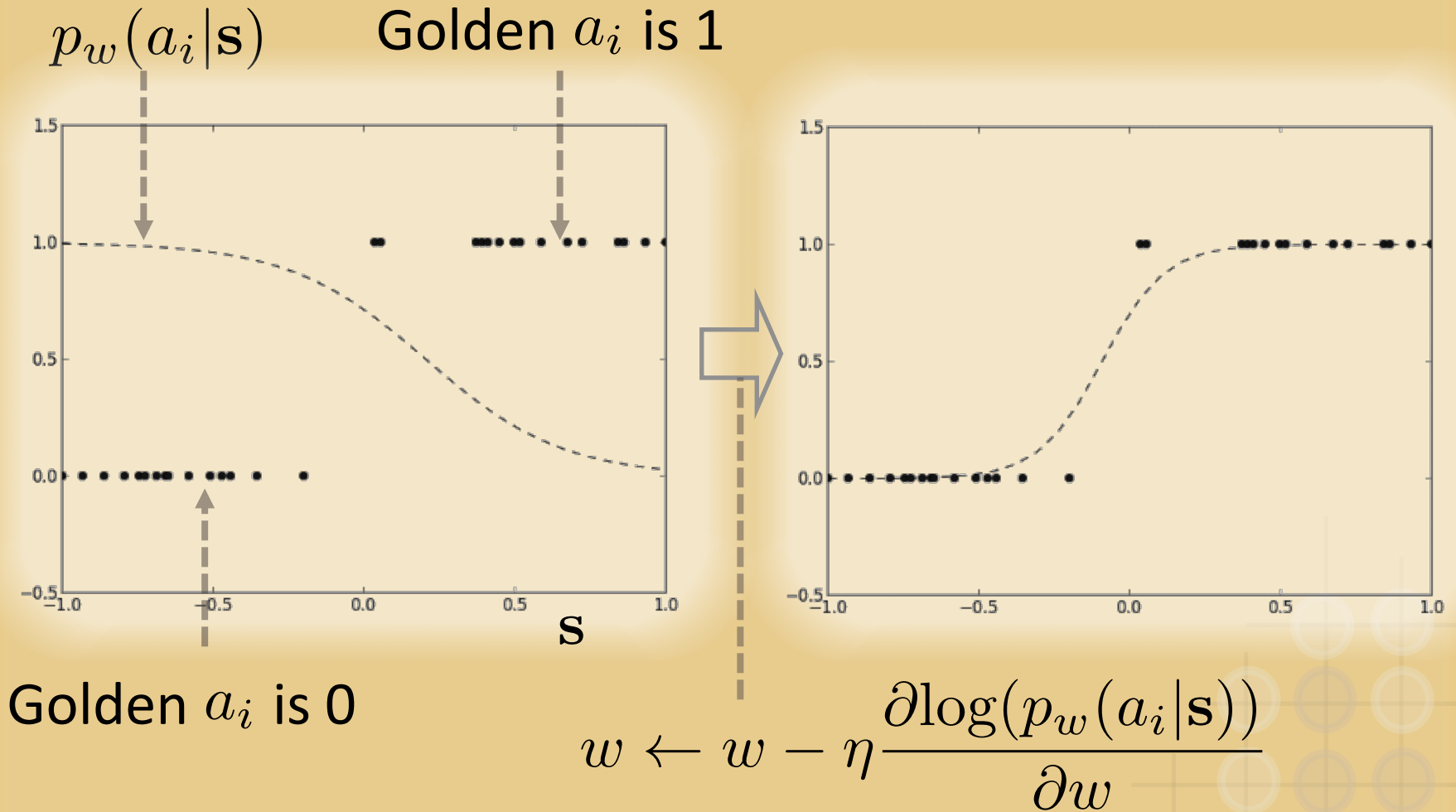
Cost Function



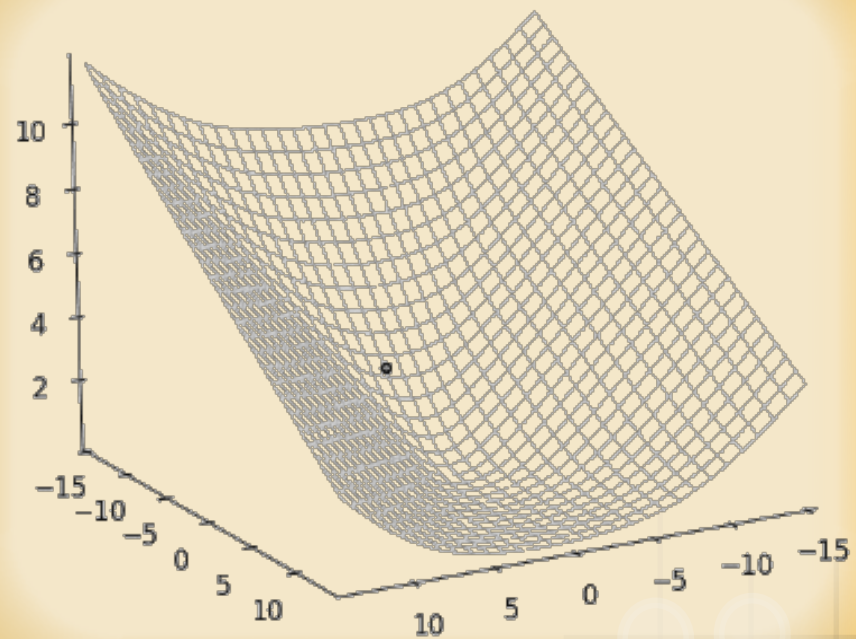
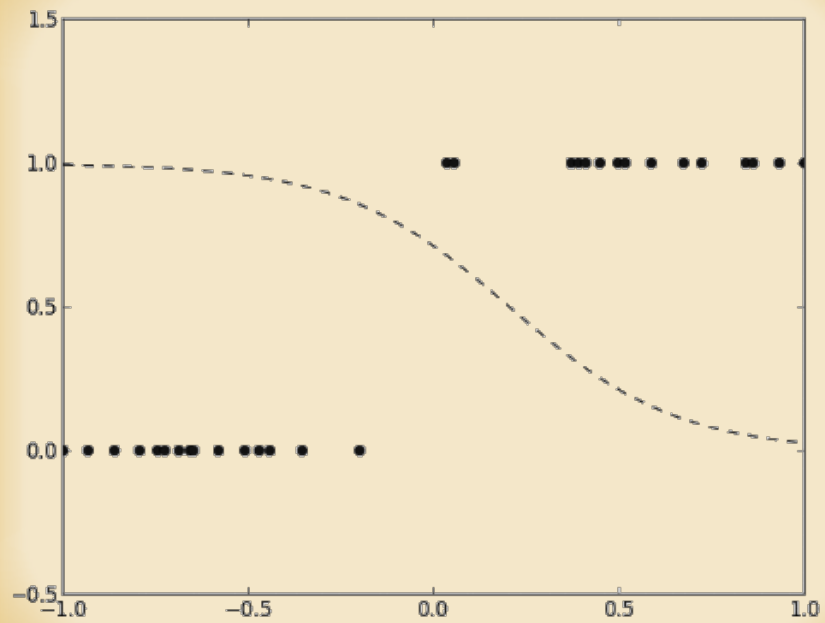
Gradient Descent



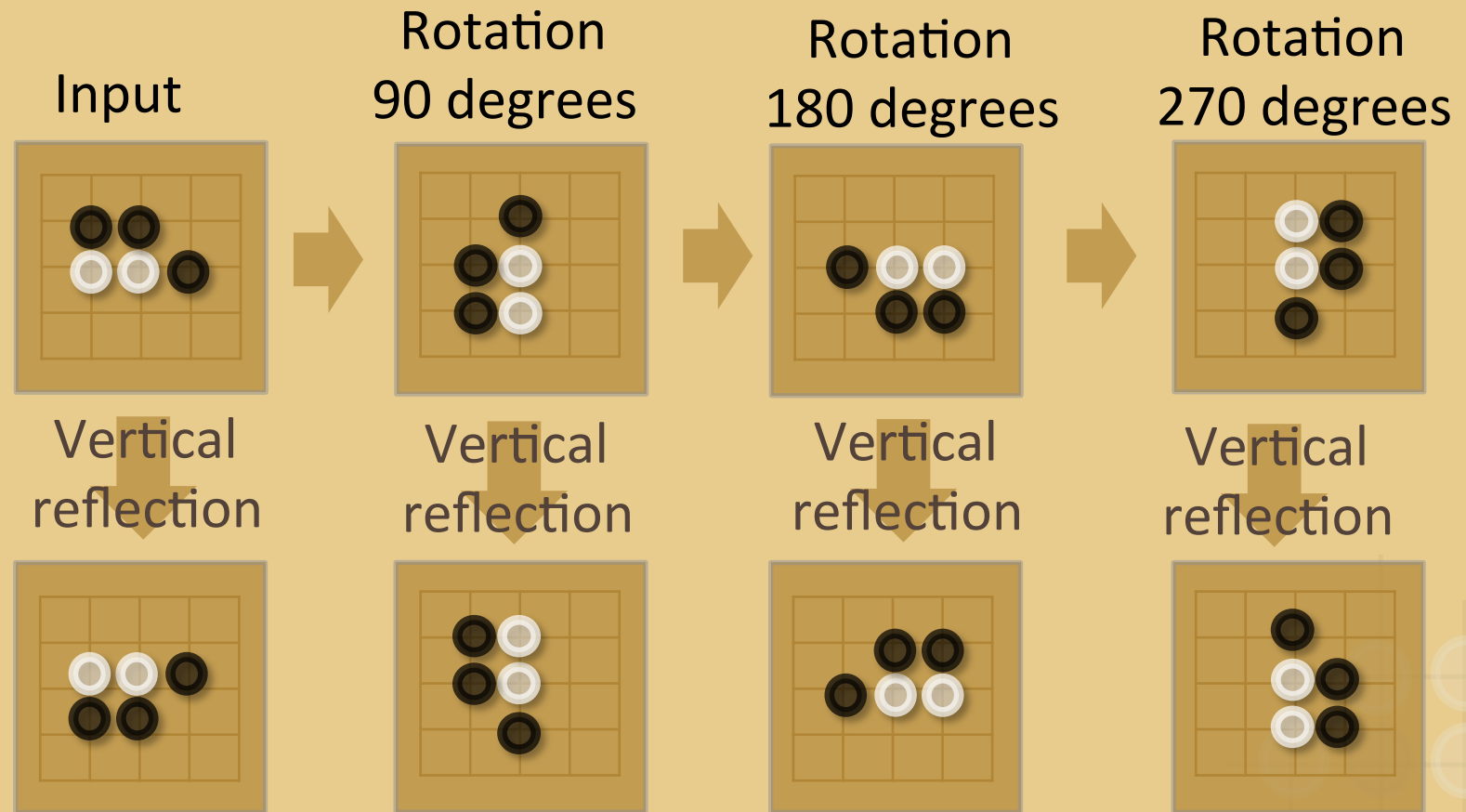
Gradient Descent



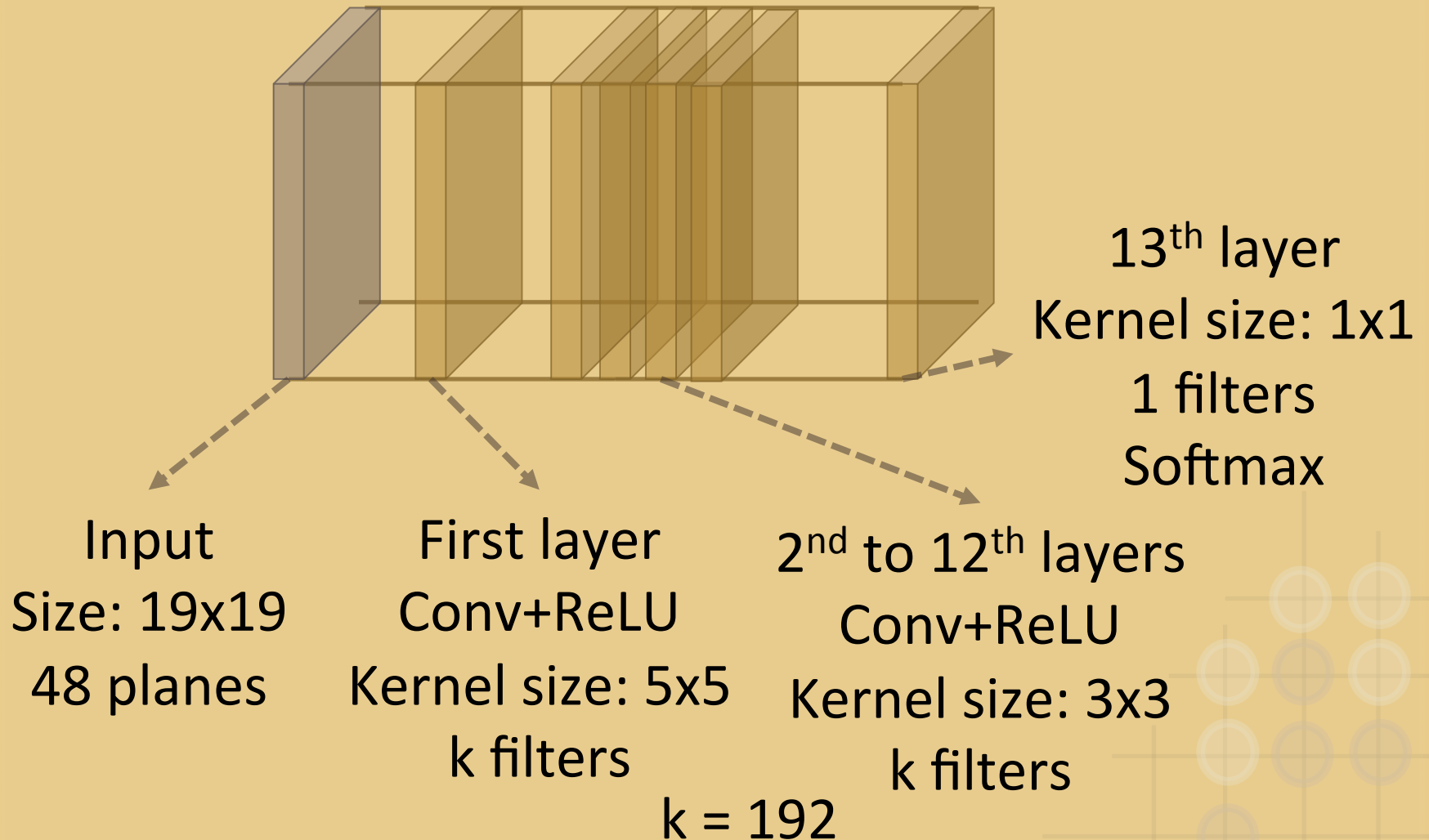
Gradient Descent



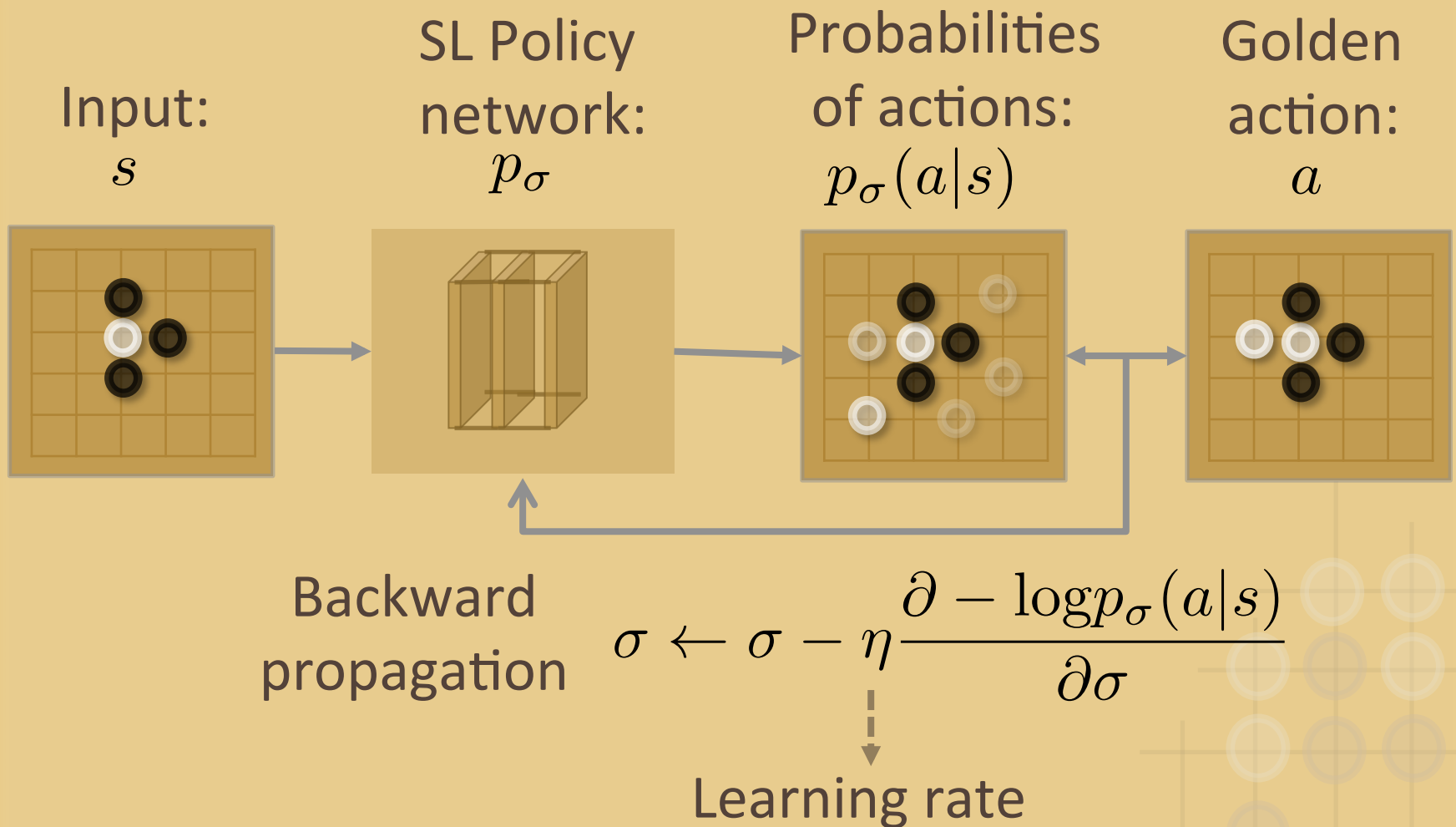
Symmetries



SL Policy Network



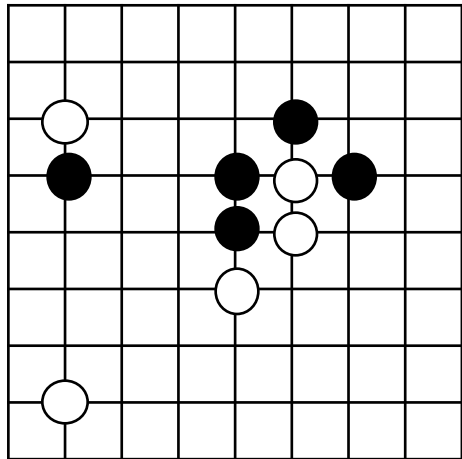
Supervised Learning : Classification



1. Reducing “action candidates”

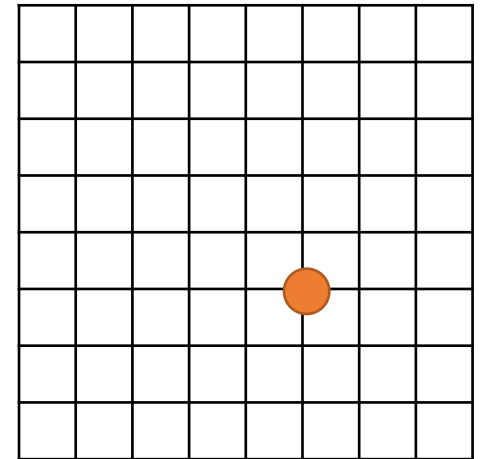
(1) Imitating expert moves (supervised learning)

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”

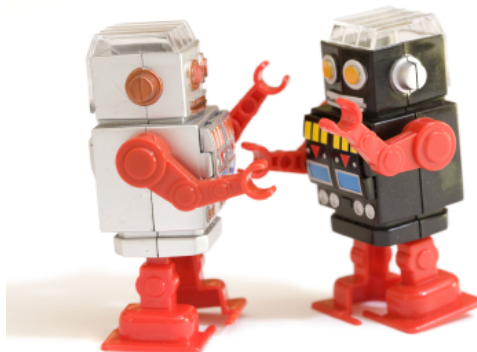
(2) Improving through self-plays (reinforcement learning)

Improving by playing against itself

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

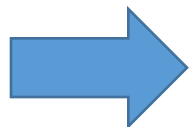
vs

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



1. Reducing “action candidates”

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

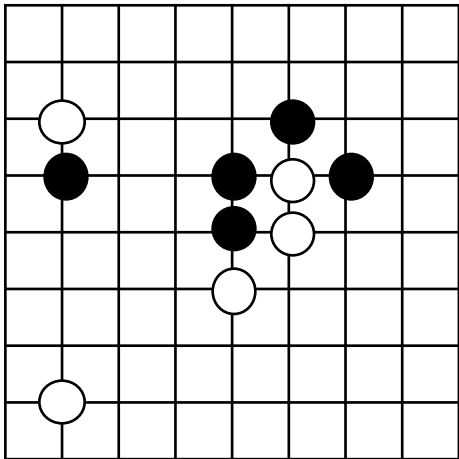


Return: board positions, win/lose info

1. Reducing “action candidates”

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

Board position



win/loss

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

Loss

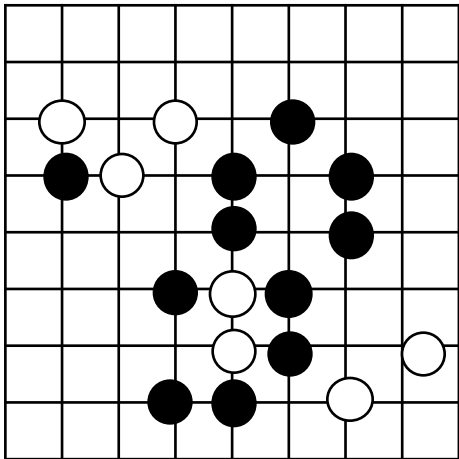
$z = -1$

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

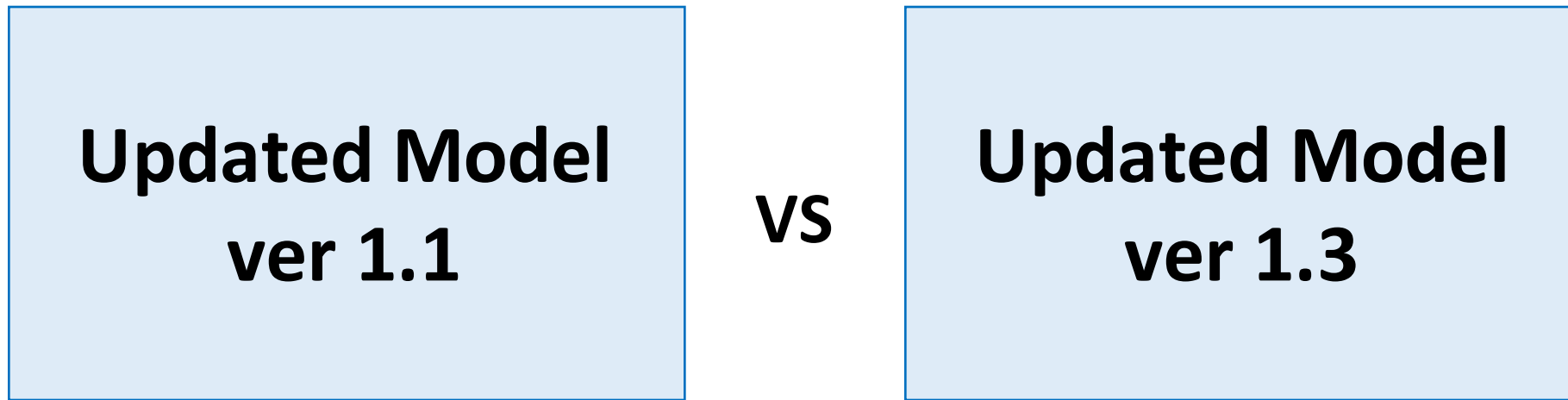
$z = +1$

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

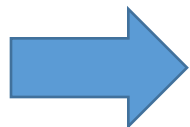
1. Reducing “action candidates”

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

Older models vs. newer models



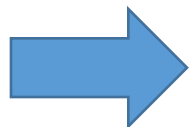
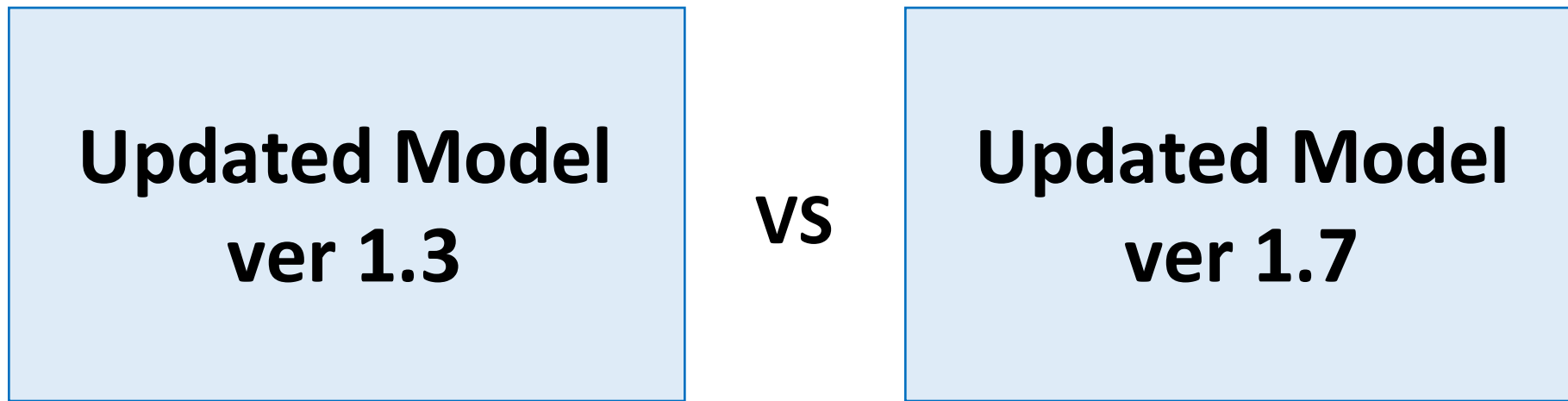
It uses the same topology as the expert moves imitator model, and just uses the updated parameters



Return: board positions, win/lose info

1. Reducing “action candidates”

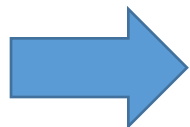
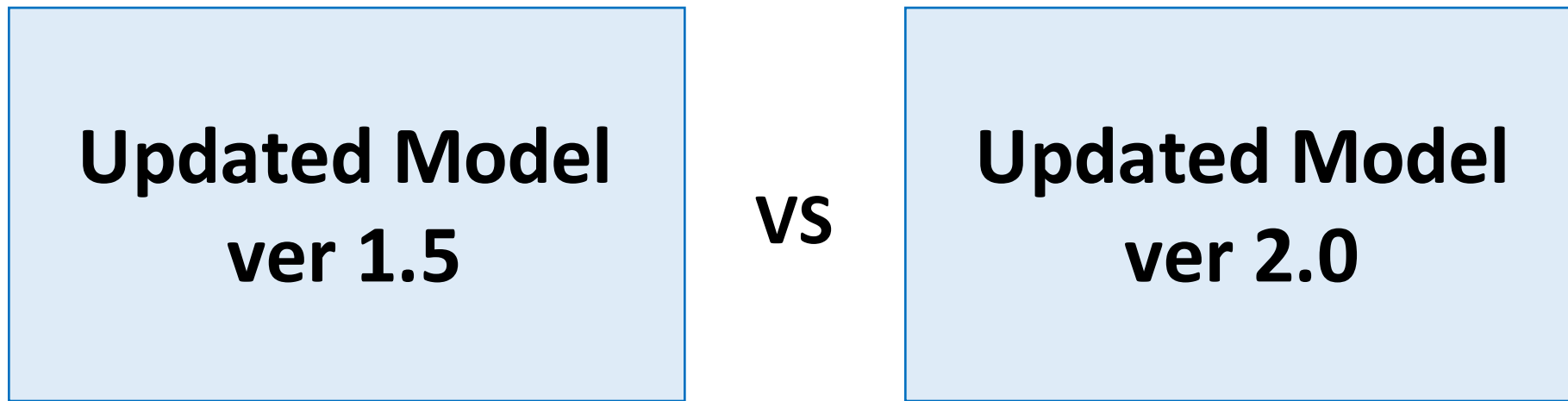
(2) Improving through self-plays (reinforcement learning)



Return: board positions, win/lose info

1. Reducing “action candidates”

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



Return: board positions, win/lose info

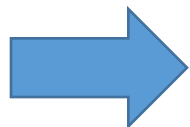
1. Reducing “action candidates”

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

**Updated Model
ver 3204.1**

VS

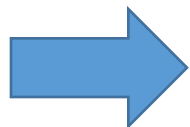
**Updated Model
ver 46235.2**



Return: board positions, win/lose info

1. Reducing “action candidates”

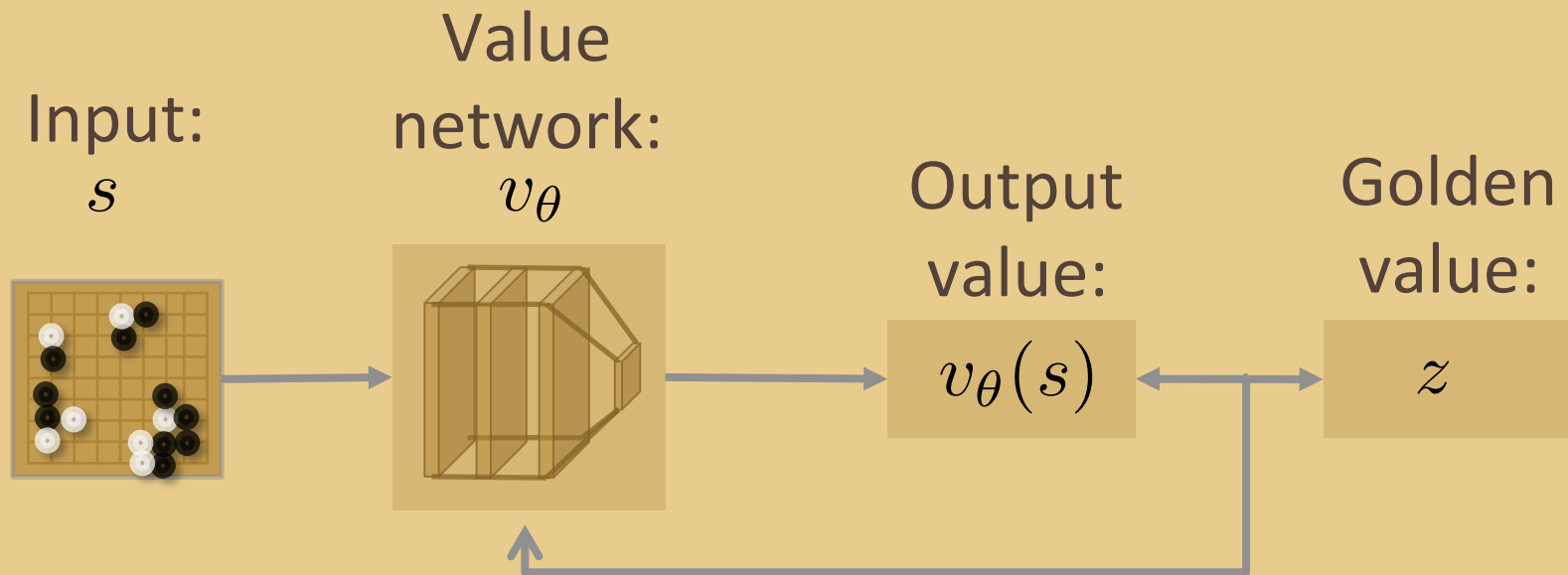
(2) Improving through self-plays (reinforcement learning)



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

2. Board Evaluation

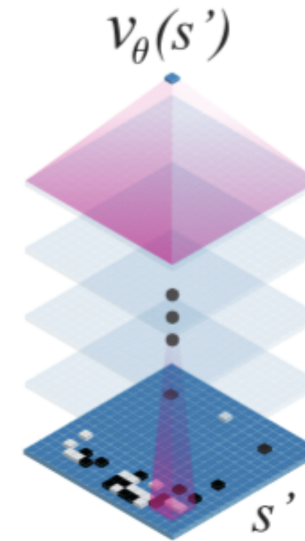
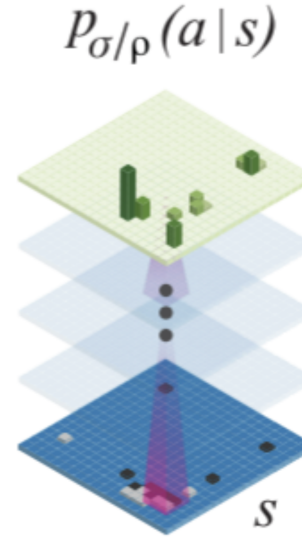
Supervised learning : Regression



Backward propagation

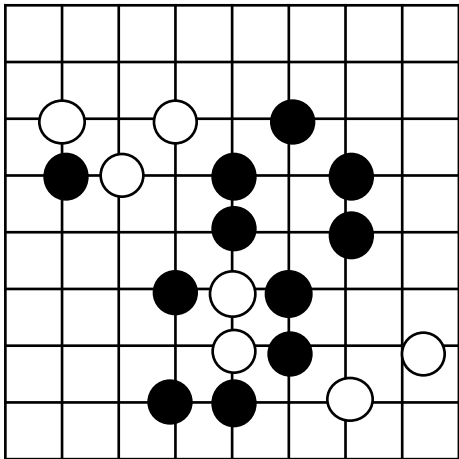
$$\theta \leftarrow \theta + \eta(z - v_\theta(s)) \frac{\partial v_\theta(s)}{\partial \theta}$$

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



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

**Value
Prediction
Model
(Regression)**

Win / Loss

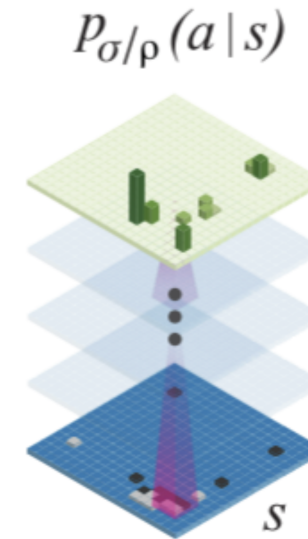
**Win
(0~1)**

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

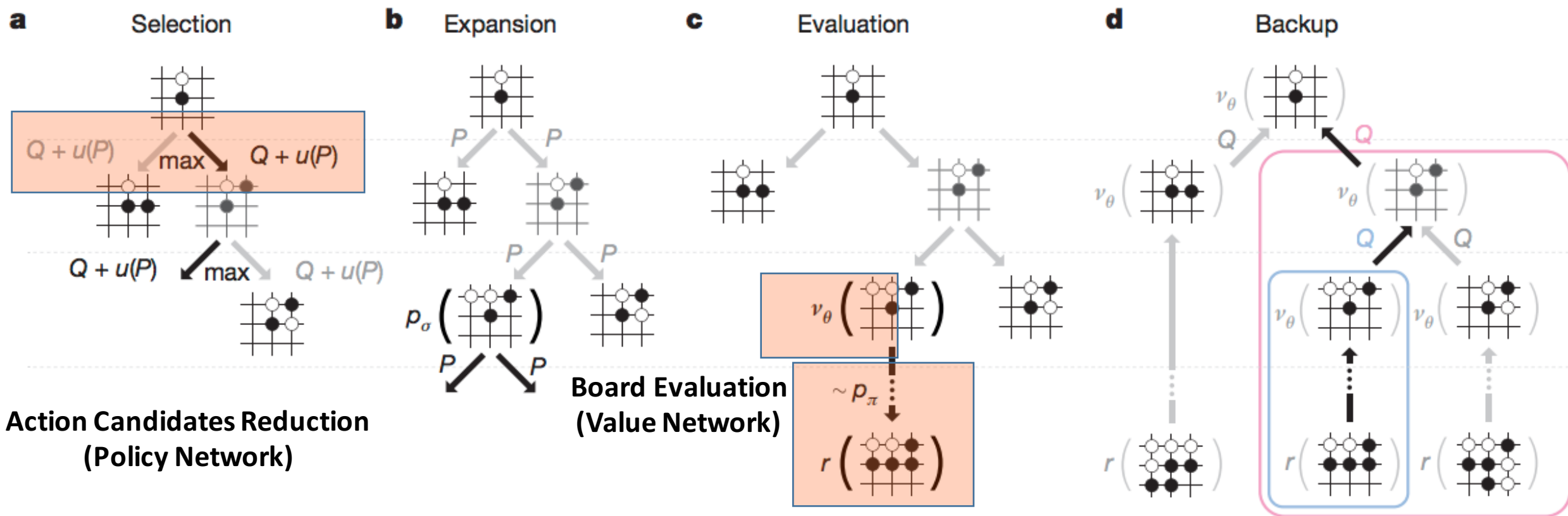


2. Board Evaluation (Depth Reduction)

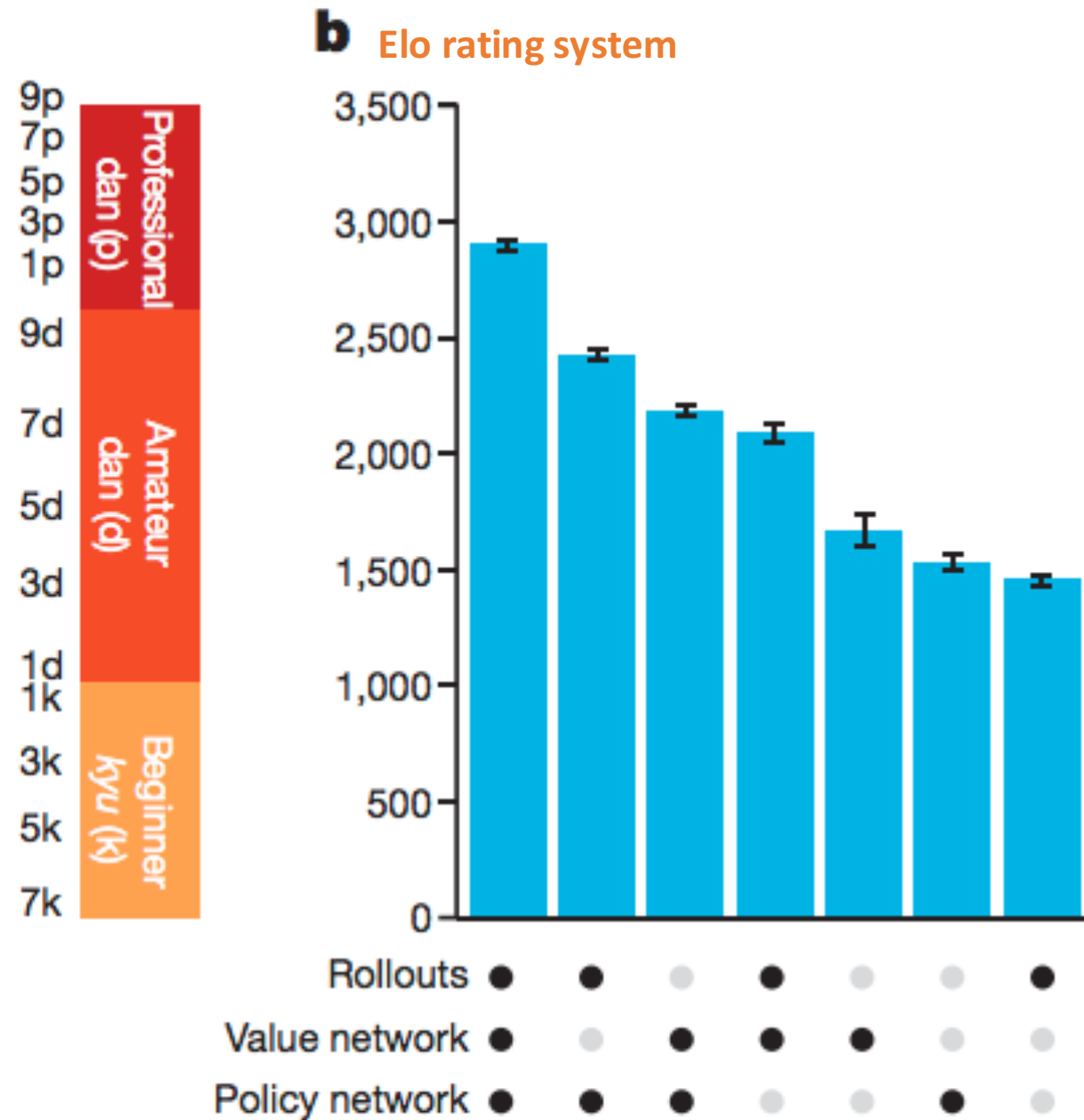
Value Network



Looking ahead (w/ Monte Carlo Search Tree)



Results

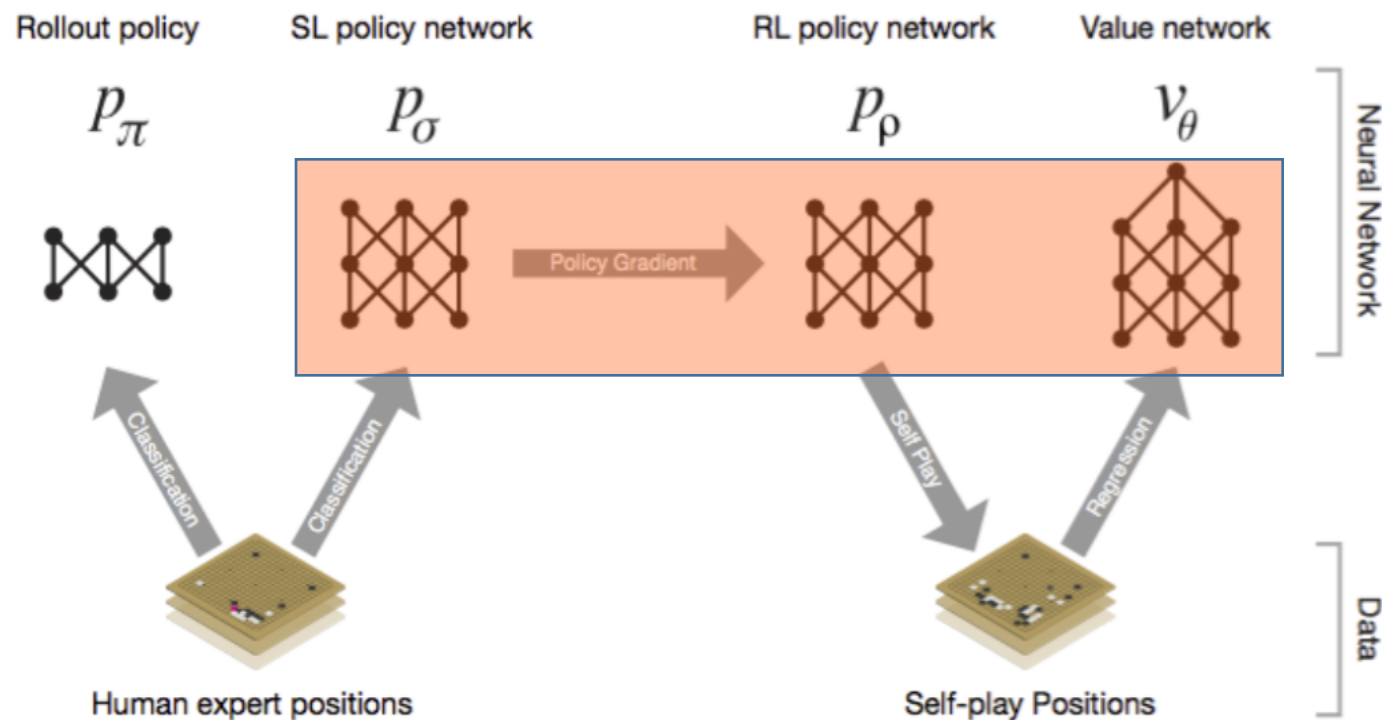


Performance with different combinations of AlphaGo components

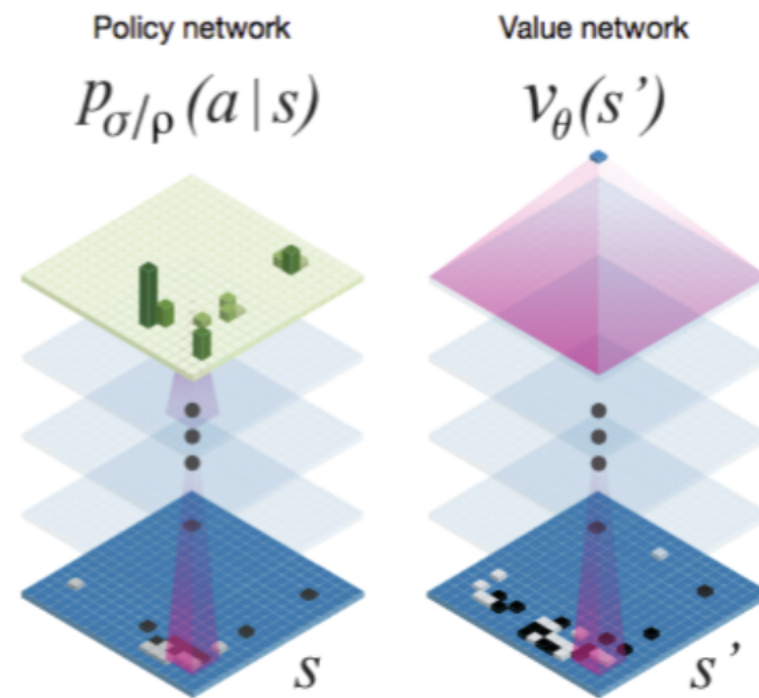
Takeaways

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

a



b



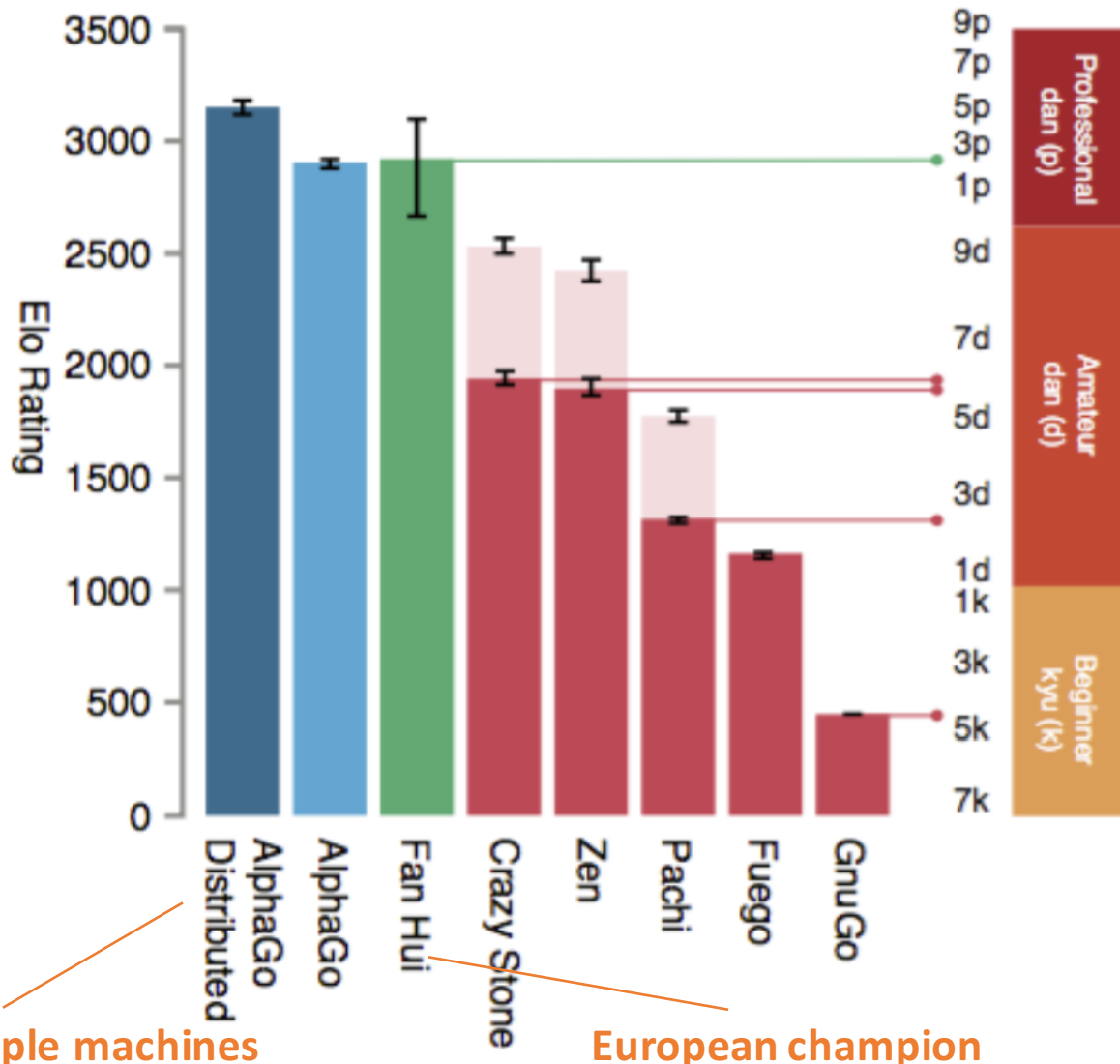
Lee Sedol 9-dan vs AlphaGo

Energy Consumption

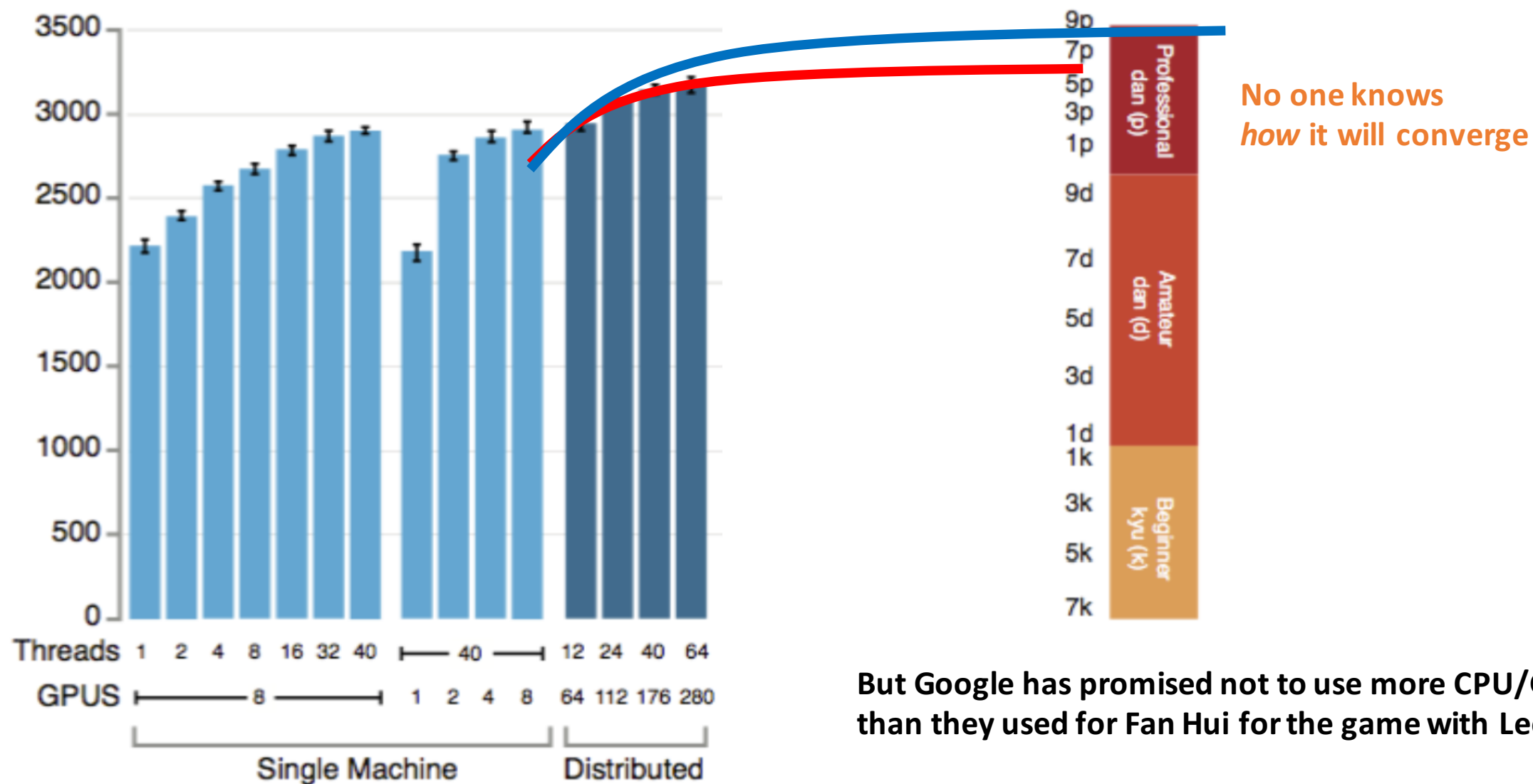
Lee Sedol	AlphaGo
- Recommended calories for a man per day : ~2,500 kCal - Assumption: Lee consumes the entire amount of per-day calories in this one game 2,500 kCal * 4,184 J/kCal $\approx 10\text{M [J]}$	- Assumption: CPU: ~100 W, GPU: ~300 W - 1,202 CPUs, 176 GPUs $170,000 \text{ J/sec} * 5 \text{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

A very, very rough calculation ;)

AlphaGo is estimated to be around ~5-dan



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

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?