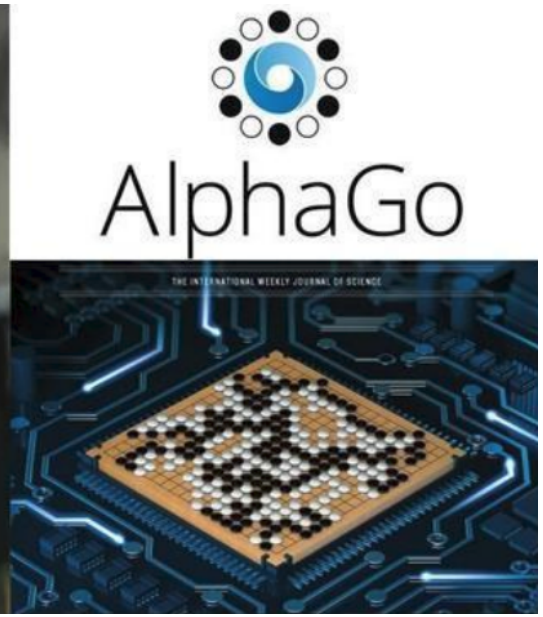
Mastering Games with Deep Learning

Tomáš Brázdil



AI in Games

Game playing = natural benchmark for AI

Tests human like intelligence in a controlled and measurable environment.

AI in Games

Game playing = natural benchmark for AI

Tests human like intelligence in a controlled and measurable environment.

Board games (chess, go, etc.):

- Ancient, widely studied and well understood by humans
- Simple and unified "interface"
- Deep strategic reasoning

AI in Games

Game playing = natural benchmark for AI

Tests human like intelligence in a controlled and measurable environment.

Board games (chess, go, etc.):

- Ancient, widely studied and well understood by humans
- Simple and unified "interface"
- Deep strategic reasoning

Some games completely solved: English draughts (ends in draw)

Some played by AI with super-human skills: Chess

Some have been considered too hard: **Go**

AI in Games

Game playing = natural benchmark for AI

Tests human like intelligence in a controlled and measurable environment.

Board games (chess, go, etc.):

- Ancient, widely studied and well understood by humans
- Simple and unified "interface"
- Deep strategic reasoning

Some games completely solved: English draughts (ends in draw)

Some played by AI with super-human skills: Chess

Some have been considered too hard: **Go**

Computer games (**Starcraft**, Call of Duty, Civilization, etc.):

- AI present from their advent

  Quality of AI is crucial in assessment of computer games.
- Typically complex graphical interface and possible behavior

Learning AI

The Goal: AI learns to play without any "hardwired" rules.

Learning AI

The Goal: AI learns to play without any "hardwired" rules.

Example: Tesauro's TD-Gammon

- Developed in 90s
- Neural networks based reinforcement learning algorithm playing backgammon
- Reached expert level by self-play
- Except input-output, no part of the algorithm specific for backgammon

Compare with DeepBlue, master level chess program, which uses lots of chess-specific heuristics.

Learning AI

The Goal: AI learns to play without any "hardwired" rules.

Example: Tesauro's TD-Gammon

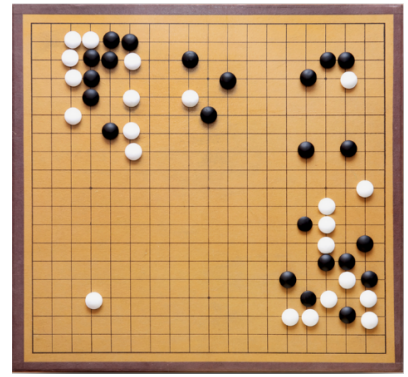
- Developed in 90s
- Neural networks based reinforcement learning algorithm playing backgammon
- Reached expert level by self-play
- Except input-output, no part of the algorithm specific for backgammon

Compare with DeepBlue, master level chess program, which uses lots of chess-specific heuristics.

This talk:

- Self-playing learning algorithm for Go
- Starcraft 2 challenge

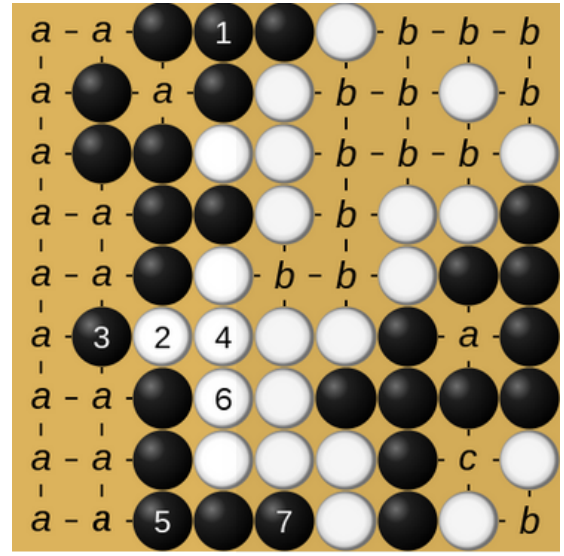
GO



Rules (roughly):

- Two players
- Take turns placing stones on the vacant intersections ("points") of a board with a 19x19 grid of lines
- Stones cannot be moved but are removed from the board when surrounded by opposing stones (captured)

- The game proceeds until neither player wishes to make another move
- The surrounded area (territory) is counted along with captured stones

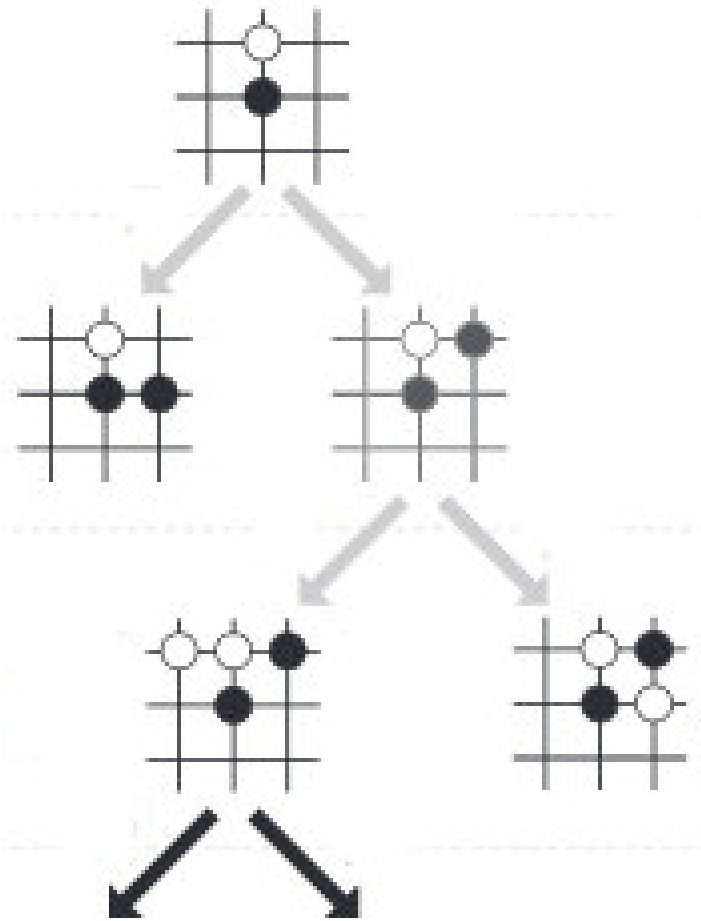


Basic notation

- States

  Each state is controlled by one of the players, **Li** and **Wang**

  GO: Current board configuration



Basic notation

- States

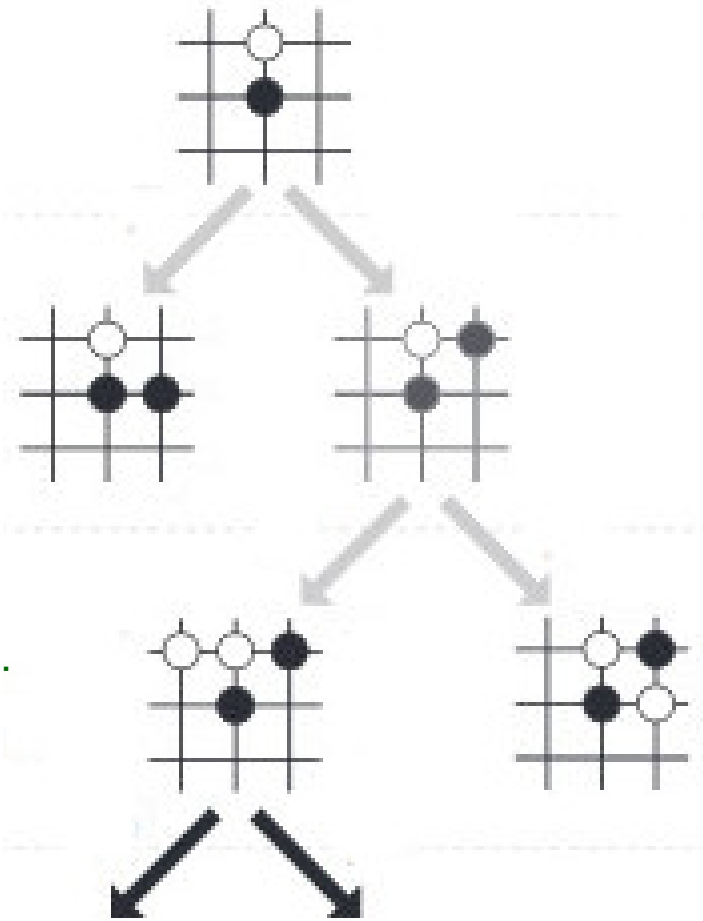
  Each state is controlled by one of the players, **Li** and **Wang**

  GO: Current board configuration

- Actions

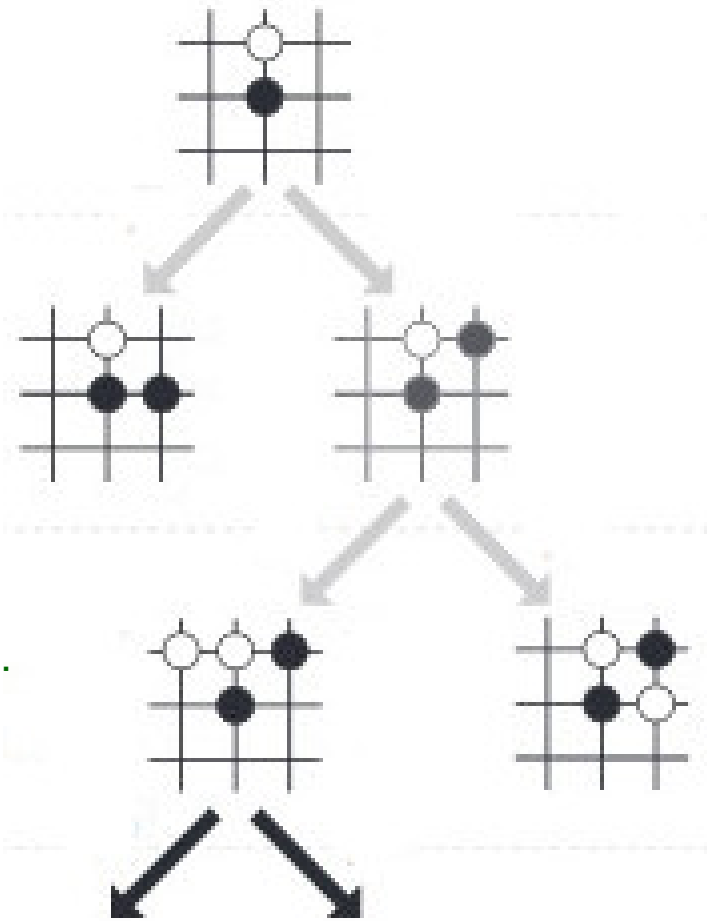
  Each state is assigned a set of *enabled actions*

  GO: Possible placements of a new stone, pass.



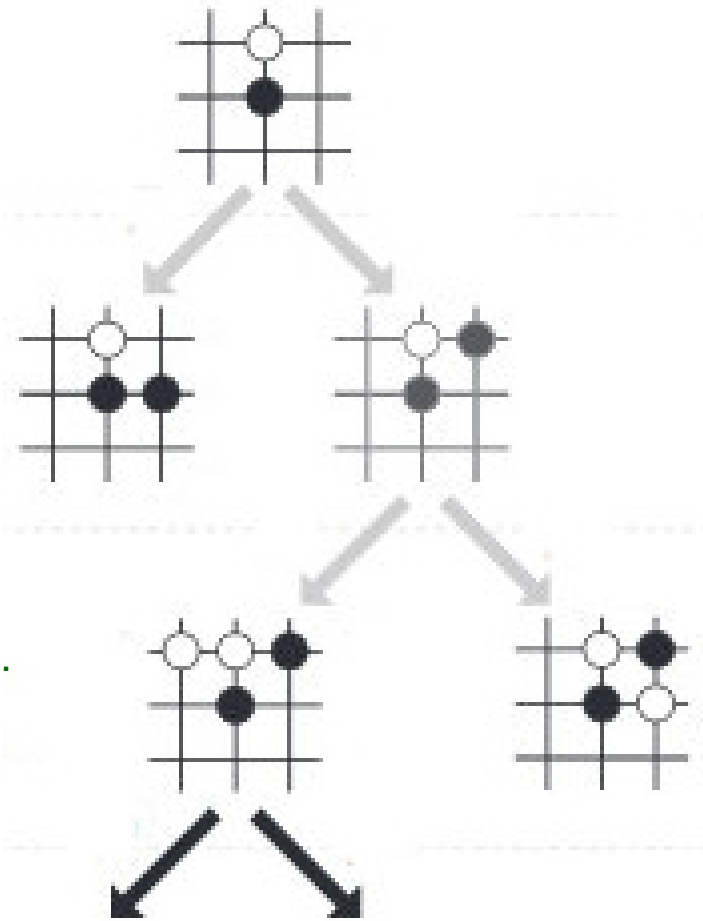
Basic notation

- States
  Each state is controlled by one of the players, **Li** and **Wang**
  GO: Current board configuration
- Actions
  Each state is assigned a set of *enabled actions*
  GO: Possible placements of a new stone, pass.
- Transition function assigning to each state and action the resulting state
  GO: New board configuration.



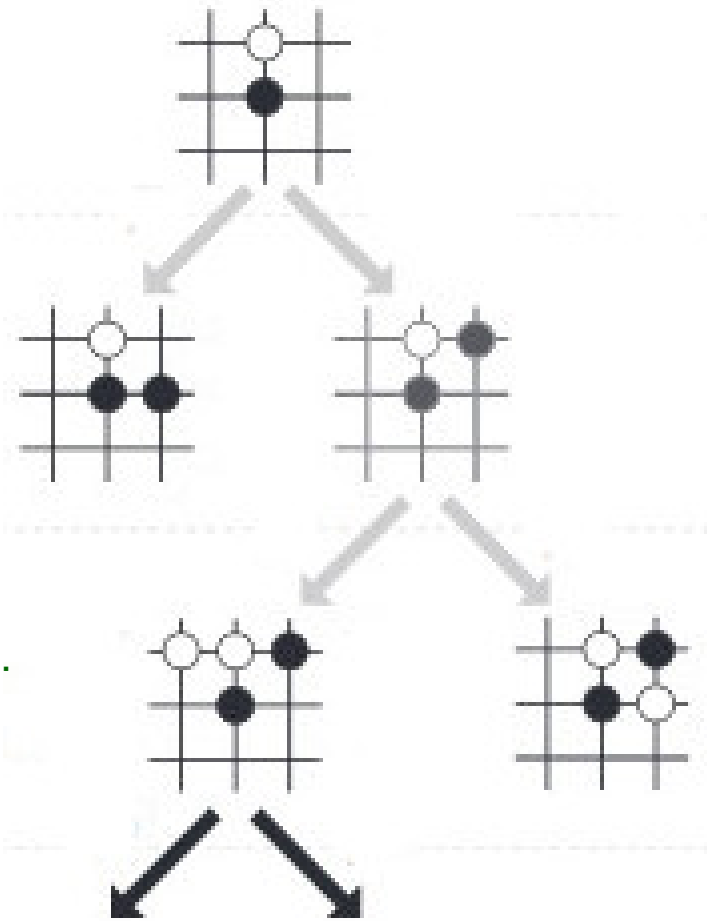
Basic notation

- States
  Each state is controlled by one of the players, **Li** and **Wang**
  GO: Current board configuration
- Actions
  Each state is assigned a set of *enabled actions*
  GO: Possible placements of a new stone, pass.
- Transition function assigning to each state and action the resulting state
  GO: New board configuration.
- Terminal states, with winner assigned
  GO: Both players pass, one of them wins.



Basic notation

- States
  Each state is controlled by one of the players, **Li** and **Wang**
  GO: Current board configuration
- Actions
  Each state is assigned a set of *enabled actions*
  GO: Possible placements of a new stone, pass.
- Transition function assigning to each state and action the resulting state
  GO: New board configuration.
- Terminal states, with winner assigned
  GO: Both players pass, one of them wins.



Strategy of a player is a function which to every state controlled by the player assigns an enabled action (possibly in random).

Backward induction

How to find an optimal strategy?

Evaluate possible transitions from the end to the start

Backward induction

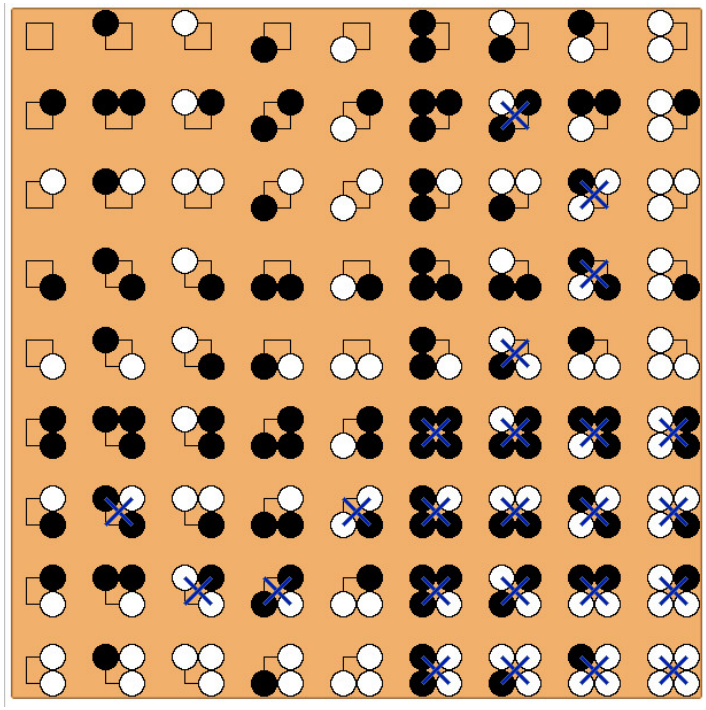
How to find an optimal strategy?

Evaluate possible transitions from the end to the start

... but backward induction does not work for larger games ...

Size of GO

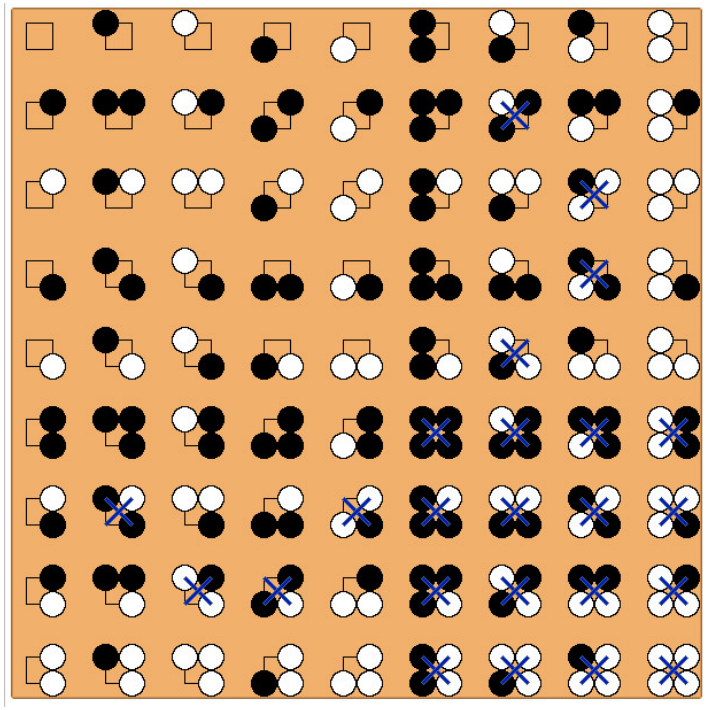
2x2 variant: 81 possible board positions, out of which 57 legal.



How many possible plays?

Size of GO

2x2 variant: 81 possible board positions, out of which 57 legal.

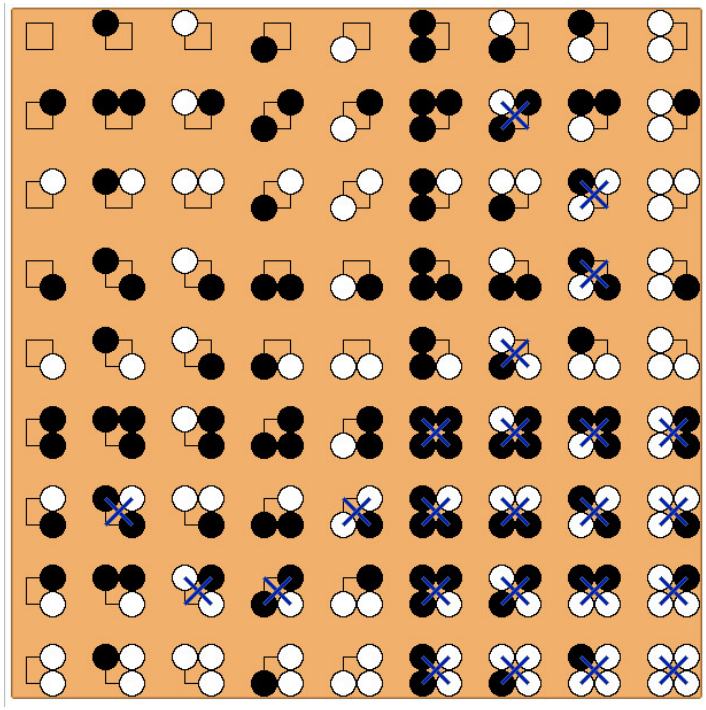


How many possible plays? Under Tromp-Taylor rules 386,356,909,593.

What about 19x19 ?

Size of GO

2x2 variant: 81 possible board positions, out of which 57 legal.



How many possible plays? Under Tromp-Taylor rules 386,356,909,593.

What about 19x19 ?

Legal game positions: $2.08168199382 * 10^{170}$
Branching degree approx. 250

How to find a good strategy in GO?

Exhaustive search (backward induction) impossible.

Heuristic/approximate/learning solutions possible but only (strong) amateur level.

How to find a good strategy in GO?

Exhaustive search (backward induction) impossible.

Heuristic/approximate/learning solutions possible but only (strong) amateur level.

Silver et al. Mastering the game of Go with deep neural networks and tree search. Nature 529(7587): 484-489 (2016)

achieved grand-master level using three well known concepts:

- Monte Carlo tree search

  search for best actions
- Reinfocement learning

  where search infeasible
- Deep learning (neural networks)

  represent intermediate values, strategies, etc.

How to find a good strategy in GO?

Exhaustive search (backward induction) impossible.

Heuristic/approximate/learning solutions possible but only (strong) amateur level.

Silver et al. Mastering the game of Go with deep neural networks and tree search. Nature 529(7587): 484-489 (2016)

achieved grand-master level using three well known concepts:

- Monte Carlo tree search
  search for best actions
- Reinfocement learning
  where search infeasible
- Deep learning (neural networks)
  represent intermediate values, strategies, etc.

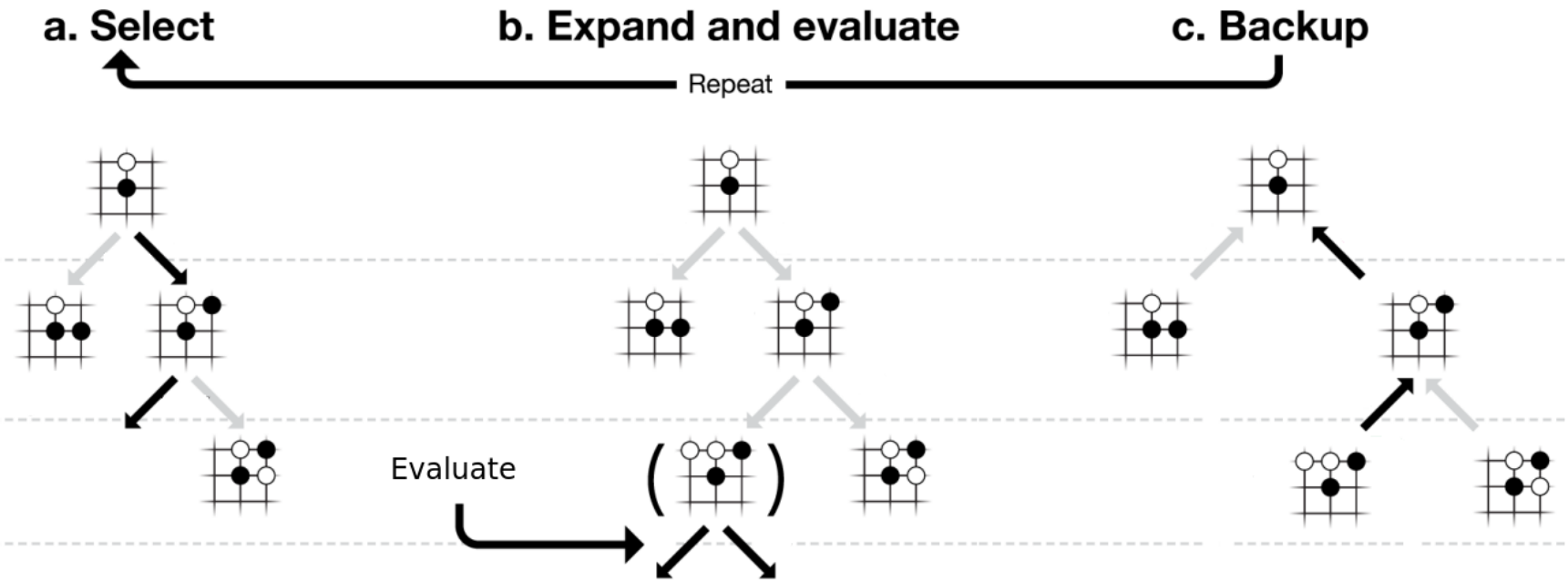
Beaten by
Silver et al. Mastering the game of Go without human knowledge. Nature 550: 354–359 (September 2017)

Monte Carlo tree search

Combines backward induction with simulations.

Monte Carlo tree search

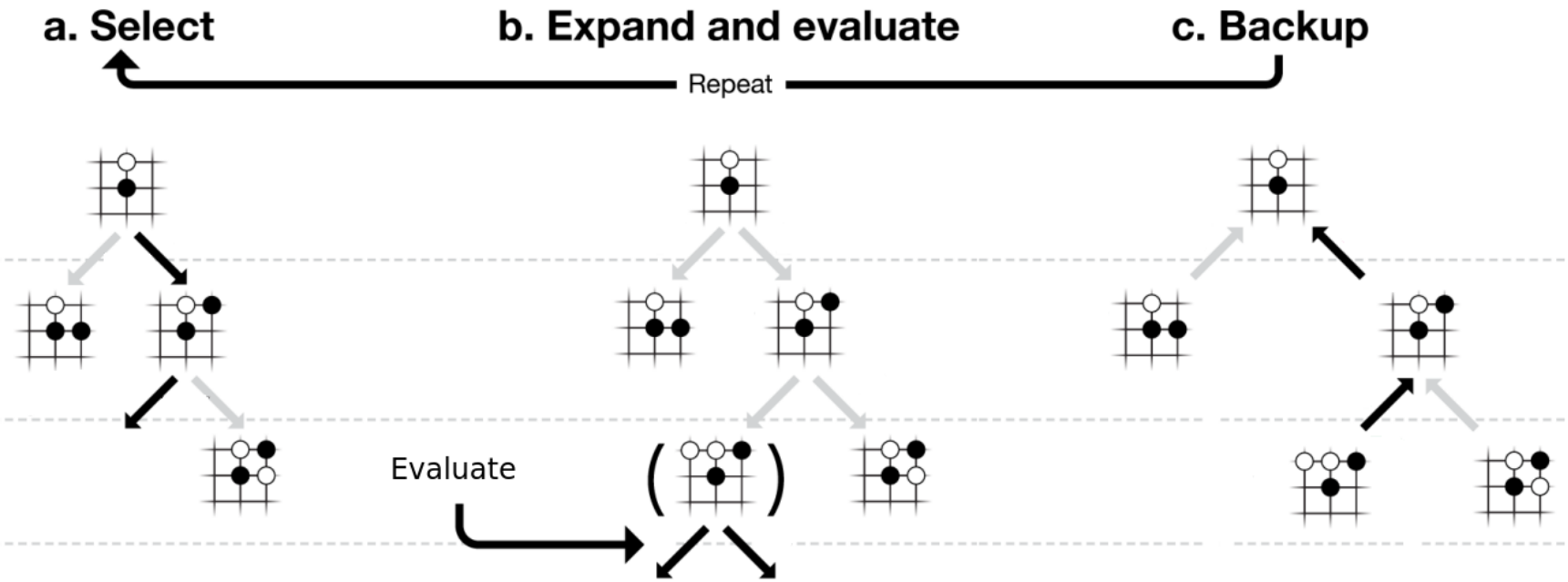
Combines backward induction with simulations.



To make a decision in a particular state s_{root}

Monte Carlo tree search

Combines backward induction with simulations.

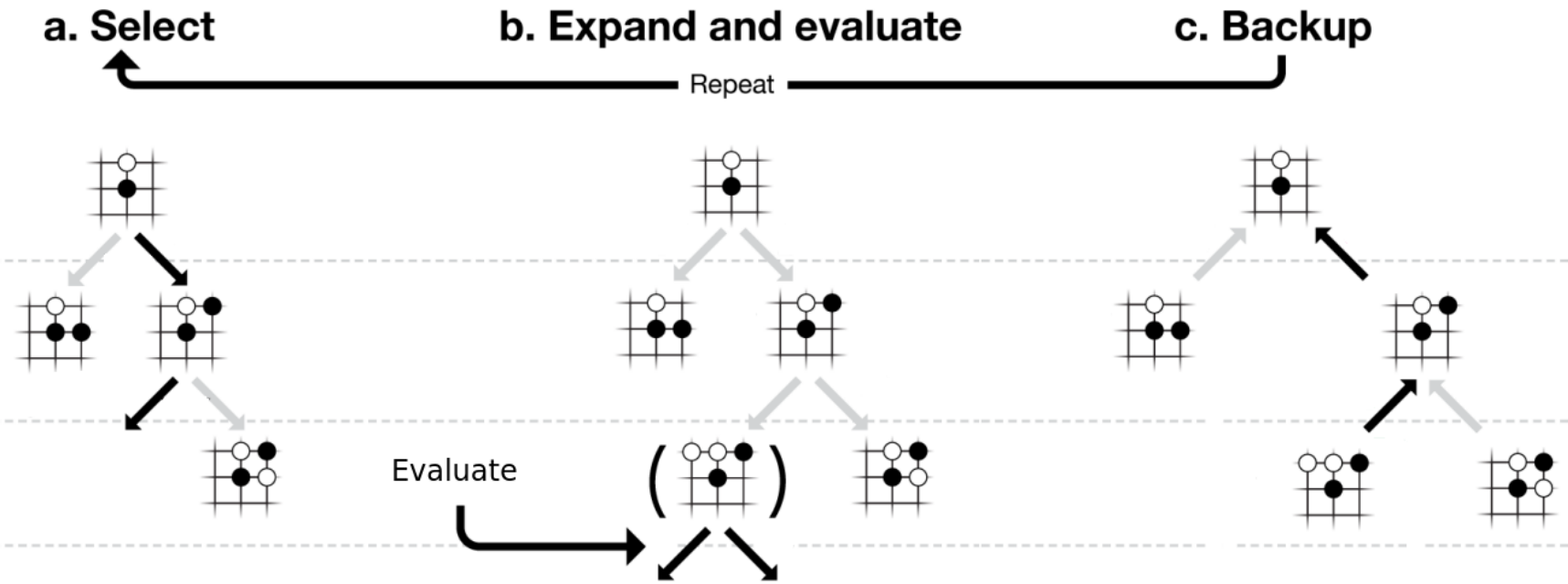


To make a decision in a particular state s_{root}

- iteratively grow a lookahead tree rooted in s_{root}

Monte Carlo tree search

Combines backward induction with simulations.

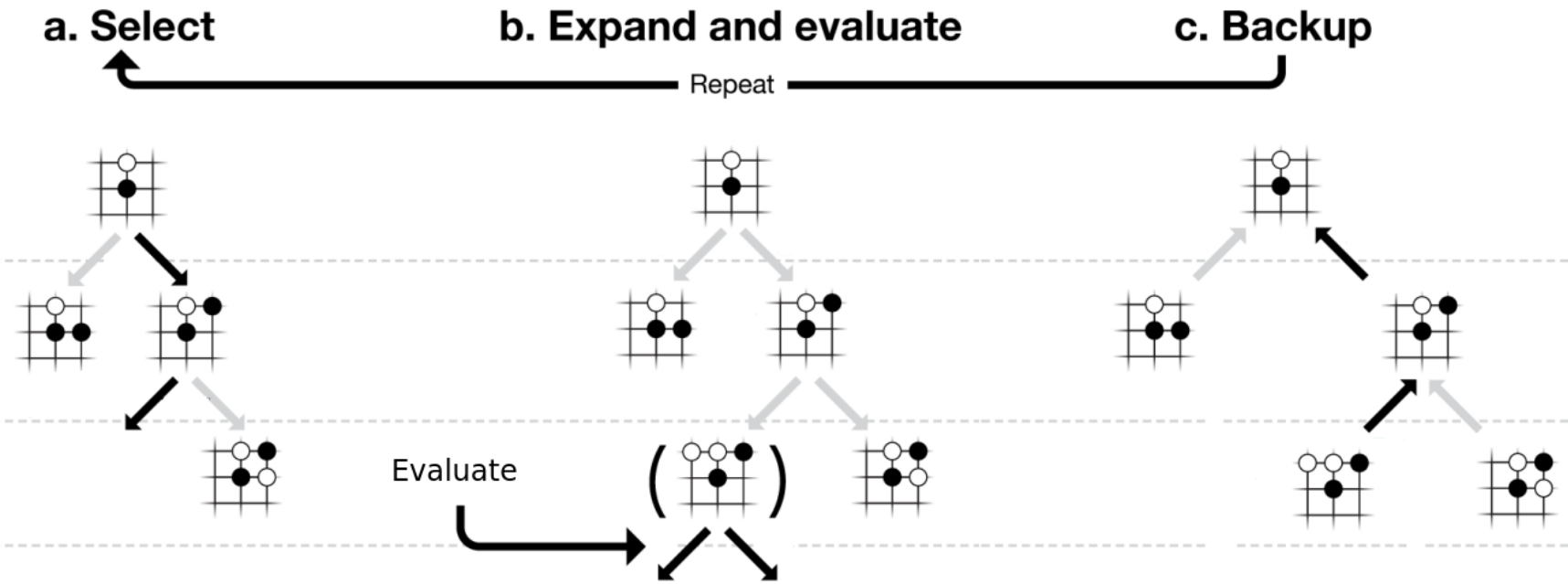


To make a decision in a particular state s_{root}

- iteratively grow a lookahead tree rooted in s_{root}
- in every iteration
  - add a new leaf at the end of the "most promising path"
  - estimate the value of the game in the added leaf
  - backpropagate the value to the root of the tree

Monte Carlo tree search

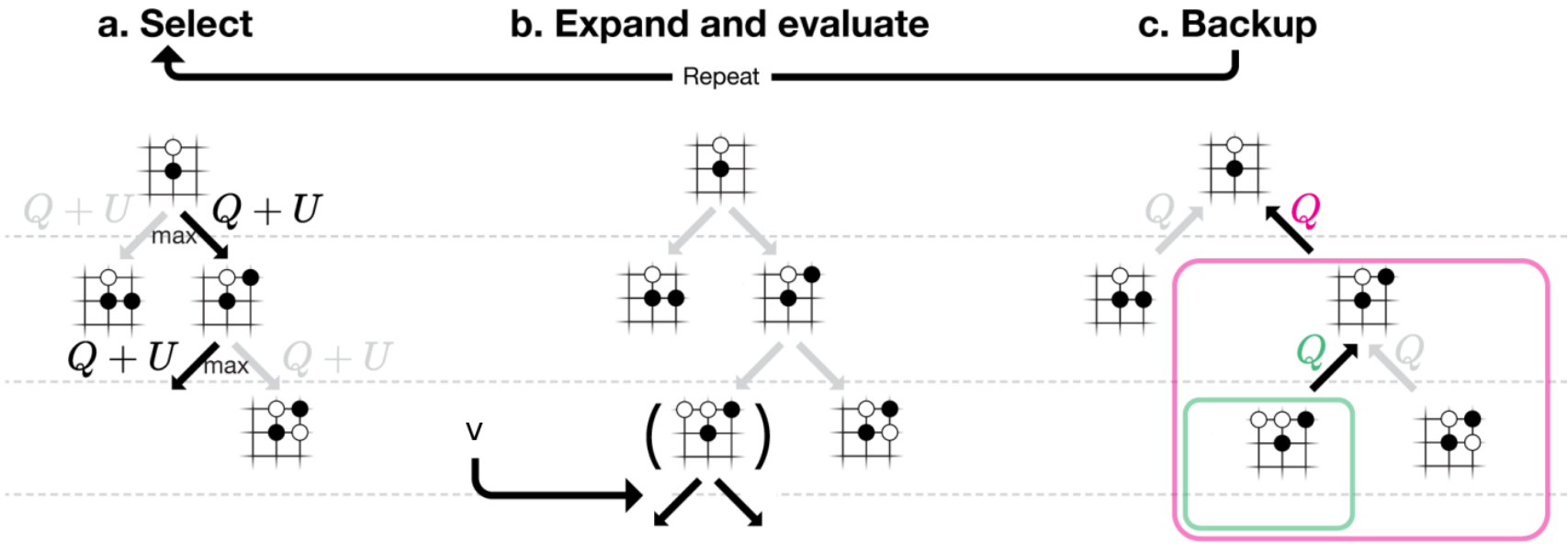
Combines backward induction with simulations.



To make a decision in a particular state s_{root}

- iteratively grow a lookahead tree rooted in s_{root}
- in every iteration
  - add a new leaf at the end of the "most promising path"
  - estimate the value of the game in the added leaf
  - backpropagate the value to the root of the tree

How to select the nodes? How to evaluate?

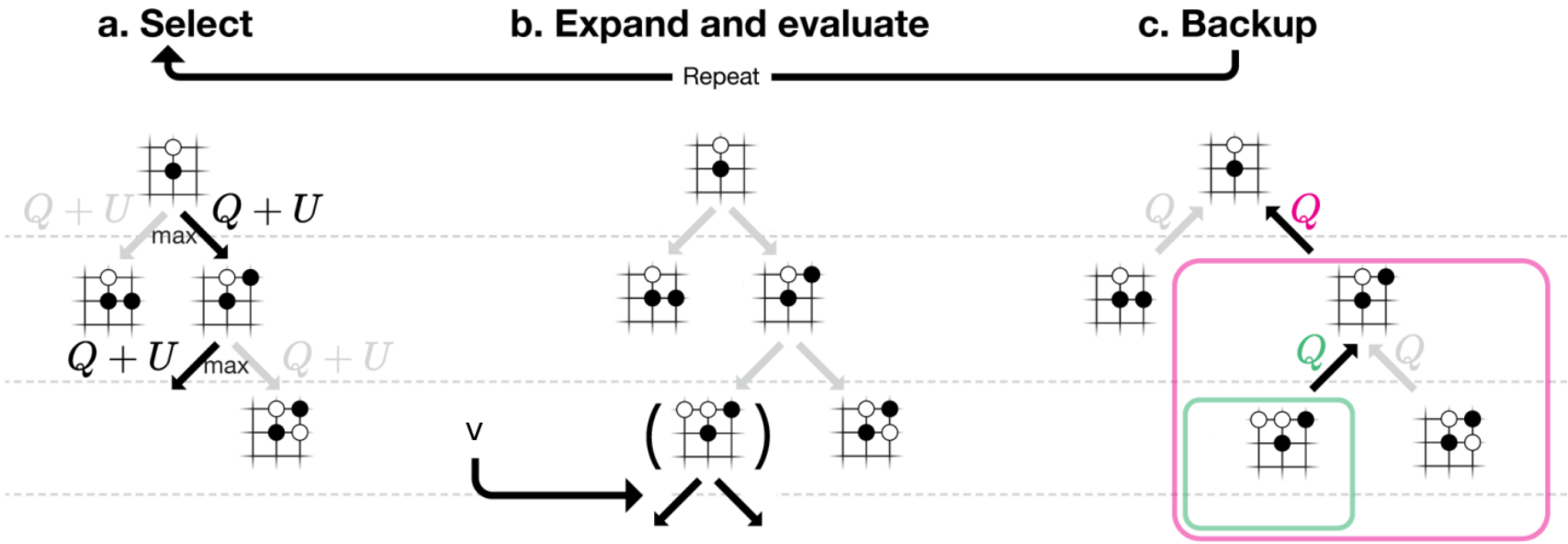


f_θ = parametrized function such that for every state s

$$(\mathbf{p}, v) = f_\theta(s)$$

- $\mathbf{p}$ = probability distribution on actions enabled in s
- v = prior value estimate for s

f_θ computed by a deep neural network, θ weights of the network



f_θ = parametrized function such that for every state s

$$(\mathbf{p}, v) = f_\theta(s)$$

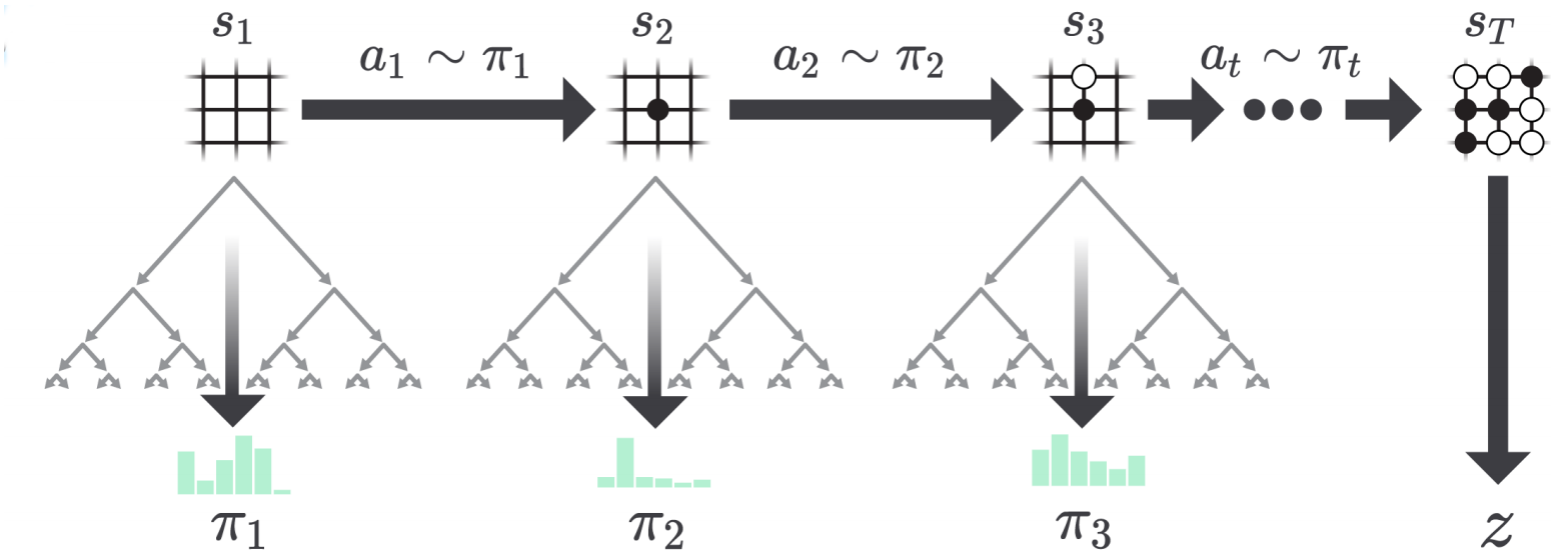
- $\mathbf{p}$ = probability distribution on actions enabled in s
- v = prior value estimate for s

f_θ computed by a deep neural network, θ weights of the network

MCTS uses v to get *mean action value* Q and uses $\mathbf{p}$ to get an *upper bound* U which influences the selection.

Strategy

In every step perform the MCTS.



Action a_t enabled in state s_t is chosen with the probability

$$\pi_t(a_t) = \frac{N(s_t, a_t)^{1/\tau}}{N(s_t)} \quad \text{where} \quad N(s_t) = \sum_{a'} N(s_t, a')^{1/\tau}$$

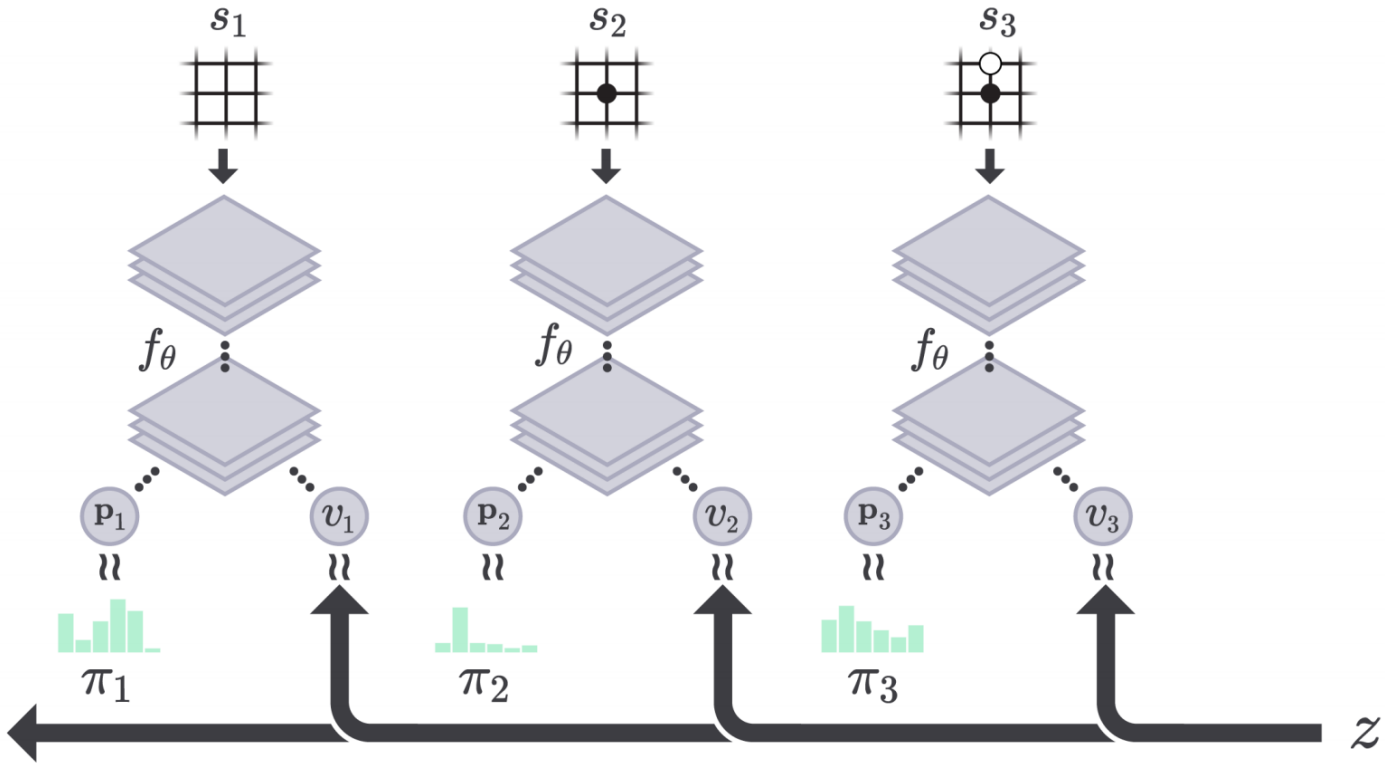
$N(s_t, a_t)$ = the number of times MCTS rooted in s_t takes the transition (s_t, a_t).

$\tau \to 0 \approx$ maximum probability action

Training

f_θ is trained in self-play reinforcement learning using MCTS.

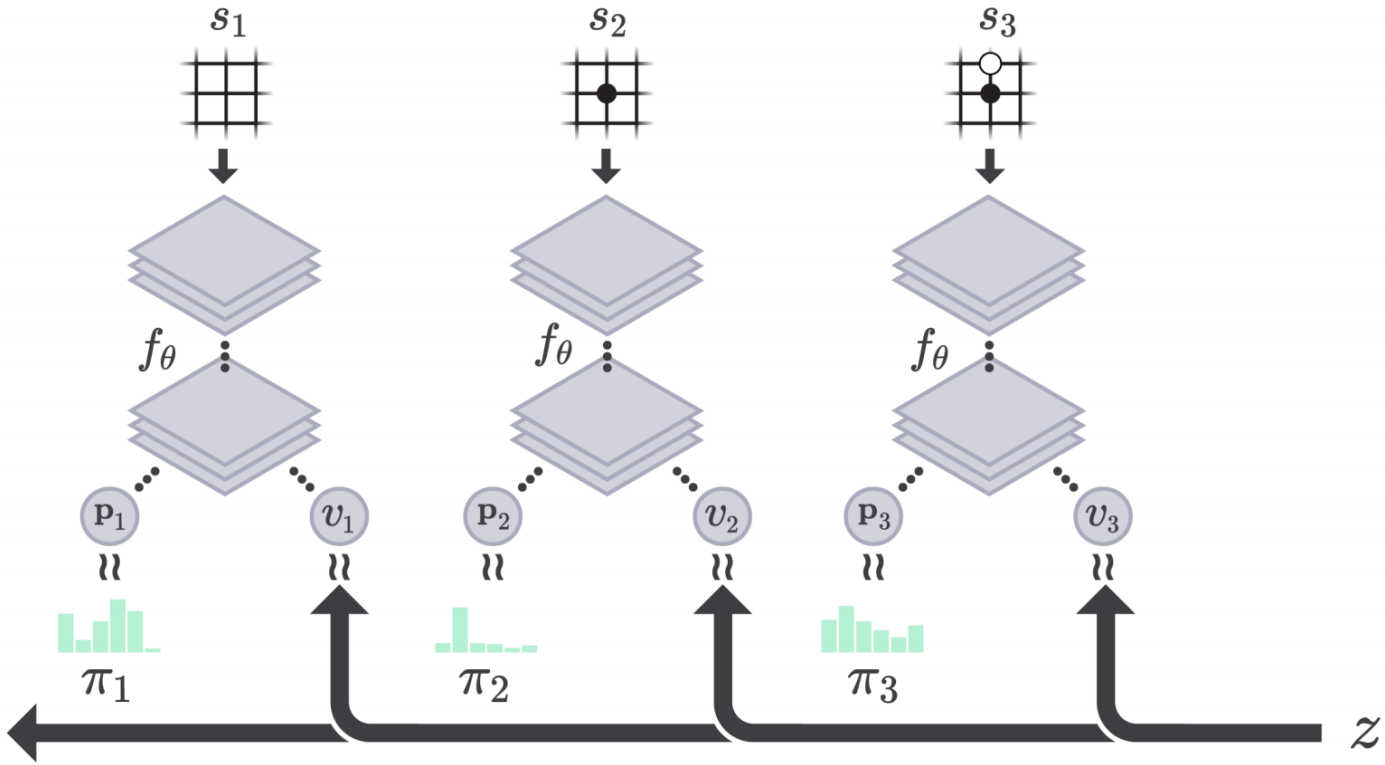
Play according to f_θ using MCTS till the end of the game.



Training

f_θ is trained in self-play reinforcement learning using MCTS.

Play according to f_θ using MCTS till the end of the game.

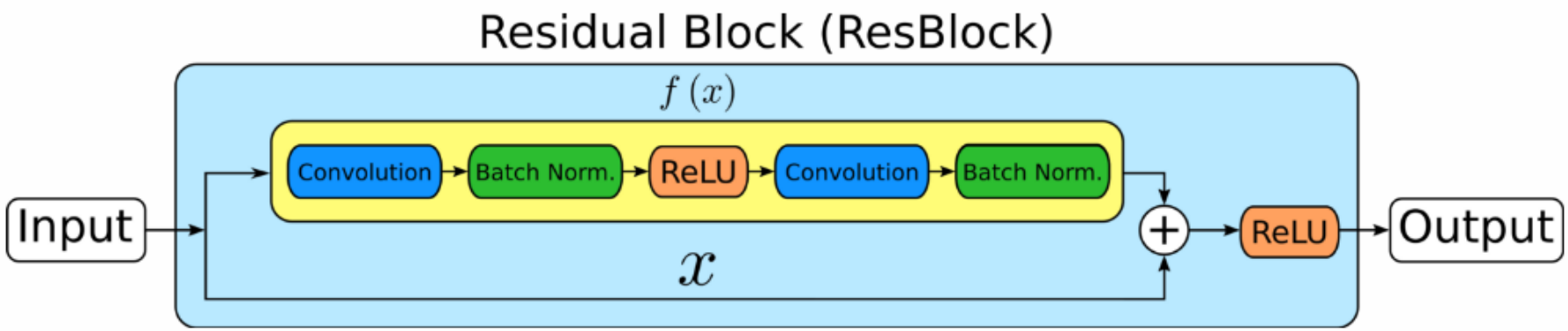


Adjust θ so that for $(\mathbf{p}_t, v_t) = f_\theta(s_t)$

- $\mathbf{p}_t$ gets closer to π_t
- v_t gets closer to the true result z of the game

  $z = \pm 1$ from the point of view of the owner of s_t

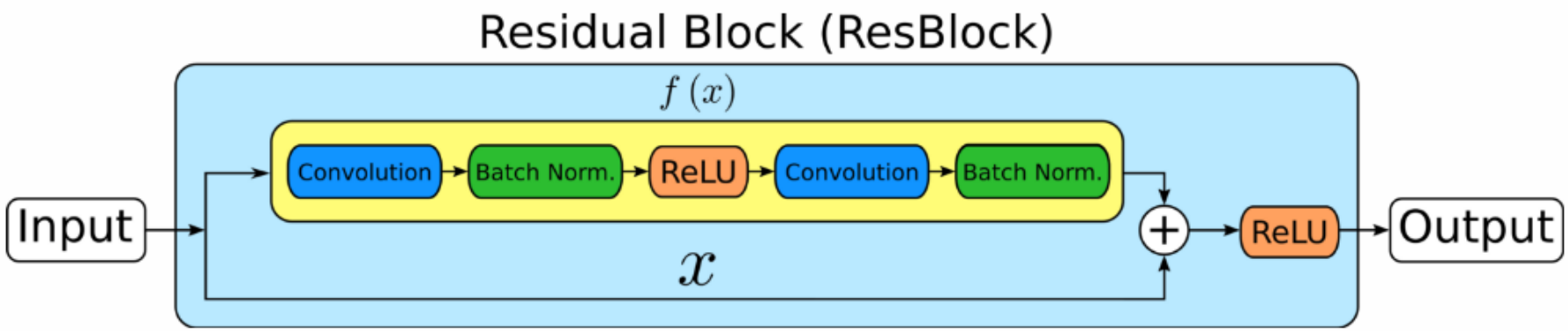
Network & Training



Network f_θ = 1 convolutional followed by 39 residual blocks

Input: 8 last raw board configurations

Network & Training



Network f_θ = 1 convolutional followed by 39 residual blocks

Input: 8 last raw board configurations

Training:

- Randomly initialized
- 4.9 million games of self-play
- 1,600 simulations for each MCTS, approximately 0.4s thinking time per move

Comparison

	AlphaGo Zero (new version)	AlphaGo Lee (older version)
Prior info	None, initialized randomly	database of human plays
Network	1 conv. (256 filters each) + 39 residual	12 conv. (256 filters each)
Training HW	64 GPU + 19 CPU	50 GPU
Training	36 hours	month
Playing HW	one machine, 4 TPUs	many machines, 48 TPUs
Defeated	AlphaGo Lee, 100:0	best players in the world, best computer programs: Crazy Stone, Zen, Pachi and Fuego

AlphaGo in action

AlphaGo Lee (older version) defeated Lee Sedol

9. dan, ranked second in international titles

- Chinese rules with a 7.5-point komi

  7.5 points added to the score of the white player for playing second
- Two-hour set time limit for each player followed by three 60-second byo-yomi overtime periods
- AlphaGo Lee won four games, Lee Sedol one

Lee apologized for his losses, stating after game three that he misjudged the capabilities of AlphaGo and felt powerless.

AlphaGo in action

AlphaGo Lee (older version) defeated Lee Sedol

9. dan, ranked second in international titles

- Chinese rules with a 7.5-point komi

  7.5 points added to the score of the white player for playing second
- Two-hour set time limit for each player followed by three 60-second byo-yomi overtime periods
- AlphaGo Lee won four games, Lee Sedol one

Lee apologized for his losses, stating after game three that he misjudged the capabilities of AlphaGo and felt powerless.

AlphaGo Lee (older version) won 494 out of 495 games against best computer programs, each program given 5s for each move.

AlphaGo in action

AlphaGo Lee (older version) defeated Lee Sedol

9. dan, ranked second in international titles

- Chinese rules with a 7.5-point komi

  7.5 points added to the score of the white player for playing second
- Two-hour set time limit for each player followed by three 60-second byo-yomi overtime periods
- AlphaGo Lee won four games, Lee Sedol one

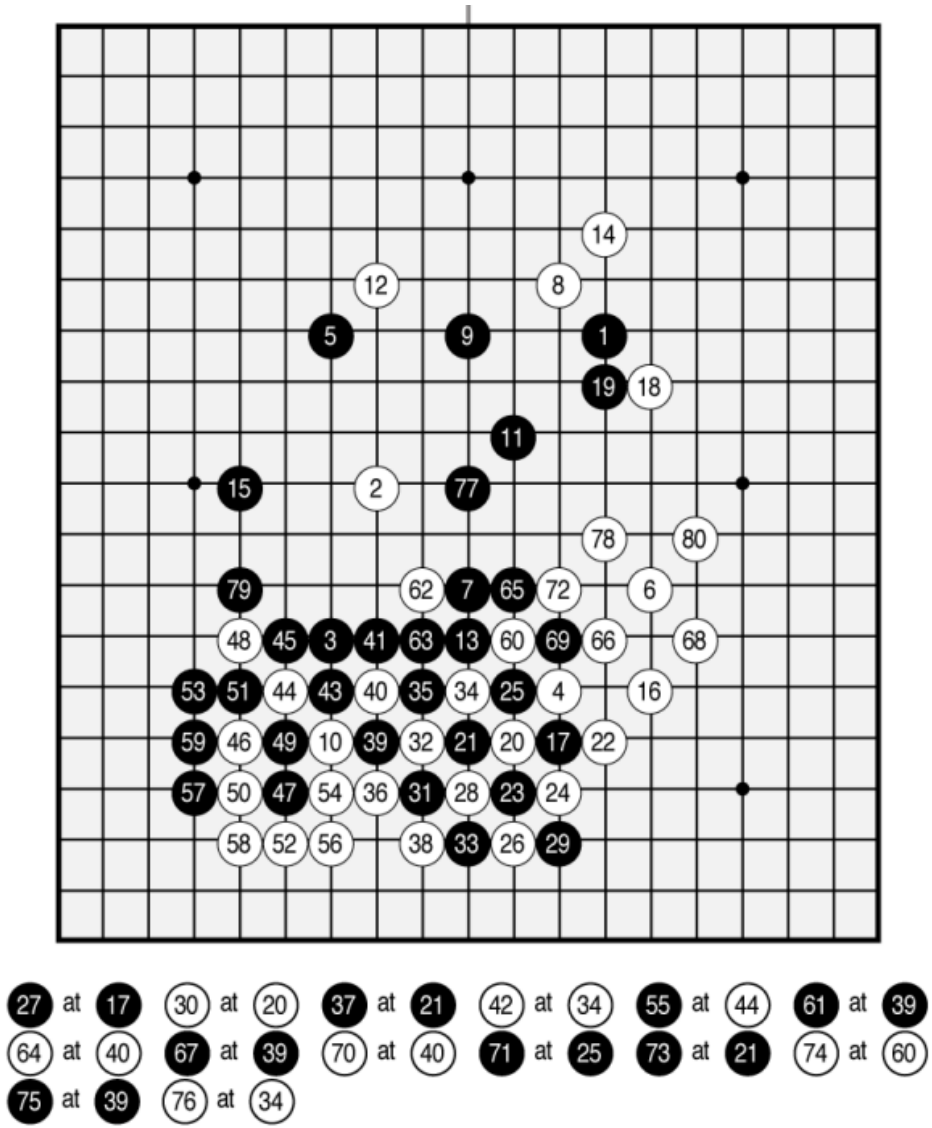
Lee apologized for his losses, stating after game three that he misjudged the capabilities of AlphaGo and felt powerless.

AlphaGo Lee (older version) won 494 out of 495 games against best computer programs, each program given 5s for each move.

Alpha Go Zero (new version) won 100 out of 100 games against **AlphaGo Lee** (older version) under the conditions of the match with Lee Sedol.

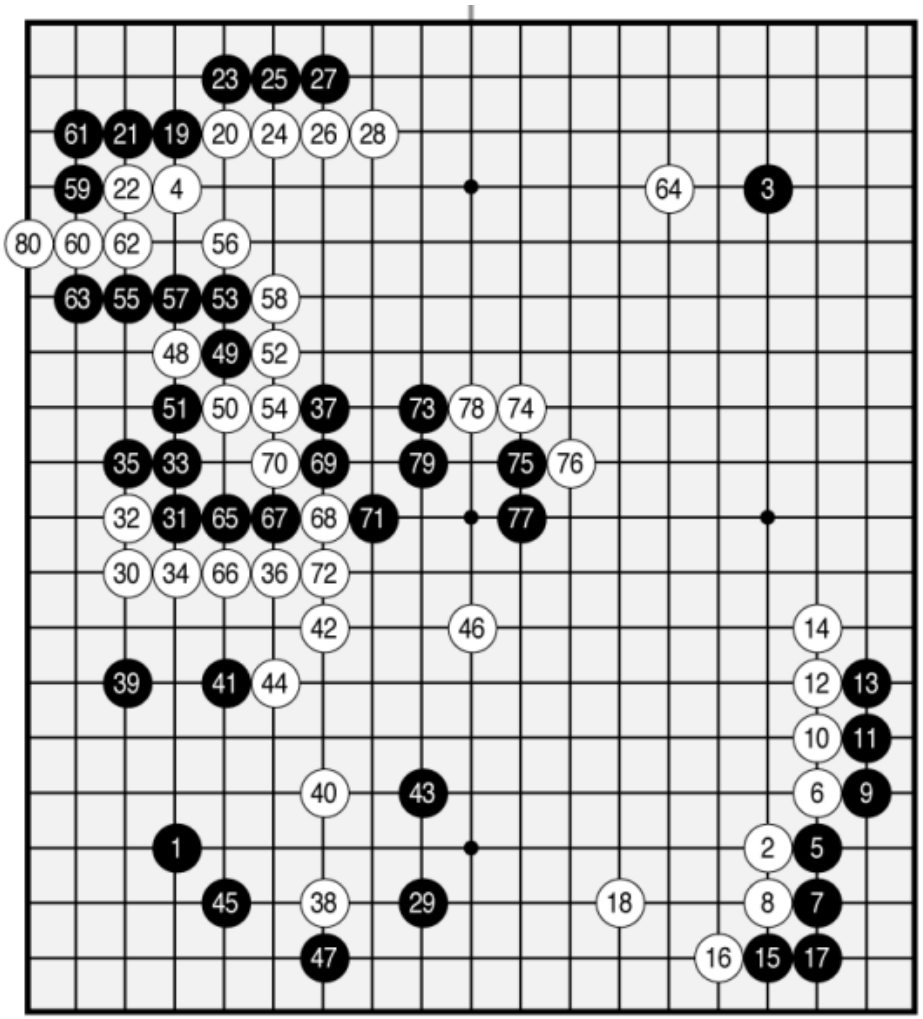
Learning behavior

After 3 hours of learning:



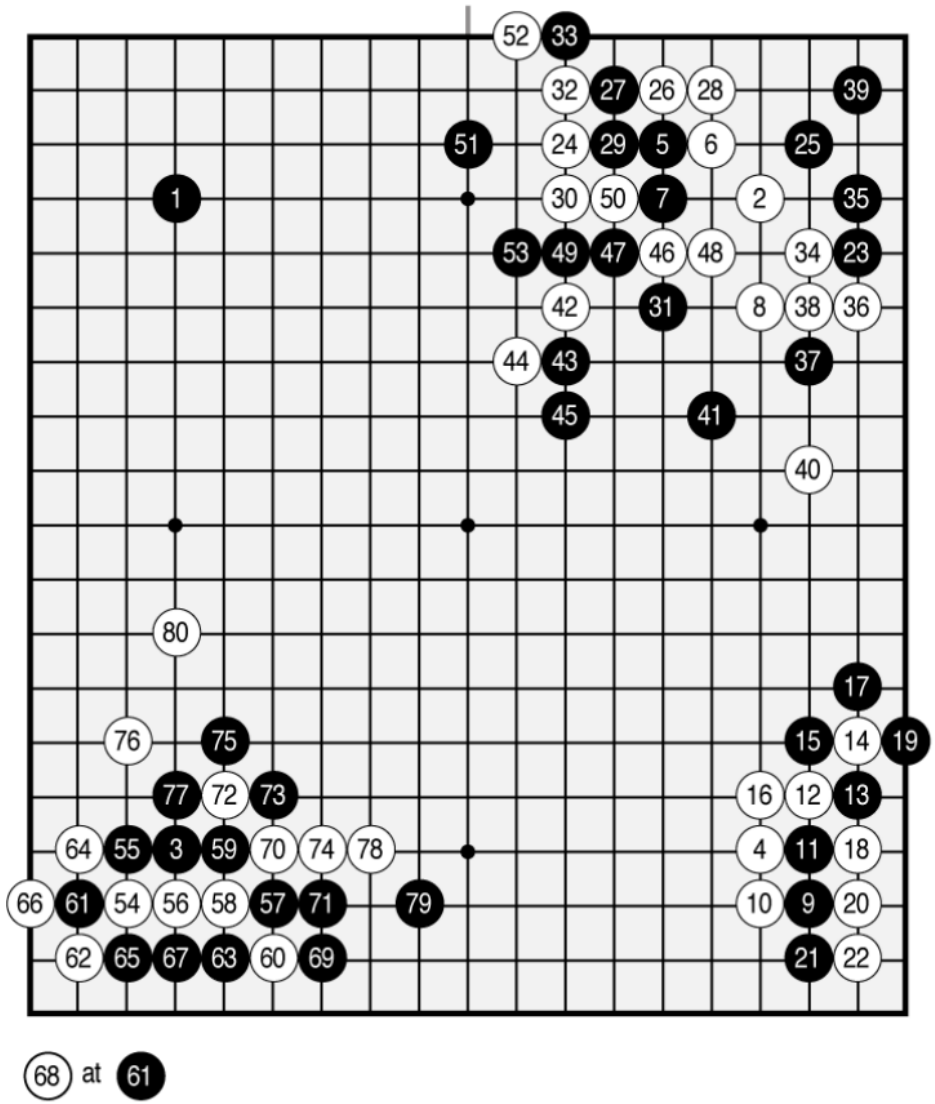
Learning behavior

After 19 hours of learning:



Learning behavior

After 70 hours of learning:



Shall human race be exterminated by super-human AI soon?

Shall human race be exterminated by super-human AI soon?

Hawking:
"The development of full artificial intelligence could spell the end of the human race."

Shall human race be exterminated by super-human AI soon?

Hawking:
"The development of full artificial intelligence could spell the end of the human race."

Musk:
"something seriously dangerous happening" as a result of machines with artificial intelligence, could be in as few as five years.

"I think we should be very careful about artificial intelligence. If I had to guess at what our biggest existential threat is, it's probably that."

Shall human race be exterminated by super-human AI soon?

Hawking:
"The development of full artificial intelligence could spell the end of the human race."

Musk:
"something seriously dangerous happening" as a result of machines with artificial intelligence, could be in as few as five years.

"I think we should be very careful about artificial intelligence. If I had to guess at what our biggest existential threat is, it's probably that."

Brázdil:
"Don't Panic! AI still sucks in real-world strategic reasoning."

Starcraft 2

Real-time strategic game produced by Blizzard Entertainment.

Played by milions of people, tournaments, etc.

Starcraft 2 - principles & challenges

To win a game players must:

- Accumulate resources (such as minerals)
- Construct production buildings
- Amass an army
- Eliminate all of the opponent's buildings

Starcraft 2 - principles & challenges

To win a game players must:

- Accumulate resources (such as minerals)
- Construct production buildings
- Amass an army
- Eliminate all of the opponent's buildings

Main challenges for AI:

- Multi-agent: Possibly several players, many independent units

Starcraft 2 - principles & challenges

To win a game players must:

- Accumulate resources (such as minerals)
- Construct production buildings
- Amass an army
- Eliminate all of the opponent's buildings

Main challenges for AI:

- Multi-agent: Possibly several players, many independent units
- A game typically lasts from few minutes to an hour (22 fps)
  Win/lose signal is *very* sparse

Starcraft 2 - principles & challenges

To win a game players must:

- Accumulate resources (such as minerals)
- Construct production buildings
- Amass an army
- Eliminate all of the opponent's buildings

Main challenges for AI:

- Multi-agent: Possibly several players, many independent units
- A game typically lasts from few minutes to an hour (22 fps)
  Win/lose signal is *very* sparse
- Long term: Early decisions may have long-term consequences

Starcraft 2 - principles & challenges

To win a game players must:

- Accumulate resources (such as minerals)
- Construct production buildings
- Amass an army
- Eliminate all of the opponent's buildings

Main challenges for AI:

- Multi-agent: Possibly several players, many independent units
- A game typically lasts from few minutes to an hour (22 fps)
  Win/lose signal is *very* sparse
- Long term: Early decisions may have long-term consequences
- Short term: In skirmishes, short term decisions matter

Starcraft 2 - principles & challenges

To win a game players must:

- Accumulate resources (such as minerals)
- Construct production buildings
- Amass an army
- Eliminate all of the opponent's buildings

Main challenges for AI:

- Multi-agent: Possibly several players, many independent units
- A game typically lasts from few minutes to an hour (22 fps)
  Win/lose signal is *very* sparse
- Long term: Early decisions may have long-term consequences
- Short term: In skirmishes, short term decisions matter
- Imperfect information: players do not see whole map, scouting

Starcraft 2 - principles & challenges

To win a game players must:

- Accumulate resources (such as minerals)
- Construct production buildings
- Amass an army
- Eliminate all of the opponent's buildings

Main challenges for AI:

- Multi-agent: Possibly several players, many independent units
- A game typically lasts from few minutes to an hour (22 fps)
  Win/lose signal is *very* sparse
- Long term: Early decisions may have long-term consequences
- Short term: In skirmishes, short term decisions matter
- Imperfect information: players do not see whole map, scouting
- Huge action space (GUI, many "moving parts"): Approx. 10^8

So far, learning AI solves only trivial minigames.

Starcraft 2 Learning Environment

- Machine Learning API developed by Blizzard (even for Linux).
- Dataset of 65,000 anonymised game replays

  promised to grow to millions soon
- Open source version of DeepMind's toolset PySC2
  - Simplified interface for RL agents to interact with StarCraft 2, getting observations and sending actions
  - Simple minigames to test basic AI

Results on full game

- Agents trained with sparse ternary rewards managed to avoid constant losses by using the Terran ability to lift and then move buildings out of attack range
- Agents trained with the Blizzard score converged to trivial strategies that simply preserved the initial mining process without building further units or structures

Conclusions

I have shown you

- super-human AI,

  ... playing Go
- trained by self-play, i.e. completely on its own,
- using amazingly simple algorithms,
- and relatively cheap hardware.

I have also shown you current limits of such AI

... it fails miserably in Starcraft 2