

Intro to Deep Reinforcement Learning

Luckeciano Melo

AI Engineer @ Deep Learning Brasil

Patrocínio:



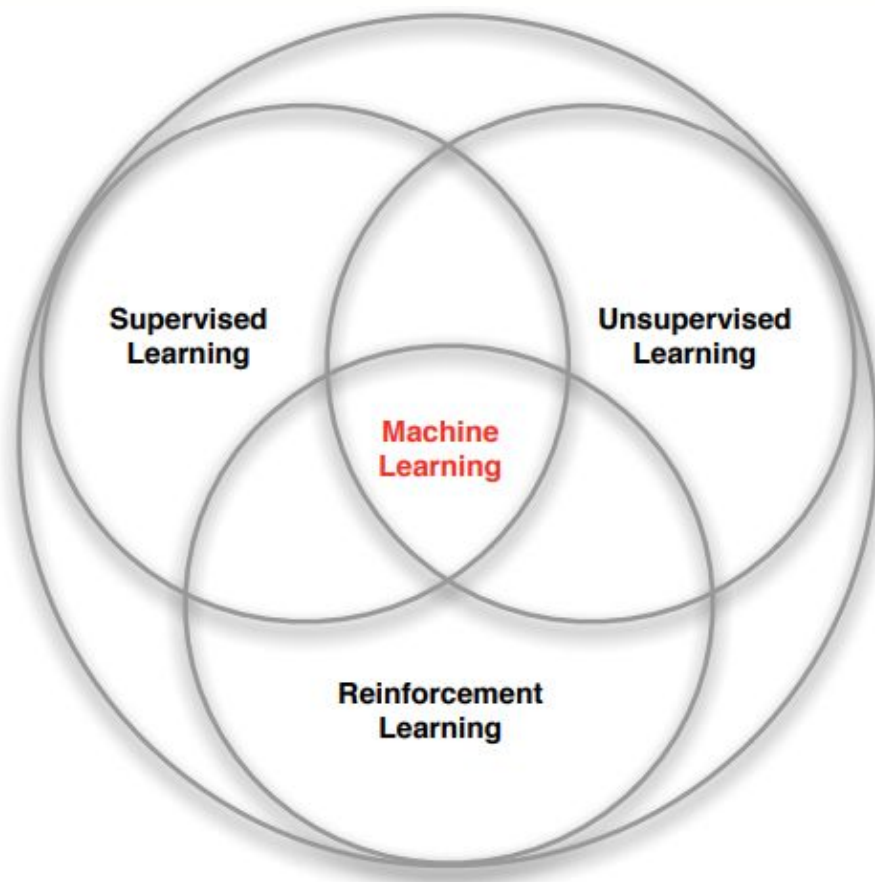
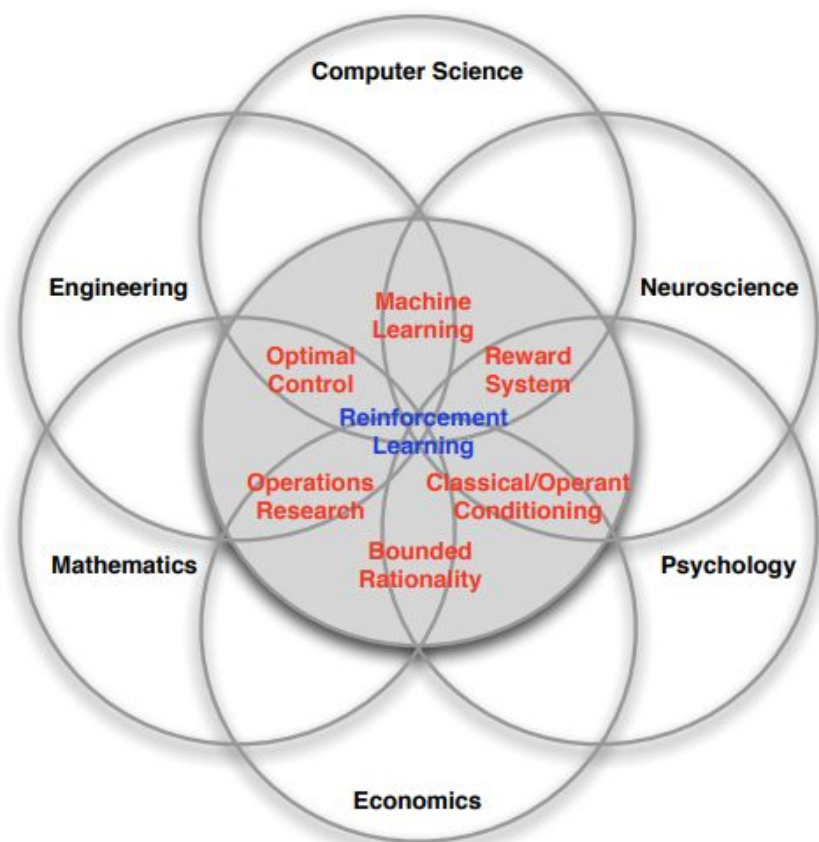
www.deeplearningbrasil.com.br



Apoio:



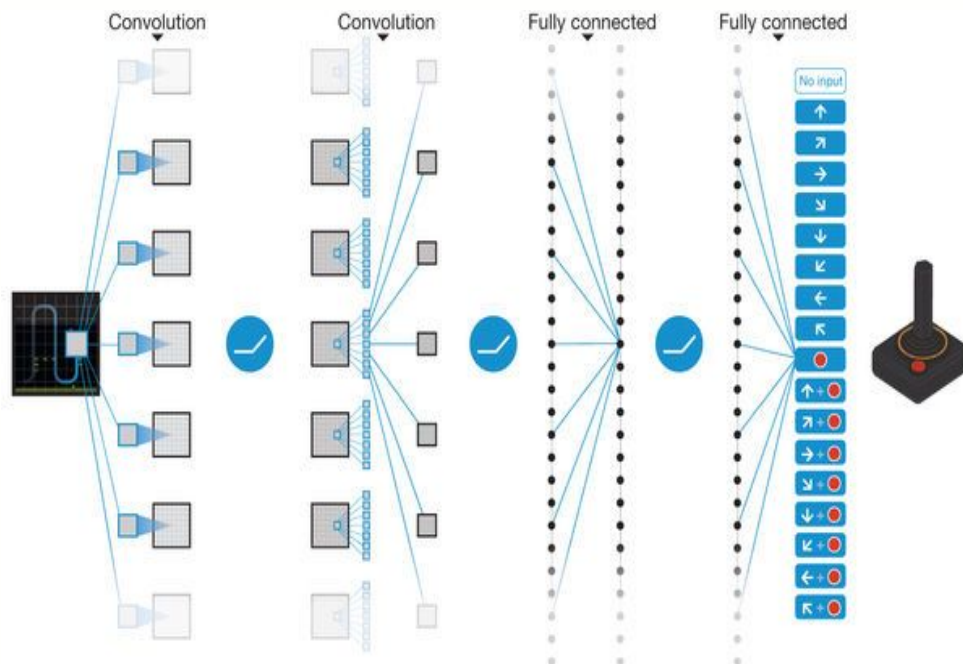
What is Reinforcement Learning?



What is Reinforcement Learning?

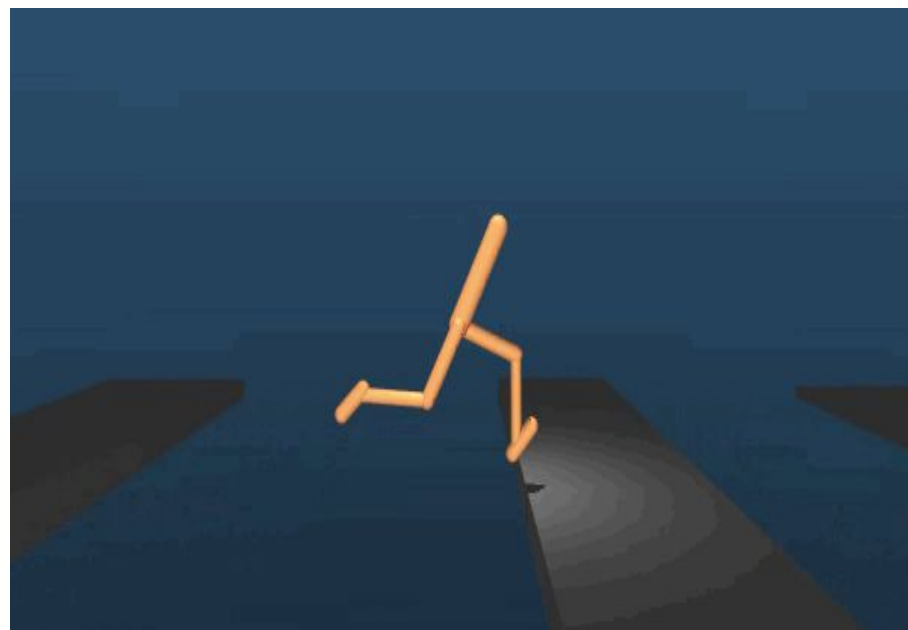
- No supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters - data is sequential
- Agent's actions affect the subsequent data it receives

Examples of Reinforcement Learning



Play Atari Games from raw image

Examples of Reinforcement Learning



Humanoid Walk (and Parkour)

Examples of Reinforcement Learning

AlphaGo seals 4-1 victory over Go grandmaster Lee Sedol

DeepMind's artificial intelligence astonishes fans to defeat human opponent and offers evidence computer software has mastered a major challenge

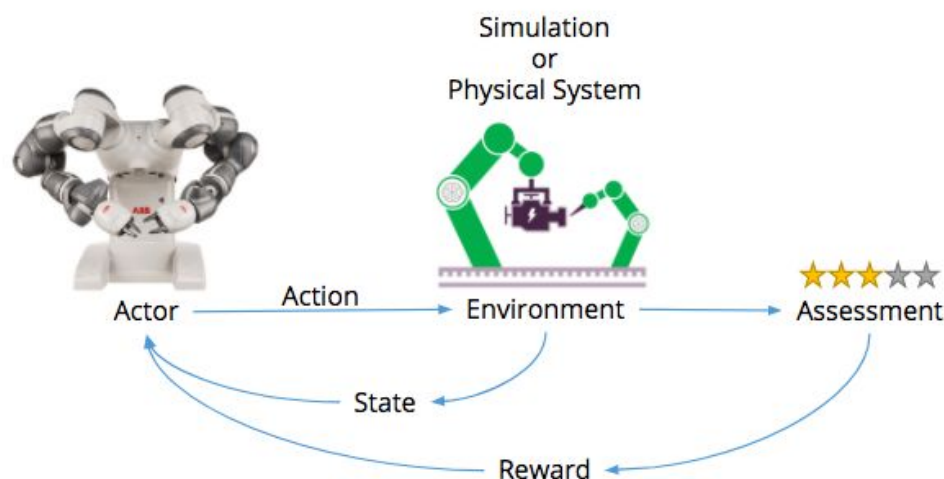


Google DeepMind's AlphaGo program triumphed in its final game against South Korean Go grandmaster Lee Sedol to win the series 4-1, providing further evidence of the landmark achievement for an artificial intelligence program.



Play the game of Go very well and without human knowledge

Reinforcement Learning System



- Every time step the agent:
 - Executes Action;
 - Receives an Observation from environment State;
 - Receives scalar Reward.
- Every time step the environment:
 - Receives Action;
 - Emits Observation from its State;
 - Emits Reward signal.

Reinforcement Learning Concepts

- **Reward: Scalar feedback signal that indicates how well agent is doing at step t**
 - Agent's job is select actions to maximize cumulative reward
 - Actions may have long term consequences or the reward may be delayed
 - Ex.: A financial investment may take months to mature
- **State (Markov State): Contains all useful data from history (actions + observations + rewards)**
 - State is Markov if “the future is independent of the past given the present”.
 - Ex .: Board game state like Go or Chess

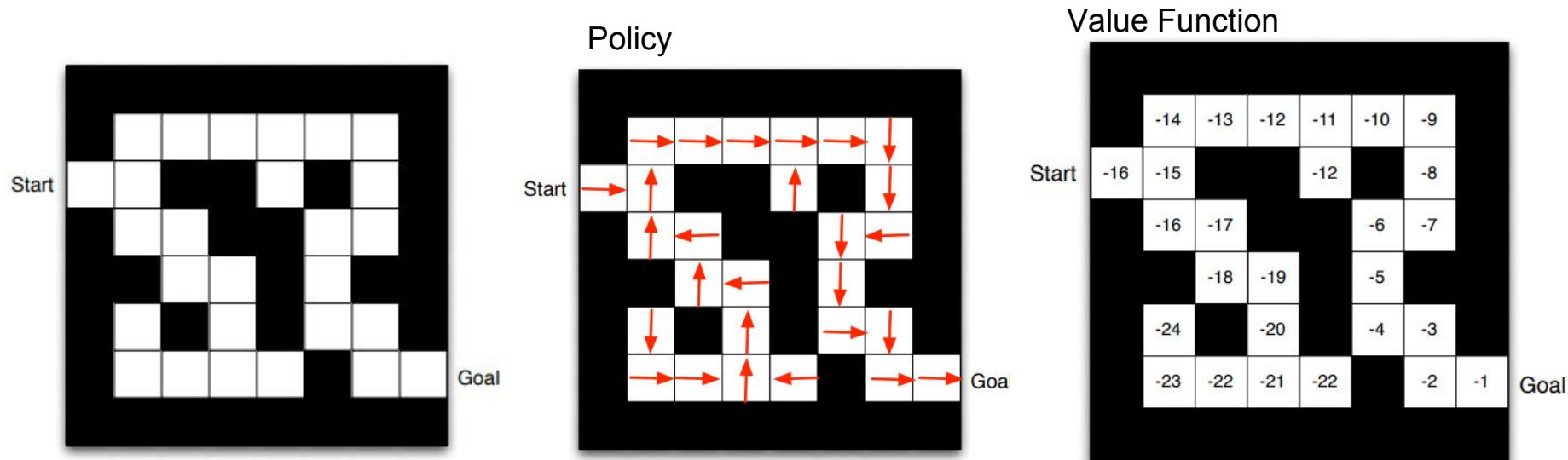
Major Components of RL Agent

- **Policy: Agent's behaviour function**
 - It's a map from state to action
 - Deterministic Policy: $a = \text{Policy}(s)$
 - Stochastic Policy: $a = \text{Policy}(a | s) = P(\text{Action} = a | \text{State} = s)$
- **Value function: Prediction of future reward**
 - Used to evaluate how good/bad the state is
 - And therefore to select between actions:

$$v_{\pi}(s) = \mathbb{E}_{\pi} [R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$$

- **Model: predicts what the environment will do next**

Maze Example

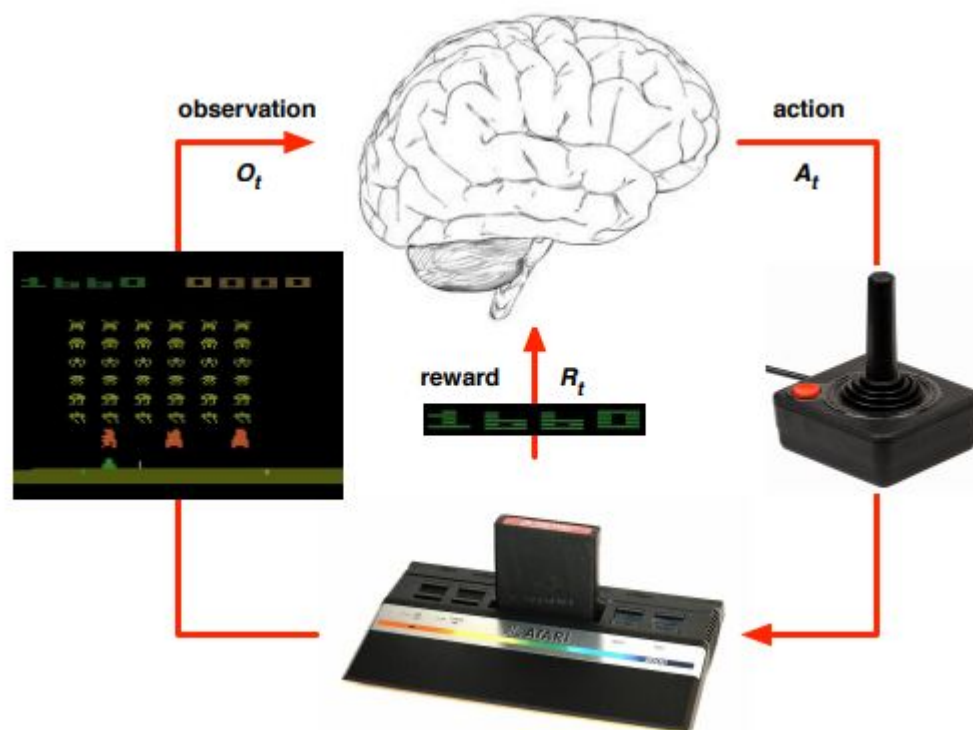


Reward: -1 per timestep

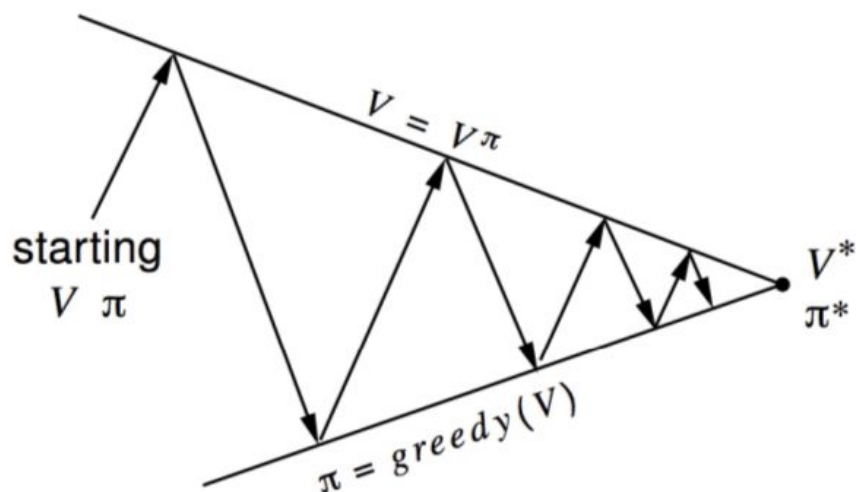
Actions: N, S, W, E

State: Agent's location

Atari Example

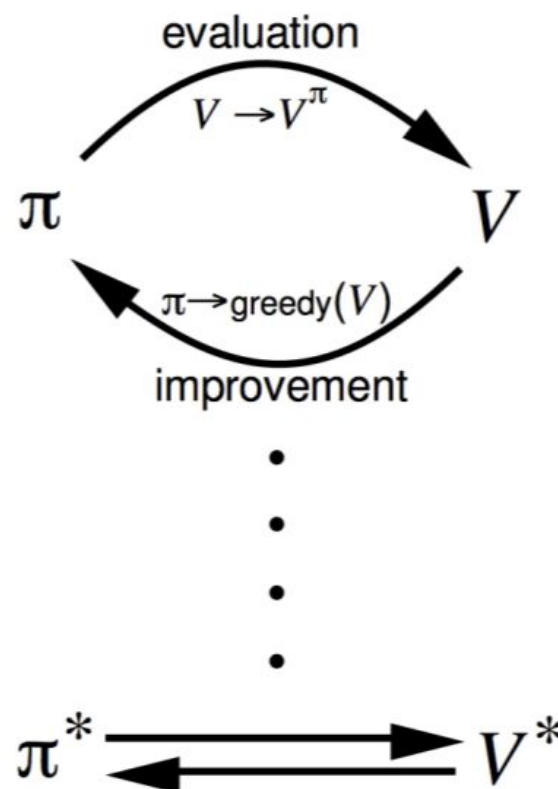


Generalized Policy Iteration - Control



Policy evaluation Estimate v_π
 e.g. Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
 e.g. Greedy policy improvement



Case Study: AlphaGo

- RL can be used to solve large problems:
 - Backgammon: 10^{20} possible states
 - Chess: 10^{45} possible states
 - Go: 10^{170} possible states
- AlphaGo defeated the best human Go player, Lee Sedol
- AlphaGo is based on Machine Learning and Monte-Carlo Techniques

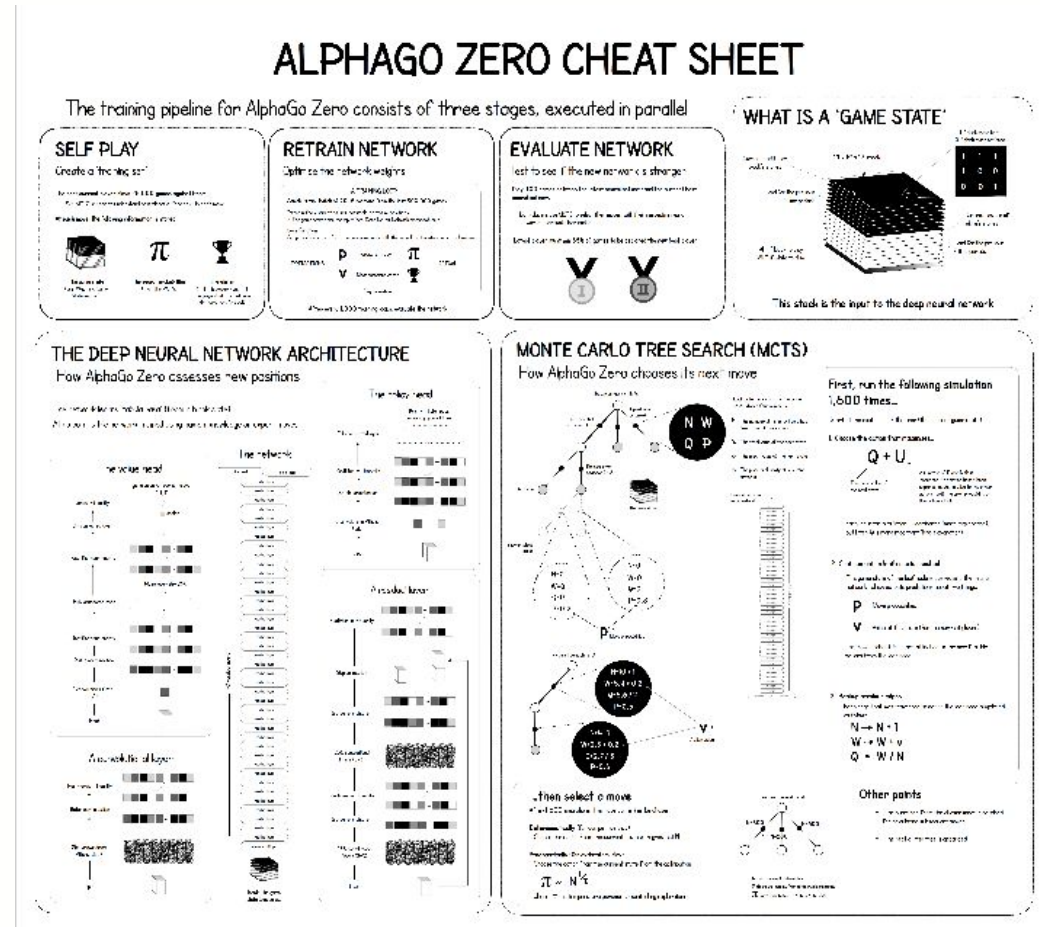
AlphaGo → AlphaGo Zero → AlphaZero

- AlphaGo uses a supervised learning step before the reinforcement learning
- This supervised learning phase bases on human knowledge from millions of games from Go Experts
- AlphaGo Zero is a variant who mastered go just with self-playing - **without human knowledge**
 - Defeated AlphaGo by 100-0
- AlphaZero generalizes AlphaGo Zero model for Shogi and Chess (and, probably, for any discrete-space game)



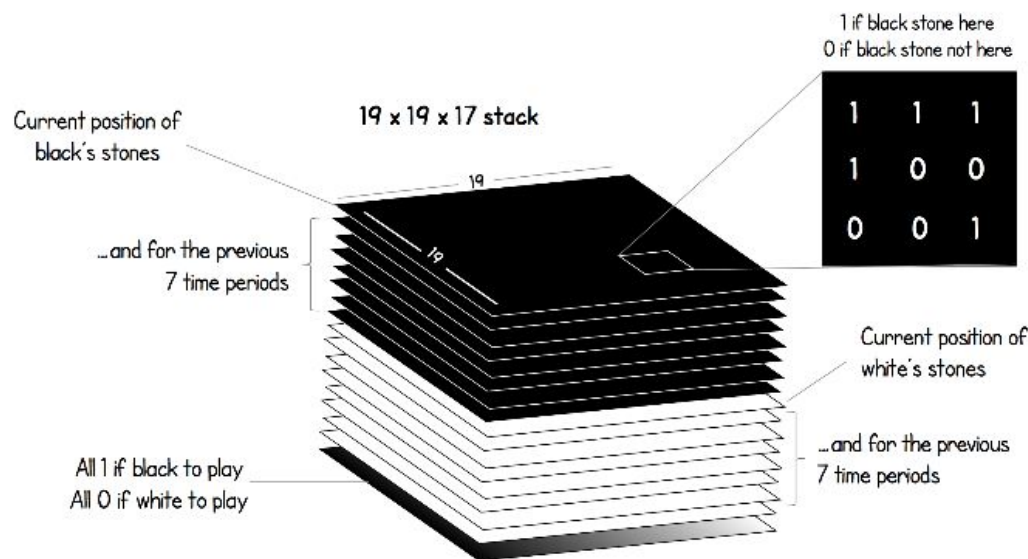
AlphaGo Zero Model

Deep Neural Network
for evaluate states
and actions + Monte
Carlo Tree Search for
“exploring” the game



AlphaGo Zero - RL Model

WHAT IS A 'GAME STATE'



This stack is the input to the deep neural network

- **Reward:** Win = +1, Loss = -1
- **Action:** New stone in the board
- **State:** Configuration of stones in the board

AlphaGo Zero - Pipeline - Self Play

SELF PLAY

Create a 'training set'

The best current player plays 25,000 games against itself

See MCTS section to understand how AlphaGo Zero selects each move

At each move, the following information is stored



The game state
(see 'What is a Game
State section')



The search probabilities
(from the MCTS)



The winner
(+1 if this player won, -1 if
this player lost - added once
the game has finished)

AlphaGo Zero - Pipeline - Retrain NN

RETRAIN NETWORK

Optimise the network weights

A TRAINING LOOP


Sample a mini-batch of 2048 positions from the last 500,000 games

Retrain the current neural network on these positions

- The game states are the input (see 'Deep Neural Network Architecture')

Loss function

Compares predictions from the neural network with the search probabilities and actual winner

	p	Cross-entropy	π	
PREDICTIONS		+		ACTUAL
	v	Mean-squared error		
		+		
		Regularisation		

After every 1,000 training loops, evaluate the network

AlphaGo Zero - Pipeline - Evaluate NN

EVALUATE NETWORK

Test to see if the new network is stronger

Play 400 games between the latest neural network and the current best neural network

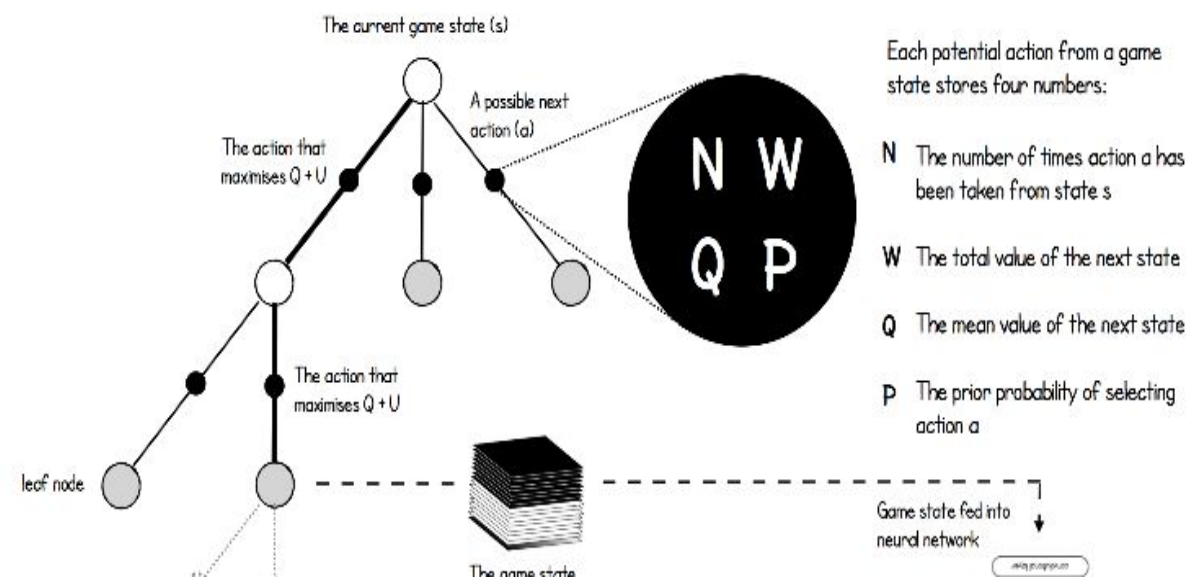
Both players use MCTS to select their moves, with their respective neural networks to evaluate leaf nodes

Latest player must win 55% of games to be declared the new best player



MONTE CARLO TREE SEARCH (MCTS)

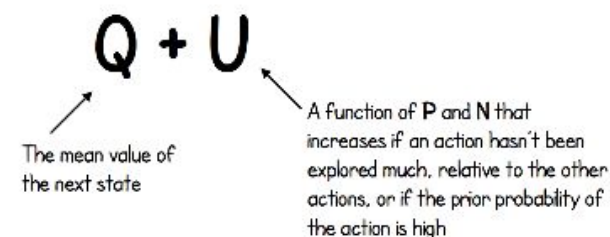
How AlphaGo Zero chooses its next move



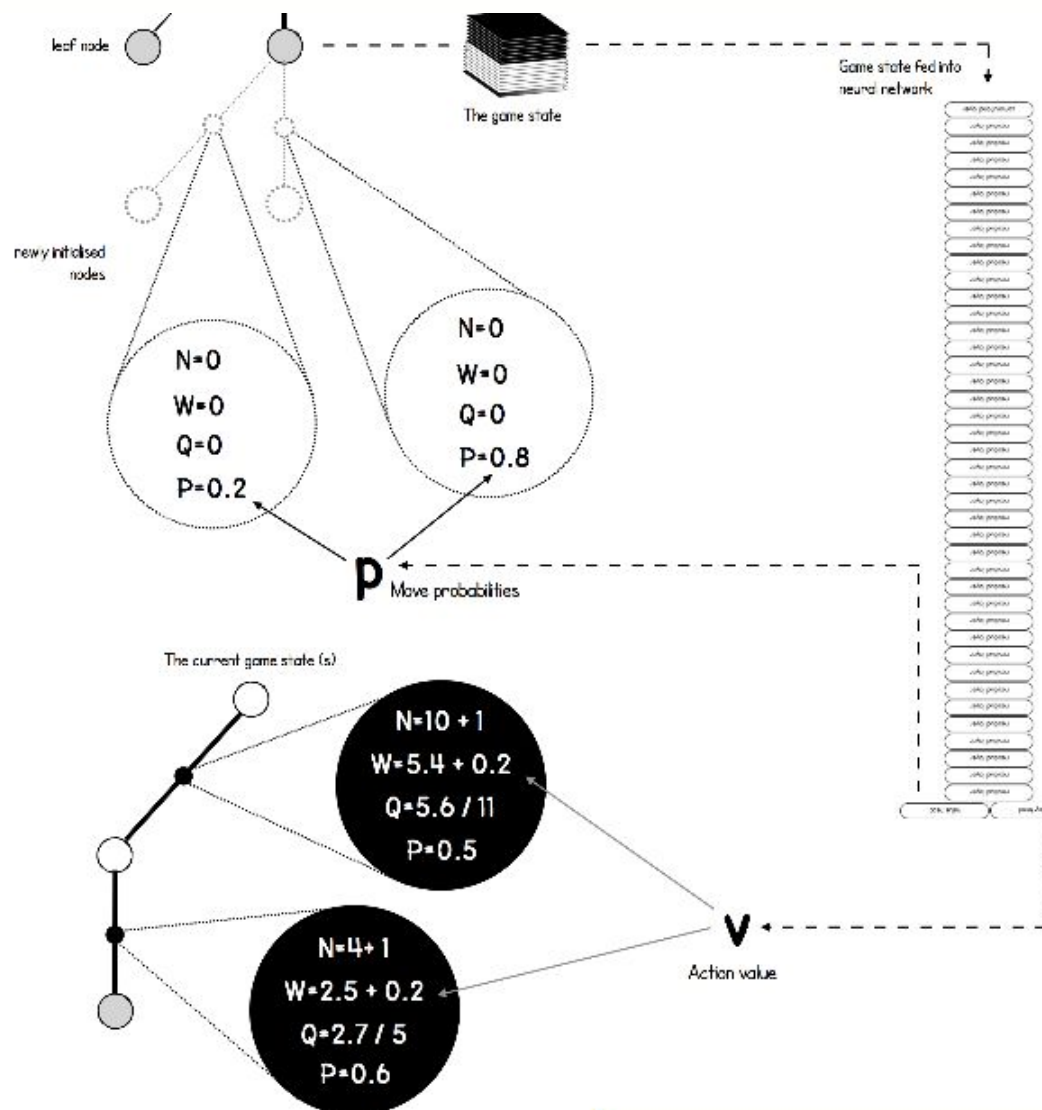
First, run the following simulation 1,600 times...

Start at the root node of the tree (the current game state)

1. Choose the action that maximises...



Early on in the simulation, U dominates (more exploration), but later, Q is more important (less exploration)



2. Continue until a leaf node is reached

The game state of the leaf node is passed into the neural network, which outputs predictions about two things:

P Move probabilities

V Value of the state (for the current player)

The move probabilities p are attached to the new feasible actions from the leaf node

3. Backup previous edges

Each edge that was traversed to get to the leaf node is updated as follows:

$$N \rightarrow N + 1$$

$$W \rightarrow W + v$$

$$Q = W / N$$

...then select a move

After 1,600 simulations, the move can either be chosen:

Deterministically (for competitive play)

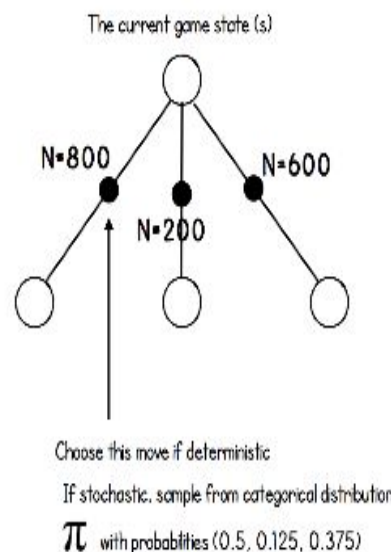
Choose the action from the current state with greatest N

Stochastically (for exploratory play)

Choose the action from the current state from the distribution

$$\pi \sim N^{1/\tau}$$

where τ is a temperature parameter; controlling exploration



Other points

- The sub-tree from the chosen move is retained for calculating subsequent moves
- The rest of the tree is discarded

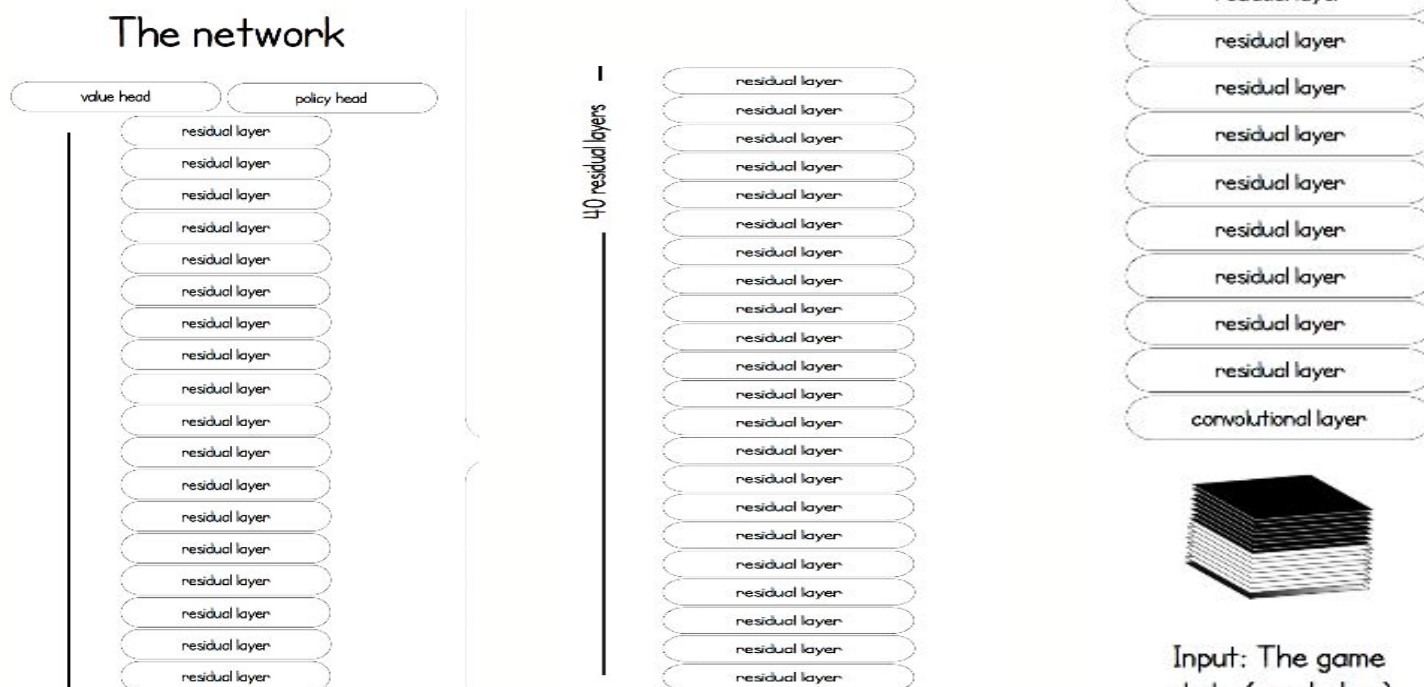
THE DEEP NEURAL NETWORK ARCHITECTURE

How AlphaGo Zero assesses new positions

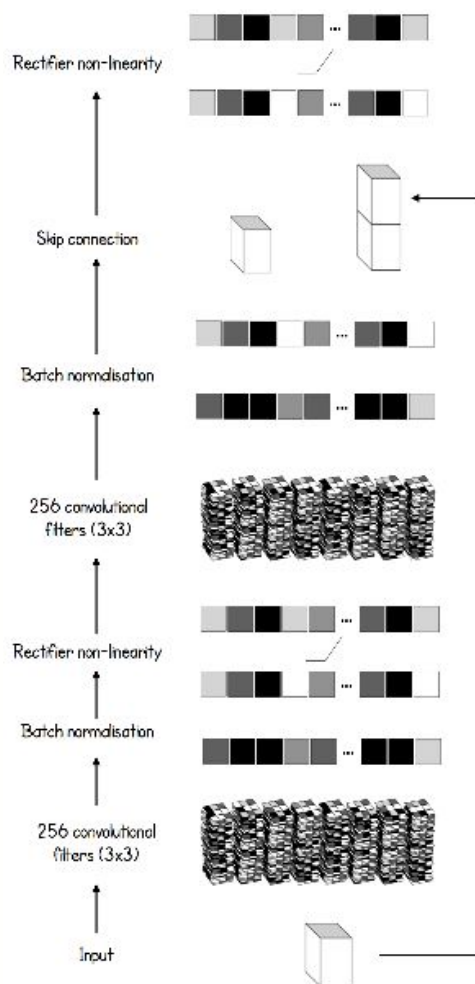
The network learns 'tabula rasa' (from a blank slate)

At no point is the network trained using human knowledge or expert moves

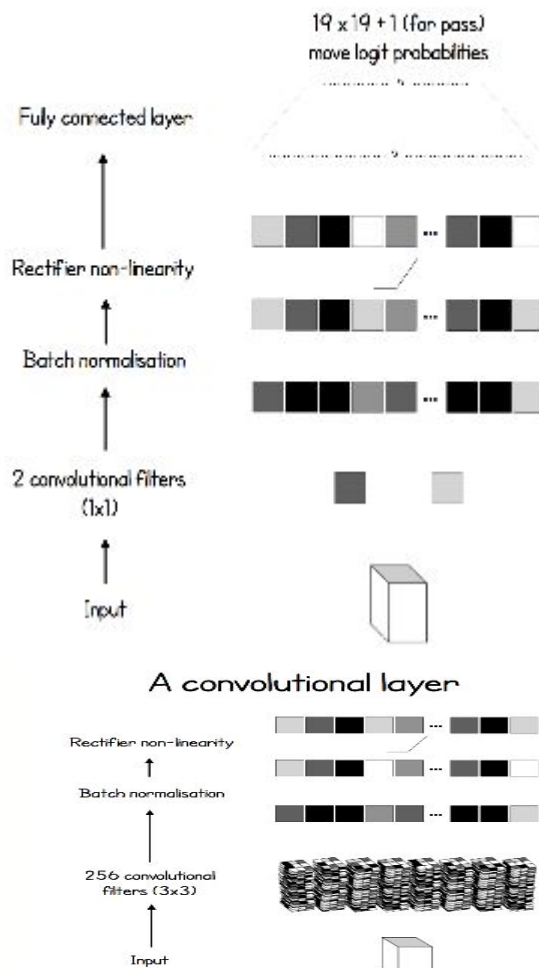
The network



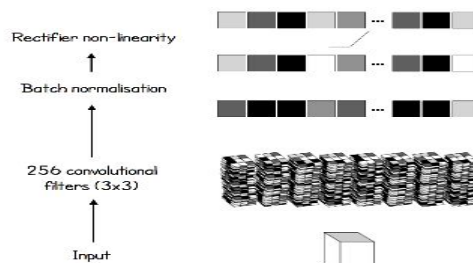
A residual layer



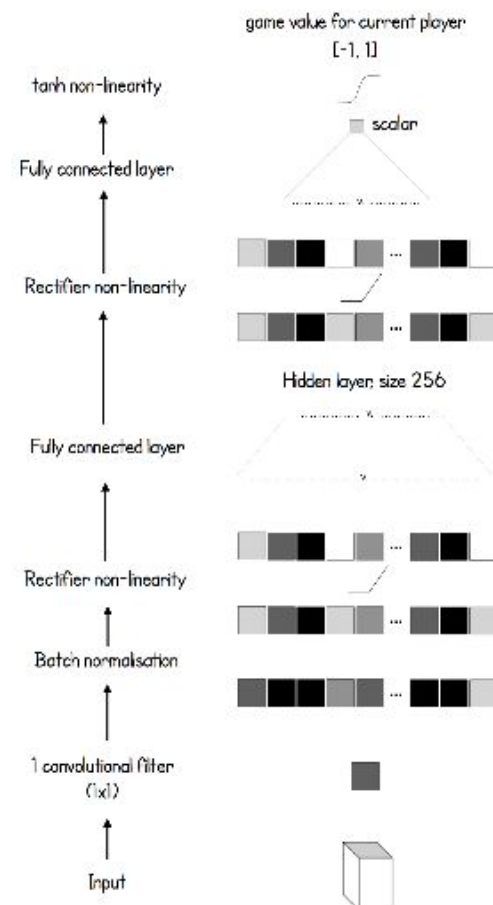
The policy head



A convolutional layer



The value head

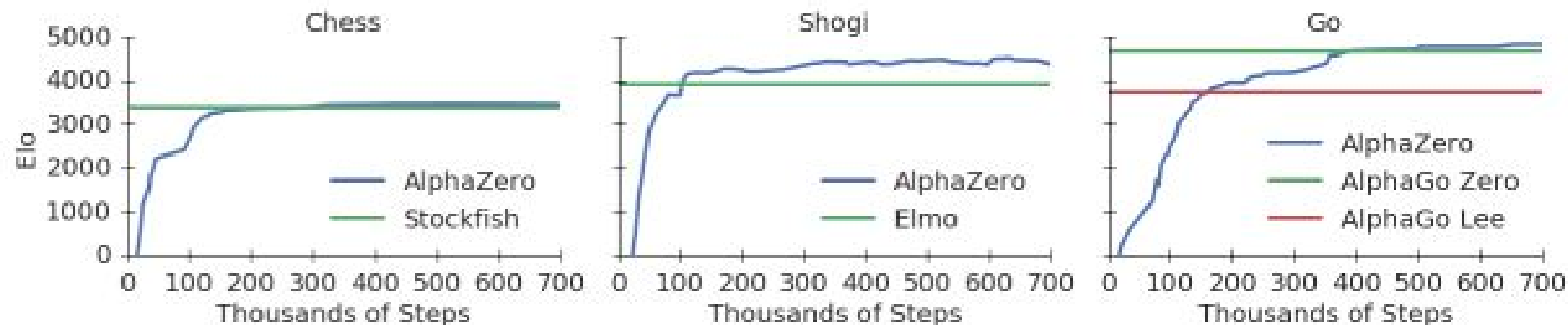


AlphaZero: Generalization

- Just 48 days after “Mastering Go without Human Knowledge” (Dec 2017)
- Just Self-Play and with General Reinforcement Learning Algorithm
- Some modifications about AlphaZero Go:
 - Reward: Generalized from Win/Loss for potentially draws or other outcomes
 - Remove data augmentation specific for Go (board symmetry)
 - Simplifications about evaluation step and selection of best player

AlphaZero: Generalization

- AlphaZero outperformed Stockfish (Chess) after just 4 hours of training
- AlphaZero outperformed Elmo (Shogi) after just 2 hours of training
- AlphaZero outperformed AlphaGo Lee in 8 hours



AlphaZero: Generalization

MCTS Search Performance:

- AlphaZero
 - Shogi: 40k positions in a second
 - Chess: 80k positions in a second
- Stockfish (Chess): 70 million positions in a second
- Elmo (Shogi): 35 million positions in a second

AlphaZero: Keras Notebook

Bonus: My Research :p



Deep Reinforcement Learning for Humanoid Robot Walking and Kicking



Luckeciano Melo

Unknown

In this project, we apply modern Deep Reinforcement Learning techniques for optimizing Walking Engine and Kicking in Soccer 3D Simulation.

Robotics, Artificial Intelligence

[Tweet](#) [Share](#) [in Share](#)

0

Collaborators

0

Followers

Follow