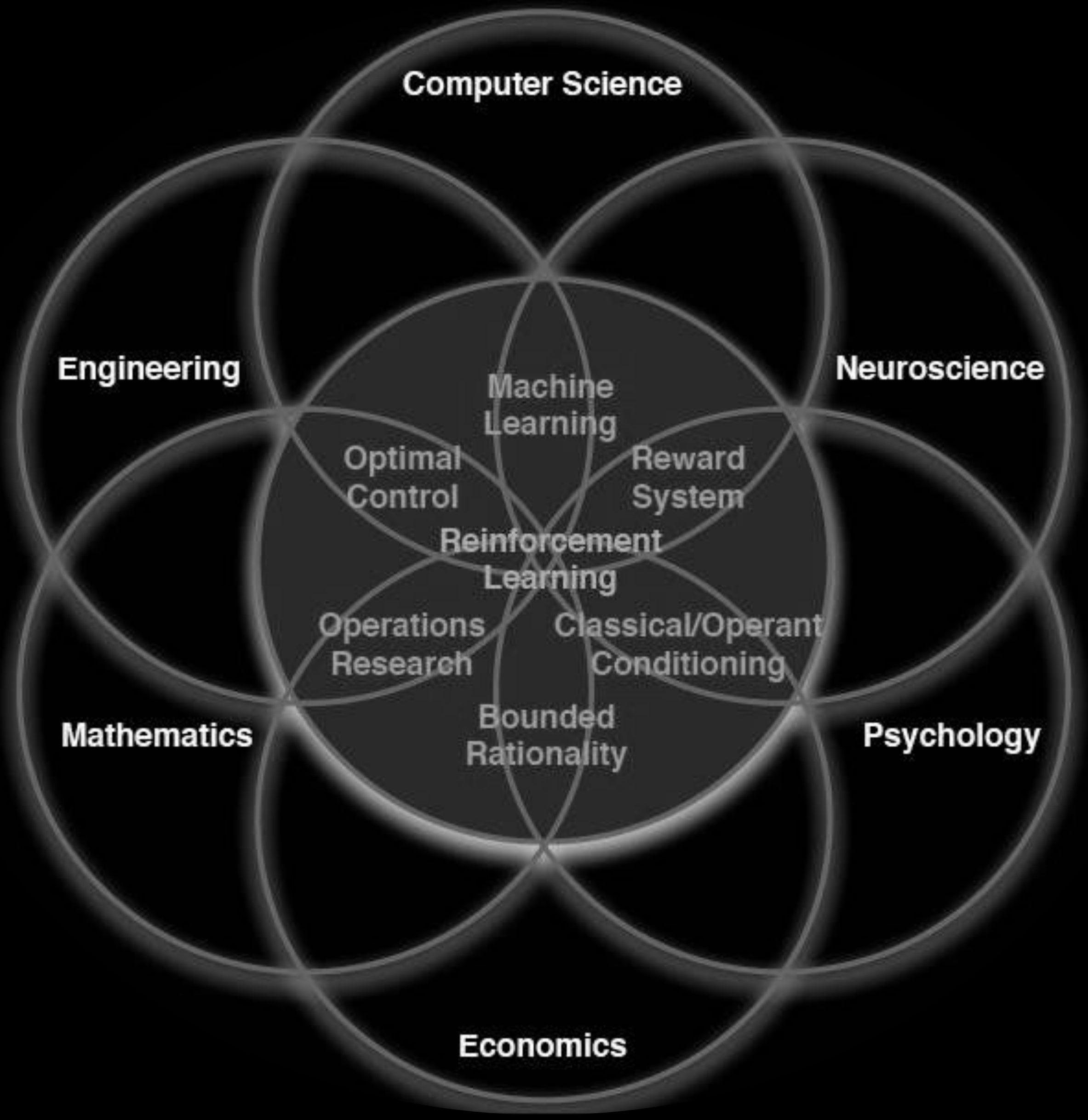
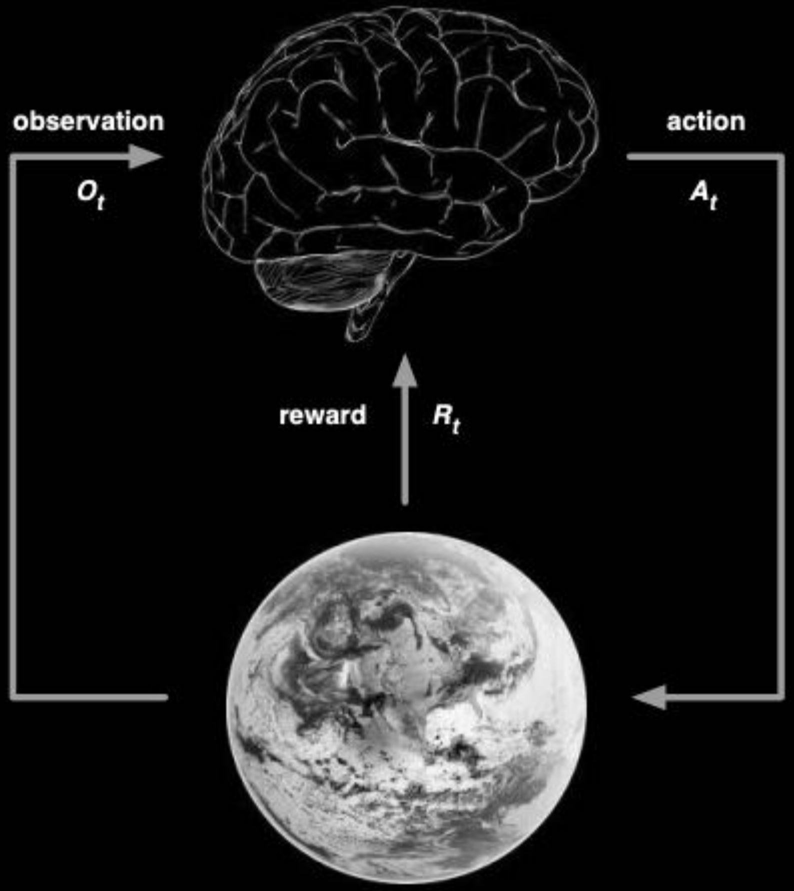
Reinforcement Learning

Overview of RL and Deep RL


Computer Science
Engineering
Neuroscience
Machine Learning
Optimal Control
Reward System
Reinforcement Learning
Operations Research
Classical/Operant Conditioning
Bounded Rationality
Mathematics
Psychology
Economics



R_t feedback at time t

SEQUENTIAL PARTIALLY OBSERVABLE ENVIRONMENT

Agent Goal: select actions to maximise total future reward

At each step t:

the agent:
Executes action A_t
Receives observation O_t
Receives scalar reward R_t

The environment:
Receives action A_t
Emits observation O_{t+1}
Emits scalar reward R_{t+1}

t increments at env. step

$H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t$

$S_t = f(H_t)$

environment state S_t^e - real state

agent state S_t^a - the agent's representation of state

TOOLS USED TO CHOOSE ACTIONS

Policy π function mapping state to action

Deterministic: $a = \pi(s)$
Stochastic: $\pi(a|s) = P[At = a|St = s]$

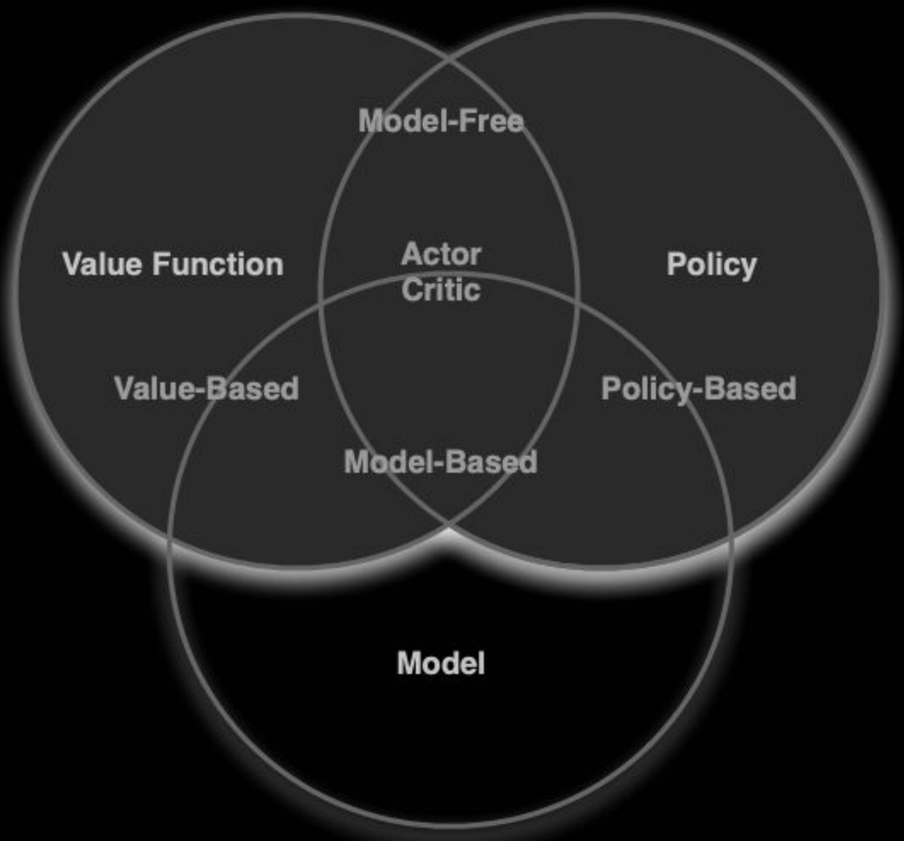
Value $v_\pi(s)$ a prediction of future reward

$v_\pi(s) = E_\pi[R_t+1 + \gamma R_t+2 + \gamma^2 R_t+3 + \ldots \mid St = s]$

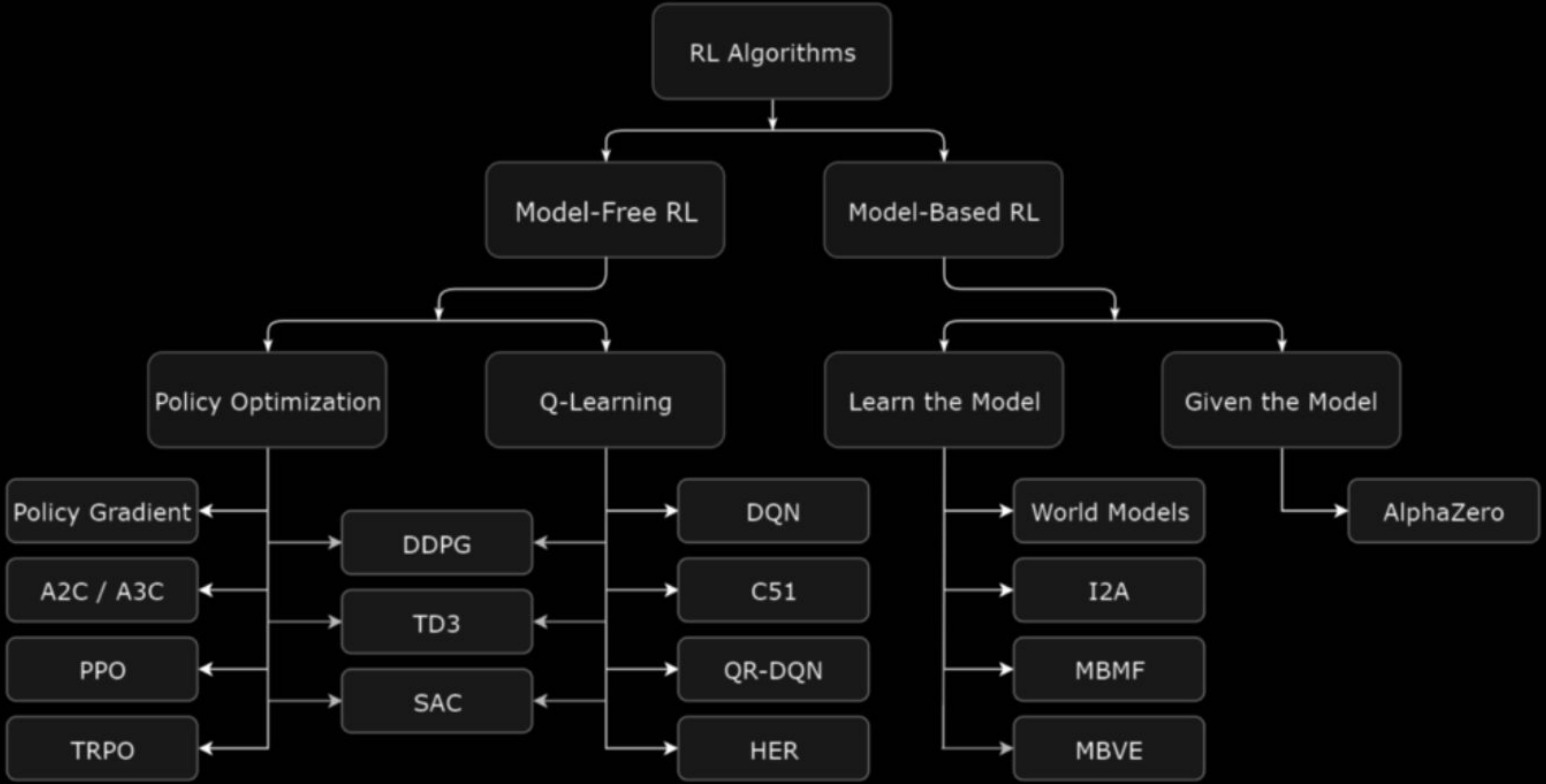
Model predicts what the next state *P* and the next reward *R*

$P^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
$R^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$

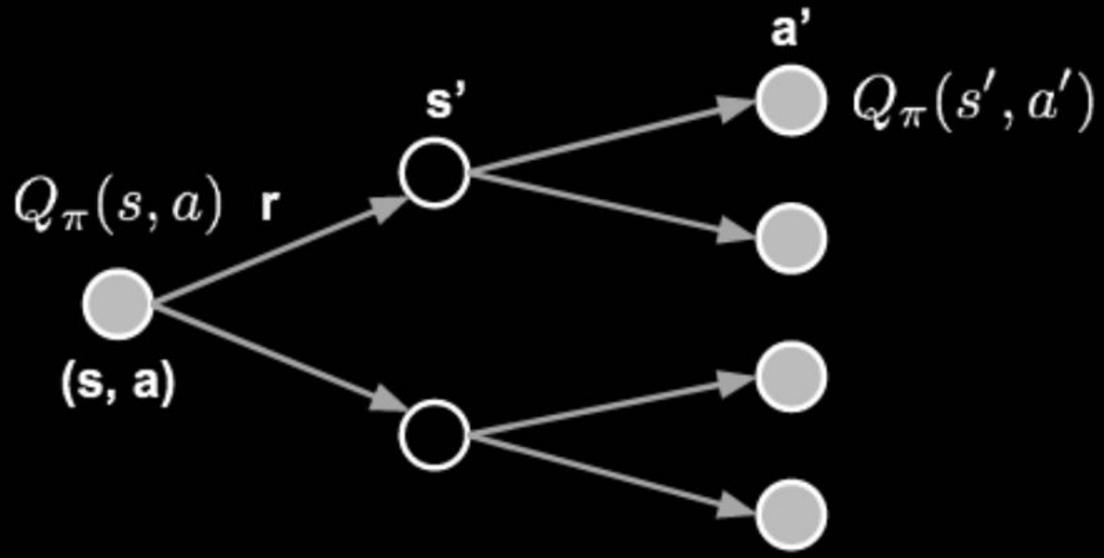
DIFFERENT TYPES OF RL



EXPLORATION VS EXPLOITATION


RL Algorithms
Model-Free RL
Model-Based RL
Policy Optimization
Q-Learning
Learn the Model
Given the Model
Policy Gradient
A2C / A3C
PPO
TRPO
DDPG
TD3
SAC
DQN
C51
QR-DQN
HER
World Models
I2A
MBMF
MBVE
AlphaZero

Q learning to AlphaZero



$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q_\pi(s, a)$$

$$Q_\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} P^a_{ss'} V_\pi(s')$$

$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \Big(R(s, a) + \gamma \sum_{s' \in S} P^a_{ss'} V_\pi(s') \Big)$$

$$Q_\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} P^a_{ss'} \sum_{a' \in \mathcal{A}} \pi(a'|s') Q_\pi(s', a')$$

Q-learning: Learn function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Require:

Sates $\mathcal{X} = \{1, \ldots, n_x\}$

Actions $\mathcal{A} = \{1, \ldots, n_a\}$, $\quad A : \mathcal{X} \Rightarrow \mathcal{A}$

Reward function $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Black-box (probabilistic) transition function $T : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$

Learning rate $\alpha \in [0, 1]$, typically $\alpha = 0.1$

Discounting factor $\gamma \in [0, 1]$

procedure QLEARNING($\mathcal{X}$, A, R, T, α, γ)

  Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrarily

  while Q is not converged **do**

    Start in state $s \in \mathcal{X}$

    while s is not terminal **do**

      Calculate π according to Q and exploration strategy (e.g. $\pi(x) \leftarrow \arg\max_a Q(x, a)$)

      $a \leftarrow \pi(s)$

      $r \leftarrow R(s, a)$ ▷ Receive the reward

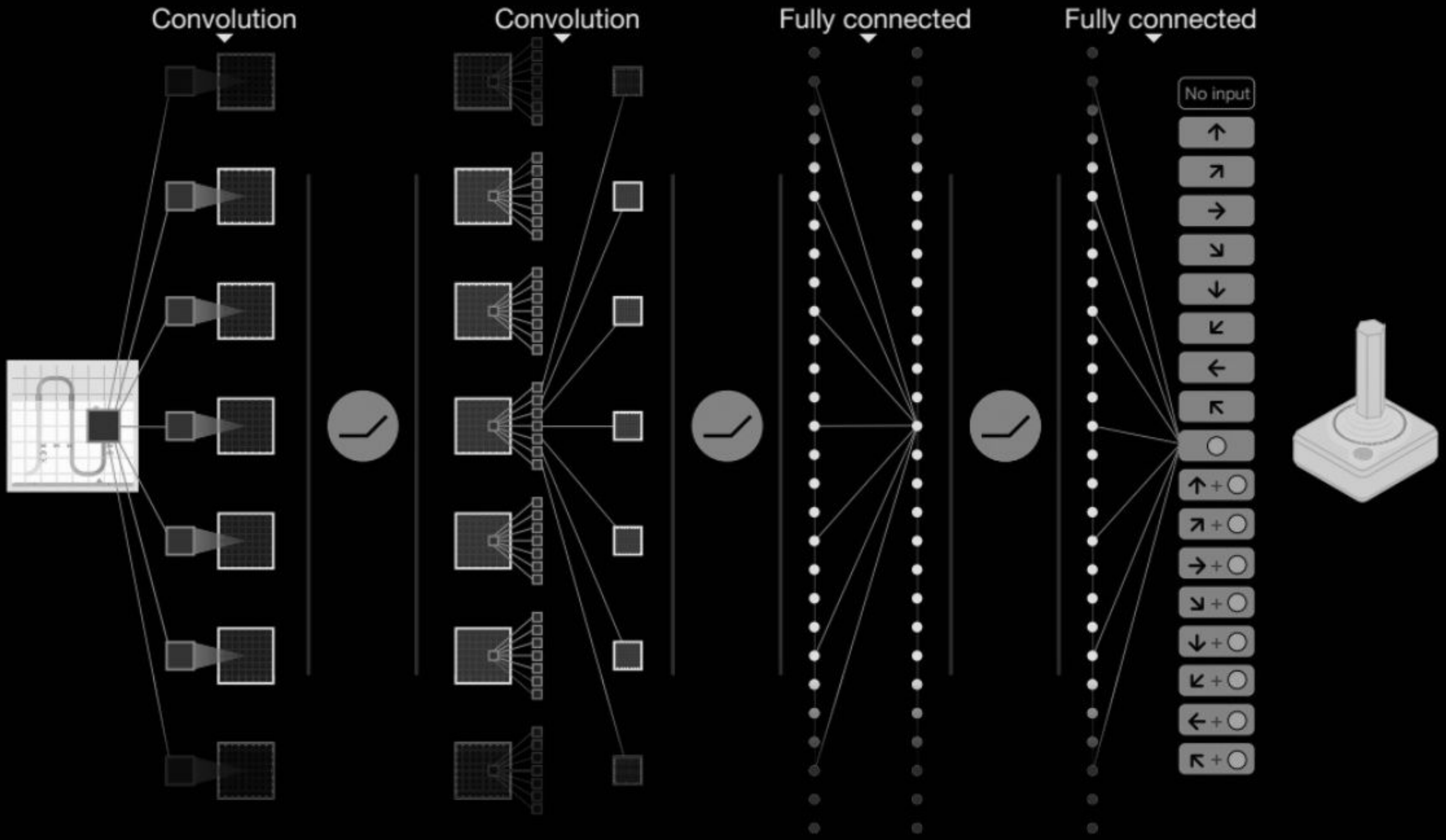
      $s' \leftarrow T(s, a)$ ▷ Receive the new state

      $Q(s', a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$

      $s \leftarrow s'$

  return Q

PROBLEM: DOES NOT HAVE GENERALITY


Convolution
Convolution
Fully connected
Fully connected
No input

Algorithm 1: deep Q-learning with experience replay.
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode $= 1, M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

EXPERIENCE REPLAY

SAMPLE TARGET NETWORK

ALPHA GO

User Player knowledge and learning to play

Knows the model of GO

Not generalizable

ALPHA GO Zero

learns to play from scratch

Knows the model of GO

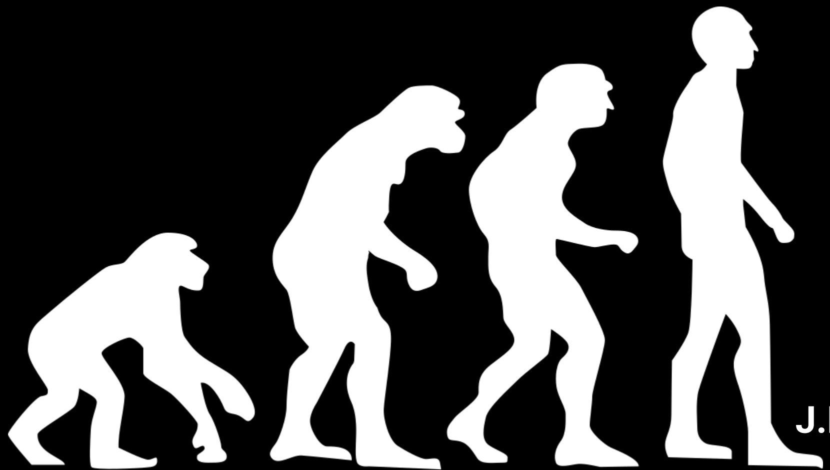
Not generalizable

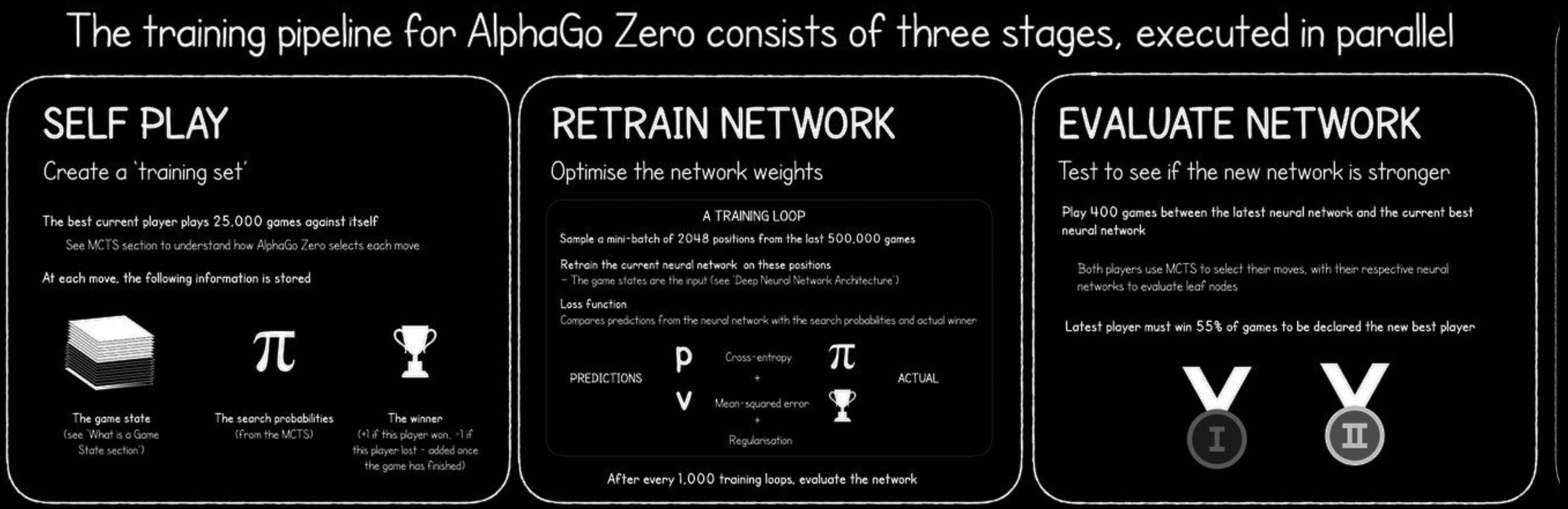
ALPHA Zero

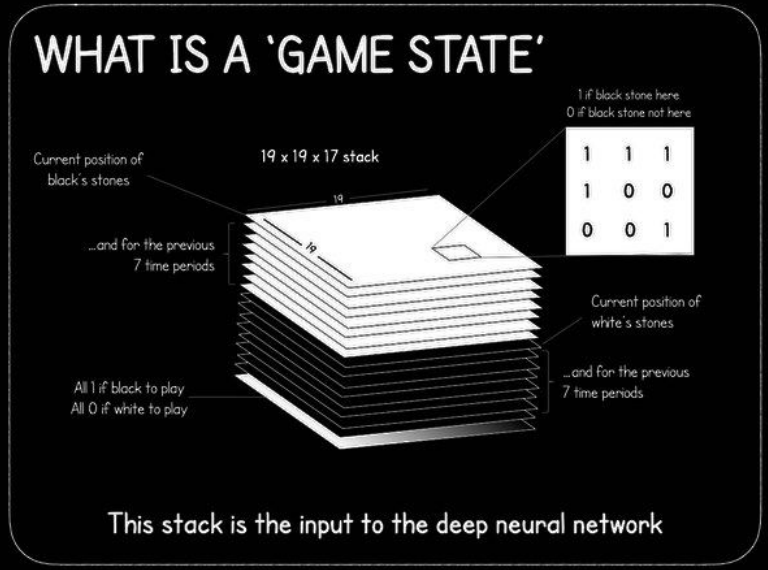
learns to play from scratch

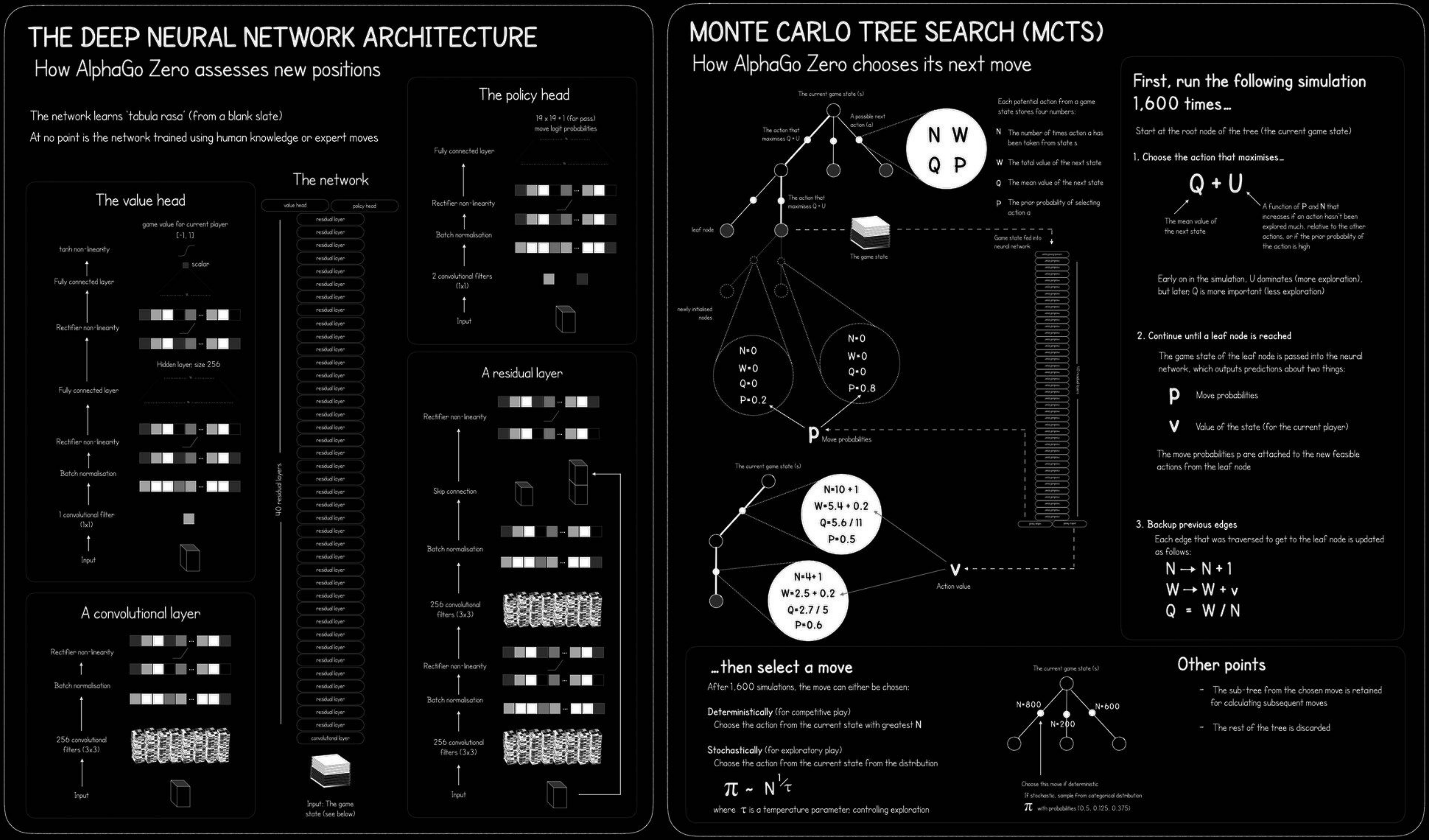
Learns the model for any game

Generalizable to minimax games

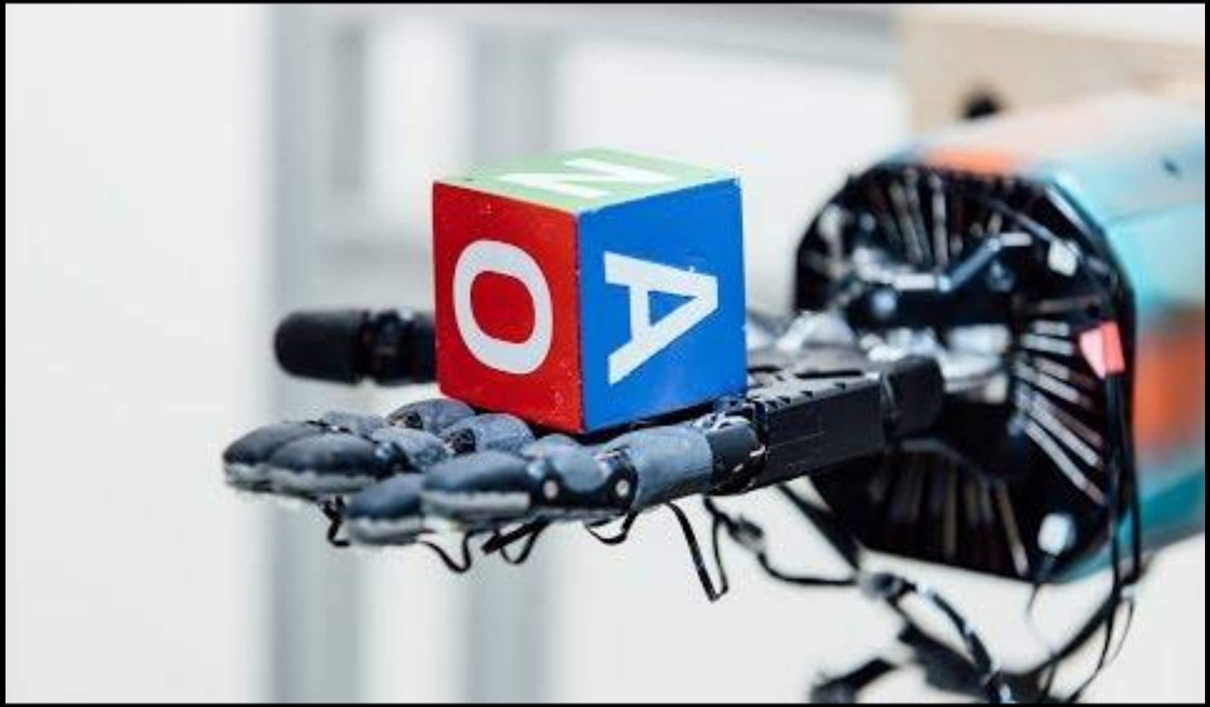



The training pipeline for AlphaGo Zero consists of three stages, executed in parallel
SELF PLAY
Create a 'training set'
The best current player plays 25,000 games against itself
See MCTS section to understand how AlphaGo Zero selects each move
At each move, the following information is stored
The game 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)
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
PREDICTIONS
p
v
Cross-entropy
+
Mean-squared error
+
Regularisation
π
ACTUAL
After every 1,000 training loops, evaluate the network
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
I
II


WHAT IS A 'GAME STATE'
19 x 19 x 17 stack
Current position of black's stones
...and for the previous 7 time periods
All 1 if black to play
All 0 if white to play
1 if black stone here
0 if black stone not here
1 1 1
1 0 0
0 0 1
Current position of white's stones
...and for the previous 7 time periods
This stack is the input to the deep neural network


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 value head
game value for current player [-1, 1]
tanh non-linearity
scalar
Fully connected layer
Rectifier non-linearity
Hidden layer size 256
Fully connected layer
Rectifier non-linearity
Batch normalisation
1 convolutional filter (1x1)
Input
A convolutional layer
Rectifier non-linearity
Batch normalisation
256 convolutional filters (3x3)
Input
The network
value head
policy head
residual layer
40 residual layers
convolutional layer
Input: The game state (see below)
The policy head
19 x 19 + 1 (for pass) move logit probabilities
Fully connected layer
Rectifier non-linearity
Batch normalisation
2 convolutional filters (1x1)
Input
A residual layer
Rectifier non-linearity
Skip connection
Batch normalisation
256 convolutional filters (3x3)
Rectifier non-linearity
Batch normalisation
256 convolutional filters (3x3)
Input
MONTE CARLO TREE SEARCH (MCTS)
How AlphaGo Zero chooses its next move
The current game state (s)
A possible next action (a)
The action that maximises Q + U
leaf node
The game state
Game state fed into neural network
newly initialised nodes
N W
Q P
Each potential action from a game state stores four numbers:
N The number of times action a has been taken from state s
W The total value of the next state
Q The mean value of the next state
P The prior probability of selecting action a
N=0
W=0
Q=0
P=0.2
N=0
W=0
Q=0
P=0.8
p Move probabilities
The current game state (s)
N=10 + 1
W=5.4 + 0.2
Q=5.6 / 11
P=0.5
N=4+1
W=2.5 + 0.2
Q=2.7 / 5
P=0.6
v Action value
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...
Q + U
The mean value of the next state
A function of P and N that increases if an action hasn't been explored much, relative to the other actions, or if the prior probability of the action is high
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 → N + 1
W → 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
π ~ N^(1/τ)
where τ is a temperature parameter, controlling exploration
The current game state (s)
N=800
N=200
N=600
Choose this move if deterministic
If stochastic, sample from categorical distribution
π with probabilities (0.5, 0.125, 0.375)
Other points
- The sub-tree from the chosen move is retained for calculating subsequent moves
- The rest of the tree is discarded

APPLICATIONS

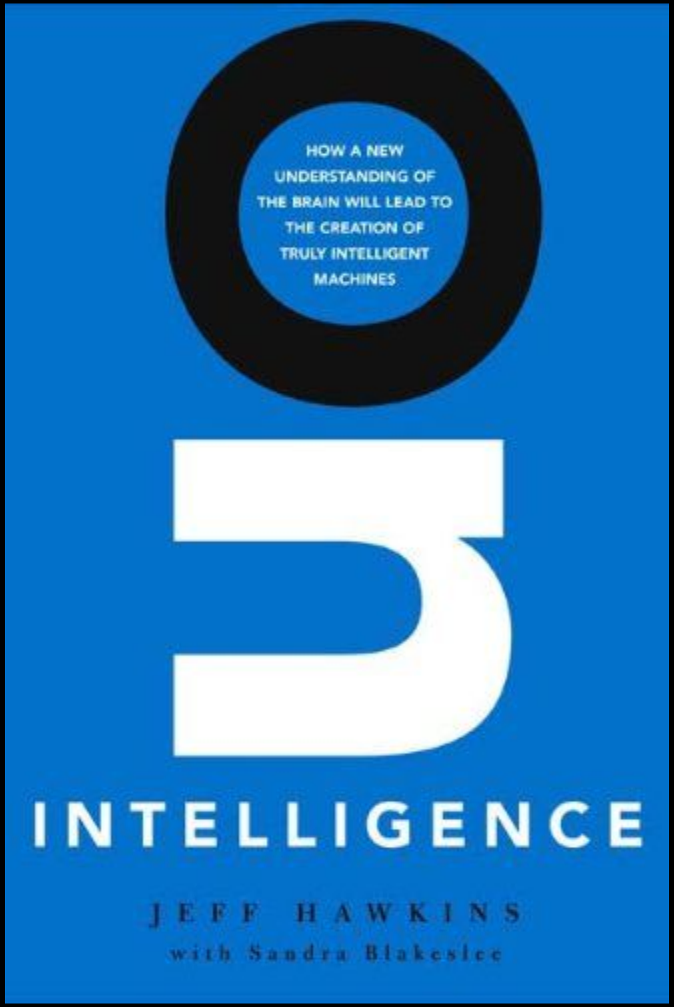

O
A
N

Artificial General Intelligence

How Much Information is the Machine Given during Learning?

Y. LeCun

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**


HOW A NEW UNDERSTANDING OF THE BRAIN WILL LEAD TO THE CREATION OF TRULY INTELLIGENT MACHINES
INTELLIGENCE
JEFF HAWKINS
with Sandra Blakeslee