AlphaGo

Nick Greenquist and Alex Hedges

What Is AlphaGo?

What Is AlphaGo?

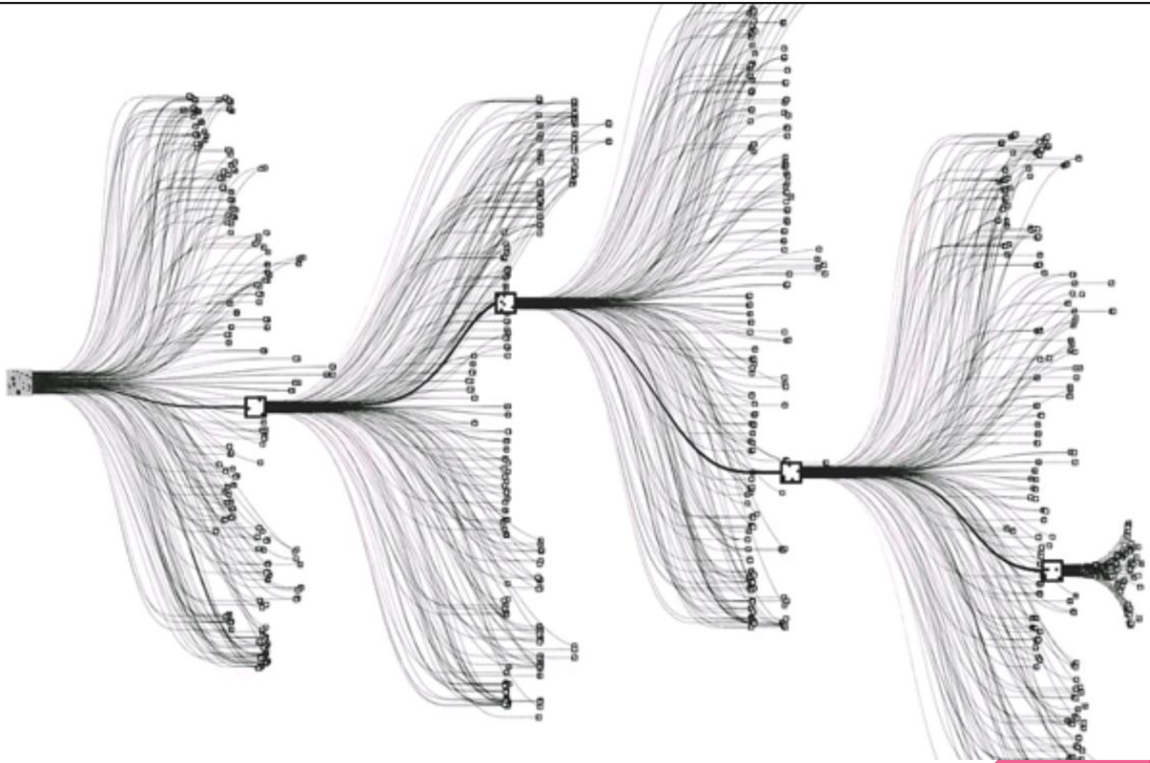
- AI developed by DeepMind that plays the board game Go
    - DeepMind is owned by Google
- Was created to test how effective DeepMind's **deep learning neural network** algorithm was

AI’s Struggle with Go

AI’s Struggle with Go

- 10^{761} possible game states, compared to 10^{120} for chess
- More legal moves per turn than chess and longer games (about ×2 more total moves per game)
- Needed a new method
    - Old methods relied on **Monte Carlo search trees**

Go Game Tree



Monte Carlo Search Trees

- **MCST** is an alternative to searching a game tree like we did with tic-tac-toe
  - Run simulation games from **current state** until someone wins
  - Simulations use random moves at the beginning
- Simulations **return values** like if a certain player won

Monte Carlo Search Trees

- **Return values** are used in later simulations
- There is a direct correlation with the **more simulations** you run, the **better** the algorithm gets at winning
- To combat the algorithm from becoming too narrow (picking the same moves over and over again) a **random component** is added to promote **exploration** of new states

Hello, AlphaGo

AlphaGo

- DeepMind’s AlphaGo research paper
    - “Mastering the Game of Go with Deep Neural Networks and Tree Search” from *Nature*

Big Picture

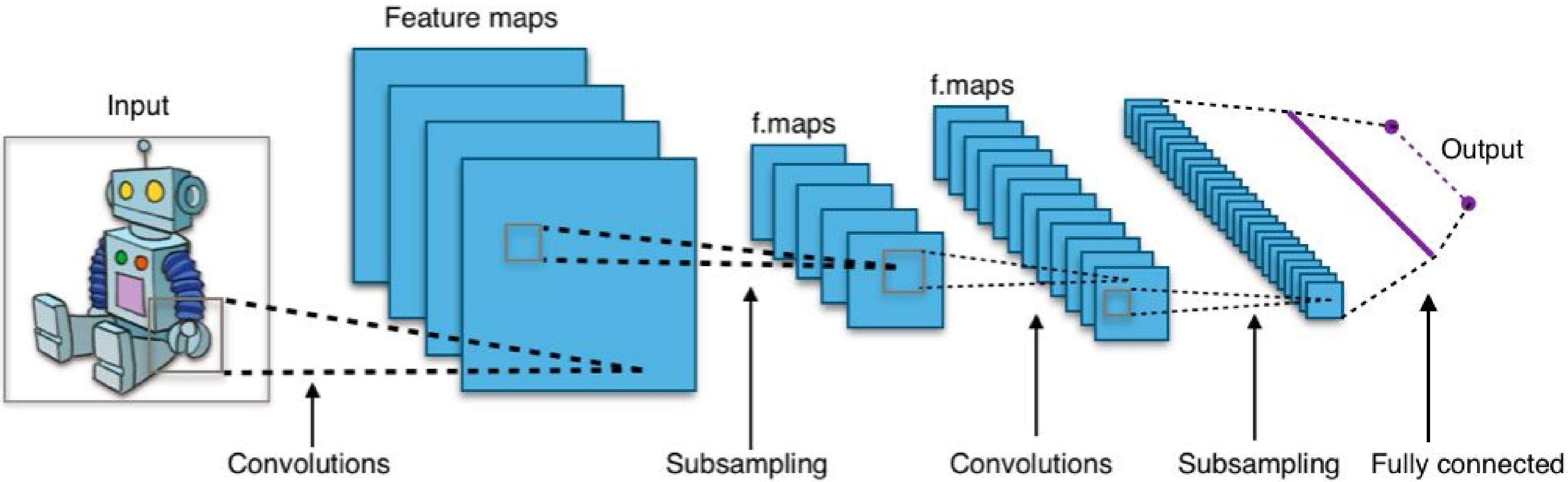
Big Picture

- DeepMind took the existing MCST approach and added **machine learning** to the mix
  - **Neural net** approach
  - Good at handling a lot of data
- Two main components
  - **Tree search** (MCST)
  - **Convolutional networks** that guide MCST

What Is a Convolutional Network?

- Used extensively in **image processing**
- Takes an image as an input and applies filters to it
- At every game state, AlphaGo passes in the actual image of the board to the algorithm
- AlphaGo trained 4 separate networks: **3 policy networks** and **1 value network**

Convolutional Network Diagram



Convolutional Network

- **Policy:** guides which **action** to take **from** current state
  - Out of all legal moves from this state, which has the highest odds of resulting in a win?
- **Value:** assigns **value** to **this** current game state
  - What are the odds of this player winning from this state?

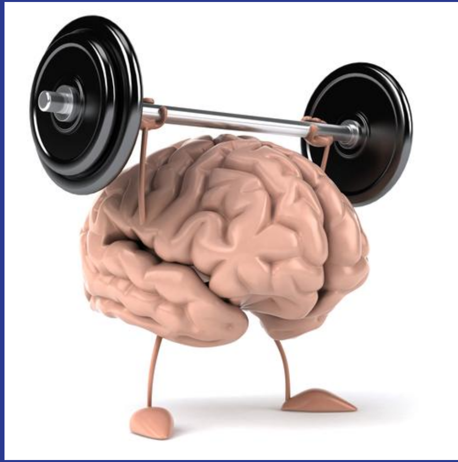
Different Convolutional Networks

- 3 policy networks
  - **SL policy** or supervised learning policy
  - **RL policy** or reinforcement learning policy
  - Fast or **rollout policy**
- 1 value network

Combining Approaches

- **Combine** tree search and convolutional networks is similar to using both reflection and intuition
- **Reflection** and **intuition**: closer to how a human thinks
  - **Reflection**: We look back at how we handled similar situations (i.e. tree search)
  - **Intuition**: We use our gut to make decisions (i.e. heuristics)

Training



Training: Policy

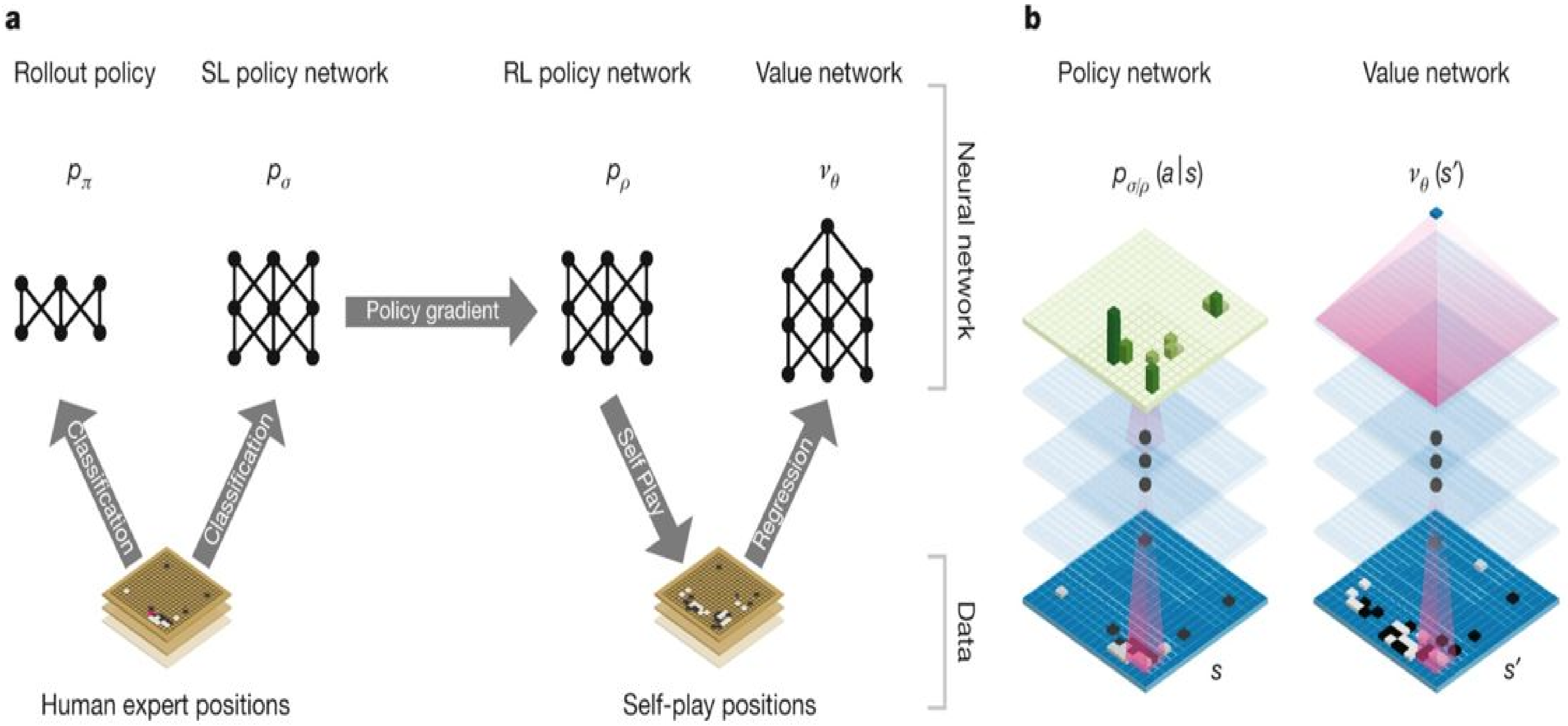
- First, **SL policy network** was trained on 30 million game positions (played by experts) obtained from a Go server
  - 13 layers deep
  - **Deep learning**: a neural net with many layers
- Next, a faster **rollout policy** network was trained
  - 1500x faster than SL policy

Training: Policy

- The **RL policy network** is trained by having the SL network play against **random past iterations** of itself
  - Prevents **overfitting** from if you just play the SL network against one with identical weights
  - 1.2 million total games
- This RL policy is the strongest policy

Training: Value

- Next, the **value network** was trained using 30 million unique positions collected from the RL policy playing itself
- This value network essentially acts as an **evaluation function** at each state
  - However, evaluations functions are designed by hand
  - This value network is **learned**
  - This is one key difference from DeepBlue’s chess algorithm that relied on **hand-crafted evaluation**



a
Rollout policy
SL policy network
RL policy network
Value network
p_π
p_σ
p_ρ
v_θ
Policy gradient
Classification
Classification
Self Play
Regression
Human expert positions
Self-play positions
Neural network
Data
b
Policy network
Value network
$p_{\sigma/\rho}(a|s)$
$v_\theta(s')$
s
s′

How It Plays

How It Plays

- **Combines** a policy network (rollout policy being the best), value network, and MCST together
  - Using the SL or RL in real gameplay was **too slow**
- This MCST keeps track of how many times a node (game state) has been visited
  - The **more times a node is visited**, the **less likely to pick** it in the future
  - Encourages **exploration**

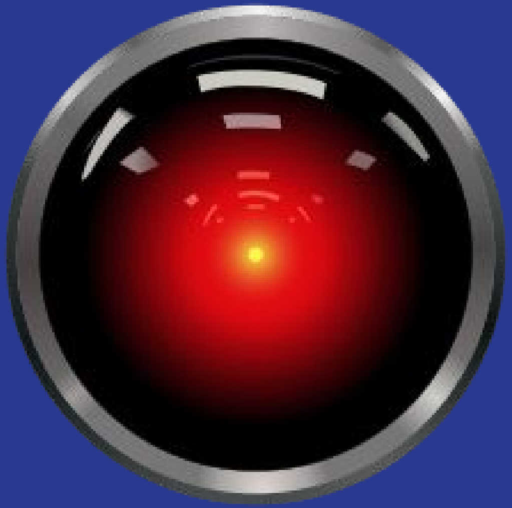
How It Plays

- Every turn, AlphaGo is running simulations in its “head” from the current game state
  - It uses the two neural networks to **guide** the moves picked in these simulations
  - After running many simulations, it will pick that move that led to the most victories in these simulations

How It Plays

- AlphaGo uses **multi-threading** to evaluate the policy networks at the same times as running MCST simulations
  - **Simulations** are run on **CPUs**
  - Policy and value **networks** are computed on **GPUs**
- **Final** version used 40 search threads, 48 CPUs, 8 GPUs
- **Distributed** version used 40 search threads, 1,202 CPUs, and 176 GPUs

Conclusion



Conclusion

- Only one human left with a higher Elo
    - He was unofficially beaten in online Go when DeepMind secretly unleashed **distributed AlphaGo** on the site
- AlphaGo gets us closer to true AI as more parts of these algorithms are becoming **learned rather than designed** by hand

Pros of the Research

- AlphaGo shines a light on a key insight: combining multiple approaches seems to have the best outcome
  - **Combining** brute force **search** (i.e. MCST) with **intuition** (values derived by machine learning, i.e. neural nets)
  - This is more similar to how humans approach problems, combining reflection of past experiences and also using their “gut”
- This research points to a trend of more “self taught” algorithms and less on human crafted evaluation functions/heuristics
  - AlphaGo made the most leaps in performance after the RL policy was created from self play

Cons of the Research

- AlphaGo is still heavily reliant on humans
    - First policy network trained on 30 million expert game positions
- Convolutional networks are a specialized form of neural nets
    - Won’t work on every problem
- Effective AlphaGo iterations were the ones that relied on massive hardware specs
    - Also, the effective policy and value networks were the ones derived from the millions of games fed into it
        - Very long to train

Questions?