ALPHA GO

$\alpha - \beta$ search + heuristics

$\downarrow$

too hard!

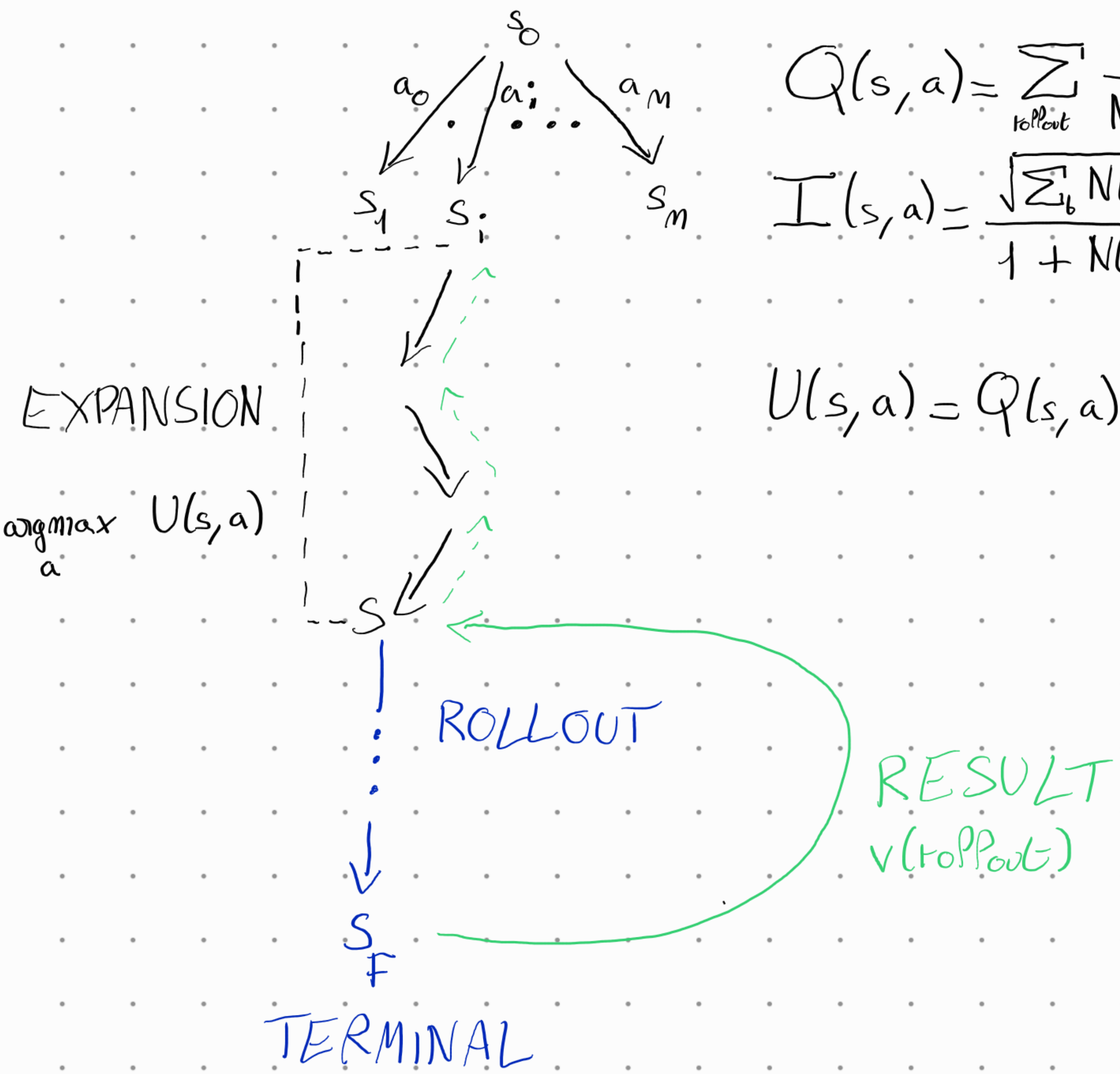
Solutions?

-
-

When branching factor is high

MCTS (1940s) is great!

(UCB, 2002)



$$Q(s,a) = \sum_{\text{rollout}} \frac{1}{N(s,a)} \cdot v(\text{rollout})$$

$$I(s,a) = \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)}$$

$$U(s,a) = Q(s,a) + c_T \cdot I(s,a)$$

We play with $\pi_s(a) = \dfrac{N(s,a)^T}{\sum_b N(s,b)^T}$

ISSUES:

- random policy
    - lot of games
    -
-

So let's use "modern tech."

Neural Networks

Basic Idea:

game state → $\boxed{\text{NN}}$ → predict probability of black winning

Supervised Training on PRO Games

David Silver (DeepMind)

- 4 weeks on 50 GPUs
- 12 layer CNN
- 57% accuracy on test set
  (SOTA was 44%)

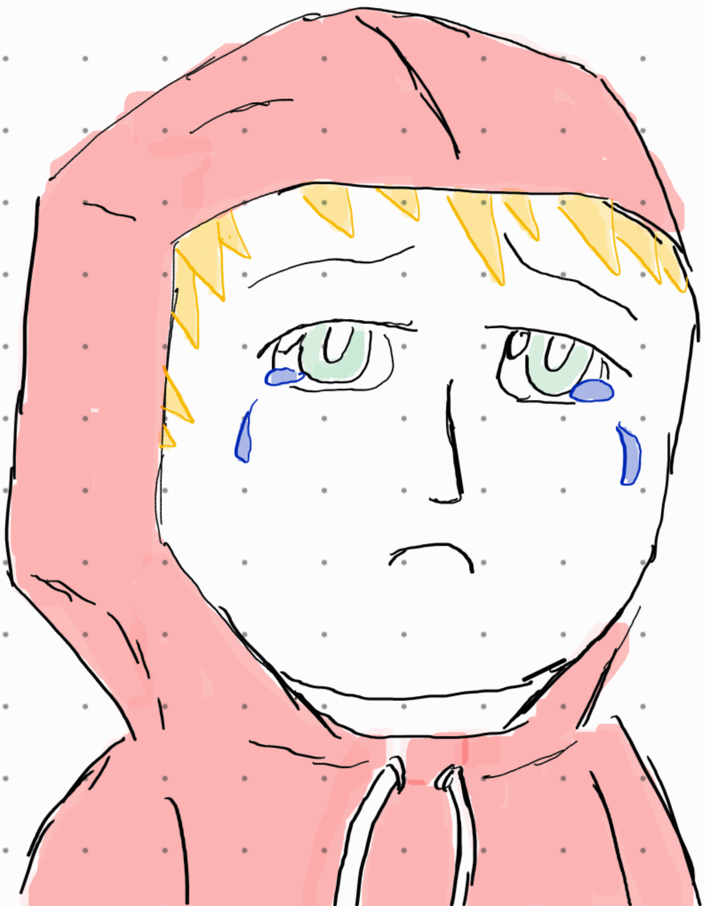
Reinforcement Learning

Train with RL with policy gradient

Self - Play (bootstrap f previous NN)

David Silver

- 1 week on 50 GPUs
- 80% accuracy
  ↳ amateur level



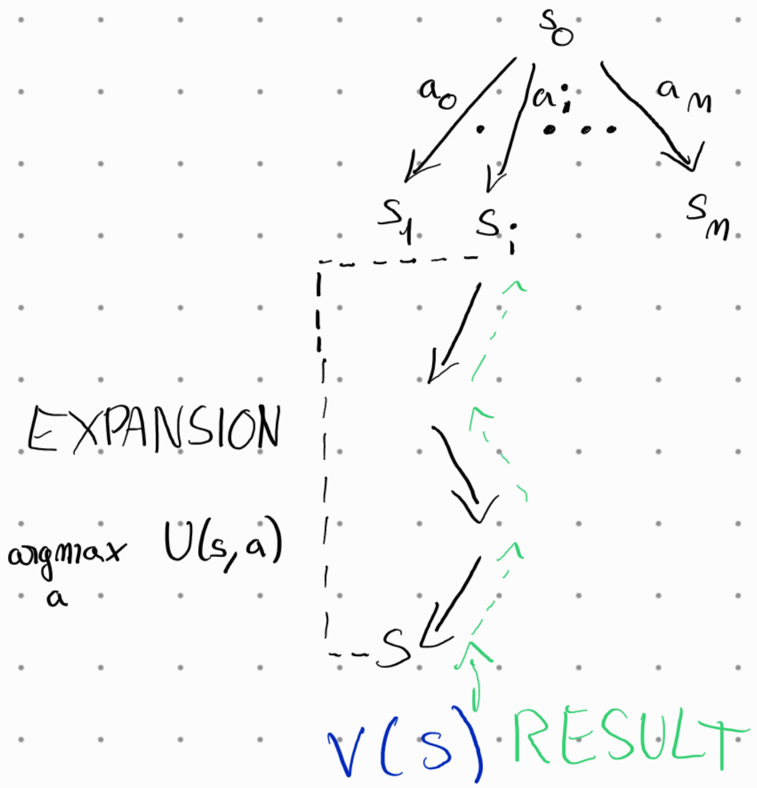
SOLUTION

?

Gotta Mix'em all!

MCTS + NN + RL = Alpha Go

game state $\rightarrow \boxed{NN} \rightarrow$ $\forall a.\ p(a|s)$
$V(s)$



$$Q(s,a) = \sum_{s'} \frac{1}{N(s,a)} \cdot V(s')$$

$$I(s,a) = \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)} \, p(a|s)$$

$$U(s,a) = Q(s,a) + c_T \cdot I(s,a)$$

(PUCT)

We play with $\pi_s(a) = \dfrac{N(s,a)^T}{\sum_b N(s,b)^T}$

$V(s)$: value function

↳ better rollouts estimations

$p(a|s)$: policy function

↳ bias exploration towards promising branches