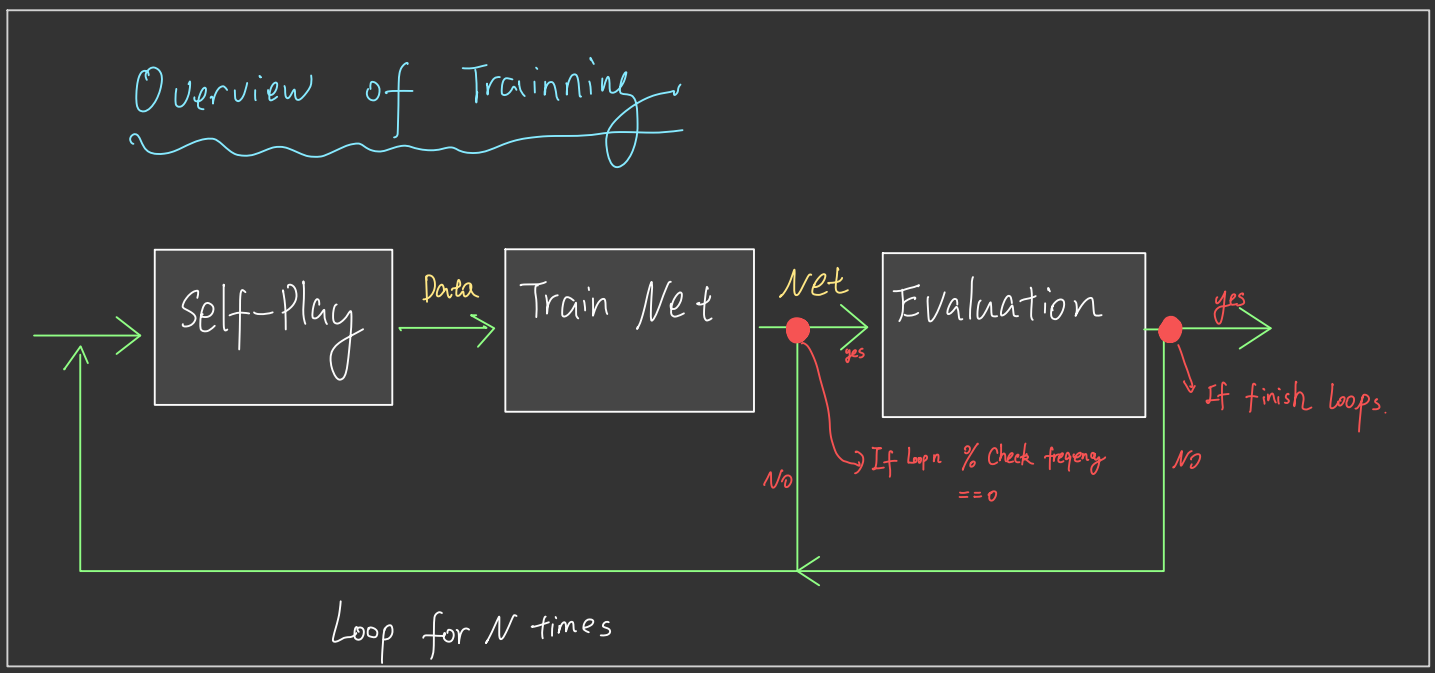
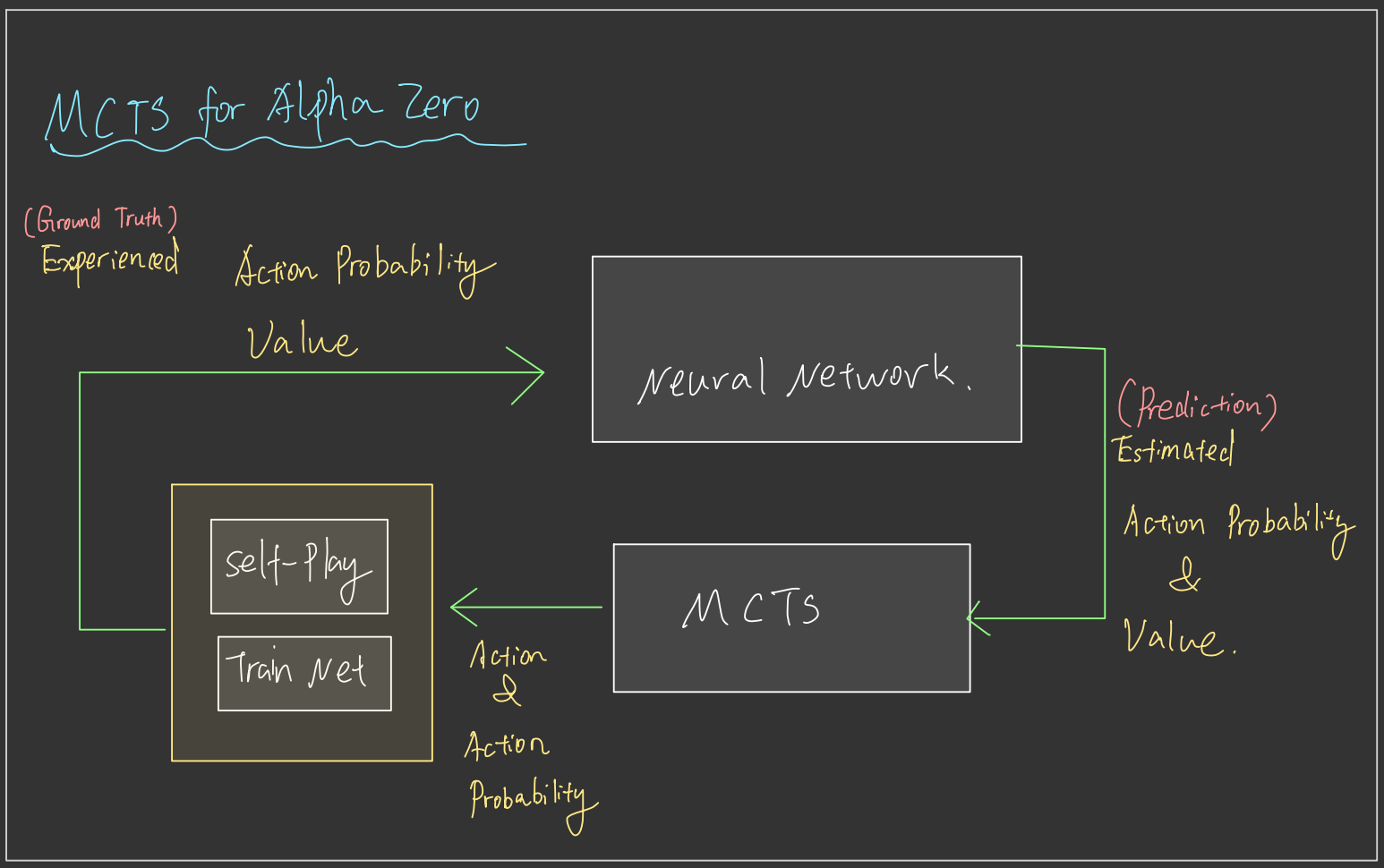

Alphazero
TOP 10 OF THE 2010S
#3


Overview of Trainning
Self-Play
Data
Train Net
Net
yes
Evaluation
yes
If finish loops.
If loop n % Check freqeny == 0
No
No
Loop for N times


MCTS for Alpha Zero
(Ground Truth) Experienced
Action Probability
Value
Neural Network.
(Prediction) Estimated
Action Probability
&
Value.
self-Play
Train Net
MCTS
Action
&
Action
Probability

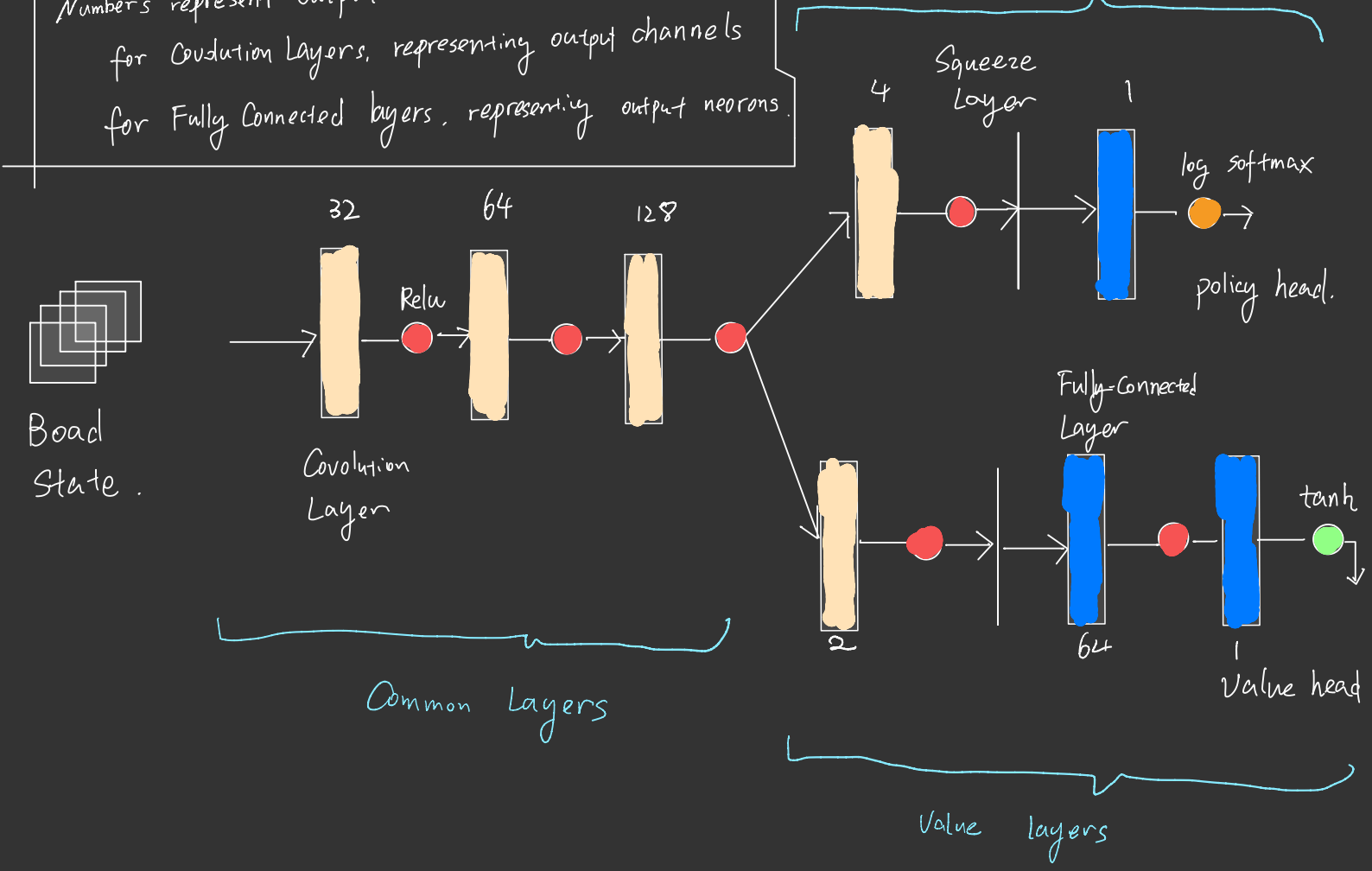
Architechture of Deep Neural Network

Note:

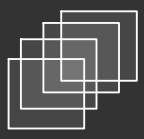
Numbers represent output neorons.

for Coudution Layers, representing output channels

for Fully Connected bayers, representing output neorons.



①



Boad State.

Input board state represent the current state of board.

- Shape : 4d tensor (batch size , 4, height , width)
- Meaning of 4 layers:

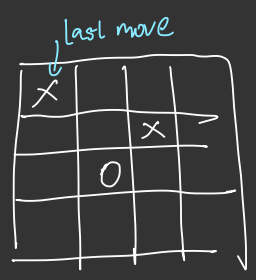
Layer 1 : All the moves for your own.

Layer 2 : All the moves for opponent

Layer 3 : The last move

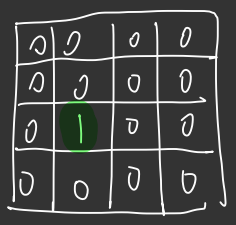
Layer 4 : Indicat the player to play for current turn.

For example

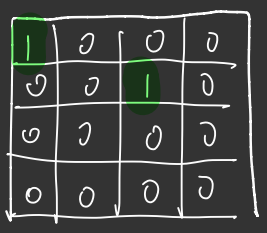


O
↳ current turn

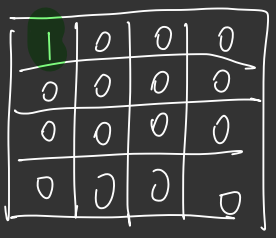
Layer 1



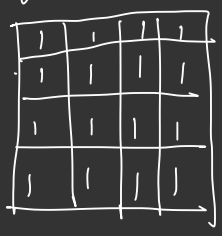
Layer 2



Layer 3



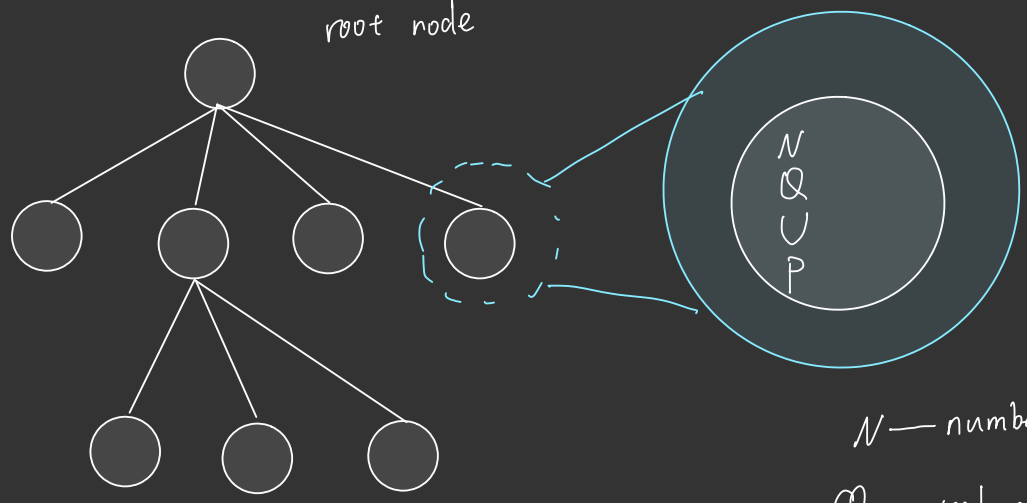
Layer 4



1 — for "O"

0 — for "X"

Monte Carlos Tree Search (MCTS) for Alpha Zero

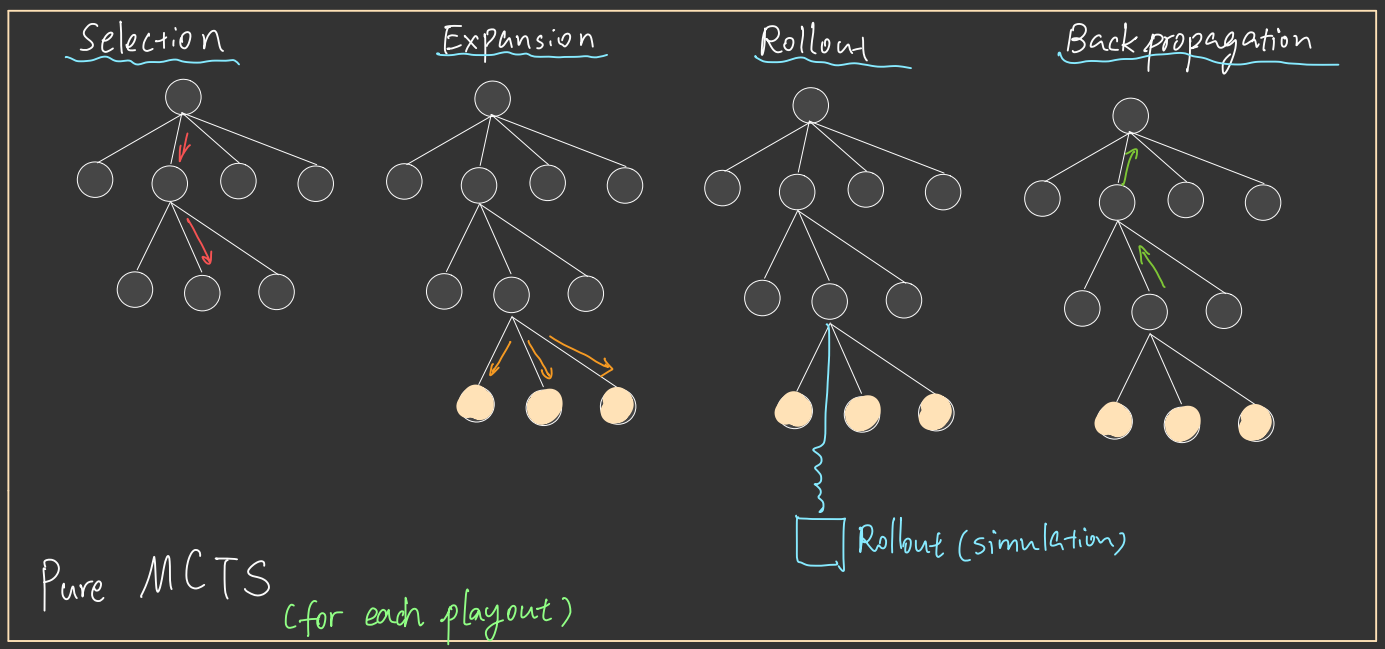


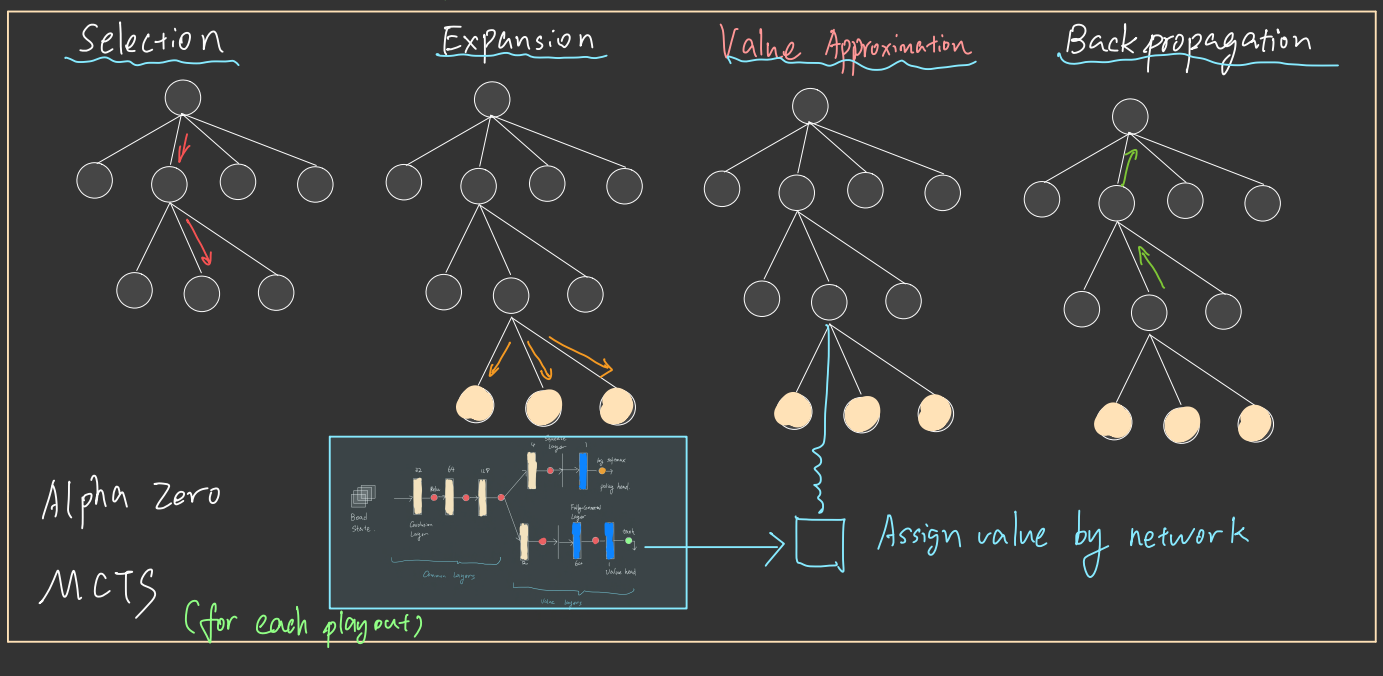
N — number of visits

Q — value

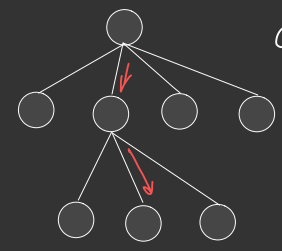
U — Exploration function given P and N

P — Action Prior





Selection



choose the node with $\underset{\text{nodes}}{\text{argmax}}(Q+U)$ until reaching the leaf node.

Q is the average of Q_t

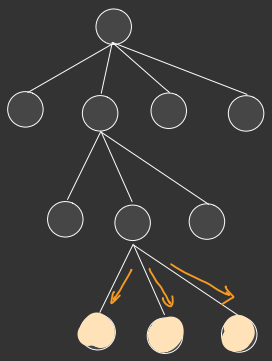
Q_t is updated Q value for the t times.

$$U = C_{puct} \cdot P \sqrt{\frac{N_{parent}}{N+1}}$$

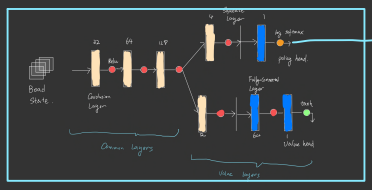
N_{parent} – N for parent node

C_{puct} – Constant

Expansion

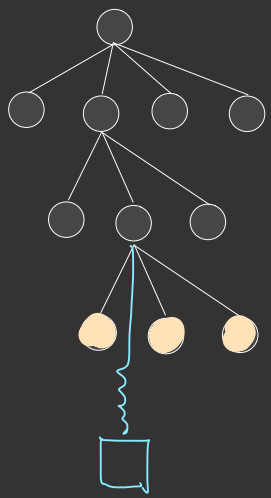


- Expand nodes corresponding to available actions.
- Assign the action prior P.
  - For pure MCTS, P is in uniform distribution.
  - For Alpha zero,



→ P ↘ Assigned with policy head.

Rollout / Value Approximation



For Pure Mcts:

Rollout by creating a game simulation with random policy starting from current state.

Q_t = result (1, 0, -1)

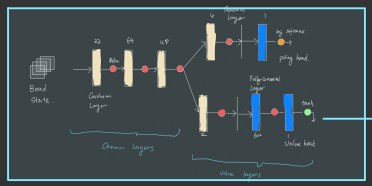
For Alpha Zero:

if game is terminated.

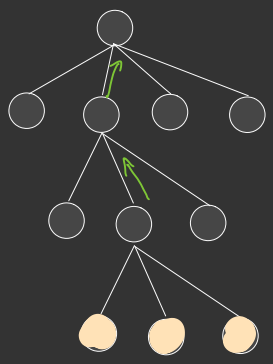
Q_t = result (1, 0, -1)

else (not terminated)

Q_t = value head of network.



Backpropagation (B.P.)



For the current node, its parent node and its ancestral nodes (parent's parents ....):

$$Q = Q + (-1)^{k+1} \frac{Q_t}{N}$$

$$N = N + 1$$

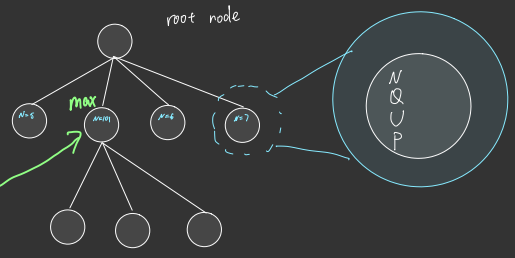
where k is the kth step of backpropagation.

Compute Action with MCTS (given current state)

For Pure MCTS:

① playout for n times

② Choose the next move by.



argmax (N)

children nodes for the root node.

③ Set the chosen move node as root node and reset the tree

For Alpha Zero:

① playout for n times

② Action probability = $\text{Softmax}\left(\frac{1}{\text{temp}} \times \log(N) + 1 \cdot 10^{-10}\right)$

temp ↓ temperature (hyper-param)

$1 \cdot 10^{-10}$ ↓ small number

③ If self-play training:

a. P = Action probability × 0.7 + Dirichlet Noise × 0.3.

b. Sample one action from P.

c. Set the chosen move node as root and reuse the subtree.

Else (when not training)

a. P = Action probability

b. Sample one action from P.

c. Set the chosen move node as root and reset the tree.

Self-play for alpha zero.

Self-play for n times.

For each self-play:

① Experience Data = self-play (Alpha Zero)

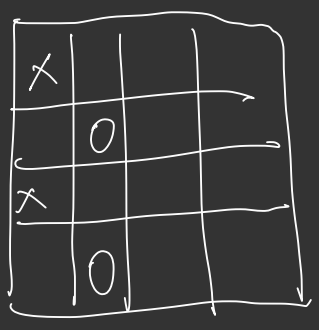
- Experienced Data includes:
  - Experienced States.
  - Experienced MCTS probability
  - Final Winner.

② Augment data

③ Extend to buffer.

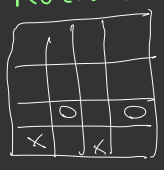
Augmented Data

Original State

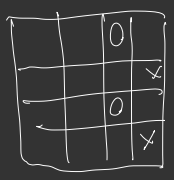


Rotate

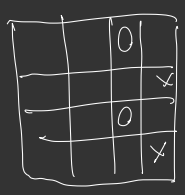
90°



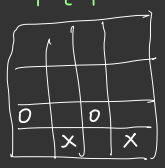
180°



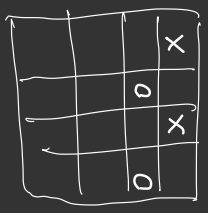
270°



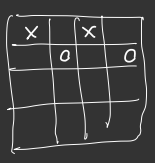
Flip



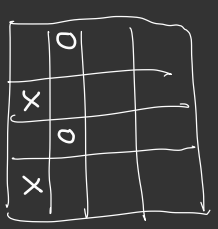
Flip 90°



Flip 180°



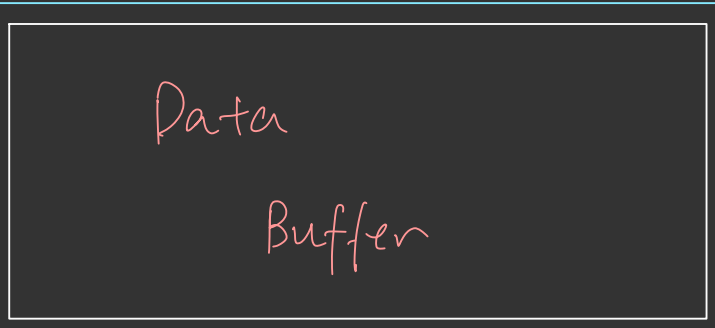
Flip 270°



Flip 0°

Data Buffer.

Experienced Data →



- Can be sampled randomly for training.
- For storing data.

Train Network for Alpha Zero.

① Sample experienced data from data buffer.

② Compute origin action probability with network.

③ Train network with experienced data in a loop. for n times

(a) train network

i) forward pass: $\hat{prob}$, $\hat{value}$ = network (states)

ii) value loss = Mean Square Error ($\hat{values}$, $value_{Data}$)

iii) Policy loss = KL Divergence ($\hat{prob}$, $prob_{Data}$)

iv) loss = value loss + Policy loss

v) Back propagation with loss (Optimization)

(with learning rate: Fixed learning Rate × Lr_mul)

(b) ∇ = KL Divergence (action-$prob_{origin}$, action-$prob_{new}$)

(c) if $\nabla > 4 \cdot \nabla_{target}$, Early Stopping, break the loop.

④ Adjust learning Rate Multiplier: lr_mul.

if $\nabla > 2\,\nabla_{target}$ and lr_mul > 0.1

lr_mul = lr_mul / 1.5

if $\nabla < \frac{1}{2}\nabla_{target}$ and lr_mul < 10

lr_mul = lr_mul × 1.5

Evaluation for Alpha Zero.

① Initialize Player Alpha Zero and Player Pure MCTS.

② Evaluation Loop for n times.

(a) Rollout Simulation (Alpha Zero vs Pure MCTS)

(b) Record Result.

③ Compute winning ratio.

$$\text{Winning ratio} = \frac{(\text{winning times} + 0.5\ \text{tie times})}{\text{total times}}.$$

④ Save Model