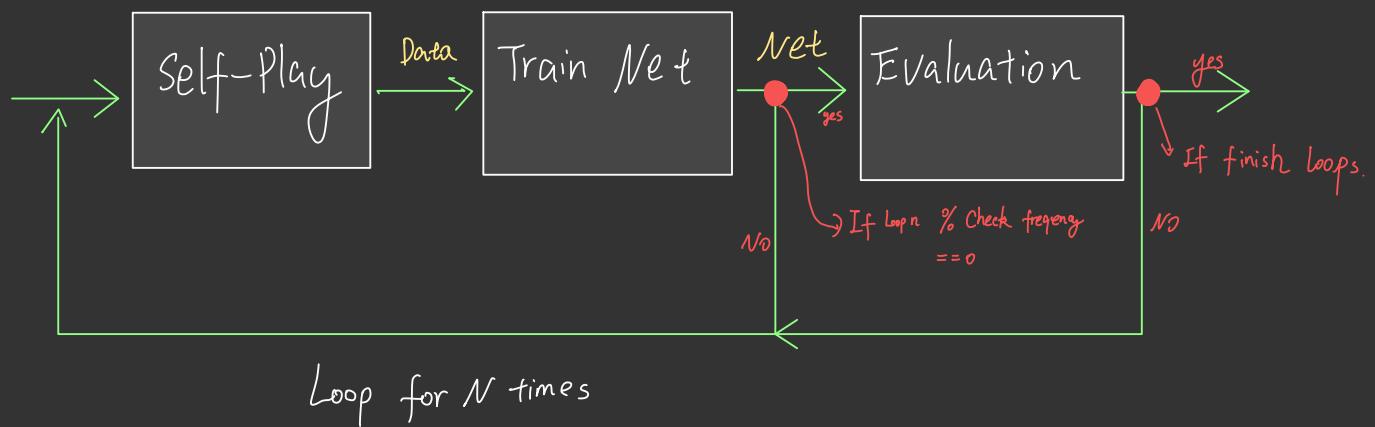
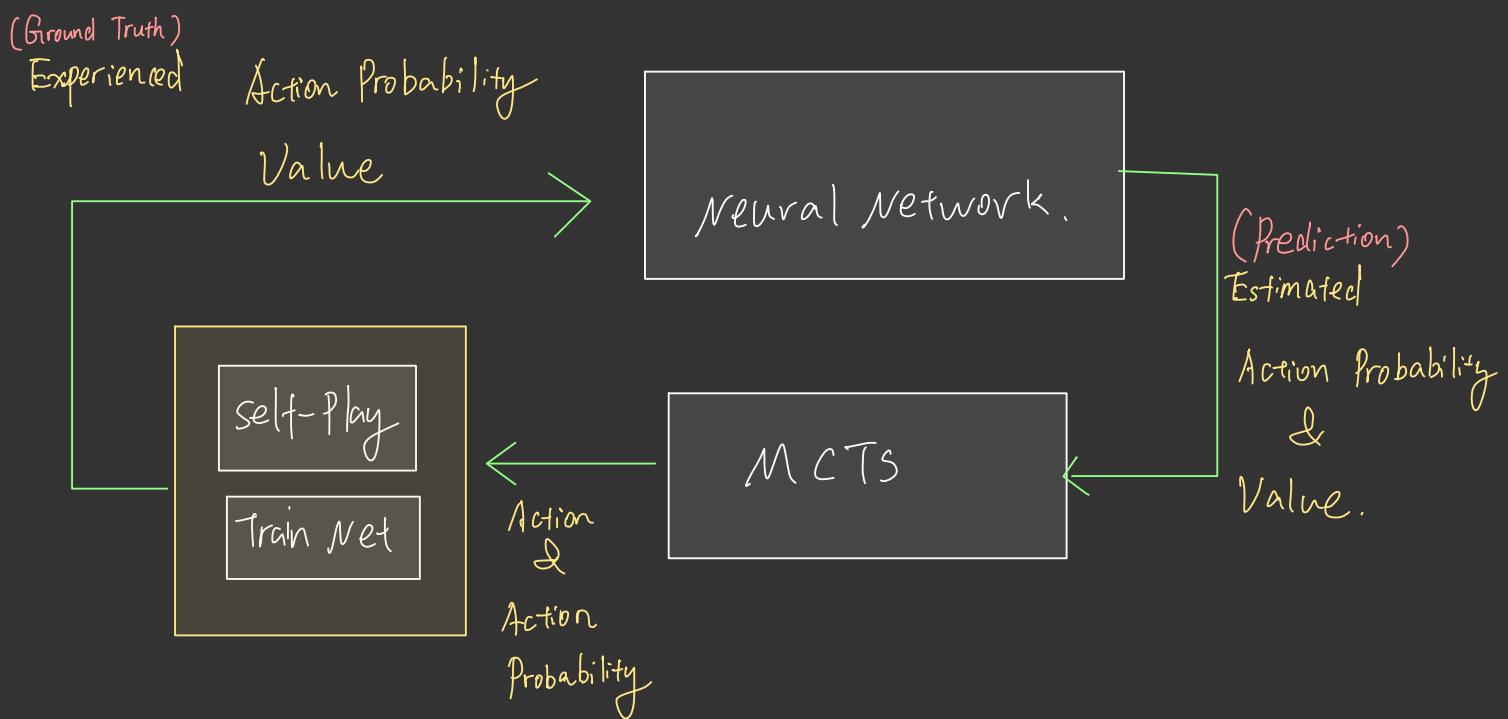




Overview of Training



MCTS for Alpha Zero



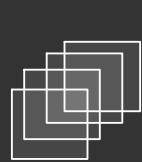
Architecture of Deep Neural Network

Note:

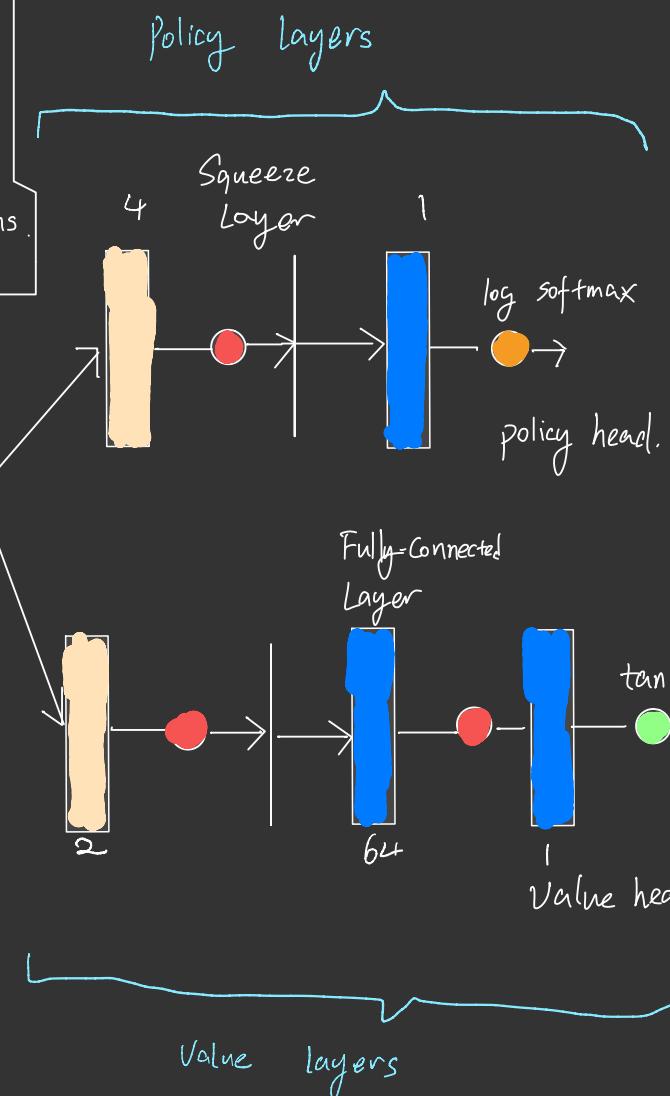
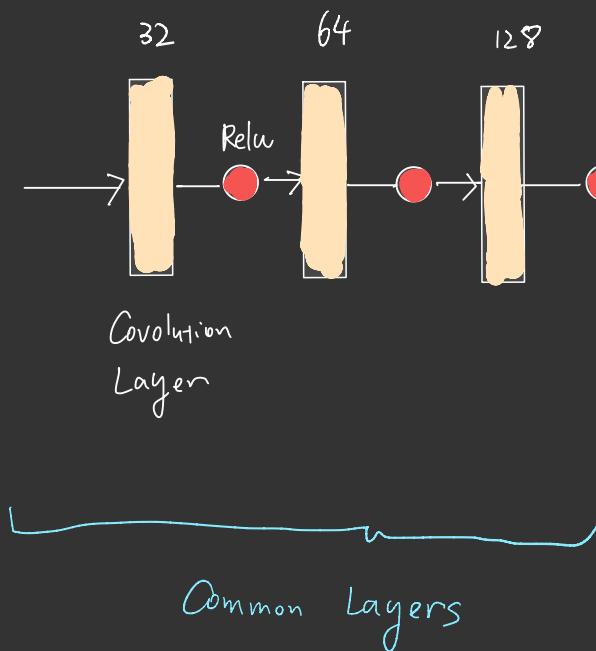
Numbers represent output neurons.

for Convolution Layers, representing output channels

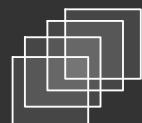
for Fully Connected layers, representing output neurons.



Board State.



①



Board State.

Input board state represent the current state of board.

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

- Meaning of 4 layers :

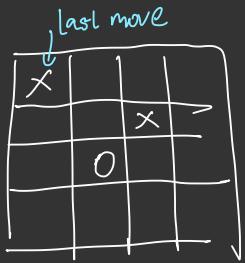
Layer 1 : All the moves for your own .

Layer 2 : All the moves for opponent

Layer 3 : The last move

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

For example



Layer 1

0	0	0	0
0	0	0	0
0	1	0	0
0	0	0	0

Layer 2

1	0	0	0
0	0	1	0
0	0	0	0
0	0	0	0

Layer 3

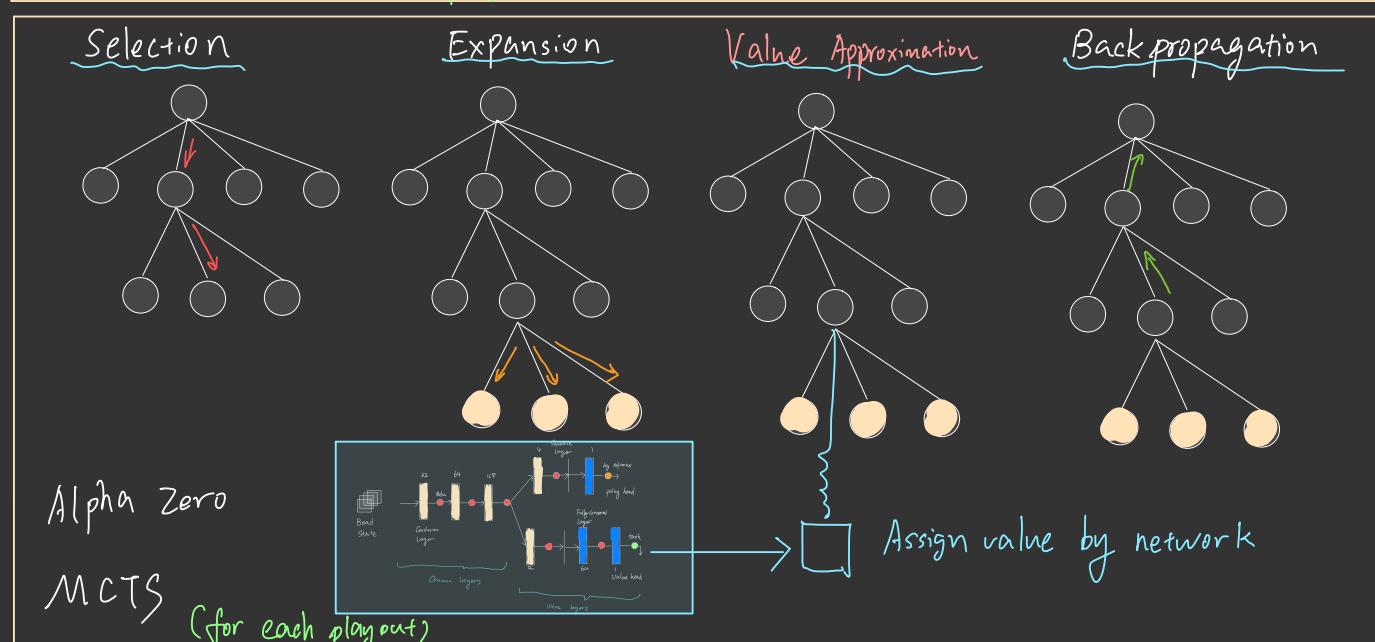
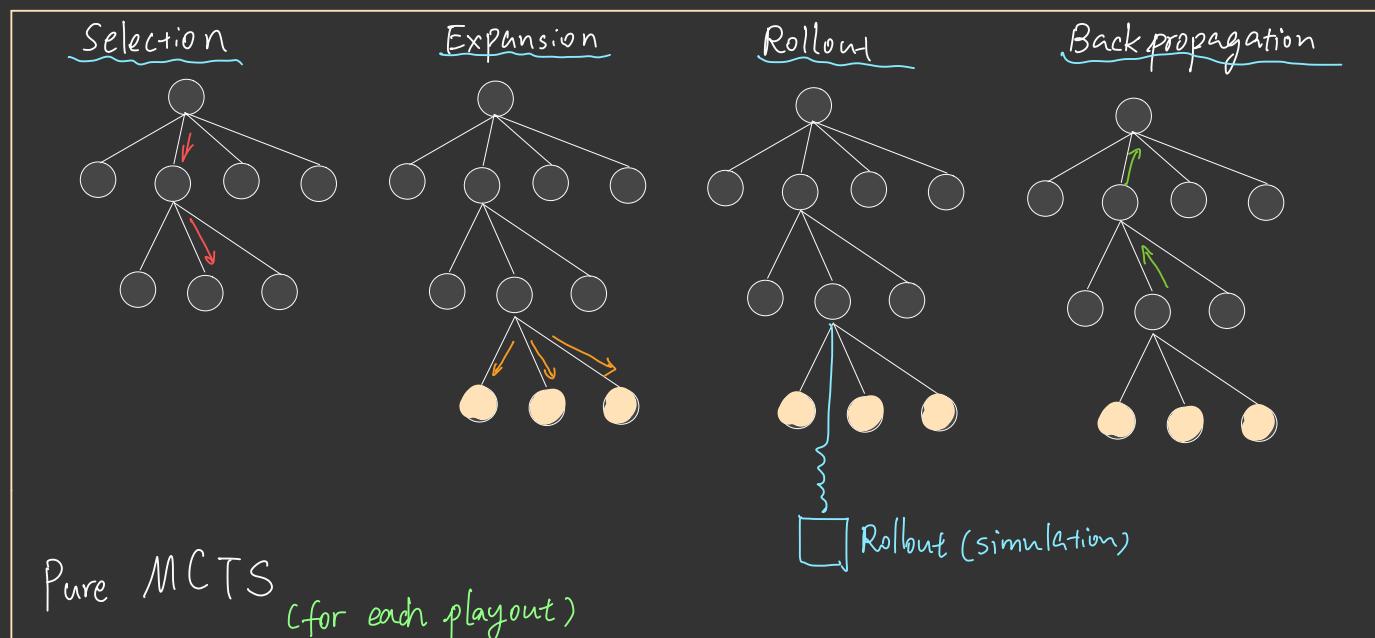
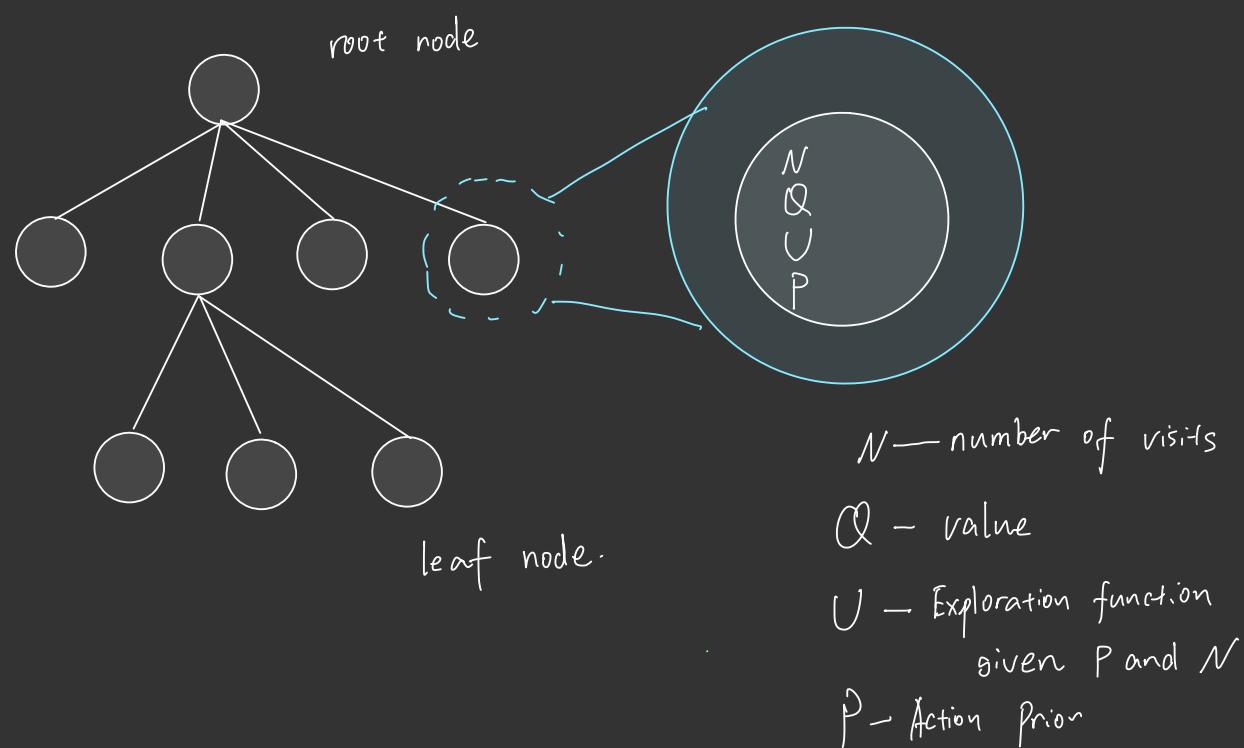
1	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Layer 4

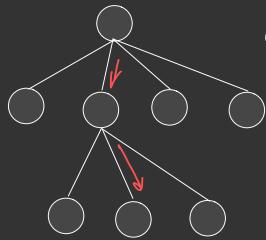
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

1 - for "0"
0 - for "X"

Monte Carlos Tree Search (MCTS) for Alpha Zero



Selection



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

Q is the average of Q_t

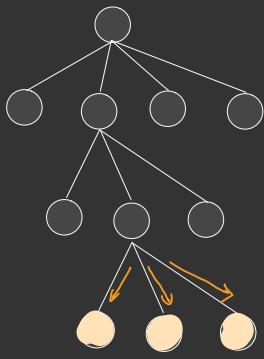
Q_t is updated Q value for the t times.

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

N_{parent} - N for parent node

C_{puct} - Constant

Expansion

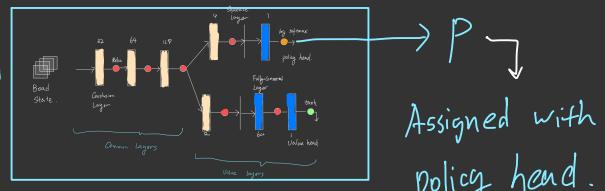


- Expand nodes corresponding to available actions.

- Assign the action prior P .

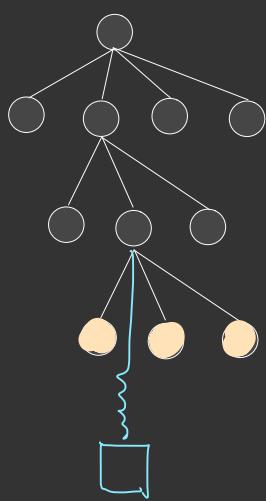
- For pure MCTS, P is in uniform distribution.

- For Alpha zero,



Assigned with policy head.

Rollout / Value Approximation



For Pure Mcts:

Rollout by creating a game simulation

with random policy starting from current state.

$$Q_t = \text{result}(1, 0, -1)$$

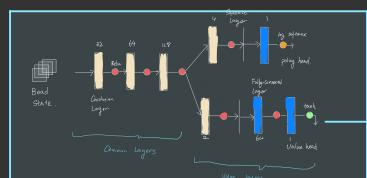
For Alpha zero:

if game is terminated.

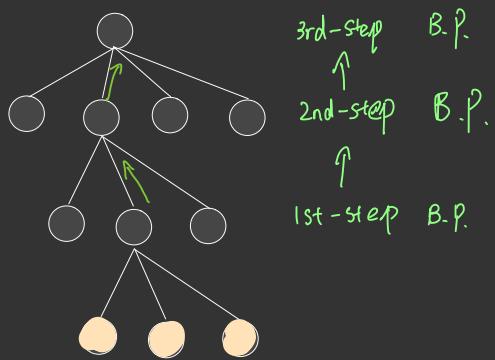
$$Q_t = \text{result}(1, 0, 1)$$

else (not terminated)

$$Q_t = \text{value head of network.}$$



Backpropagation (B.P.)



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

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

$$N = N + 1$$

where k is the k th step
of backpropagation.

Compute Action with MCTS (given current state)

For Pure MCTS :

① playout for n times

② Choose the next move by -

$\arg \max (N)$

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

\downarrow
temperature
(hyper-param)
 \downarrow
small number

③ If self-play training :

a. $P = \text{Action probability} \times 0.7 + \text{Dirichlet Noise} \times 0.3$,

b. Sample one action from P.

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

Else (when not training)

a. $P = \text{Action probability}$

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:

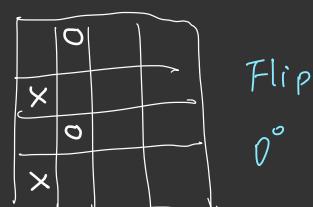
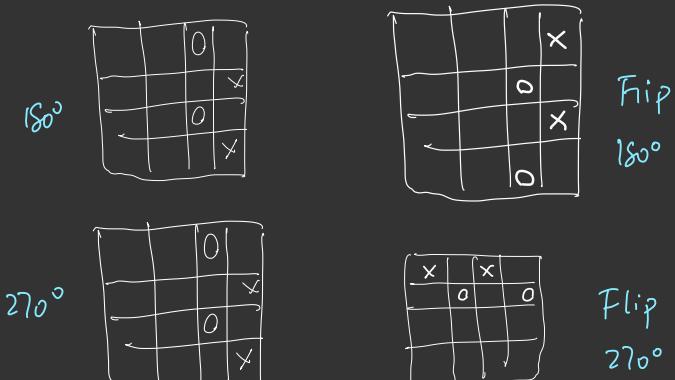
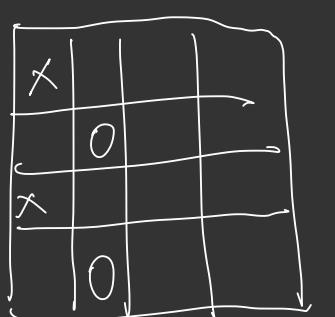
① Experience Data = self-play (Alpha Zero)

② Experienced Data includes:

- Experienced States.
- Experienced MCTS probability
- Final Winner.

③ Augment data

④ Extend to buffer.



Experienced
Data



Data Buffer

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

Train Network for AlphaZero.

Train Net

- 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 k
 - i) forward pass: \hat{prob} , $\hat{value} = \text{network}(\text{states})$
 - ii) value loss = Mean Square Error (\hat{values} , $\text{value}_{\text{data}}$)
 - iii) policy loss = KL Divergence (\hat{prob} , $\text{prob}_{\text{data}}$)
 - iv) loss = value loss + policy loss
 - v) Backpropagation with loss (Optimization)
(with learning rate: Fixed learning rate $\times Lr_mul$)
 - (b) $\nabla = \text{KL Divergence}(\text{action-prob}_{\text{origin}}, \text{action-prob}_{\text{new}})$
 - (c) if $\nabla > 4 \cdot \nabla_{\text{target}}$, Early Stopping, break the loop.

④ Adjust learning Rate Multiplier: Lr_mul .

if $\nabla > 2 \cdot \nabla_{\text{target}}$ and $Lr_mul > 0.1$

$$Lr_mul = Lr_mul / 1.5$$

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

$$Lr_mul = Lr_mul \times 1.5$$

Evaluation for Alpha Zero .



① Initialize Player AlphaZero 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