Research of AlphaGo technique

My job is to research the article: Mastering the game of Go with deep neural networks and tree search. The article is from:
https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf

The world famous AlphaGo, which is the master-piece of Google, had defeated all human Go players in this earth. What technic does AlphaGo use? I hope this simple review helps reader to understand the AlphaGo skills.

AlphaGo uses 3 neural networks and "Fast Rollout Policy", and combines them using Monte Carlo tree search (MCTS).

The 3 neural works are **Supervised Learning of policy networks**, **Reinforcement Learning of Policy Networks** and **reinforcement learning of value networks**.

- Supervised Learning of policy networks (SL): This network is supervised trained using KGS data set (30 million positions) to predict human expert moves and get 55.7% accuracy.
- Reinforcement Learning of Policy Networks (RL): The RL policy network is identical in structure to the SL policy network. And RL weights are initialized to the same values of SL. They play games between the current policy network and a randomly selected previous iteration of the policy network. In the end, the RL policy network won more than 80% of games against the SL policy network. Using no search at all, the RL policy network won 85% of games against Pachi, which is the one of strongest open source Go programs.
- Reinforcement learning of value networks: This neural network has a similar architecture to the RL policy network, but outputs a single prediction instead of a probability distribution. The prediction is outcome from position s of games played by using policy p for both players.

Fast rollout Policy is the technique based on Local pattern matching and Logistic regression. The time spend in fast rollout policy is only 2 microsecond comparing to 3 millisecond in policy networks. But the accuracy of fast rollout is only 24.2%.

Now we have all the components AlphaGo needs, AlphaGo will use below process to play game:

(1) Evaluate all successors s' of the root position s. Each edge(s, a) of the search tree stores an action value Q(s,a), visit count N(s,a), and prior probability P(s,a). In the beginning, all Q(s,a) =0;

(2) At each time step t of each simulation, an action a_t is selected from state s_t

$$a_t = \arg\max_a (Q(s_t, a) + u(s_t, a))$$

u(s,a) is a bonus:

$$u(s,a) \propto \frac{P(s,a)}{1+N(s,a)} \quad .$$

When the traversal reaches a leaf node s_L at step L, the leaf node may be expanded. The leaf

position s_L is processed just once by the SL policy network $p_\sigma(a|s)$. The leaf node is evaluated in two very different ways: first ,by the value network $v_\theta(s_L)$; and second, by the outcome z_l of a random rollout played out until terminal step T using the fast rollout policy. We use below function to calculate Q(s,a):

$$V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda z_L$$

$$N(s,a) = \sum_{i=1}^{n} 1(s,a,i)$$

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i)V(s_L^i)$$

where s_L^i is the leaf node from the ith simulation.

(3) Once the search is complete, the algorithm chooses the most visited move from the root position.

From above the progress, we know the object of function "$a_t = \arg\max_a (Q(s_t,a) + u(s_t,a))$" is to decide which action should be expanded. And the most visited move is to decide which action should be chosen.

The result of the tournament suggest that single-macine AlphaGo is many dan ranks stronger than any previous Go program, winning 494 out of 495 games (99.8%) against other Go grogram. To provide a greater challenge to AlphaGo, they also played games with four handicap stones (that is, free moves for the opponent); AlphaGo won 77%, 86% and 99% of handicap games against Crazy Stone, Zen and Pachi, respectively. The distributed version of AlphaGo was significantly stronger, winning 77% of games against single-machine AlphaGo and 100% of its games against other programs.