# Research Review:

# Mastering the Game of Go with Deep Neural Networks and Tree Search

*Publication by David Silver et al. – January 28, 2016*

*Review by David Hariton Katz – June 19, 2017*

In their seminal research study[1], David Silver and the Google DeepMind team create an intelligent agent capable of achieving super-human performance in the game of Go. Applying deep neural networks and reinforcement learning to a Monte-Carlo tree search algorithm, "AlphaGo" attained a 99.8% winning rate against other Go programs and defeated the European Go Champion 5 games to 0 in standard match play.

**Research Goals**:

- **Achieve a New Milestone for Artificial Intelligence**: Many view Go as the "final frontier" for intelligent computer agents in the realm of classic board games. Two of the game's key challenges include:
  i. **Massive search space**: On a standard 19x19 board, a single game of Go has roughly $2x10^{170}$ possible board positions[2].
  ii. **Difficulty evaluating board positions and moves**: Professional Go players often cite "intuition" as their rationale behind moves. Thus, researchers have found it difficult to identify the characteristics of board positions and moves that correlate to winning and losing performances.
- **Demonstrate the Effectiveness of Deep Learning**: Deep learning has become a rapidly expanding area of research thanks to advances in computing power and the availability of large datasets. Neural networks lay the foundation for AlphaGo's move selection and evaluation functions, and the agent is strong example of how deep learning can be applied to enhance computer performance in a classic test of intelligence.

**Design Architecture**:

- **Input**: *Convolutional Neural Network*
  - **Input**: 19x19 images that illustrates the position of the board
  - **Output**: Numerical representation of board state
  - **Significance**: Highlights key positional information and makes board states easier to process
- **Training**: *Policy Networks – Supervised Learning (SL) & Fast Rollout*
  - **Input**: Numerical representation of 30 million different board states from KGS Go Server (Full list of input features in Extended Data Tables 2, 4)
  - **Output**: Probability distribution of likely next move based on playing patterns of professional Go players (57.0% test accuracy using all input features)
  - **Significance**: Provides agent with a high-quality baseline intuition (gradients) for moves – Processing Speed: SL Policy Network (3ms) vs. Fast Rollout (2$\mu$s)
- **Training**: *Policy Network – Reinforcement Learning (RL)*
  - **Input**: Board state (s) for each turn during games of self-play – Agent plays games against randomly selected previous iteration of the itself and uses gradient ascent to update gradients

[1] Silver, David et al. "Mastering the game of Go with deep neural networks and tree search." https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf. January 28, 2016.
[2] Tromp, John. "Counting Legal Positions in Go." http://tromp.github.io/go/legal.html. February, 2016.

  - **Output**: Probability distribution of best next move to select for player in game – Reinforcement provided at terminal states: (z = +1) for winning game, (z = -1) for losing game, (z = 0) for all non-terminal states
  - **Significance**: The RL policy network updates the gradients from the SL policy network to maximize the probability of picking moves that lead to the agent winning instead of matching professional players' moves
- **Training**: *Value Network – Regression*
  - **Input**: State-Outcome pairs (s, z) from the games of self-play executed by the RL policy network
  - **Output**: Scalar $v_\theta(s)$ that approximates the expected game outcome (z) given board state (s) → (loss) -1 <= $v_\theta(s)$ <= +1 (win)
  - **Significance**: Approximates a value function for each state that AlphaGo visits
- **Search**: *Monte-Carlo Tree Search (MCTS)*
  - **Input**: Board state in real competition (s)
  - **Output**: The most visited move from the root position, which approximates the move with the maximum likelihood of AlphaGo's victory
  - **Significance**: This is how AlphaGo decides which move to select in competition. It applies its trained understanding of board states (s) relative to game outcomes (z) to prioritize which nodes to search and determine which move maximizes the expected outcome z = +1

**Notable Results**:

- Defeated the European Go Champion, Fan Hui, 5 games to 0 in match play
- 99.8% winning rate against state-of-the-art Go programs (494/495 games; 5s time limit/move)
- Consistent winning performances against state-of-the-art Go programs despite handicapped starting position
- AlphaGo evaluated thousands of times fewer moves than did Deep Blue, IBM's superhuman chess program, even though Go is more complex than chess