

# Detailed Specification of Participating Program “BeeBee”

Seungmin Yum

## 1. Algorithm of the Thinking Part (Search, Machine Learning Mechanism)

The decision-making mechanism of this program is based on a policy–value integrated neural network architecture. Given a board state as input, the network simultaneously outputs (1) a probability distribution over all legal moves (policy) and (2) the expected outcome of the position (value). The current implementation focuses on pure neural-network-based inference without tree search.

### 1.1 Overall Network Architecture

The input is a multi-channel tensor (currently NUM\_PLANE = 1) representing the Go board, structured as a 19×19 spatial grid. The network processes the input through the following steps:

#### Initial Convolution Layer

- A 5×5 convolution expands the input from 1 channel to 128 channels (feature maps).
- This layer extracts low-level spatial patterns such as stone placement, basic territorial areas, and local clusters of stones.

#### Nine Pre-activation Attentive Residual Blocks

Each block consists of the following components:

- Batch Normalization → SiLU activation → DSCConv (3×3)
- Batch Normalization → SiLU activation → DSCConv (3×3)
- Spatial Attention module
- Identity skip connection

By repeating this block nine times, the network gradually learns higher-level Go-specific abstractions beyond stone placement, such as local fights, thickness relationships, weaknesses, and territorial boundaries.

#### Policy Head

- A 1×1 convolution reduces the 128 channels to 1.
- Batch Normalization is applied, followed by flattening the 19×19 feature map into a 361-dimensional vector.
- A log\_softmax layer produces the final log-probability distribution for each board coordinate.
- By avoiding fully connected layers and using only a 1×1 convolution, the model reduces parameter count while preserving the spatial structure of the board.

## Value Head

- A  $1 \times 1$  convolution, Batch Normalization, and SiLU activation.
- Adaptive Average Pooling reduces the spatial dimension ( $19 \times 19$ ) to  $1 \times 1$  (Global Average Pooling).
- Two fully connected layers (FC\_v1, FC\_v2) with Dropout are applied.
- A tanh activation outputs a scalar in the range  $[-1, 1]$ , representing the expected win value of the position.

This architecture enables the model to output both policy and value simultaneously, making it applicable for Go evaluation regardless of whether tree search is used.

## 1.2 Depthwise Separable Convolution (DSConv2d) and Its Effects

The network uses Depthwise Separable Convolution (DSConv) instead of standard  $3 \times 3$  convolutions to maximize computational efficiency. DSConv consists of two stages:

### Depthwise Convolution

- Applies  $3 \times 3$  filtering independently to each channel.
- Parameter count:  $\text{in\_channels} \times \text{kernel\_size} \times \text{kernel\_size}$ .
- Extracts spatial patterns (adjacent shapes, local fights, tactical structures) channel-wise.

### Pointwise Convolution ( $1 \times 1$ )

- Combines information across channels via a  $1 \times 1$  convolution.
- Parameter count:  $\text{in\_channels} \times \text{out\_channels}$ .
- Preserves spatial dimensions while integrating semantic information between channels.

Compared to standard convolutions with identical channel dimensions, DSConv significantly reduces parameter count and FLOPs.

In the Go domain—where the model repeatedly processes  $19 \times 19$  board grids—DSConv offers the following benefits:

- Substantial reduction in inference time and memory usage, even in environments with limited GPU resources (e.g., integrated GPUs).
- Ability to maintain channel size or increase network depth under the same computational budget, achieving a balance between efficiency and expressive power.

Thus, the model becomes a lightweight Go network capable of running in low-resource environments such as laptops and mobile-class hardware.

### 1.3 Role of Spatial Attention and Its Significance in the Go Domain

Each Residual Block incorporates a Spatial Attention module, which assigns learnable importance scores to different spatial locations in the feature map.

**The module operates as follows:**

- Compute the channel-wise average map (avg\_out) and max map (max\_out):
  - $\text{avg\_out} \in \mathbb{R}^{B \times 1 \times H \times W}$
  - $\text{max\_out} \in \mathbb{R}^{B \times 1 \times H \times W}$
- Concatenate the two maps to form a 2-channel feature map, then apply a  $7 \times 7$  convolution to produce a single-channel importance map.
- A sigmoid function generates a spatial attention mask in the range  $[0, 1]$ .
- The mask is multiplied with the original feature, strengthening important regions and suppressing less relevant ones.

**In Go, this mechanism helps emphasize:**

- Local fighting zones and life-and-death battles
- Boundaries of thickness or weak groups
- Cutting points or liberties under threat
- Key intersections for territorial formation

Thus, unlike standard CNNs that treat the entire board uniformly, this network includes an internal mechanism for emphasizing “critical regions” in the position. Because Go fundamentally depends on local fights and tactical hotspots, this design introduces a domain-specific structural advantage over traditional ResNet-based Go models.

### 1.4 Effects of Pre-activation Residuals, Zero-initialized BatchNorm, and SiLU on Stability and Expressiveness

The model’s Residual Blocks combine Pre-activation (ResNet v2 style), Zero-initialized BatchNorm, and SiLU (Swish) to enhance training stability and expressive capability.

#### Pre-activation Structure

- Applies Batch Normalization and SiLU activation before the convolution (“BN → SiLU → Conv”).
- Improves gradient flow and yields more stable residual connections in deep networks.

#### Zero-initialized BatchNorm

- The second BatchNorm layer in each block is initialized with scale parameter  $\gamma = 0$ .
- This forces the block to behave almost like an identity mapping at the start of training.
- Helps prevent early-stage instability or divergence, enabling residual signals to emerge gradually during learning.

SiLU (Swish) Activation

- Defined as  $\text{SiLU}(x) = x \cdot \text{sigmoid}(x)$ .
- Provides smoother gradients than ReLU and retains non-zero gradients in the negative region.
- Enhances sensitivity to subtle differences—important for evaluating fine-grained Go positions.

The combination of these techniques allows the network to train stably despite its depth while maintaining a highly expressive function space for evaluating Go positions.

2. Hardware Details (Prior Learning and Competition)

Prior Learning

The model was trained using Google Colab Pro, leveraging NVIDIA L4 and A100 GPUs for large-scale parallel computation.

Competition

Component	Specification
CPU	Intel Core Ultra 5 125H @ 3.60GHz
GPU	Intel Arc Graphics (Integrated GPU)
RAM	16GB
Storage	477GB SSD
OS	Windows 11 64-bit

3. Use of Other Programs and Unique Contributions

The foundational Go AI model, named HongGo, was provided by Professor Sounman Hong from Yonsei University.

Program Base

The basic code structure and core data-handling components draw from the HongGo model, a ResNet-based Go AI developed by Professor Sounman Hong.

Unique Contributions

Compared to traditional ResNet-based architectures used in open-source Go engines such as Leela Zero and KataGo, this model introduces several structural distinctions. These differences represent meaningful advantages in terms of computational efficiency, domain suitability, and training stability.

## **(1) Maximizing Computational Efficiency Using Depthwise Separable Convolution**

Most Go AIs rely on standard  $3 \times 3$  convolutions within Residual Blocks.

As channels and depth increase, computational and memory costs rise significantly.

By constructing all major convolution layers using DSConv, this model achieves:

- Significant reduction in parameters and FLOPs
- Ability to build deeper networks or operate under constrained environments
- Practical inference even on integrated GPUs or low-power devices

This means the model is not only suitable for high-end GPUs (L4, A100) but also usable on local, lightweight devices, enabling future deployment on mobile or embedded platforms.

## **(2) Domain-specific Spatial Attention Mechanism for Go**

While ResNet-based Go AIs typically treat all spatial locations evenly, this model leverages Spatial Attention at the end of every Residual Block.

- The mechanism identifies positions with strong cross-channel responses
- Learns spatial importance maps
- Enhances tactical hotspots, weaknesses, and strategic boundaries

This effectively transforms the network into one that "focuses more deeply on important regions and less on irrelevant areas," aligning closely with human Go intuition.

Because such systematic use of Spatial Attention is rarely seen in existing Go AIs, this design can be regarded as a domain-tailored and research-significant structural contribution.

## **(3) Improved Training Stability with Pre-activation, Zero-init BN, and SiLU**

- Pre-activation enhances gradient behavior and residual efficiency
- Zero-init BN ensures safe initial behavior by forcing early identity mapping
- SiLU improves subtle value estimation performance

Together, these techniques represent an advanced stabilization approach beyond the widely used ResNet v1 + ReLU designs. They support deeper yet more stable architectures.

#### (4) Lightweight Policy/Value Heads and Structural Simplicity

The heads are optimized as follows:

- **Policy Head:**  $1 \times 1$  Conv + BN + log\_softmax, without fully connected layers
- **Value Head:** Global Average Pooling + compact MLP + Dropout

This preserves structural simplicity, interpretability, and consistency with the design goal of a lightweight model.

#### 4. Other Appointing Points About the Program

The model is designed around three core goals: structural simplicity, extensibility, and resource efficiency.

##### Research and Analysis Utility

The modular architecture makes it highly suitable for experimentation:

- Testing changes in Residual Block structure
- Ablation studies for Spatial Attention
- DSConv vs standard convolution comparisons
- Comparing activation functions (SiLU vs ReLU)
- Extending input planes (ko status, forbidden moves, liberties, etc.)

This makes the model valuable not only as a Go engine but also as an analysis platform for Go AI research.

##### Design Philosophy

The program is based on the following principles:

##### Efficiency

- Minimize unnecessary computation through DSConv
- Reduce reliance on high-end GPUs

##### Domain Awareness

- Emphasize critical regions of the board using Spatial Attention
- Reflect the localized and tactical nature of Go

##### Stability

- Combine Pre-activation, Zero-init BN, and SiLU
- Achieve stable learning and improved value estimation in deep networks

This integrated approach moves beyond conventional ResNet expansion and represents a meaningful effort to merge Go-specific inductive biases with modern deep learning techniques.