Deep Learning Project – Game of Go

Master 2 Intelligence Artificielle, Systèmes et Données (IASD)

Mellissa HAFIS and Ilona Le Drogoff

Université Paris Dauphine PSL

Janvier 2026

Contents

1 Introduction

This report presents the work conducted as part of a Deep Learning project aimed at training a neural network to play Go on a 19×19 board. The primary challenge is a strict constraint limiting model parameters to fewer than 100,000, which severely restricts architectural complexity.

The training data consists of one million games generated by KataGo's self-play. The network processes board representations as 31 input planes (19×19) and produces two outputs: a policy head predicting the next move across 361 possible positions, and a value head estimating White's winning probability.

We explored multiple architectures including ResNet variants, MobileNet-inspired designs, and hybrid approaches, each designed to maximize performance while respecting parameter constraints. Models were evaluated through training metrics and tournament play, where networks compete using a PUCT search engine with 2 seconds per move.

This report details the architectures tested, training results, tournament performance, and the rationale behind our final model selection.

2 State of the Art

The breakthrough in game-playing neural networks for Go came with AlphaGo Zero [1], which achieved superhuman performance using only self-play reinforcement learning, without human game data or domain knowledge beyond the rules.

AlphaGo Zero employed a ResNet architecture with approximately 20 residual blocks, combining both policy and value predictions in a single network. The network takes the raw board position as input and outputs move probabilities (policy) and a position evaluation (value). This dual-head architecture became the standard for game-playing neural networks.

The training process relied entirely on self-play: the network played against itself, using Monte Carlo Tree Search (MCTS) to guide move selection during play. The MCTS was enhanced by the neural network's policy and value predictions, creating a virtuous cycle in which improved play generated higher-quality training data, which in turn further improved the network.

Several design principles introduced by AlphaGo Zero are particularly relevant to this project:

- **Residual connections**: These enable the training of deeper networks by mitigating the vanishing gradient problem, allowing the model to learn hierarchical representations ranging from local tactics to global strategy.
- **Dual-head architecture**: Separating policy and value predictions allows the network to jointly learn move selection and position evaluation, sharing representations in a common trunk while specializing in the output heads.
- **Batch normalization**: This stabilizes training and permits the use of higher learning rates, which is especially important when training deep neural networks.
- **Supervised learning from strong play**: While AlphaGo Zero relied on reinforcement learning from self-play, supervised learning from KataGo games, as adopted in this project, provides a strong baseline since target policies and values are derived from MCTS-enhanced play.

Our project differs significantly from AlphaGo Zero in scale. Whereas AlphaGo Zero employed networks with millions of parameters, our model is constrained to fewer than 100,000 parameters. This limitation necessitates specific architectural adaptations, including:

- Reducing network depth and the number of convolutional filters
- Exploring more parameter-efficient convolutional variants, such as depthwise separable convolutions
- Carefully balancing model capacity between the policy and value heads
- Investigating whether the parameter overhead of batch normalization is justified at this scale

The central challenge is to retain the core principles of deep game-playing neural networks while dramatically reducing model capacity.

3 Network architectures tested

As part of this project, we tested different neural network architectures in order to find the best model capable of playing Go while respecting the strict constraint on the number of parameters (100,000). The objective was to study the impact of different architectural choices, such as network depth, the use of residual connections, normalization, and separable convolutions, on model performance. This section presents the different models that were implemented and tested.

3.1 ResNet without Batch Normalization

3.1.1 Model Objective

The objective of this first architecture was to evaluate a residual network (ResNet-type) while keeping the model simple and lightweight, in particular by removing Batch Normalization. The idea was to leverage residual connections to facilitate learning—allowing for a deeper network without significantly degrading backpropagation—while maintaining compatibility with the project's parameter constraints.

3.1.2 Architecture Description

The network takes as input a tensor of size $19 \times 19 \times 31$ (31 feature planes). It begins with a 1×1 convolution with 32 filters and ReLU activation, which serves as an adaptation layer.

The backbone then consists of 5 residual blocks. Each block contains:

- a 3×3 convolution (32 filters, ReLU),
- a 3×3 convolution (32 filters, linear activation),
- an addition with the shortcut (residual connection),
- followed by a ReLU activation.

After the backbone, the network branches into two outputs. The **policy head** is obtained through two 1×1 convolutions, followed by a flattening operation and a softmax over 361 positions. The **value head** follows the same initial 1×1 convolution, then passes through a dense layer of 50 neurons with dropout (0.3) before a sigmoid output.

Training is performed using the Adam optimizer (1×10^{-4}), with cross-entropy loss for the policy head and mean squared error (MSE) for the value head. Policy accuracy and value mean absolute error (MAE) are used as evaluation metrics.

Model Size:

- Total parameters: 111,791 (436.68 KB)
- Trainable parameters: 111,791 (436.68 KB)
- Non-trainable parameters: 0

Note: This initial version exceeds the 100,000 parameter constraint and would require further optimization for tournament submission.

Compared to the baseline architecture provided by the instructor, this model introduces several modifications:

Residual blocks in feature extractor: The simple sequential Conv2D layers are replaced with residual blocks. This architectural choice enables more efficient feature reuse across layers and mitigates the vanishing gradient problem, allowing the network to learn deeper representations despite the parameter constraints.

Dropout in value head: A dropout layer (rate=0.3) is added before the final sigmoid activation in the value head. This regularization technique reduces overfitting by preventing the network from relying too heavily on specific neurons during training, improving generalization to unseen positions.

3.1.3 Advantages and Limitations

This architecture benefits from residual connections, which facilitate learning and allow the training of a deeper network compared to a standard CNN of similar size. The model remains relatively simple while retaining enough structure to learn meaningful patterns on the Go board.

However, the model has several limitations. The absence of Batch Normalization makes training less stable and more sensitive to hyperparameter choices. The parameter count (111,791) exceeds the project constraint, requiring architectural adjustments. Furthermore, the model capacity is constrained by the limited number of filters and residual blocks, which may reduce performance compared to more complex architectures.

3.2 ResNet with Batch Normalization

3.2.1 Model Objective

This architecture aims to improve upon the previous ResNet by incorporating Batch Normalization, which stabilizes training and enables better gradient flow. The goal was to determine whether the parameter overhead of Batch Normalization (through additional learnable parameters) would be offset by improved learning efficiency and final performance, while staying within the 100,000 parameter constraint.

3.2.2 Architecture Description

The network takes as input a tensor of size $19 \times 19 \times 31$ (31 feature planes).

The architecture begins with an initial convolution block:

- 1×1 convolution (32 filters, no bias)
- Batch Normalization
- ReLU activation

The backbone consists of 4 residual blocks with Batch Normalization. Each residual block follows the structure:

- 3×3 convolution (32 filters, no bias, same padding)
- Batch Normalization
- ReLU activation
- 3×3 convolution (32 filters, no bias, same padding)
- Batch Normalization
- Addition with shortcut connection
- ReLU activation

After the backbone, the network branches into two specialized heads:

Policy head:

- 1×1 convolution (2 filters, ReLU)
- 1×1 convolution (1 filter, L2 regularization = 0.0001)
- Flatten
- Softmax activation over 361 positions

Value head:

- 1×1 convolution (2 filters, ReLU)
- 1×1 convolution (1 filter, L2 regularization = 0.0001)

- Flatten
- Dense layer (50 neurons, ReLU, L2 regularization = 0.001)
- Dropout (rate = 0.3)
- Dense layer (1 neuron, sigmoid, L2 regularization = 0.0001)

Training is performed using the Adam optimizer with learning rate 9.9×10^{-4} (last value), categorical cross-entropy loss for policy, and MSE for value. Both heads have equal loss weights (1.0).

Model Size:

- Total parameters: 94,159 (367.81 KB)
- Trainable parameters: 93,583 (365.56 KB)
- Non-trainable parameters: 576 (2.25 KB)

This architecture successfully meets the 100,000 parameter constraint.

Key Modifications from Previous Model, compared to the ResNet without Batch Normalization:

Batch Normalization integration: BN layers are added after each convolution in the residual blocks (before activation). Convolutions use `use_bias=False` since BN includes learnable bias parameters. The 576 non-trainable parameters correspond to BN moving mean and variance statistics.

Reduced residual blocks: The number of residual blocks is reduced from 5 to 4 to accommodate BN parameters while staying under the constraint.

Enhanced policy head: An additional intermediate convolution with 2 filters is added before the final policy output, providing more expressiveness.

Checkpoint system: Training includes checkpoint saving every 20 epochs and resumption capability, enabling longer training sessions.

3.2.3 Advantages and Limitations

This architecture demonstrates several advantages. Batch Normalization stabilizes training by normalizing activations, reducing sensitivity to initialization and learning rate. The model achieved better results than the previous ResNet without BN after the same number of training epochs, validating the trade-off of fewer residual blocks for improved training stability. The parameter count (94,159) comfortably fits within the project constraint.

However, limitations emerged during extended training. We pushed this model to 760 epochs with periodic learning rate adjustments (varying between 6×10^{-4} and 9.9×10^{-4}) to maximize performance. While the model initially showed good improvement, performance eventually plateaued around epoch 440 and remained stable thereafter despite further training and hyperparameter tuning. This stagnation suggested that the model had reached its representational capacity limits, motivating exploration of alternative architectures with different efficiency trade-offs, as presented in the following sections. Detailed training metrics and performance curves will be discussed in the Results section.

Additional technical limitations include the computational overhead of BN's moving statistics during inference and the reduced depth (4 blocks vs. 5) constraining the model's ability to learn hierarchical features.

3.3 ResNet with Separable Convolutions

3.3.1 Model Objective

This architecture explores the use of depthwise separable convolutions to maximize network depth under parameter constraints. Separable convolutions decompose standard convolutions into depthwise and pointwise operations, significantly reducing parameters per layer. The goal was to determine whether a much deeper network (27 blocks vs. 4) with efficient convolutions could outperform the shallower standard ResNet.

3.3.2 Architecture Description

The network takes as input a tensor of size $19 \times 19 \times 31$ (31 feature planes).

The architecture begins with an initial convolution block:

- 1×1 convolution (32 filters, no bias)
- Batch Normalization
- ReLU activation

The backbone consists of 27 residual blocks with separable convolutions—the maximum number achievable under the 100,000 parameter constraint. Each residual block follows the structure:

- Depthwise separable 3×3 convolution (32 filters, no bias, same padding)
- Batch Normalization
- ReLU activation
- Depthwise separable 3×3 convolution (32 filters, no bias, same padding)
- Batch Normalization
- Addition with shortcut connection
- ReLU activation

The policy and value heads follow the same structure as the previous ResNet with BN model.

Training uses the Adam optimizer with learning rate 1×10^{-4}, categorical cross-entropy for policy, and MSE for value.

Model Size:

- Total parameters: 97,167 (379.56 KB)
- Trainable parameters: 93,647 (365.81 KB)
- Non-trainable parameters: 3,520 (13.75 KB)

Key Modifications from Previous Model The primary modification is replacing standard Conv2D layers with SeparableConv2D in all residual blocks. This enables dramatically increasing network depth from 4 to 27 blocks while remaining under the parameter budget. The separable convolution factorizes a standard $k \times k$ convolution into a depthwise convolution (applying a single filter per input channel) followed by a 1×1 pointwise convolution, reducing parameters by approximately a factor of $k^2/(k^2 + \text{channels})$.

3.3.3 Advantages and Limitations

The separable convolution approach enables exceptional network depth (27 blocks) while respecting parameter constraints, theoretically allowing the model to learn more complex hierarchical representations through increased receptive field and feature abstraction layers.

However, training results revealed a critical limitation: despite being 6.75 times deeper, this architecture underperformed the 4-block standard ResNet with Batch Normalization. At comparable training epochs(........), the shallower standard ResNet achieved superior validation metrics (......). This suggests that for Go position evaluation, the representational power of standard convolutions outweighs the benefits of additional depth through separable convolutions. The factorization in separable convolutions may reduce the model's ability to learn spatial feature interactions critical for board game understanding.

This finding motivated shifting our architectural exploration toward MobileNet-inspired designs that balance efficiency with feature learning capacity, as presented in the following sections.

3.4 Hybrid ResNet–MobileNet Architecture

3.4.1 Model Objective

This architecture combines the representational power of standard ResNet blocks with the parameter efficiency of MobileNetV2-style inverted residual blocks. The goal was to leverage the strengths of both approaches: using efficient MobileNet blocks for early feature extraction and standard residual blocks for final feature refinement, while incorporating attention mechanisms to focus on critical board positions.

3.4.2 Architecture Description

The network takes as input a tensor of size $19 \times 19 \times 31$ (31 feature planes).

Initial Feature Extraction:

- 3×3 convolution (24 filters, no bias)
- Batch Normalization
- ReLU activation

Hybrid Backbone: The backbone consists of three stages combining different block types:

Stage 1 - Early MobileNet blocks (2 blocks):

- Inverted residual blocks with expansion factor $t = 2$, output channels $= 24$
- Each block: 1×1 expansion $\rightarrow$ depthwise 3×3 conv $\rightarrow$ 1×1 projection

Stage 2 - Mixed architecture (4 blocks):

- Inverted residual block ($t = 3$, channels $= 32$)
- Standard residual block (32 filters) for feature refinement
- Inverted residual block ($t = 3$, channels $= 32$)
- Standard residual block (32 filters)

Attention mechanism:

- Squeeze-excitation style channel attention
- Global average pooling $\rightarrow$ Dense(8, ReLU) $\rightarrow$ Dense(32, sigmoid)
- Element-wise multiplication with feature maps

Stage 3 - Final refinement (1 block):

- Standard residual block (32 filters)
- Final 1×1 convolution to 40 channels

Policy Head:

- Separable 3×3 convolution (16 filters) + BN + ReLU
- 1×1 convolution (2 filters) + BN + ReLU
- 1×1 convolution (1 filter, L2 reg $= 0.0001$)
- Flatten $\rightarrow$ Softmax (361 outputs)

Value Head:

- 1×1 convolution (8 filters) + BN + ReLU
- Global average pooling
- Dense(32, ReLU, L2 reg = 0.001)
- Dropout(0.3)
- Dense(1, sigmoid, L2 reg = 0.0001)

Training uses Adam optimizer with initial learning rate 1×10^{-3}.

Model Size:

- Total parameters: 83,467 (326.04 KB)
- Trainable parameters: 80,855 (315.84 KB)
- Non-trainable parameters: 2,612 (10.20 KB)

3.4.3 Key Architectural Innovations

This model introduces several innovations:

Strategic block placement: MobileNet blocks handle early feature extraction efficiently, while standard residual blocks refine high-level features where representational capacity matters most.

Attention mechanism: The squeeze-excitation style attention allows the network to focus on important board regions, particularly valuable for Go where key groups and influence areas determine position evaluation.

Efficient heads: The policy head uses separable convolutions for parameter efficiency, while the value head employs global average pooling to aggregate spatial information without dense flattening.

Channel progression: Gradual channel expansion ($24 \rightarrow 32 \rightarrow 40$) balances capacity with parameter budget.

3.4.4 Advantages and Limitations

This architecture achieved the best performance among all tested models, reaching a validation policy accuracy of 0.45044 at epoch 920 and 0.45750 at epoch 1680— a significant improvement over previous architectures. The parameter count (83,467) provides comfortable margin under the constraint, and the hybrid design successfully balances efficiency with representational power.

However, extended training revealed performance stagnation. Between epochs 920 and 1720, validation accuracy oscillated narrowly between 0.45044 and 0.45750 despite continued training and learning rate adjustments. This plateau suggested the model had reached its capacity limits, with the hybrid approach potentially introducing optimization challenges due to mixing different block types with varying gradient flow characteristics.

These observations motivated exploring a pure MobileNet architecture to determine whether eliminating architectural heterogeneity and fully committing to the MobileNet paradigm could break through the performance ceiling, as discussed in the following section.

3.5 Simple MobileNet Architecture

3.5.1 Model Objective

After observing the hybrid architecture's plateau, we tested a pure MobileNet approach to evaluate whether eliminating architectural heterogeneity could improve parameter efficiency and training dynamics. This simplified design uses only inverted residual blocks throughout the backbone.

3.5.2 Architecture Description

The network takes as input a tensor of size $19 \times 19 \times 31$ (31 feature planes).

Initial Feature Extraction:

- 3×3 convolution (24 filters, no bias)
- Batch Normalization
- ReLU activation

Backbone - Three Stages of Inverted Residual Blocks: **Stage 1:** 2 inverted residual blocks ($t = 2$, output channels = 24)
Stage 2: 2 inverted residual blocks ($t = 3$, output channels = 32)
Attention mechanism: Squeeze-excitation style channel attention (32 channels)
Stage 3: 2 inverted residual blocks ($t = 3$, output channels = 32)
Final refinement: 1×1 convolution to 40 channels + BN + ReLU

Policy Head:

- Separable 3×3 convolution (16 filters) + BN + ReLU
- 1×1 convolution (2 filters) + BN + ReLU
- 1×1 convolution (1 filter, L2 reg = 0.0001)
- Flatten $\rightarrow$ Softmax (361 outputs)

Value Head:

- 1×1 convolution (8 filters) + BN + ReLU
- Global average pooling
- Dense(32, ReLU, L2 reg = 0.001) $\rightarrow$ Dropout(0.3) $\rightarrow$ Dense(1, sigmoid)

Training uses Adam optimizer with learning rate 1×10^{-3}.

Model Size:

- Total parameters: 46,091 (180.04 KB)
- Trainable parameters: 43,735 (170.84 KB)
- Non-trainable parameters: 2,356 (9.20 KB)

3.5.3 Key Design Principles

This architecture removes all standard residual blocks, creating a homogeneous design using only MobileNetV2 inverted residual blocks. The attention mechanism remains for spatial focus. With only 46,091 parameters—approximately half the hybrid model's count—this design maximizes parameter efficiency.

3.5.4 Performance Analysis and Insights

This simplified architecture demonstrated remarkable parameter efficiency:

- Achieved 0.43290 validation policy accuracy at epoch 480
- The hybrid model (83,467 params) reached 0.43104 at epoch 320
- All previous ResNet variants never exceeded 0.43 validation accuracy

Most significantly, this model matched the hybrid architecture's performance using only 55% of its parameters. The homogeneous MobileNet design proved more efficient than mixing architectural paradigms, likely due to:

- Consistent gradient flow through uniform block types
- Better optimization dynamics without conflicting inductive biases
- Efficient parameter utilization through inverted residuals

These promising results—achieving competitive performance with substantial parameter headroom—motivated increasing the parameter budget to fully utilize the 100,000 parameter constraint. The next section presents an enhanced MobileNet architecture designed to maximize capacity while maintaining the architectural principles that proved successful here.

3.6 Winner - Enhanced MobileNet Architecture

3.6.1 Model Objective

Building on the simple MobileNet's success, this architecture scales up to utilize the full 100,000 parameter budget. The goal was to increase model depth and capacity through additional inverted residual blocks and multiple attention mechanisms while maintaining the efficient MobileNet paradigm.

3.6.2 Architecture Description

The network takes as input a tensor of size $19 \times 19 \times 31$ (31 feature planes).

Initial Feature Extraction:

- 3×3 convolution (28 filters, no bias) + BN + ReLU

Backbone - Three Stages with Multiple Attention: **Stage 1:** 3 inverted residual blocks ($t = 2$, channels = 28)

- Attention mechanism (28 channels)

Stage 2: 3 inverted residual blocks ($t = 3$, channels = 36)

- Attention mechanism (36 channels)

Stage 3: 3 inverted residual blocks ($t = 3$, channels = 40)

- Attention mechanism (40 channels)

Final refinement: 1×1 convolution to 44 channels + BN + ReLU

Enhanced Policy Head:

- Separable 3×3 convolution (20 filters) + BN + ReLU
- 1×1 convolution (3 filters) + BN + ReLU
- 1×1 convolution (1 filter, L2 reg = 0.0001)
- Flatten $\rightarrow$ Softmax (361 outputs)

Enhanced Value Head:

- 1×1 convolution (12 filters) + BN + ReLU
- Global average pooling
- Dense(48, ReLU, L2 reg = 0.001) $\rightarrow$ Dropout(0.3)
- Dense(24, ReLU, L2 reg = 0.001) $\rightarrow$ Dropout(0.3)
- Dense(1, sigmoid, L2 reg = 0.0001)

Training uses Adam optimizer with learning rate 1×10^{-3}.

Model Size:

- Total parameters: 88,526 (345.80 KB)
- Trainable parameters: 84,424 (329.78 KB)
- Non-trainable parameters: 4,102 (16.02 KB)

3.6.3 Key Enhancements

Compared to the simple MobileNet:

Increased depth: 9 inverted residual blocks (vs. 6), organized in three stages of 3 blocks each.

Multiple attention mechanisms: Three attention blocks placed after each stage, allowing the network to refine feature focus at different hierarchical levels.

Progressive channel expansion: More granular progression ($28 \rightarrow 36 \rightarrow 40 \rightarrow 44$) provides better capacity distribution.

Enhanced heads: The policy head uses more filters (20 vs. 16), and the value head employs a deeper two-layer dense architecture ($48 \rightarrow 24 \rightarrow 1$) for better position evaluation.

3.6.4 Performance and Results

This architecture achieved the best performance across all tested models:

- Reached 0.46132 validation policy accuracy at epoch 960
- Achieved maximum accuracy of 0.47776 at epoch 4500
- Substantial improvement over simple MobileNet (0.43290) despite using only 88,526 parameters

The enhanced architecture demonstrated that scaling MobileNet principles with strategic capacity allocation yields consistent improvements. The multiple attention mechanisms proved particularly effective, allowing the network to focus on critical board regions at different feature abstraction levels. Extended training to 4500 epochs showed continued learning without severe overfitting, validating the regularization strategy.

This model represents the optimal balance found between parameter efficiency and representational capacity under the 100,000 parameter constraint, making it the selected architecture for tournament submission.

3.7 ConvNeXt

3.7.1 Model Objective

To conclude, the last model tested in this project is a ConvNeXt-inspired architecture designed for the Game of Go under a strict constraint of fewer than 100,000 parameters. ConvNeXt introduces modern convolutional design choices such as depthwise convolutions, Layer Normalization, and simplified residual blocks.

3.7.2 Architecture Description

The network takes as input a tensor of size $19 \times 19 \times 31$ (31 feature planes).

Initial Feature Extraction:

- 3×3 convolution with 56 filters (L2 regularization = 0.0001)
- Layer Normalization

ConvNeXt Backbone: The backbone consists of 5 identical ConvNeXt blocks with 56 channels. Each block includes:

- Depthwise 5×5 convolution
- Layer Normalization
- 1×1 pointwise convolution (expansion factor = 2)
- GELU activation
- 1×1 pointwise convolution (projection)
- Trainable LayerScale parameter
- Dropout (rate = 0.05)
- Residual connection

Policy Head:

- 1×1 convolution (32 filters, GELU)
- 1×1 convolution (1 filter, L2 regularization = 0.0001)
- Flatten
- Softmax activation over 361 positions

Value Head:

- 1×1 convolution (56 filters, GELU)
- Global average pooling
- Dense layer (64 neurons, GELU, L2 regularization = 0.0001)
- Dense output layer (1 neuron, sigmoid)

Training Setup:

- Optimizer: AdamW (learning rate 2×10^{-3}, weight decay 10^{-4})
- Policy loss: categorical cross-entropy with label smoothing (0.05)
- Value loss: mean squared error
- Loss weights: policy = 1.0, value = 0.5

Model Size:

- Total parameters: 95,954 (under the 100,000 parameter constraint)

3.7.3 Key Design Principles

This ConvNeXt architecture is based on the following design principles:

- **Depthwise convolutions for parameter efficiency:** Depthwise 5×5 convolutions are used to capture spatial patterns on the Go board while significantly reducing the number of parameters compared to standard convolutions.
- **Layer Normalization instead of Batch Normalization:** Layer Normalization is preferred to Batch Normalization to ensure stable training with small batch sizes and to avoid the additional parameter overhead and batch-dependent statistics.
- **Residual connections with LayerScale:** Residual connections help preserve gradient flow across multiple layers, while trainable LayerScale parameters stabilize training by controlling the contribution of each block during early optimization.
- **Balanced allocation between backbone and heads:** Model capacity is primarily allocated to the shared backbone, while the policy and value heads remain lightweight to respect the strict parameter constraint.
- **Parameter-efficient value estimation:** Global average pooling is used in the value head instead of flattening to reduce parameter count while maintaining access to global board information.

3.7.4 Advantages and Limitations

This ConvNeXt-based model shows relatively stable training thanks to the use of Layer Normalization and residual connections. The architecture remains fully compliant with the 100,000 parameter constraint while using modern convolutional design choices such as depthwise convolutions.

Despite these advantages, ConvNeXt is not particularly well suited for the Game of Go under such a strict parameter constraint. The architecture was originally designed for much larger models, and its benefits are limited when scaled down below 100,000 parameters. In our experiments, the best validation performance was reached at epoch 265, with a policy accuracy of 0.3410, which remains significantly lower than the results obtained with MobileNet-based architectures. The model also showed early performance saturation, suggesting insufficient capacity to capture complex Go patterns in this constrained setting.

4 Results and Comparison

This section presents a comprehensive analysis of all tested architectures, comparing their performance across training epochs and evaluating their effectiveness under the 100,000 parameter constraint.

4.1 Overall Performance Summary

Table 1 and the fig. 1 summarizes the performance of all six architectures tested during this project. The models are ranked by their best validation policy accuracy achieved during training.

Table 1: Comparison of Model Performance

Model	Parameters	Best Val Acc	Epoch	Final Val Acc	Training Epochs
Enhanced MobileNet	88,526	**0.47776**	4500	0.47462	4540
Hybrid ResNet-MobileNet	83,467	0.45750	1680	0.45358	1720
Simple MobileNet	46,091	0.43334	540	0.43218	560
ResNet (BN)	94,159	0.40692	720	0.40618	760
ConvNeXt	95,954	0.34103	265	0.34103	265
ResNet (Separable)	97,167	0.33394	60	0.33394	60
ResNet (No BN)	111,791*	0.33364	40	0.33364	40

*Exceeds 100K parameter constraint

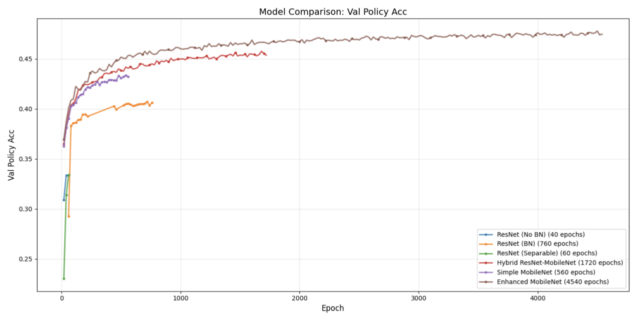

Figure 1: Validation policy accuracy of all the models.

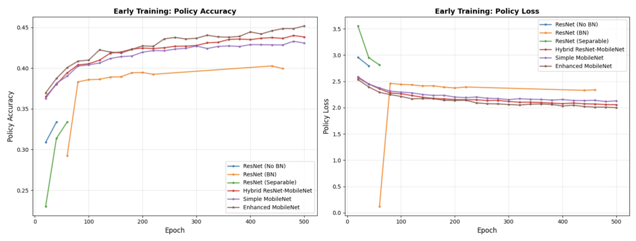

Figure 2: Early training results

The Enhanced MobileNet architecture achieved the highest validation policy accuracy of 47.776%, representing a 17.4% relative improvement over the best ResNet variant. This demonstrates the superiority of the MobileNet paradigm for parameter-constrained Go networks.

4.2 Training Dynamics and Convergence

4.2.1 Early Training Performance

Figure 2 shows the first 500 epochs of training for models that reached this milestone. The MobileNet-based architectures demonstrated faster initial learning:

- **Enhanced MobileNet**: Reached 0.45170 accuracy at epoch 500
- **Hybrid ResNet-MobileNet**: Reached 0.43826 accuracy at epoch 500
- **Simple MobileNet**: Reached 0.43078 accuracy at epoch 500

In contrast, ResNet with Batch Normalization plateaued below 0.41 accuracy even after 760 epochs, demonstrating that architectural efficiency matters more than extended training duration under strict parameter constraints.

4.2.2 Long-Term Training Behavior

Analysis of convergence patterns revealed distinct behaviors across architectures:

ResNet variants showed limited capacity for improvement:

- ResNet (BN) improved 0.11366 over 760 epochs but plateaued around 0.40-0.41 accuracy
- ResNet with separable convolutions failed to learn effectively, reaching only 0.33394
- The absence of batch normalization severely hampered learning (0.33364 accuracy)

Hybrid architecture demonstrated strong early performance but eventual stagnation, fig. 3:

- Achieved 0.45750 peak accuracy at epoch 1680

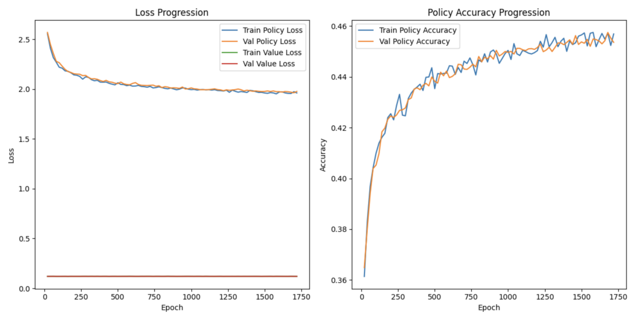

Figure 3: Validation and training curves of Hybrid architecture

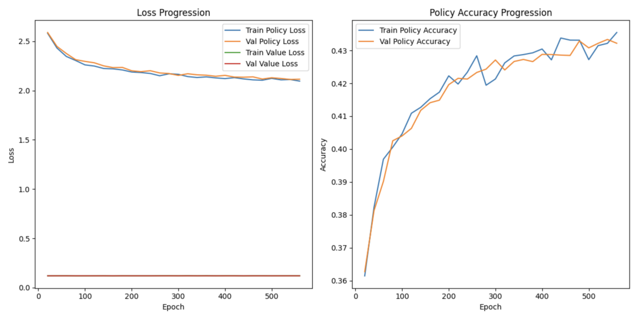

Figure 4: Validation and training curves of the simple MobileNet architecture

- Improvement over final 1000 epochs: only 0.00978
- Mixed architectural paradigms may introduce optimization challenges

Simple MobileNet showed efficient learning despite minimal parameters, fig. 4:

- Reached 0.43334 with only 46,091 parameters
- Demonstrated consistent improvement over 560 epochs (0.06946 gain)
- Validated the MobileNet approach's parameter efficiency

Enhanced MobileNet exhibited sustained learning over extended training, fig. 5:

- Continued improving through 4540 epochs, peaking at 0.47776
- Plateau detection around epoch 2360, but occasional improvements continued
- Final 1000 epochs still showed 0.00382 improvement
- Successfully utilized 88,526 parameters without severe overfitting

4.3 Architectural Insights

4.3.1 Parameter Efficiency vs. Representational Power

The results reveal a critical trade-off between parameter efficiency and representational capacity:

- **Standard convolutions** (ResNet BN): 94,159 parameters → 0.407 accuracy
- **Separable convolutions alone** (ResNet Separable): 97,167 parameters → 0.334 accuracy
- **Pure MobileNet** (Simple): 46,091 parameters → 0.433 accuracy

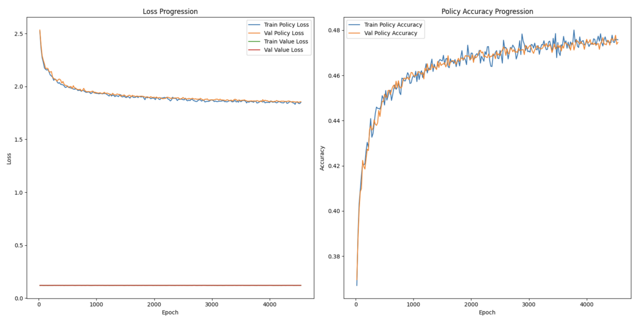

Figure 5: Validation and training curves of the Enhanced MobileNet architecture

- **Scaled MobileNet** (Enhanced): 88,526 parameters → 0.478 accuracy

Separable convolutions in isolation proved insufficient, but when combined with MobileNet's inverted residual structure and multiple attention mechanisms, they enabled superior performance with fewer parameters.

4.3.2 Impact of Depth and Attention

Comparing the Simple and Enhanced MobileNet architectures isolates the impact of increased depth and attention:

- Scaling from 6 to 9 inverted residual blocks: +4.4% absolute accuracy
- Adding two additional attention mechanisms: improved feature focus
- Enhanced heads (deeper policy/value networks): better move selection and evaluation
- Parameter increase of 92% yielded 10.3% relative accuracy improvement

The diminishing returns suggest that 88,526 parameters approaches the effective capacity for this dataset and input representation under the current training regime.

4.4 Final Model Selection

The Enhanced MobileNet architecture was selected for tournament submission based on:

1. **Superior performance**: 47.776% validation accuracy, 7.5% absolute improvement over the second-best model
2. **Compliance with constraints**: 88,526 parameters (11.5% under the 100K limit)
3. **Robust training**: Continued learning over 4500+ epochs without catastrophic overfitting
4. **Architectural coherence**: Homogeneous MobileNet design avoids mixing conflicting paradigms

This model represents the optimal balance between parameter efficiency, representational capacity, and training stability achieved in our architectural exploration.

5 Conclusion

This project explored the challenge of training a neural network for Go under severe parameter constraints (under 100,000 parameters), testing six different architectures ranging from standard ResNet to MobileNet-inspired designs.

Our systematic exploration revealed that architectural paradigm matters more than network depth when parameters are limited. While ResNet with Batch Normalization achieved only 40.7% validation accuracy despite 94,159 parameters, the Enhanced MobileNet architecture reached 47.8% accuracy with 88,526 parameters—a 17.4% relative improvement. The key insight is that inverted residual blocks, combined with strategic attention mechanisms and progressive channel expansion, provide superior parameter efficiency for board game position evaluation.

The project also demonstrated that pure architectural approaches outperform hybrid designs. The Hybrid ResNet-MobileNet model, despite initial promise (45.8% accuracy), stagnated earlier than the pure Enhanced MobileNet, suggesting that mixing standard and depthwise separable convolutions introduces optimization challenges that limit long-term performance.

Extended training proved valuable for MobileNet architectures. The Enhanced MobileNet continued improving through 4500+ epochs, reaching its peak at epoch 4500, while ResNet variants plateaued much earlier. This indicates that efficient architectures can better leverage additional training data without overfitting under parameter constraints.

The final model—Enhanced MobileNet with 88,526 parameters achieving 47.8% policy accuracy—demonstrates that careful architectural design can effectively compensate for limited model capacity. This work validates the MobileNet paradigm for resource-constrained game-playing applications and provides insights for future neural network design under strict parameter budgets.

Future work could explore additional architectural innovations such as squeeze-and-excitation modules, mixed precision training for larger effective capacity, or alternative training strategies like knowledge distillation from larger models. The consistent improvement observed suggests that further gains may be possible with continued architectural refinement within the parameter constraint.

References

[1] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017. https://doi.org/10.1038/nature24270