AlphaGateau for Breakthrough: A piece of cake?

Athmane Benarous
McGill University

Farooq Khan
McGill University

Andrew Saffar
McGill University

Abstract

We study whether a board-size-invariant graph encoder improves self-play learning and cross-board transfer in Breakthrough. Using an AlphaZero-style training loop, we compare a graph neural network (GNN) with a convolutional neural network (CNN) on 8×8 play and in 5×5→8×8 transfer. Under our training budget, the CNN reaches a 100% greedy-evaluation win rate by iteration 40, while the GNN plateaus near 42%. Pretraining on 5×5 improves early 8×8 performance from 17% to 67% at iteration 5, but does not remove the CNN's long-run advantage. In a direct 12-game match, the CNN defeats the best GNN 7 games to 5.

1 Introduction

Breakthrough is a two-player abstract strategy game played on an 8×8 board in which each side controls only pawns. Despite its simple rules, the game is strategically rich and has a large state space of approximately 1.5×10^{16} [2]. Figure 1 illustrates the starting position and the legal forward and diagonal-capture moves that define the game.

Breakthrough is a useful testbed for representation learning because strong play depends on local, directional structure and the same move rules apply across board sizes. We study whether an AlphaGateau-style graph neural network (GNN) can outperform a simpler convolutional neural network (CNN) on 8×8 Breakthrough, and whether its board-size-invariant representation makes 5×5 → 8×8 transfer meaningful. Under our training budget, the CNN is stronger from scratch, while 5×5 pretraining gives the GNN a clear early transfer benefit without closing the full gap.

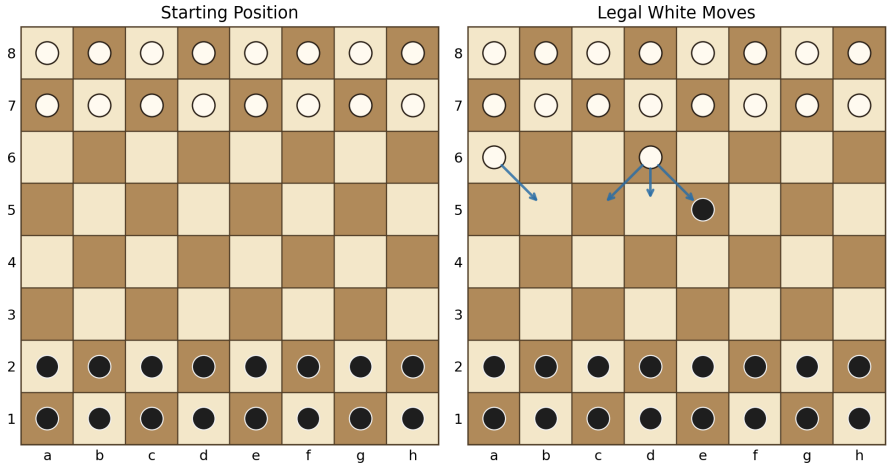


Figure 1: Breakthrough rules overview. *Left*: the standard starting position on an 8×8 board. *Right*: example legal moves for White. A pawn may move one square straight forward into an empty square, or one square diagonally forward only when capturing an opposing pawn.

2 Background

AlphaZero-style agents combine self-play reinforcement learning with Monte Carlo Tree Search (MCTS) [5]. A neural network predicts a policy over actions and a scalar value estimate; MCTS then refines action selection during play, and the search distribution becomes a policy target for training. We use Gumbel MCTS,[1] which selects actions by differentiably perturbing the policy logits with Gumbel noise [1].

AlphaGateau [3] replaces the convolutional encoder in AlphaZero with a graph neural network, aiming to better represent the relational structure of board games. It uses graph attention [6] with residual message passing layers.

Prior work on Breakthrough has focused primarily on exact game solving rather than self-play learning [4, 2]. We defer those details to Appendix A because our focus is representation learning and transfer rather than solving methodology. Our setting is different: we do not attempt to solve Breakthrough exactly, but instead study whether learned policy/value models can play effectively under self-play and transfer across board sizes.

3 Methodology

Environment and action encoding. We implemented a custom Breakthrough environment in JAX. States include the board array, an observation tensor, a legal-action mask, the side to move, rewards, and terminal game state. Actions use a fixed encoding $\texttt{action} = \texttt{from_square} \times 3 + \texttt{move_type}$, where $\texttt{move_type} \in \{\texttt{forward}, \texttt{diag_left}, \texttt{diag_right}\}$. We canonicalize the board so the player to move always advances upward, keeping both the tensor and graph interfaces board-size agnostic. By canonicalizing the perspective, the GNN's filters don't need to care about the absolute coordinates or the total size of the board. They only need to learn the local rule.

Model representations. The CNN baseline uses a three-channel tensor (self pieces, opponent pieces, empty squares) fed into a residual convolutional network. The GNN constructs a graph with one node per square and directed move edges for both players; see Appendix Figure 4 for a visual example of the canonical encoding.[2] Node features encode occupancy, normalized coordinates, and edge-of-board indicators; edge features encode move direction, owner, legality, capture structure, and goal-rank information. This graph construction preserves the same local feature schema across board sizes, which is the basis for our transfer experiments.

Learning architectures and training procedure. For the AlphaZero-style agents, both the CNN and GNN use hidden size 64 and four residual layers. The GNN adds attention pooling over node embeddings to produce a global representation. Self-play training uses 32 MCTS simulations per move, a 50k-sample replay buffer, and 32-sample minibatches. Unless otherwise noted, the learning rate is 10^{-3}.

Experiment configurations. We ran three main experiment groups. First, in the 8×8 scratch comparison, the CNN and GNN were trained from scratch for 40 iterations with 64 self-play games per iteration and a 192-ply move cap. We used 40 iterations as the standard budget because of our single-GPU compute constraints. Second, in the transfer setup, we pretrained the GNN on 5×5 for 40 iterations using 128 self-play games per iteration, then fine-tuned it on 8×8 for 30 iterations.[3] Third, we repeated selected GNN runs with cosine decay and explored a progressive $5\times5 \rightarrow 6\times6 \rightarrow 8\times8$ transfer ablation. We also evaluated a value-only TD(λ) baseline, deferred to Appendix B.

Evaluation protocol. Every five iterations we play 12 games against a greedy evaluator and record the win rate. For the policy-based agents, the greedy evaluator always picks the highest-policy action. After training, we run a 12-game head-to-head match between the final GNN and CNN

[1] using the `mctx` library

[2] This is in line with the AlphaGateau paper.

[3] We used 30 fine-tuning iterations to measure early 8×8 transfer gains rather than fully optimize the transferred model.

Run	Iters	Peak win rate	Final win rate
GNN 8×8 scratch	40	0.67 @ iter 20	0.42 @ iter 40
CNN 8×8 scratch	40	1.00 @ iter 40	1.00 @ iter 40
GNN 5×5 pretrain	40	1.00 @ iter 20	1.00 @ iter 40
GNN 8×8 fine-tune	30	0.75 @ iter 15	0.67 @ iter 30
GNN 5×5 pretrain (cosine decay)	40	1.00 @ iter 20	0.83 @ iter 40
GNN 8×8 scratch (cosine decay)	40	0.58 @ iter 40	0.58 @ iter 40
GNN 8×8 fine-tune (cosine decay)	30	0.83 @ iter 15	0.83 @ iter 30
GNN 6×6 fine-tune (cosine, progressive)	30	1.00 @ iter 20	0.92 @ iter 30
GNN 8×8 fine-tune (cosine, progressive)	30	0.92 @ iter 5	0.58 @ iter 30

Table 1: Win rates against the greedy evaluator over 12 games. Cosine and progressive-transfer runs are included because they change the transfer outcome.

Final GNN variant	Wins	Losses	Win rate
Scratch GNN	2	10	0.17
Scratch GNN (cosine decay)	1	11	0.08
Direct 5×5→8×8 transfer (cosine)	4	8	0.33
Progressive 5×5→6×6→8×8 transfer (cosine)	5	7	0.42

Table 2: Direct 12-game matches against the final CNN 8×8 scratch checkpoint. Despite the small sample, they show how much each GNN variant narrows the final gap.

scratch checkpoints. For the progressive transfer experiment, we evaluate each stage only on its current board size; we do not re-evaluate the final 8×8 checkpoint on earlier boards, so these results reflect stagewise transfer rather than catastrophic forgetting. Because of compute constraints, this evaluation sample size is small. Consequently, exact win rate percentages and specific peaks are subject to high variance and should be interpreted as general performance trends rather than highly precise measurements.

4 Results

Table 1 summarises the main training runs, Table 2 reports the direct 12-game matches, and Figure 2 shows the 8×8 scratch comparison.

Scratch comparison. The CNN and GNN diverge sharply after iteration 15. By iteration 40, the CNN reaches 1.00 greedy-evaluation win rate while the scratch GNN finishes at 0.42 (Table 1), and the CNN wins the final head-to-head 10 games to 2 (Table 2). Figure 2 shows the same pattern: the CNN improves steadily while the GNN peaks earlier and plateaus.

Transfer learning. The 5×5 pretrained GNN converges cleanly, reaching 1.00 win rate by iteration 20 and maintaining it through iteration 40. Applied zero-shot to the 8×8 board, it achieves only 17% win rate (2/12 games), showing that the 5×5 policy does not transfer directly to the larger board.

Fine-tuning from the 5×5 checkpoint gives a substantial early-stage advantage. At iteration 5, the GNN pretrained on 5×5 and then fine-tuned directly on 8×8 with cosine decay wins 75% of games versus a greedy evaluator, compared to 17% for the fixed-LR GNN trained from scratch at the same iteration.[4] This shows meaningful cross-board transfer. By iteration 30, this direct 5×5 → 8×8 fine-tuning run reaches 83%, while the fixed-LR scratch GNN at the same iteration is at 58%, a persistent advantage. The cosine run is the strongest transfer run gest transfer run in Table 1 by final greedy-evaluator win rate, so we use it as the main transfer result. The final transfer checkpoint still trails the final CNN checkpoint overall, but in direct final-checkpoint play it narrows the gap from 2 wins in 12 games for the scratch GNN to 4 wins in 12 games against the same CNN (Table 2). A progressive 5×5 → 6×6 → 8×8 pipeline does not improve the final 8×8 greedy-evaluator result, so we treat it as a secondary ablation and defer its full curves to Appendix C.

[4] more than a 4× improvement

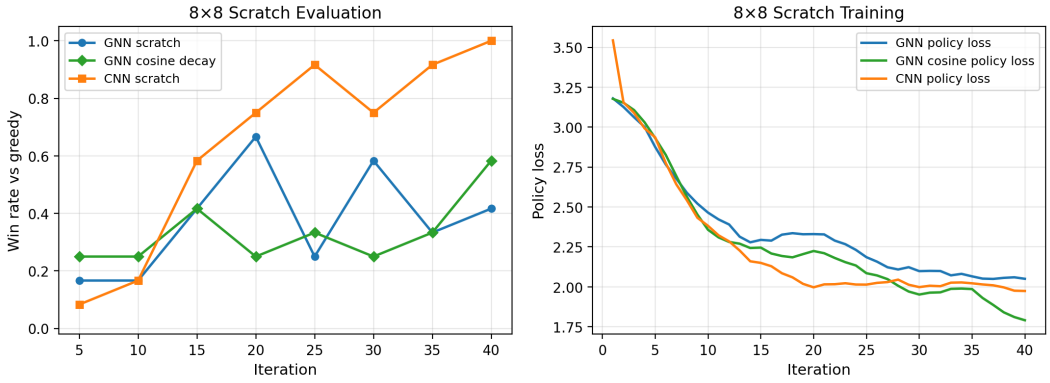


Figure 2: 8×8 scratch comparison. *Left*: greedy-evaluation win rate over training iterations; the CNN (orange) overtakes the GNN (blue) at iteration 15 and reaches 1.00 by iteration 40 while the GNN oscillates and plateaus. *Right*: policy loss curves confirming the GNN's convergence failure.

5 Discussion

Why the CNN outperforms the GNN. The CNN's advantage on 8×8 Breakthrough is a meaningful negative result. On a regular grid, convolutions share weights across spatial positions and efficiently capture the local, directional patterns that dominate Breakthrough tactics. The GNN adds extra optimization burden through attention pooling and edge-feature processing, which appears harder to train under the fixed 40-iteration budget. Figure 2 is consistent with this interpretation.

Transfer benefit and limits. Transfer from 5×5 pretraining produces a real, quantifiable early-training benefit: the fine-tuned model starts 4× higher than the scratch GNN at iteration 5. This confirms our second hypothesis: the GNN's board-size-invariant representation supports meaningful knowledge transfer. However, the benefit is largest early in fine-tuning and shrinks as both models train longer, suggesting that pretraining provides a stronger initialisation rather than a compounding structural advantage. Using cosine decay improves the final transferred checkpoint to 0.83 and reduces the late-stage drop seen with a fixed learning rate.

6 Conclusion and Future Work

We adapted AlphaGateau to Breakthrough and found that, under the current 40-iteration budget, the CNN remains the strongest model on 8×8 from scratch. GNN transfer from 5×5 meaningfully improves early 8×8 learning and narrows the final head-to-head gap to the CNN. Our main transfer result is the GNN pretrained on 5×5 and then fine-tuned directly on 8×8 with cosine decay, because it achieves the best final greedy-evaluator score among the transfer runs. The most useful next steps are to extend the fine-tuning budget, increase the evaluation game count, and run GNN ablations that simplify attention pooling or edge features.

7 Contributions and Acknowledgements

Contributions. *Athmane Benarous* focused on the JAX environment, training pipeline, Slurm orchestration, code pipeline setup, CNN model training, report drafting, and head-to-head comparisons. *Farooq Khan* focused on the transfer-learning baseline, head-to-head comparisons, presentation slides, video recording, background research, and report writing. *Andrew Saffar* focused on cosine decay, transfer learning, the TD(λ) baseline, presentation slides, video recording, figure creation, and report drafting.

Acknowledgements. We thank the AlphaGateau project for the architectural reference point and build on JAX, Flax, Optax, Jraph, MCTX, and the McGill department GPU infrastructure on `mimi.cs.mcgill.ca`.

References

[1] Ivo Danihelka, Arthur Guez, Thomas Hubert, and David Silver. Policy improvement by planning with Gumbel. In *International Conference on Learning Representations (ICLR)*, 2022.

[2] Kristinn Vikar Jonsson, Dagur Benjaminsson, and Steinthor Bjarni Steinarsson. Solving breakthrough for the 6x6 board. Bachelor's thesis, Reykjavik University, Reykjavik, Iceland, 2025. Available at Skemman: `https://skemman.is/bitstream/1946/50478/1/Solving-Breakthrough-for-the-6x6-board.pdf`.

[3] A. Kulenkampff. Enhancing chess reinforcement learning with graph representation (AlphaGateau), 2024. Preprint/Technical Report.

[4] Abdallah Saffidine, Nicolas Jouandeau, and Tristan Cazenave. Solving breakthrough with race patterns and job-level proof number search. In *Advances in Computer Games*, pages 196–207, 2011.

[5] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[6] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

A Additional Background

Prior work on Breakthrough has focused primarily on exact game solving rather than self-play learning. Early work combined race patterns with job-level proof-number search to solve the 6×5 variant as a second-player win [4]. More recently, the 6×6 game was weakly solved as a first-player win using a hybrid of endgame tablebases, race patterns, proof-number-based search, and a neural-network-guided job scheduler [2]. These results show that, despite simple rules, Breakthrough is strategically rich enough to support substantial search and solving research.

B TD(λ) Baseline

For TD(λ), we use $\lambda = 0.7$ and ϵ-greedy self-play; the 5×5 run uses 128 episodes per iteration and a 96-ply move cap, while the 8×8 run uses 64 episodes per iteration and a 256-ply move cap, with both running for 40 iterations at learning rate 10^{-3}.

Evaluation protocol. Since the TD baseline has no policy head, it is evaluated using greedy value-based play, selecting the action whose successor state has the highest predicted value rather than using MCTS.

TD(λ) baseline Results. TD(λ) remains much weaker than the MCTS-based agents against the greedy evaluator. On the 5×5 board it reaches a peak win rate of 1.00 at iteration 10 before falling back to 0.83 by iteration 40, and on the 8×8 board it only reaches 0.58 by iteration 40. This profile is consistent with a value learner that can exploit some short-horizon patterns on the smaller board but does not produce a policy strong enough to compete reliably on the full game.

TD(λ) baseline Discussion. Even when TD(λ) wins some evaluation games, it still trails the search-based models by a wide margin. We attribute this to the evaluator's explicit lookahead: TD(λ) must encode that lookahead implicitly in its value function, while the AlphaZero-style agents obtain it directly through MCTS. Under the current compute budget, the results therefore support the need for search-guided policy learning rather than value-only self-play.

C Additional Training Curves

Progressive transfer curves. Figure 3 shows the full 5×5 → 6×6 → 8×8 pipeline. Transfer to the intermediate 6×6 task is strong, but the benefit does not carry cleanly to the final 8×8 stage, which finishes below the GNN pretrained on 5×5 and then fine-tuned directly on 8×8 with cosine decay. Because we do not re-evaluate the final checkpoint on earlier board sizes, this figure should be read as stagewise transfer evidence rather than a direct measurement of catastrophic forgetting.

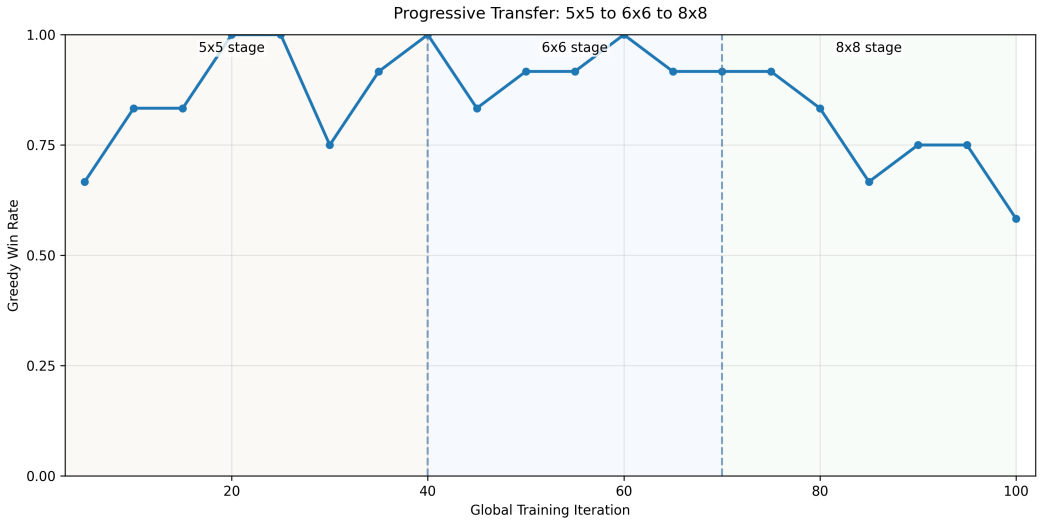


Figure 3: Progressive transfer pipeline across board sizes. Vertical dashed lines mark stage boundaries and shaded bands indicate the active board size. The model adapts well to 6×6 but does not retain that advantage through the final 8×8 fine-tuning stage. Each win-rate point is measured over 12 greedy-evaluation games.

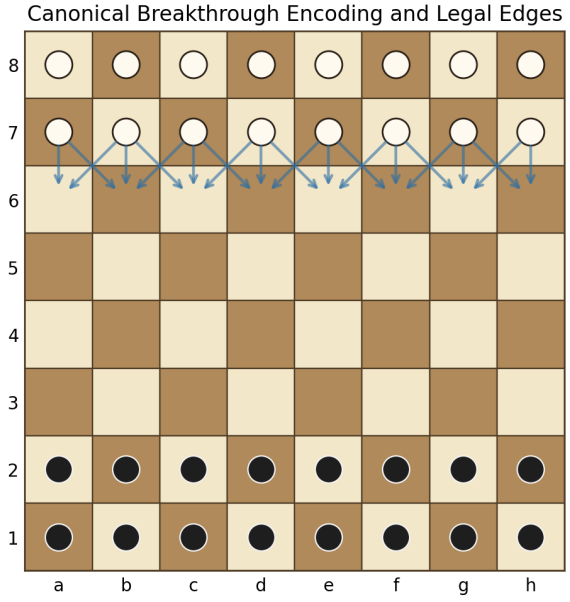


Figure 4: Canonical Breakthrough encoding showing legal directed edges for the current player at game start. Each square is a graph node; directed arrows indicate legal forward and diagonal captures.