Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A distilled value head breaks search, but search remains an effective teacher, not a move selector: a case study in 19x19 Go

Taiki Kojima

koji3744@hotmail.com

Kojima Family Clinic https://orcid.org/0000-0003-2069-6079

Research Article

Keywords: Monte Carlo tree search, knowledge distillation, value function, AlphaZero, reinforcement learning from human feedback, reward model overoptimisation, computer Go

Posted Date: August 11th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10631240/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Abstract

Background Knowledge distillation is expected to produce a policy that becomes even stronger when combined with Monte Carlo Tree Search (MCTS). We report a case in which the opposite occurred.

Methods We distilled a 19x19 Go network (192 channels, 18 residual blocks, 11.98M parameters) from KataGo. All decisions were made from head-to-head games (272 games from the empty board, adjudicated by

KataGo at 100 visits, reported with 95% Wilson confidence intervals). Every decision criterion was recorded in a pre-registration document before the corresponding experiment was run.

Results MCTS at 400 simulations scored 0.317 against the same network playing policy argmax without search. The search machinery was intact: replacing only the leaf evaluation with KataGo raised the score to

0.983 (p = 1.9x10^-14). The defect was localised to the value head, which was well calibrated in absolute terms (mean residual -0.0091) but could not rank sibling positions (Spearman +0.005, against +0.146 for the policy head). Rebuilding the value head raised sibling ranking accuracy from 0.54 to 0.64 but did not make search useful (difference from policy argmax +0.000, n = 60). Two generations of self-play training in which the policy was held in place by a KL anchor produced no distinguishable improvement (generation 2 vs generation 1: 0.526, 95% CI [0.466, 0.584]). Changing a single component - training the policy by cross-entropy to the MCTS visit-count distribution - produced 0.824 [0.774, 0.864] against the preceding generation and

0.941 [0.907, 0.963] against the distillation baseline (z = +3.04, p = 0.0024), using less than half the self-play data.

Conclusions The inference-time performance of an improvement operator and the teaching value of its output are distinct properties. MCTS selected a different move from the policy in 73% of positions without winning more games, yet its visit distribution was a substantially better training target than the policy itself. We note the same structural failure - a learned evaluator that remains calibrated while losing ranking fidelity, and an improvement operator that consequently runs backwards - in reinforcement learning for large language models.

1. Introduction

AlphaZero-style systems couple a policy network and a value network to a tree search [1]. The search is used twice: at inference time to select moves, and at training time to generate targets for the policy. These two roles are usually not distinguished, because in a healthy system the search is better than the policy in both senses at once.

This report documents a system in which the two roles came apart. The network studied here is the product of a prior line of work in which a small 19x19 network was trained by distillation from KataGo [2] rather than by self-play, because self-play stopped improving at this parameter scale [3]. Adding search to the resulting

network made it weaker. After localising the defect to the value head and rebuilding it, search still did not help. What eventually worked was to abandon search as a move selector and retain it only as a source of policy targets.

The contribution of this work is twofold. First, we show that search no longer improves play. Second, we show that the same search remains an excellent teacher for policy learning. The central question of this paper is therefore simple. Can search remain useful as a teacher after it has ceased to improve play? The following sections answer this question through a series of controlled experiments.

2. Methods

2.1 Network and distillation baseline

The network (internally "PVSNetVBO") has 192 channels and 18 residual blocks, 11.98M parameters, with policy, value, score and ownership heads. It was distilled from KataGo using the same pipeline as our earlier work [3] but at a larger student scale (that study used a 7M-parameter student): KataGo self-play positions were generated and analysed at 4,000 visits to produce policy, value, score and ownership labels, and the network was trained on those labels rather than on self-play outcomes.

The teacher network is $kata1 - b18c384nbt - s9791399168 - d4261348054$ (18 blocks, 384 trunk channels), the checkpoint shipped as KataGo's default model; the same network, at 100 visits, adjudicates all games reported here. The policy label is KataGo's **raw network policy**; value, score lead and ownership are taken from the

searched root (see Section 4.2). The distilled checkpoint is referred to throughout as the **distillation baseline**; it is held read-only and serves as a fixed opponent.

The deployed configuration is policy argmax with a self-atari filter and no search.

2.2 Search

MCTS with PUCT selection: 400 simulations, c_puct = 1.6, value mixing weight w_win = 1.0, score weight = 0, first-play-urgency reduction 0, leaf batch size 1. Tree reuse across moves was enabled for self-play (measured 1.68x speedup) and disabled in the evaluation harness inherited from earlier work.

2.3 Self-play and labelling

Self-play starts from the empty board with a 400-move cap. The first 30 plies are sampled at temperature 1.0; subsequent moves are argmax. Generations 1 and 2 sampled from the policy directly; generation 3

sampled from the MCTS visit distribution and stored the visit counts at every position.

All positions were labelled by KataGo at maxVisits = 1, yielding win rate, score lead and ownership from a single query. Value targets are in the current player's frame, $t = 2w - 1$, clipped to +/-0.98 for the regression loss only.

The game outcome z was **not** used as a value target: as the player strengthened, an increasing fraction of self-play games failed to terminate within the move cap (Section 3.8), leaving z undefined for roughly half of them.

2.4 Training objectives

All generations were trained using Adam with a learning rate of 1×10^{-4} for the shared trunk and 1×10^{-3} for all output heads. The batch size was 64, and early stopping was performed using a held-out calibration set. Data were split into training, calibration, and validation sets at the game level, rather than at the position level.

In generations 1 and 2, the policy head was trained using a KL-divergence anchor to the previous policy. Generation 3 instead optimized the policy using cross-entropy to the MCTS visit distribution while retaining the same value and ownership objectives.

In generation 3, the KL anchor was removed because the anchor objective and the cross-entropy target directly oppose one another. Each training batch consisted of 50% newly generated positions (including MCTS visit-count targets) and 50% positions from earlier generations (providing only value and ownership targets), ensuring that the value head continued to learn from the larger accumulated dataset.

2.5 Evaluation

Every reported decision rests on **272 games from the empty board**, colours alternating, with both engines subject to the identical rule that the first 30 plies are sampled at temperature 1.0. Both sides use the same self-atari filter. Outcomes are adjudicated by **KataGo at 100 visits**; the engine's own Tromp-Taylor scoring

agreed with this only 65% of the time and the disagreement was not a consistent-direction bias, so it was not used.

Win rates are reported with **95% Wilson confidence intervals**. At n = 272 the resolution near p = 0.5 is +/-0.059. Comparisons between two independently measured win rates use a two-proportion z test; comparisons of proportions at small n use Fisher's exact test.

A result whose interval spans 0.500 is reported as "not distinguishable", never as "refuted".

2.6 Pre-registration

Every decision criterion, stopping condition and cost estimate was written into a project document ($PREREGISTRATION.md$) **before** the corresponding experiment ran, and was not changed afterwards. Stopping conditions were expressed as multipliers or absolute values rather than qualitative judgements. Where a prediction was made in advance (Section 3.6), the ranges that would count as confirming or refuting it

were registered first.

This discipline was adopted because proxy metrics repeatedly failed to predict outcomes (Section 3.7), which made post-hoc reinterpretation an active hazard.

2.7 Hardware

One NVIDIA RTX 4000 SFF Ada (20 GB). Generation 3 self-play took 23.1 hours for 200 games / 74,908 positions across 8 processes.

3. Results

3.1 Search degraded play

Contrary to expectation, adding Monte Carlo Tree Search (MCTS) did not improve the distilled network. Instead, MCTS with 400 simulations achieved a win rate of only **0.317** against the same network playing by policy argmax without search. Increasing the number of simulations further degraded performance rather than improving it, indicating that the search procedure was systematically amplifying, rather than correcting, errors in the network evaluation.

This unexpected result suggested that the failure originated either from the search algorithm itself or from the information supplied to the search.

3.2 The defect was in the value head, not the search

To distinguish between these possibilities, we repeated the same search while replacing **only the leaf evaluation** with KataGo at a single visit. All other components of the search algorithm were left unchanged.

Under this condition, the win rate immediately increased from **0.317** to **0.983** ($p = 1.9 \times 10^{-14}$), demonstrating that the search procedure itself was not responsible for the observed degradation.

To identify the defective network component, we next evaluated the policy and value heads independently on **sibling positions**—positions reached from a common parent by one alternative move—and compared their rankings with KataGo. A marked asymmetry emerged between the two heads, as summarized in **Figure 1**.

These results indicate that the value head remained **well calibrated in an absolute sense but failed to preserve local ranking fidelity**. Because MCTS repeatedly backs up leaf evaluations during tree expansion, even small ranking errors accumulate and increasingly influence move selection as the search deepens. Consequently, consulting the value head more extensively made the search progressively worse.

Several alternative explanations were investigated and excluded. The degradation was not explained by a systematic optimistic or pessimistic bias in the value prediction (mean residual −0.0091), by the absence of first-play urgency (correct implementation changed the win rate by only −0.0063; not significant), by the choice of *c*_{puct}, or by the architecture of the value head itself.

3.3 Rebuilding the value head was not sufficient

To determine whether poor value ranking alone explained the degradation, we retrained only the value head while keeping the shared trunk frozen. Self-play positions were relabelled with KataGo, and an additional pairwise ranking loss was introduced for sibling positions.

This intervention improved sibling ranking accuracy from **0.54** to **0.64**, confirming that the value head had learned to discriminate local alternatives more effectively.

However, search still did not improve playing strength. With the rebuilt value head, MCTS at 400 simulations performed no better than policy argmax (difference = **+0.000**, n = 60).

These results indicate that improving value ranking alone was insufficient to restore the expected benefit of search.

3.4 Generations 1 and 2: protecting the policy is not improving it

We next examined whether the policy itself could be improved while remaining constrained by a KL-divergence anchor to the previous network.

In **Generation 1**, updating the shared trunk under this objective substantially improved playing strength relative to the fixed distillation baseline, achieving a win rate of **0.868** (95% CI, 0.822–0.903), corresponding to approximately **+327 Elo**.

In **Generation 2**, the identical procedure was repeated using twice as many self-play games (400 instead of 200), with the KL anchor transferred to the immediately preceding generation. Despite doubling the amount of self-play data, the resulting win rate against the previous generation was **0.526** (95% CI, 0.466–0.584), which was **not distinguishable** from chance.

Thus, increasing the amount of self-play alone produced no measurable improvement.

The common feature of both generations was that the policy was constrained to remain close to its predecessor. A KL anchor acts as a mechanism that **preserves** an existing policy rather than one that

actively **improves** it. The corresponding game results are summarized in **Figure 2**.

3.5 Generation 3: search succeeds as a teacher, despite failing as a player

The key change introduced in **Generation 3** was the policy-training target. Instead of maintaining the policy with a KL-divergence anchor, the policy head was trained by **cross-entropy to the MCTS visit distribution**.

Before training, we quantified the difference between the current policy and its proposed teacher. This measurement, together with its interpretation, was explicitly recorded in the pre-registration document before any training was performed.

MCTS selected a different move from the policy argmax in **73%** of positions. Since Section **3.3** had already shown that following those search-selected moves did **not** improve playing strength, the pre-registered expectation was deliberately two-sided: the new training targets would consist largely of moves that were **different, but not necessarily better**.

Despite this expectation, the outcome was striking. As summarized in **Table 1**, training on the MCTS visit distribution produced a substantial improvement in playing strength. Generation 3 achieved a **94.1%** win rate against the original distillation baseline and **0.824** (95% CI, 0.774–0.864) against the immediately preceding generation.

Table 1. Search remained an effective teacher despite failing as an inference-time player.

Comparison	Win Rate (95% CI)
Generation 2 vs. preceding generation	0.526 (0.466–0.584)
Generation 2 vs. distillation baseline	0.864 (0.818–0.900)
Generation 3 vs. preceding generation	**0.824 (0.774–0.864)**
Generation 3 vs. distillation baseline	**0.941 (0.907–0.963)**

Generation 3 represented an improvement to approximately **+268 Elo** relative to the preceding generation and represented a significant improvement over the distillation baseline (z = +3.04, two-sided p = 0.0024). Notably, this gain was obtained using only **200 self-play games (74,908 positions)**—less than half the self-play data used in Generation 2 (400 games, 152,311 positions).

The magnitude of policy change was also the largest observed during the project. Top-1 agreement with the previous checkpoint decreased from **1.000** (frozen trunk) to **0.797** under the KL-anchor objective and further to **0.351** under visit-distribution cross-entropy, indicating substantially greater policy movement.

These results demonstrate that the MCTS visit distribution remained a highly effective **teaching signal**, even though the same search procedure failed to improve inference-time play.

3.6 The teacher-policy gap shrinks as the policy absorbs it

After Generation 3 training, we repeated the analysis of Section **3.5** to quantify how much of the teacher signal had been absorbed by the updated policy.

The mean KL divergence between the MCTS visit distribution and the policy decreased from **1.353** to **0.592**, while top-1 agreement increased from **0.269** to **0.506**. Thus, the policy moved substantially closer to its teacher, reducing the remaining discrepancy available for further learning.

This behaviour is consistent with the expected convergence of an AlphaZero-style iterative training loop: as the student absorbs the teacher's knowledge, the teacher–policy gap progressively narrows.

Whether the remaining gap predicts the magnitude of improvement in the next generation remains under investigation. Before any Generation 4 experiments were performed, both the prediction and the quantitative ranges that would support or refute it were pre-registered. Because only **two measured data points** are currently available, one of which lies effectively at the origin, no proportional relationship can yet be established. The dashed line in **Figure 3** therefore represents a registered hypothesis rather than an experimental result.

3.7 Proxy metrics did not predict playing strength

The relationship between changes in proxy evaluation metrics and actual playing strength is summarized in **Table 2**. Playing strength was measured by 272-game matches adjudicated by KataGo.

Across multiple experiments, improvements in proxy metrics did not consistently predict improvements in playing strength. Conversely, deterioration in proxy metrics sometimes accompanied statistically significant gains in practical performance. These observations indicate that proxy metrics should be interpreted cautiously and validated against direct game outcomes.

Table 2 Relationship between proxy metrics and actual playing strength.

Experiment	Proxy Metric	272-Game KataGo-Adjudicated Match
1	Sibling ranking: **0.585 → 0.644** (**improved**)	No measurable change (policy identical by construction)
2	Policy evaluation loss: **0.105 → 0.120** (**worsened**)	No significant difference (p = 0.900)
3	Policy evaluation loss: **0.120 → 0.124** (**worsened**)	**Significant improvement** (p = 0.0024)

Proxy metrics failed in **both directions**. In Experiment 3, for example, value-related metrics changed only marginally (sibling ranking: +0.006), whereas playing strength increased by approximately **+268 Elo**.

We also examined whether the sibling-ranking metric correlated with playing strength across four checkpoints. The resulting Spearman correlation coefficient was **−0.500**. However, with only three comparable intervals, the smallest attainable one-sided *p*-value is **1/6 (= 0.167)**. Consequently, no amount of additional game-playing could have rendered this experimental design statistically significant; establishing such a relationship requires **more checkpoints rather than more games**.

3.8 Self-play increasingly fails to terminate

The characteristics of self-play games generated during successive training generations are summarized in **Table 3**.

As search was introduced and used more extensively, self-play games became progressively longer and increasingly reached the predefined **400-move limit** rather than terminating naturally by double pass.

Table 3. Self-play behaviour across successive generations.

Generation (Self-Play Player)	Search Setting	Median Game Length	Reached 400-Move Cap	Terminated by Double Pass
Generation 1 (distilled policy)	None	348	10.0%	90.5%
Generation 2	None	389	36.5%	64.3%
Generation 3	**MCTS (400 simulations)**	**400**	**51.0%**	**46.0%**

Median game length eventually reached the predefined move cap. Correspondingly, training labels became increasingly saturated, with the median absolute score lead increasing from **26.7** to **48.0** points.

As a consequence, the final game outcome (*z*) became unsuitable as a value-training target because approximately half of the self-play games no longer reached a natural termination.

Importantly, this behaviour was confined to self-play. In the evaluation matches between the Generation 2 and Generation 3 networks, **none** of the 272 games reached the move limit (0/272; median game length, 280 moves).

4. Discussion

4.1 Search as a teacher, not a move selector

The central finding of the present study is that **the inference-time performance of an improvement operator and the teaching value of its output are distinct properties**.

In the present system, Monte Carlo Tree Search (MCTS) consistently failed to improve playing strength. At 400 simulations it scored only **0.317** against the policy itself, and rebuilding the value head did not

restore any measurable advantage. Search therefore failed as an inference-time move selector.

Nevertheless, exactly the same search procedure produced the largest policy improvement observed in the project when its visit distribution was used as the training target. Generation 3 achieved a **94.1%** win rate against the original distillation baseline despite using less than half the self-play data of Generation 2.

The apparent contradiction disappears once these two roles of search are separated. During inference, search attempts to identify the single best move using the current value function. During training, however, the visit distribution records how search allocated computational effort across plausible continuations. Even when the value head is insufficiently accurate to identify the best final move, that distribution may still contain information unavailable in the policy prior alone.

This distinction has close conceptual parallels in reinforcement learning for large language models (LLMs), where the output of an imperfect selector can still provide an effective teaching signal. [4-9]

Table 4. Conceptual correspondence between the present study and reinforcement learning for large language models.

This study (Computer Go)	Approximate analogue in LLM RLHF
Value head	Reward model / Critic / Value function
Monte Carlo Tree Search (MCTS)	PPO advantage estimation [4], Best-of-*N*, or rejection sampling
Knowledge distillation from KataGo	Teacher distillation or reward-model training[5]
Policy argmax	Greedy decoding

This comparison is intended to illustrate **functional similarity rather than algorithmic equivalence**. In both settings, a learned policy interacts with a separately learned evaluator, making the overall system vulnerable to imbalance between these components.

Several recent LLM methods follow essentially the same principle. RAFT generates multiple candidates, ranks them using a reward model, and fine-tunes only on the highest-ranked outputs [10]. Rejection-sampling fine-tuning for mathematical reasoning applies an analogous strategy [11], while STaR bootstraps from self-generated reasoning trajectories that eventually produce correct answers [12]. In each case, the selector is valuable primarily as a **teacher**, not because it is required during inference.

4.2 Relation to our previous distillation study

Our previous study [3] reported a **distillation ceiling** for small Go networks. Scaling the training corpus from approximately 45,000 to 200,000 KataGo-labelled positions resulted in monotonic degradation of policy agreement with the teacher and no overall improvement in value calibration. We argued that the

ceiling reflected a structural limitation of conventional distillation: only per-position evaluations are transferred, whereas much of the teacher's playing strength resides in search.

The present work should **not** be interpreted as a controlled extension of that scaling experiment. The student architecture, training corpus, and optimisation schedule all differ substantially, and direct numerical comparison between the two studies is therefore inappropriate.

Instead, the conceptual connection lies elsewhere. Reference [3] suggested that methods based on **search-derived supervision**, rather than single-position outputs, might overcome part of the limitation. The present work implements one member of that family.

Notably, the supervision used here was **not** derived from the teacher's search. Instead, the policy was trained using the **student's own MCTS visit distribution** generated with only 400 simulations. Despite the limited search budget and an imperfect value head, this objective produced the largest improvement observed throughout the project.

This result suggests that the critical ingredient is not necessarily stronger search, but rather a supervision signal that is itself **a function of search** rather than of a single forward pass.

An additional observation further distinguishes the present work from conventional process-distillation studies. The search distribution that successfully trained the policy **did not itself produce stronger play**. Thus, the benefit cannot simply be explained by the visit distribution representing "better moves." Instead, the search distribution appears to capture useful structural information that the policy alone does not represent.

To our knowledge, this is the first demonstration within this line of Go distillation research that a **search-derived policy target** can substantially improve the student even when the corresponding search procedure fails to improve inference-time play.

4.3 Methodological implications

A second contribution of this study concerns **experimental methodology**.

Throughout the project, proxy metrics repeatedly failed to predict practical playing strength. Improvements in sibling-ranking accuracy, policy evaluation loss, and other intermediate quantities did not consistently translate into stronger play, while statistically significant gains occasionally occurred despite deterioration in the proxy metrics.

The present work therefore reinforces the importance of evaluating game-playing systems by **direct head-to-head competition with confidence intervals**, rather than relying exclusively on intermediate optimisation metrics.

A related lesson concerns experimental design. Generation 2 showed no statistically distinguishable improvement over its predecessor, whereas Generation 3—differing by only a single component of the

training objective—produced the largest gain observed in the project. A single negative result should therefore not be interpreted as evidence that an approach has been exhausted.

Finally, the project illustrates the practical value of **pre-registration** in AI systems research. Decision criteria, stopping rules, predicted effect sizes, and interpretation ranges were specified before experiments were conducted. Because several proxy metrics proved unreliable, this procedure substantially reduced the risk of post-hoc reinterpretation.

More broadly, the present study suggests that future work on search-based learning systems should distinguish explicitly between two fundamentally different questions:

1. **Does an improvement operator improve inference-time decisions?**
2. **Does the information produced by that operator improve subsequent learning?**

The results presented here demonstrate that the answers need not be the same.

5. Limitations

This work should be interpreted as a mechanistic case study rather than a benchmark comparison. Several limitations should therefore be considered when interpreting the results.

First, the results are derived from a **single architecture**, a **single random seed**, and a **single game (19×19 Go)**. The work should therefore be interpreted as a detailed case study rather than evidence of generality.

Second, although every comparison is based on 272 head-to-head games with confidence intervals, smaller effects remain below the statistical resolution of the present experiments.

Third, the self-play loop is **not fully self-contained**. Because an increasing fraction of self-play games reached the predefined move limit, value supervision continued to rely on KataGo labels rather than terminal game outcomes (z). Systems that can reliably train from game outcomes may behave differently.

Finally, this study establishes **an empirical phenomenon rather than its underlying mechanism**. We show that MCTS visit distributions provide effective supervision despite failing to improve inference-time play, but the reason why these distributions contain useful learning signals remains unresolved. Likewise, the parallels drawn with reinforcement learning for large language models are intended as conceptual analogies rather than claims of identical underlying mechanisms.

Declarations

Acknowledgements

We thank the developers of KataGo, whose released network served as the distillation source and as the adjudicator for all reported games.

Implementation of experiment scripts was carried out with the assistance of an AI coding tool under the author's direction; all experimental designs, decision criteria and interpretations are the author's.

References

1. Silver D, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*. 2018;362(6419):1140-1144. doi:10.1126/science.aar6404. Preprint: arXiv:1712.01815.
2. Wu DJ. Accelerating self-play learning in Go. arXiv:1902.10565. 2019.
3. Kojima T. Reading depth does not transfer: the distillation ceiling in computer Go. Research Square: preprint rs-9966155, version 1. Posted 9 June 2026. doi:10.21203/rs.3.rs-9966155/v1.
4. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv:1707.06347. 2017.
5. Ouyang L, Wu J, Jiang X, Almeida D, Wainwright CL, Mishkin P, et al. Training language models to follow instructions with human feedback. arXiv:2203.02155. 2022.
6. Gao L, Schulman J, Hilton J. Scaling laws for reward model overoptimization. arXiv:2210.10760. 2022.
7. Shao Z, et al. DeepSeekMath: pushing the limits of mathematical reasoning in open language models. arXiv:2402.03300. 2024.
8. Ahmadian A, Cremer C, Gallé M, Fadaee M, Kreutzer J, Pietquin O, Üstün A, Hooker S. Back to basics: revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2024: 12248-12267. arXiv:2402.14740.
9. Rafailov R, Sharma A, Mitchell E, Ermon S, Manning CD, Finn C. Direct preference optimization: your language model is secretly a reward model. arXiv:2305.18290. 2023.
10. Dong H, et al. RAFT: reward rAnked FineTuning for generative foundation model alignment. arXiv:2304.06767. 2023.
11. Yuan Z, Yuan H, Li C, Dong G, Lu K, Tan C, Zhou C, Zhou J. Scaling relationship on learning mathematical reasoning with large language models. arXiv:2308.01825. 2023.
12. Zelikman E, Wu Y, Mu J, Goodman ND. STaR: bootstrapping reasoning with reasoning. arXiv:2203.14465. 2022.

Figures

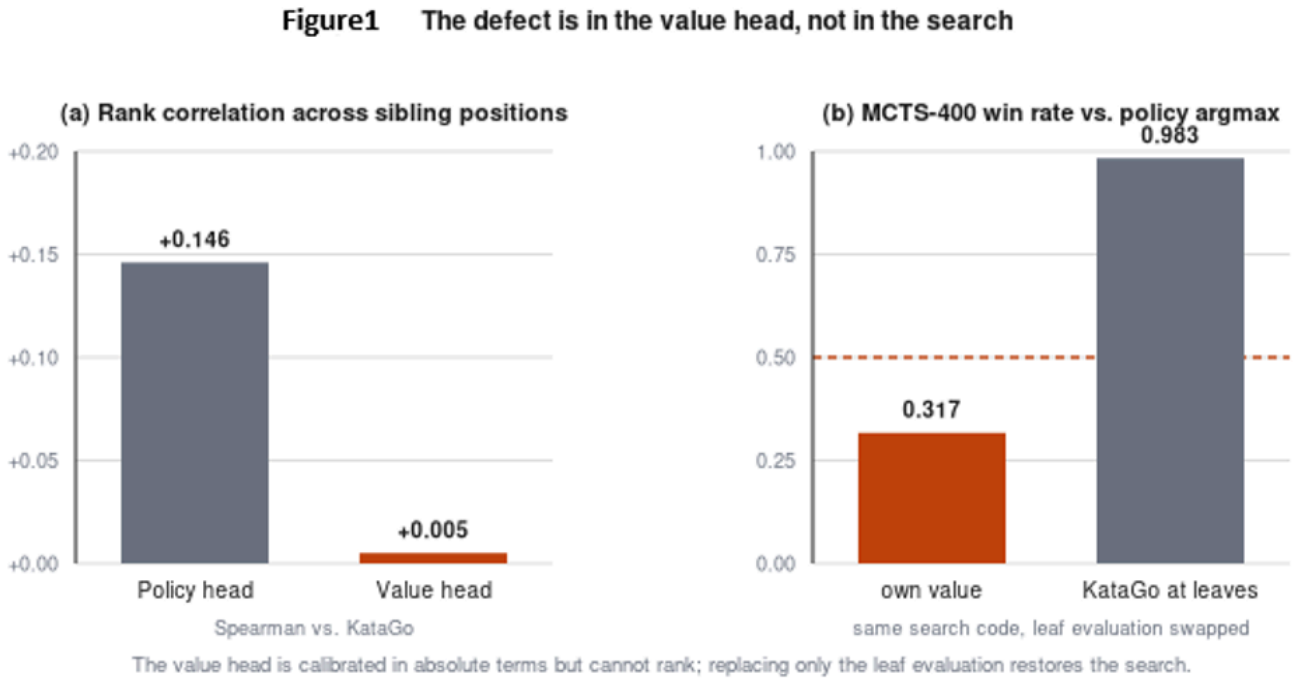


Figure 1

The defect was in the value head rather than the search procedure.
(a) Spearman rank correlation with KataGo across sibling positions. The policy head retained modest ranking ability (ρ = +0.146), whereas the value head was essentially unable to distinguish between sibling positions (ρ = +0.005).
(b) Win rate of MCTS (400 simulations) against policy argmax using the network's own value head (0.317) or replacing only the leaf evaluation with KataGo at one visit (0.983). The search algorithm was otherwise identical.

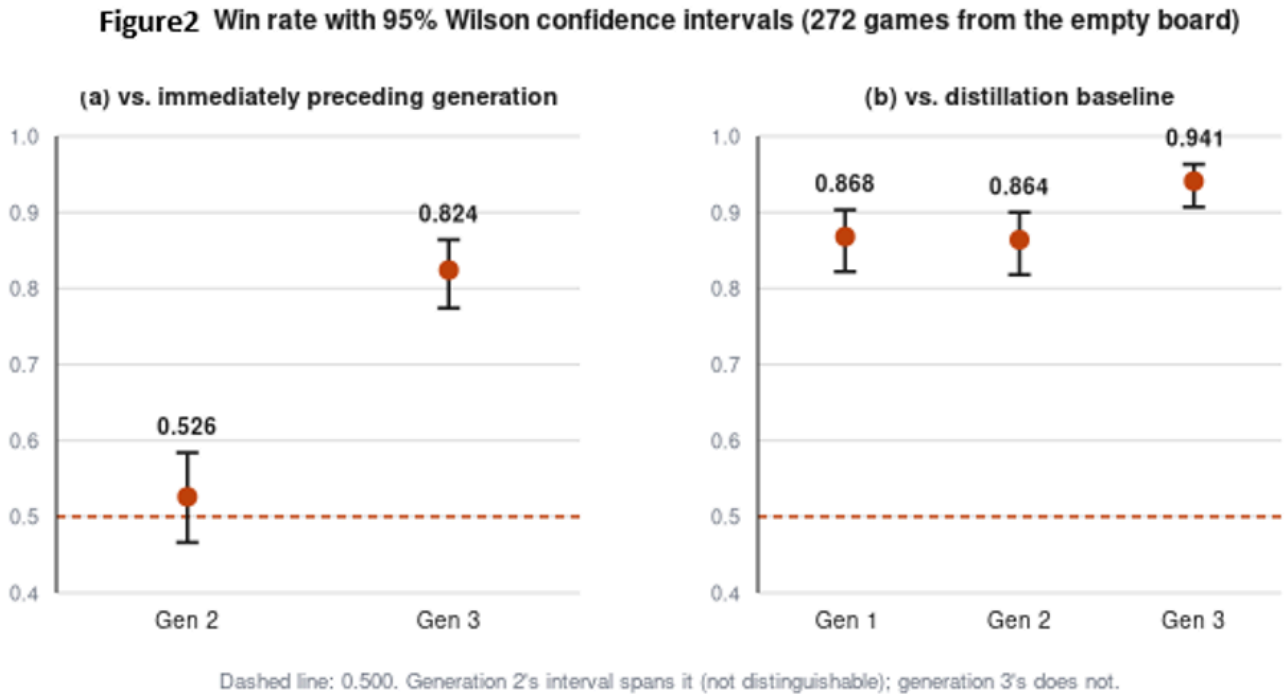


Figure 2

Win rates with 95% Wilson confidence intervals from 272 games starting from the empty board and adjudicated by KataGo at 100 visits. **(a)** Performance against the immediately preceding generation. The confidence interval for Generation 2 spans 0.500 (dashed line) and is therefore reported as *not distinguishable*, whereas Generation 3 shows a significant improvement. **(b)** Performance of each generation against the fixed distillation baseline.

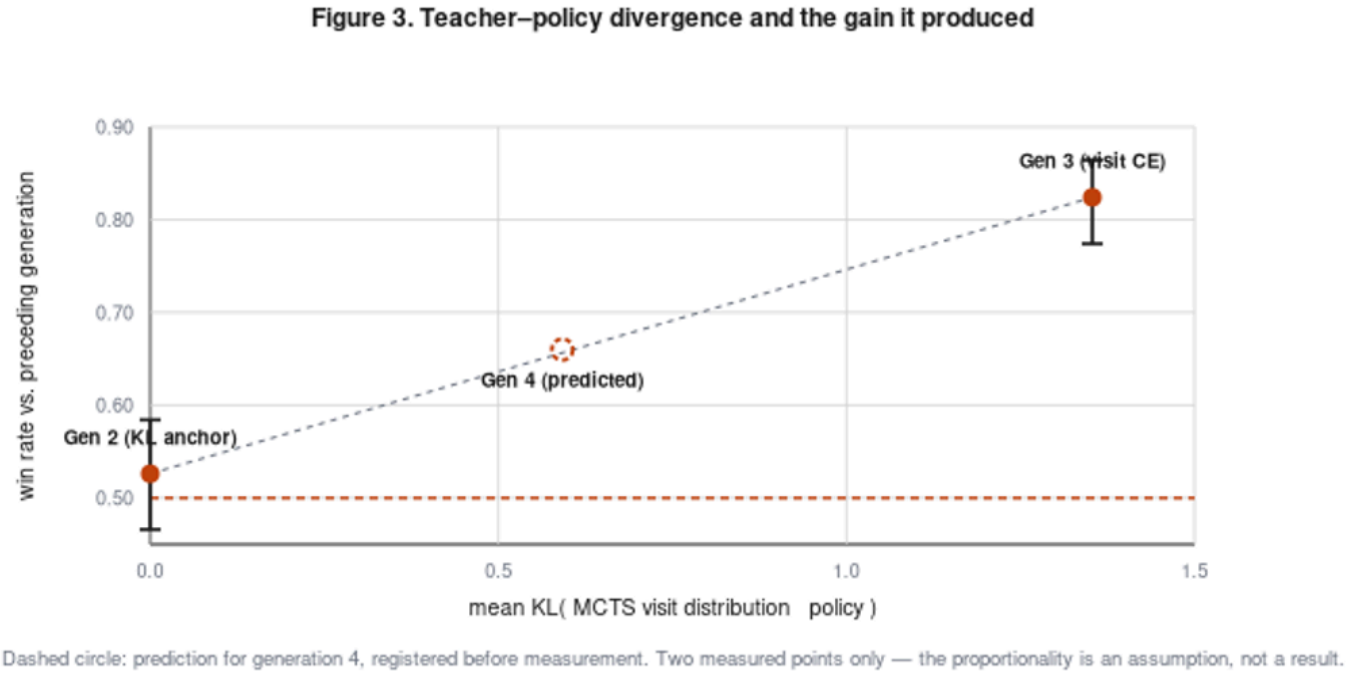


Figure 3

Relationship between the divergence of the MCTS visit distribution from the policy (mean KL divergence) and the improvement achieved over the preceding generation. Generation 2 used a KL-divergence anchor and therefore treated the policy itself as the teacher (divergence ≈ 0). The open circle represents the pre-registered prediction for Generation 4. With only two measured observations, one effectively at the origin, proportionality cannot yet be inferred; the dashed line represents the hypothesis currently under test.