Investigating AlphaZero's Value Targets Using Strongly Solved Chinese Checkers

Nicolas Ong

Supervised by Nathan Sturtevant

Department of Computing Science, University of Alberta

September 16, 2026











Motivation: AlphaZero



AlphaGo defeats Lee Sedol, March 2016.

- Superhuman in Chess, Shogi, and Go.
- Defeated the world champion of Go, Lee Sedol, in 2016.
- The same machinery has been applied to other domains: matrix multiplication (AlphaTensor), sorting routines shipped in libc++ (AlphaDev), video compression (MuZero), competition mathematics (AlphaProof).

Silver et al. (2018); AlphaTensor (2022); AlphaDev (2023); MuZero (2022); AlphaProof (2024).

Motivation: Adversarial Strategies



- An adversarial player was trained that beats **KataGo**, a superhuman Go player – yet loses to human amateurs.
- AlphaZero is a strong player, yet there are simple strategies it performs poorly against.

There is more about AlphaZero and its training process to learn about.

Wang et al., *Adversarial Policies Beat Superhuman Go AIs* (ICML 2023).

Motivation: Research Question

- Win rate and Elo are **relative**: they only say how a player does against other players.
- A strongly solved game gives an **absolute** measure: every state has ground truth.

Can we use **solve data** to better understand AlphaZero and its **training process**?

Introduction

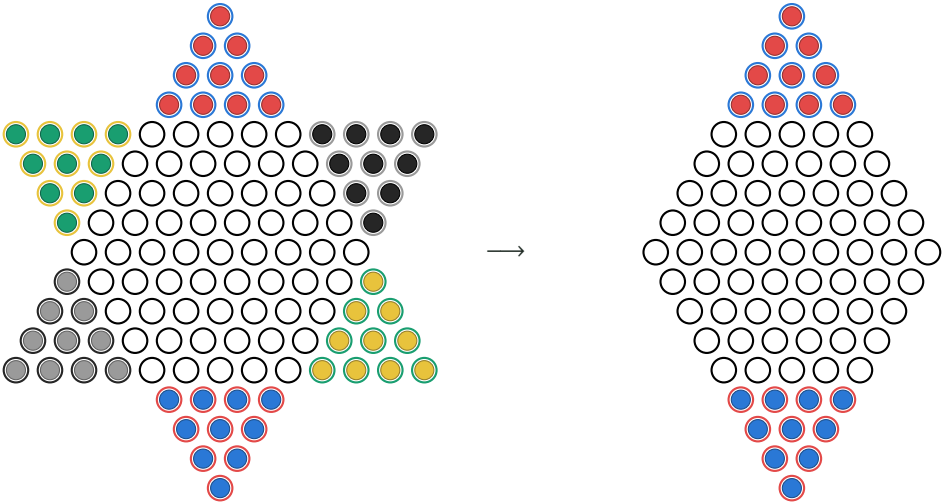
Introduction
- Motivation: Understanding AlphaZero
- Background: Chinese Checkers
- Background: AlphaZero
- Ground Truth from Solve Data
- Previous Work
- Our Implementation
- Contributions

AlphaZero with Ground Truth

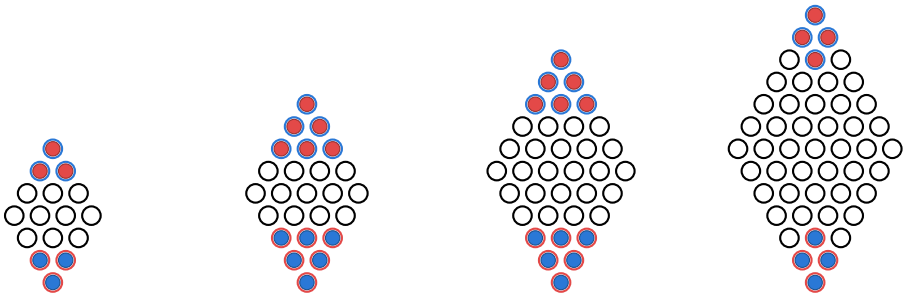
AlphaZero with TD(λ)

Conclusion

Chinese Checkers



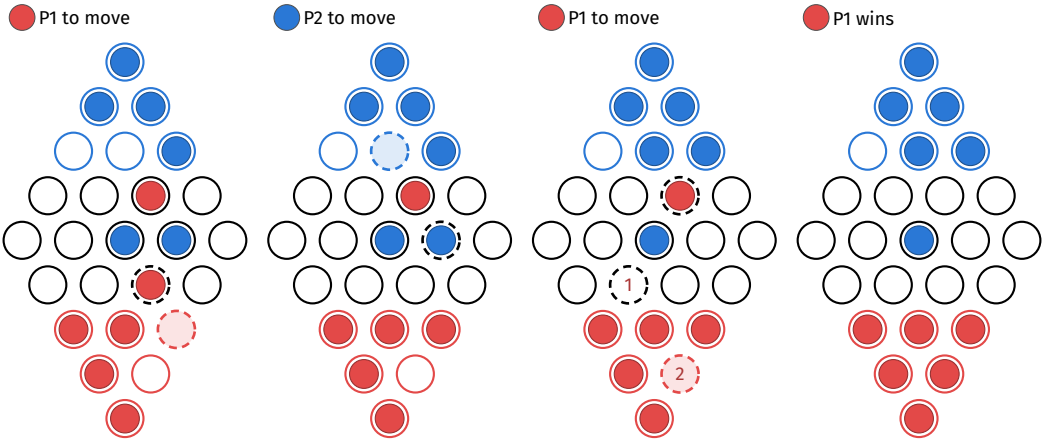
Chinese Checkers



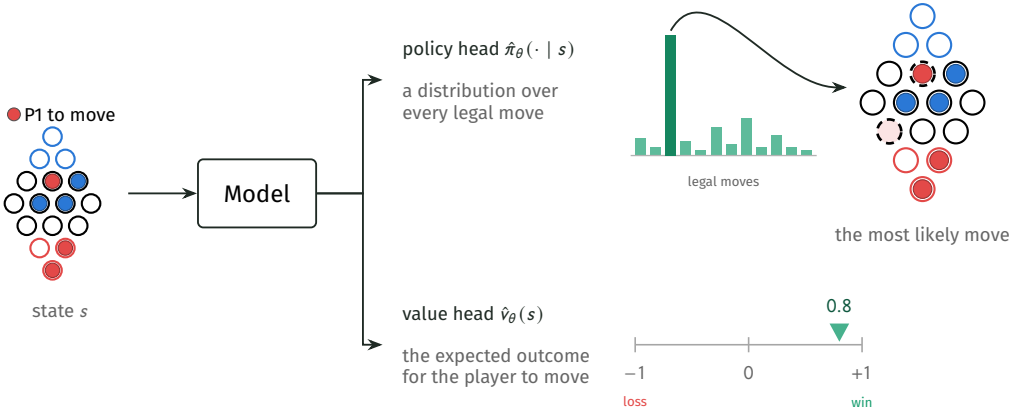
- An $n \times n$ rhombus with p pieces per player, written n^2-p.

Sturtevant, *On Strongly Solving Chinese Checkers* (2020).

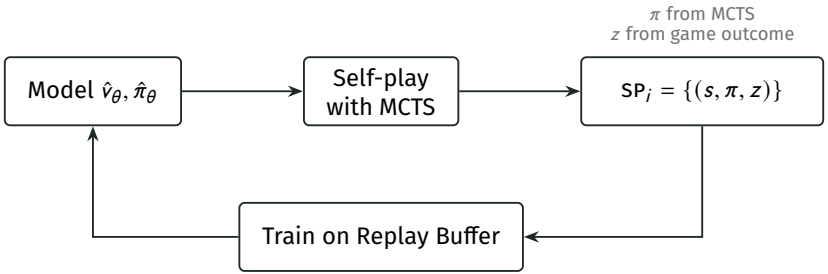
Chinese Checkers



AlphaZero: The Model

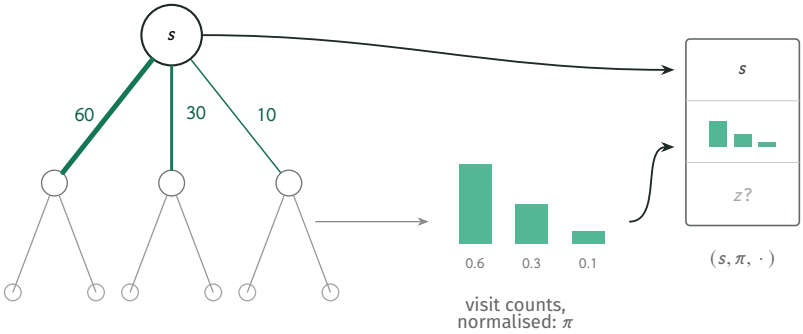


AlphaZero: The Training Loop



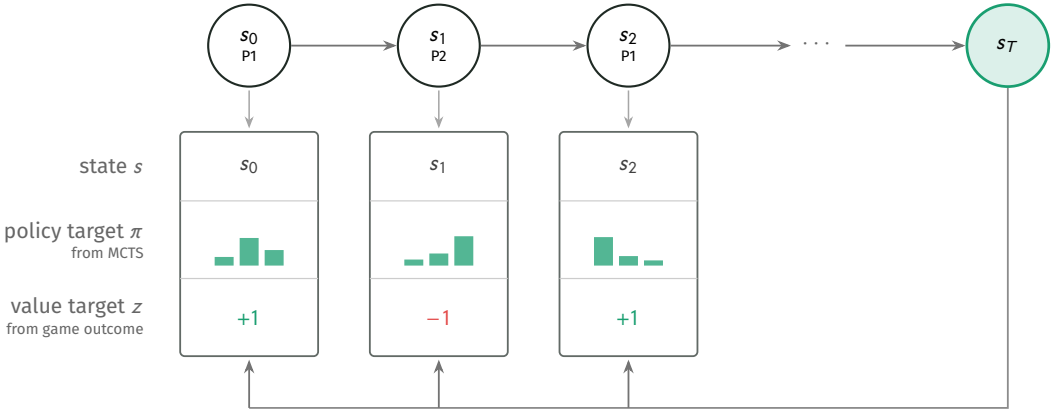
- Goal: learn a policy π that maximizes expected return v_π.
- Our experiments investigate **the value target** z.

AlphaZero: MCTS



MCTS from the root s

AlphaZero: Self-Play



Chinese Checkers: Strongly Solved

- Chinese Checkers is **strongly solved** for board sizes up to 36-6.
- The solve data gives $v_*(s) \in \{-1, 0, +1\}$, the game-theoretic value of every state s.
- We can derive an optimal policy $\pi_*(s)$ from $v_*(s)$: at every state it plays a move that achieves $v_*(s)$.

v_* and π_* allows for the direct evaluation of $\hat{v}_\theta$ and $\hat{\pi}_\theta$.

Sturtevant, *On Strongly Solving Chinese Checkers* (2020).

Chinese Checkers: The Solve Data

Board	States	Solve data
16-3	320,320	80.1 KB
25-6	9,610,154,400	2.40 GB
49-4	63,136,929,240	15.8 GB
36-6	2,313,100,389,600	578 GB

Sturtevant, *On Strongly Solving Chinese Checkers* (2020).

Evaluation Metrics with Ground Truth

Value Head Accuracy $\mathrm{Acc}(\tilde{v}_\theta \mid v_*, S)$: how often the (discretized) value head agrees with v_* on a set of states S.

Value Target Accuracy $\mathrm{Acc}(\textsc{sp} \mid v_*)$: how often the *targets* the network trains on are correct.

Other metrics include Policy Head Accuracy and Policy Target Accuracy.

Previous Work: Analysis of AlphaZero

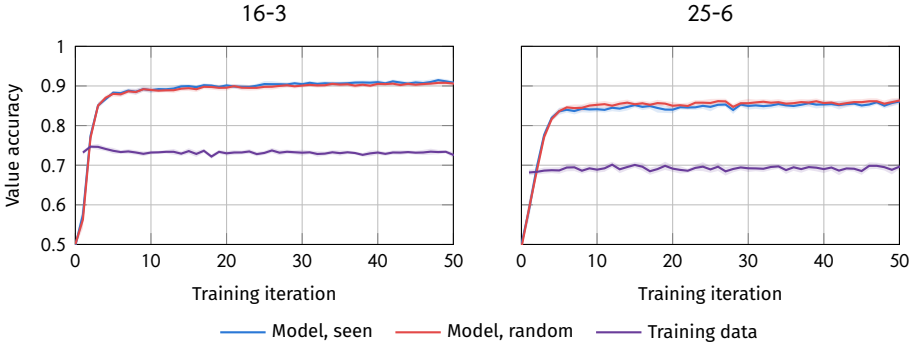
Without ground truth, analysis has still driven real improvements: engineered input features, auxiliary targets, new architectures. It has also exposed non-transitive and adversarial strategies.

With ground truth, previous work has used solve data (including tablebases) to:

- tune hyperparameters and compare training strategies on Connect Four (Prasad);
- study MCTS behaviour in Lc0 and KataGo (Haque; Husna);
- study training dynamics and how adversarial strategies exploit a trained player on Chinese Checkers (Karki) – Predecessor to this work.

Prasad (2018); Haque (2022); Husna (2024); Karki (2024).

Our Implementation Evaluated with Ground Truth



- Value head reaches 90% (16-3) and 85% (25-6)
- **The value head is consistently more accurate than the targets it learns from**, replicating an observation of Karki's.

Contributions

1. AlphaZero for Chinese Checkers, evaluated against **ground truth**.
2. We replicate a previous result: Training on ground truth labels makes AlphaZero **worse**. We find out why, and propose a fix.
3. TD(λ) value targets: more accurate training data and a stronger player.

AlphaZero with Ground Truth

Introduction

AlphaZero with Ground Truth
- Motivation
- The Experiment
- Our Replication
- Investigation
- Solution

AlphaZero with TD(λ)

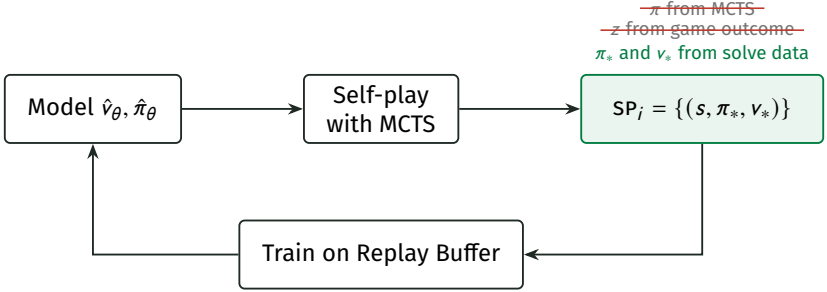
Conclusion

Motivation

- AlphaZero's value and policy targets (z and π) are imperfect.
- Idea: Replace z with v_* and π with π_*.

This gives us a clear upper bound on what better targets can buy.

The Experiment



- Only the **stored targets** change: $z \to v_*(s)$ and $\pi \to \pi_*(s)$.

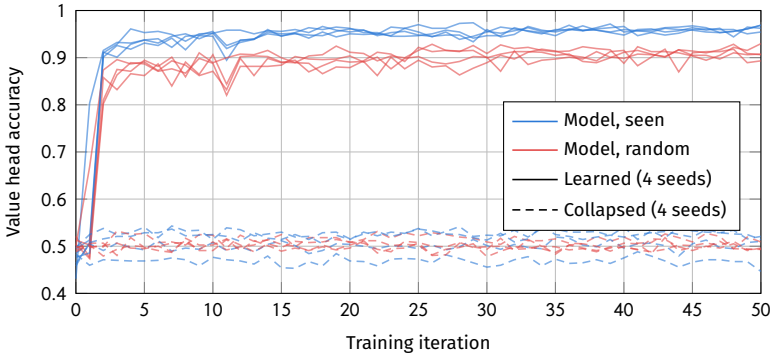
Previous Results

- Bigyan Karki, a former member of our lab, performed this experiment in his Masters research.
- The resulting player was “far worse than our original trained player without ground truth”.

Perfect labels made AlphaZero worse. Why?

Karki, *Diagnosing Weaknesses in AlphaZero* (2024, unpublished).

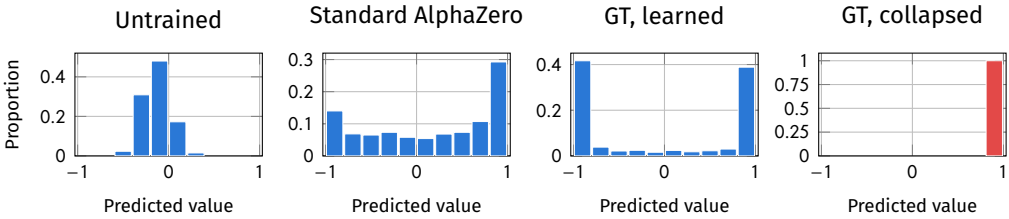
Our Replication



The outcome is bimodal.

Investigation: Value Head Collapse

- Record $\hat{v}_\theta(s)$ predictions for every state s in a set of random states.



- Collapsed runs predict a **single constant value** for every state.
- Looks like majority-class collapse.

Aside: What is Majority-Class Collapse?

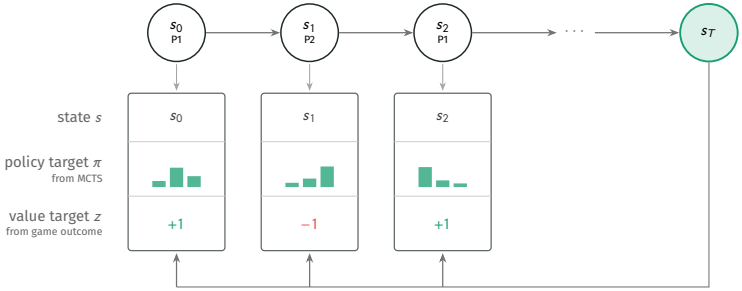
- A standard failure mode on imbalanced data: the model stops distinguishing its inputs and predicts the **majority class** for everything.
- A **local optimum**: The loss drops immediately and there is little pressure to leave.
- Well known wherever one class dominates: fraud, spam, anomaly detection, medical diagnosis.
- Established literature on mitigations: e.g. balancing the training data.

He & Garcia, *Learning from Imbalanced Data* (2009); Buda, Maki & Mazurowski (2018); Chawla et al., SMOTE (2002).

Investigation: Imbalance in the Value Targets

Define p_{maj}, the majority-class fraction of value targets (z) in SP_i.

Standard AlphaZero: $p_{\text{maj}} \approx 0.5$ **Ground Truth:** $p_{\text{maj}} \approx 0.65$



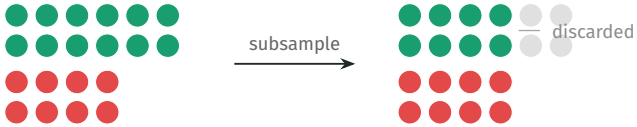
AlphaZero's default value targets are balanced by construction.

Investigation: Supervised Learning

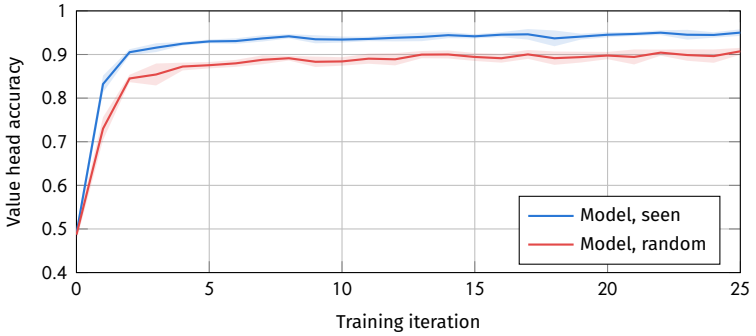
- Motivation: The imbalance looks the same in both collapsed and non-collapsed runs.
- Idea: Isolate the effects of imbalanced training data from self-play
- Method: Perform supervised learning on an imbalanced dataset of random states ($p_{\text{maj}} = 0.6$), 30 seeds.

15 of 30 collapse (50% ± 17%).
The bimodality is a property of the network and the data.

Collapse Mitigation: Balancing with Subsampling

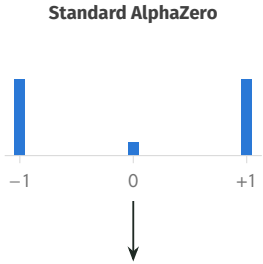


Results of Subsampling

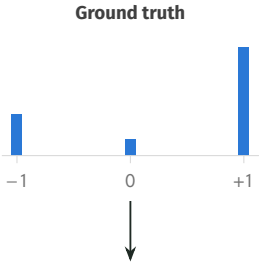


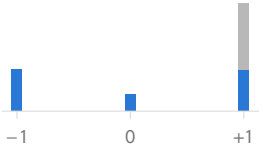
All 8 runs learn, reaching ~95% on seen and ~90% on random states – **above standard AlphaZero**.

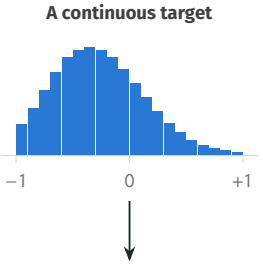
Balancing a Continuous Target



balanced by construction







?

Fix the network instead

- Goal: Make the network robust to imbalance in the training data.
- Zero-initialize the weights of the value head's final linear layer.
- Tried alongside: more residual blocks, a separate backbone per head, gradient clipping, removing draws/longer games.

Supervised control	p_{maj}	Collapsed
Standard value head	0.6	15/30
Zero-initialized head	**0.9**	**0**/**30**

- Zero-initialization eliminates collapse in all our settings, so we use it in the next section.

Lessons

- Replicated previous result where ground truth targets made AlphaZero worse.
- A diagnosis of why ground truth value targets prevented learning: imbalanced value targets and a brittle network led to majority-class collapse in the value head.
- A mitigation that makes alternative value targets viable: zero-initialization of the value head makes the network robust to imbalance in the training data.
- An instance of RL training data violating the assumptions neural networks are built on (e.g. non-i.i.d., non-stationary, imbalanced, etc.).

Sutton & Barto, *Reinforcement Learning* (2018), Ch. 9.1.

AlphaZero with TD(λ)

Introduction

AlphaZero with Ground Truth

AlphaZero with TD(λ)
- Background: TD(λ)
- TD(λ) targets in AlphaZero
- Results: Accuracy, Win Rate, Targets
- Extension: λ-interpolation

Conclusion

Motivation: TD(λ)

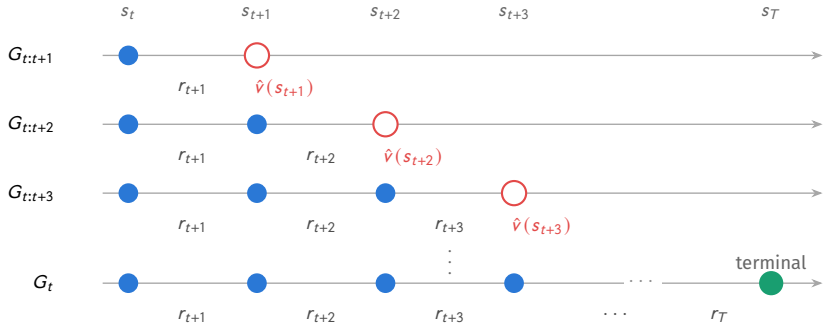
- We made AlphaZero able to train on targets it was not designed for – ground truth – and its value head accuracy improved.
- The value head achieves a higher accuracy than the targets it trains on.
- Is there an alternative value target that incorporates the model's own estimates?

$$\underbrace{z}_{\text{the game outcome}} \longrightarrow \underbrace{r_{t+1} + \hat{v}_\theta(s_{t+1})}_{\text{one step ahead}}$$

TD(λ) interpolates between the **game outcome** and **bootstrapping** off the model's own estimates.

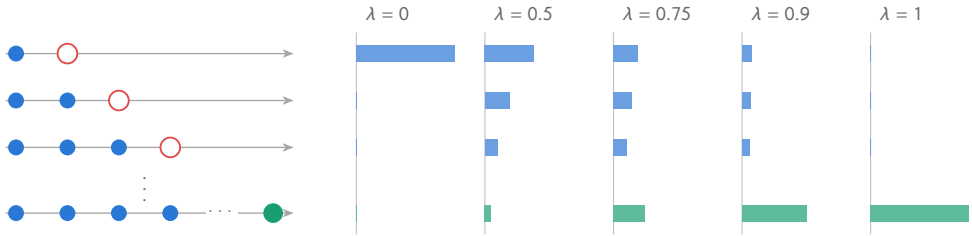
n-step targets

Bootstrap off the value estimate n steps ahead:

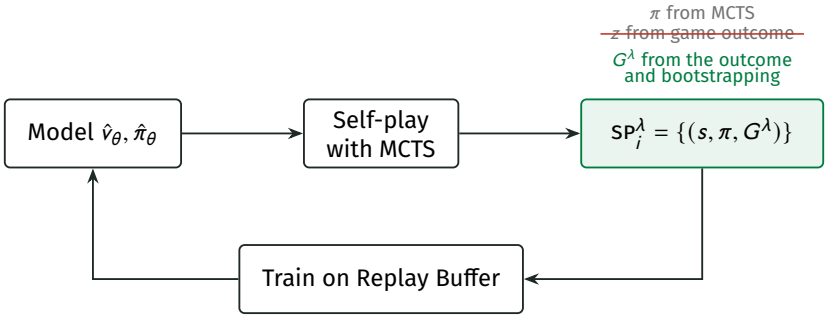


TD(λ) Targets

G^{λ}: A weighted average over *every* n-step return:

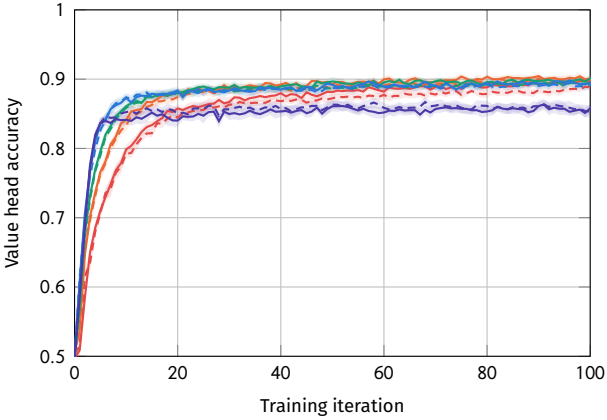


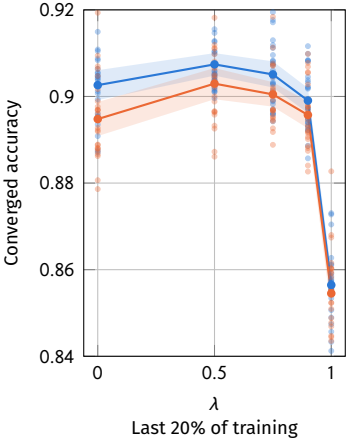
TD(λ) targets in AlphaZero



- Self-play at iteration i stores G^λ in place of z, computed from $\hat{v}_{\theta_{i-1}}$.
- Boards 16-3, 25-6, 36-6, 49-4; $\lambda \in \{0, 0.5, 0.75, 0.9, 1.0\}$.
- Zero-initialized value head throughout. Default initialization results in collapse for $\lambda = 0$.

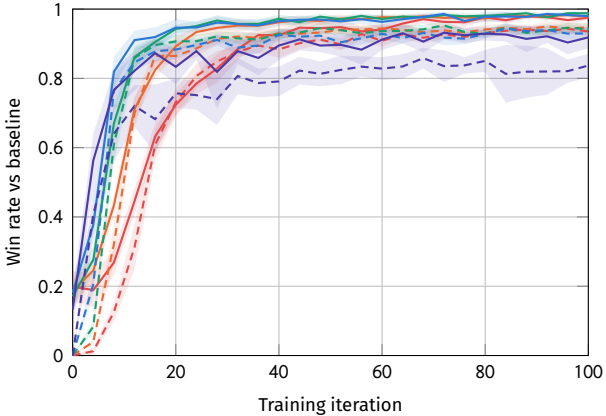
Value Head Accuracy (25-6)

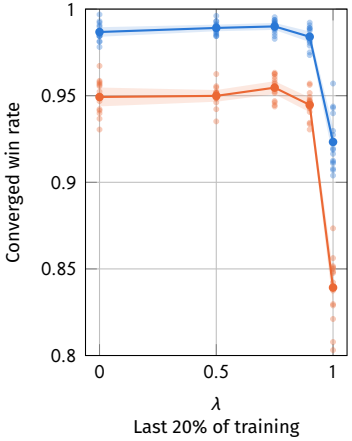


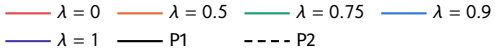


$\lambda = 0$ $\lambda = 0.5$ $\lambda = 0.75$ $\lambda = 0.9$
$\lambda = 1$ Seen Random

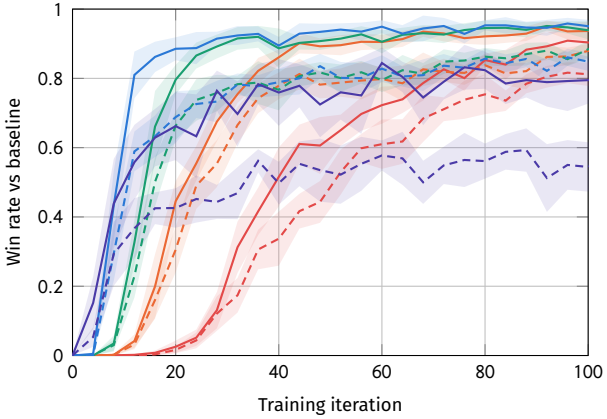
Win Rate Against a Baseline (25-6)

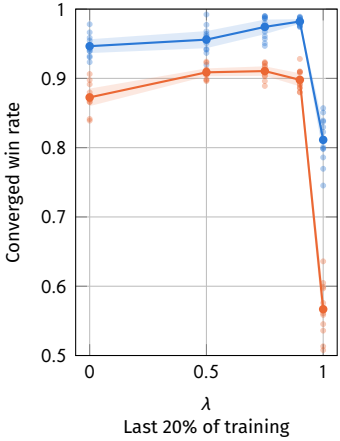


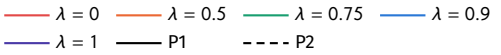




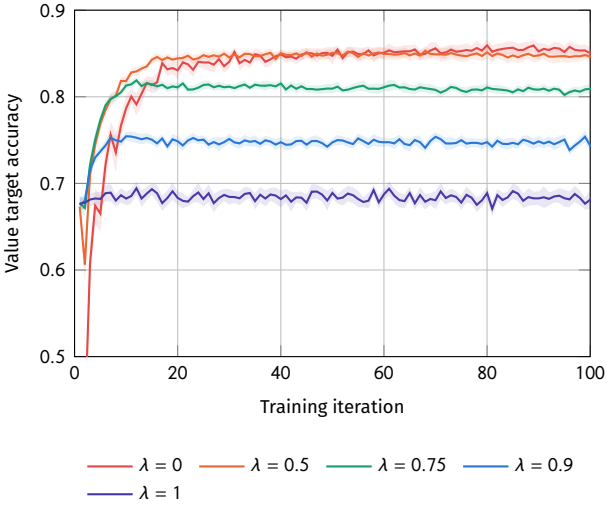
Win Rate Against a Baseline (36-6)

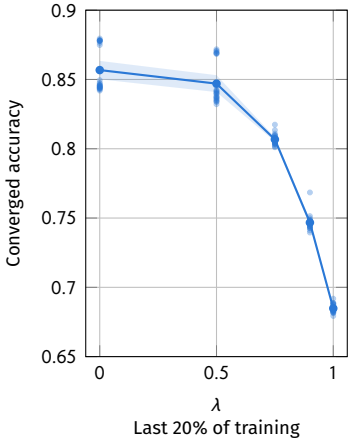




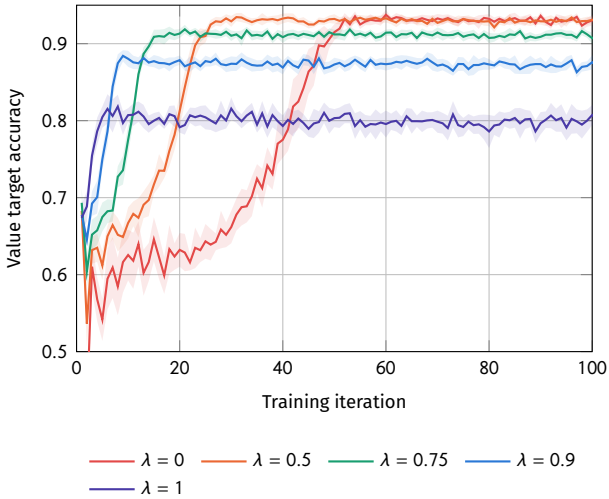


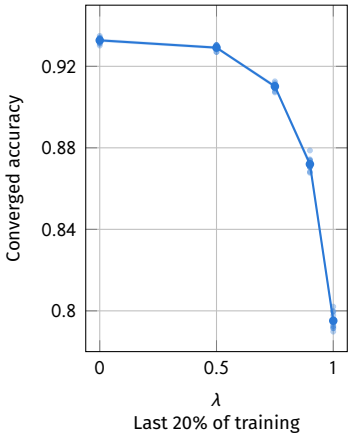
Value Target Accuracy (25-6)





Value Target Accuracy (49-4)



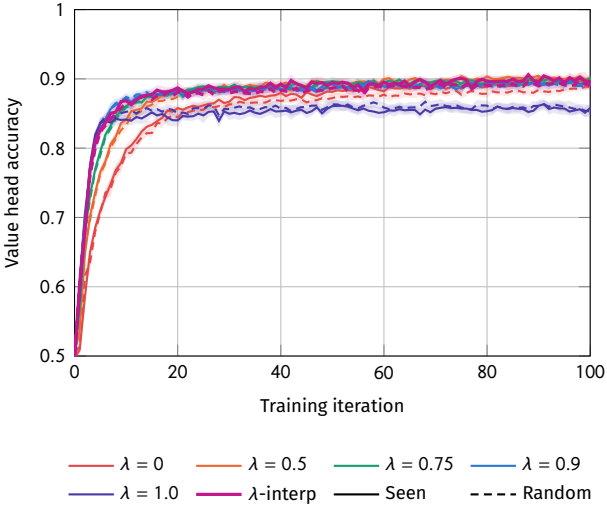


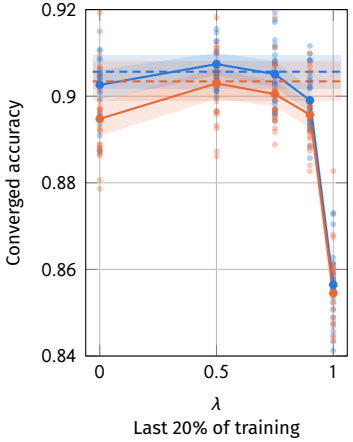
Extension: λ-interpolation

- Motivation 1: Lower λ achieves higher final model accuracy. Higher λ achieves faster convergence.
- Motivation 2: The λ with the most accurate value targets decreases over training.

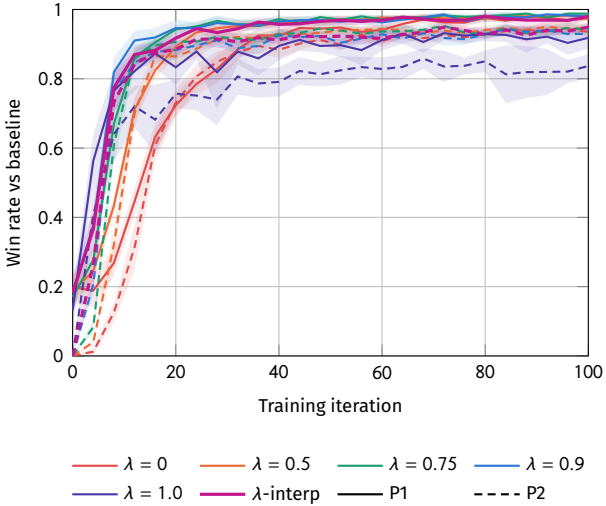
λ-interpolation: interpolate λ linearly from 1 to 0 over the first 25 iterations, then hold at 0.

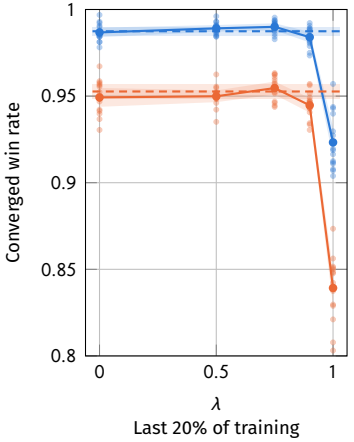
λ-interpolation: Value Head Accuracy (25-6)



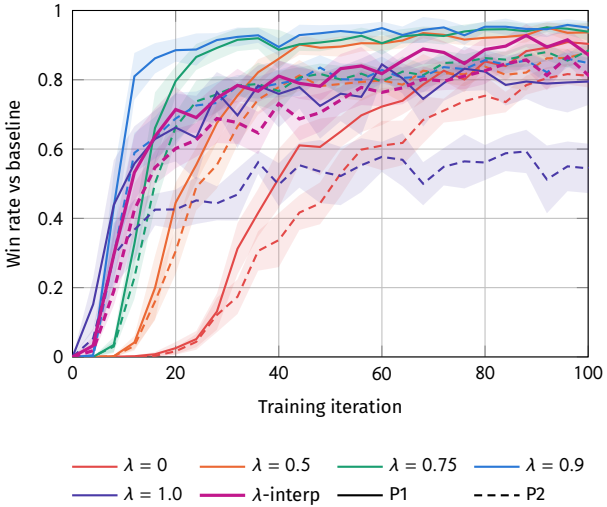


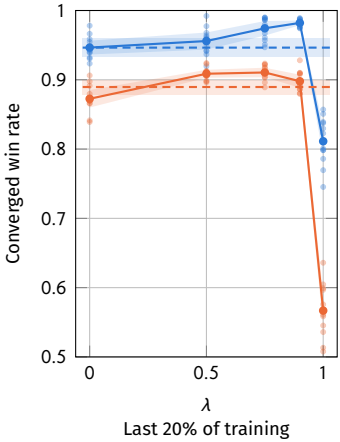
λ-interpolation: Win Rate (25-6)



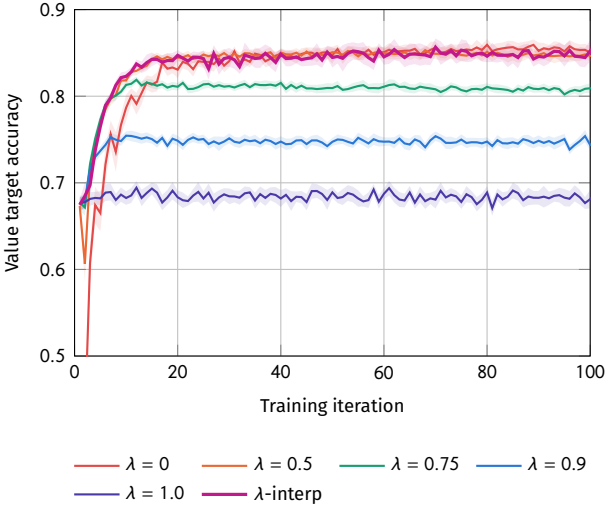


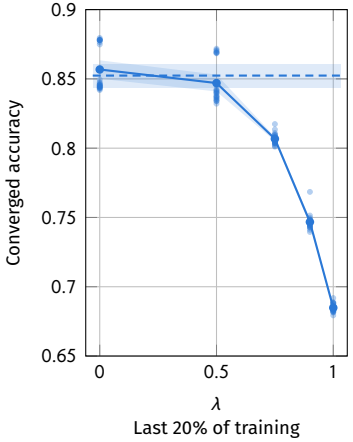
λ-interpolation: Win Rate (36-6)



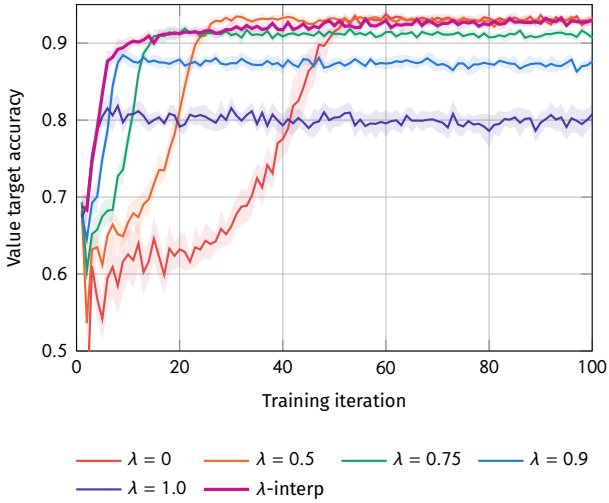


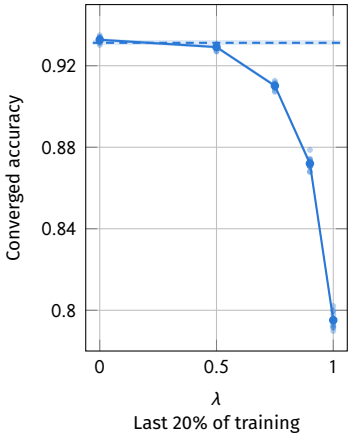
λ-interpolation: Value Target Accuracy (25-6)





λ-interpolation: Value Target Accuracy (49-4)





Conclusion: λ-interpolation

λ-interpolation gets TD(1)'s convergence speed *and* converges with the low-λ runs.

Conclusion

Introduction

AlphaZero with Ground Truth

AlphaZero with TD(λ)

Conclusion
- Contributions
- Future Work
- Acknowledgements
- Questions

Contributions

1. **An AlphaZero implementation for strongly solved Chinese Checkers, with a ground truth evaluation methodology.** Accuracy metrics for the value head, the policy head, and the training data.
2. **A diagnosis of why ground truth value targets prevented learning.** Imbalanced value targets and a brittle network led to majority-class collapse in the value head.
3. **A mitigation that makes alternative value targets viable.** Zero-initialization of the value head makes the network robust to imbalance in the training data.
4. **An evaluation of TD(λ) value targets in AlphaZero.** TD(λ) enables more accurate training data and a stronger final model at the cost of convergence speed compared to standard AlphaZero. λ-interpolation maintains both convergence speed and converged model accuracy.

Future Work

- **TD(λ)**
  - **Scope.** Do our results generalize? Repeating the experiment in other games and other AlphaZero implementations would tell us whether the TD(λ) result is general.
  - **Other value targets.** Our targets converge to the optimal behaviour policy, which accounts for exploration noise. Annealing the noise, or correcting for it with importance sampling or Tree-Backup(λ), could better aim at the optimal policy.
- **Given ground truth**
  - **The accuracy gap.** The value head is more accurate than its targets. Why?
  - **Where a trained agent is wrong.** With v_* and π_* for every state, we can inspect which states the player mis-evaluates or makes mistakes on.
  - **Non-transitivity and adversarial strategies.** Ground truth makes it possible to ask whether a cycle of exploiters is genuinely improving or just trading weaknesses.

Acknowledgements

Collaborators

Nathan Sturtevant, For his supervision
Zaheen Ahmad, For his help throughout

Committee

Nelson Amaral, Committee Chair
Nathan Sturtevant, Supervisor
Martin Müller, Examining Committee Member
James Wright, Examining Committee Member











Questions?

1. **An AlphaZero implementation for strongly solved Chinese Checkers, with a ground truth evaluation methodology.** Accuracy metrics for the value head, the policy head, and the training data.
2. **A diagnosis of why ground truth value targets prevented learning.** Imbalanced value targets and a brittle network led to majority-class collapse in the value head.
3. **A mitigation that makes alternative value targets viable.** Zero-initialization of the value head makes the network robust to imbalance in the training data.
4. **An evaluation of TD(λ) value targets in AlphaZero.** TD(λ) enables more accurate training data and a stronger final model at the cost of convergence speed compared to standard AlphaZero. λ-interpolation maintains both convergence speed and converged model accuracy.

Questions?

Appendix: Outline

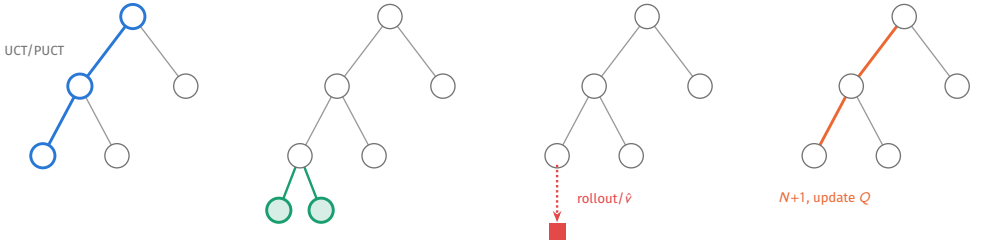
Introduction
- Motivation: Understanding AlphaZero
- Background: Chinese Checkers
- Background: AlphaZero
- Ground Truth from Solve Data
- Previous Work
- Our Implementation
- Contributions

AlphaZero with Ground Truth
- Motivation
- The Experiment
- Our Replication
- Investigation
- Solution

AlphaZero with TD(λ)
- Background: TD(λ)
- TD(λ) targets in AlphaZero
- Results: Accuracy, Win Rate, Targets
- Extension: λ-interpolation

Conclusion
- Contributions
- Future Work
- Acknowledgements
- Questions

Appendix: Monte Carlo Tree Search

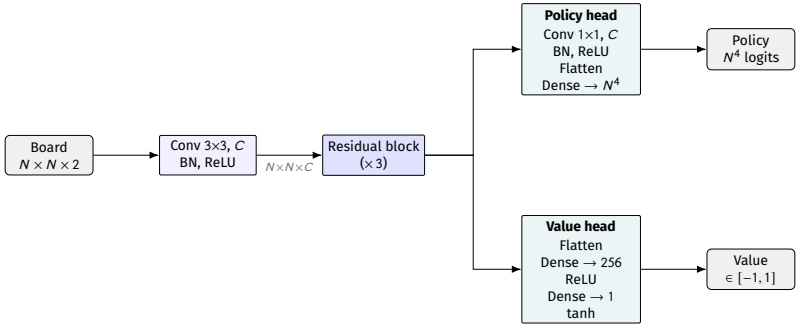


Four steps repeated until the search budget runs out. Selection uses PUCT, which balances exploitation against exploration and takes the policy head's prior:

$$\mathrm{PUCT}(s,a) = Q(s,a) + c\,\hat{\pi}_\theta(a \mid s)\frac{\sqrt{N(s)}}{1+N(s,a)}.$$

Coulom (2007); Kocsis & Szepesvári, UCT (2006); Rosin (2011).

Appendix: the network



Appendix: evaluation metrics

$$\mathrm{Acc}(\hat{v} \mid v_*, S) = \Pr_{s \sim S}\left[\hat{v}(s) = v_*(s)\right]$$

$$\mathrm{Acc}(\hat{\pi} \mid \pi_*, S) = \Pr_{s \sim S}\left[\operatorname*{argmax}_{a \in A(s)} \hat{\pi}(a|s) \in A^*(s)\right]$$

$$\mathrm{Acc}(\mathrm{SP} \mid v_*) = \Pr_{(s,\pi,z) \sim \mathrm{SP}}\left[z = v_*(s)\right]$$

- $\tilde{v}_\theta = +1$ if $\hat{v}_\theta \geq 0$, else -1.
- S^v: draws removed, classes balanced. S^π: states where every legal action is optimal removed.
- $\mathrm{Acc}(\mathrm{SP} \mid \cdot)$ scores a *list of samples*: the same state can appear twice with different targets.

Appendix: why the value head beats its training data

- Learning from noisy labels gives a reference point: the Bayes optimal classifier predicts, at each state, the most frequent target for that state in SP.
- If the model matches it, the model is just fitting the label distribution. If it exceeds it, some other structure in the data is doing the work.
- Initial evidence: both training data accuracy and value head accuracy are higher on endgame states, so generalization from the endgame is a candidate.
- Open question, and a good one to be asked about.

5 / 16

Appendix: majority-class collapse mitigations we considered

Subsampling What we used first. Discards data; infeasible when the minority class is a few percent.

Synthetic minority samples SMOTE interpolates between samples; interpolating two board states does not give a board state. Chinese Checkers symmetries give at most a factor of two, and our input representation already uses one of them.

Class weighting $w_i = N/(KN_i)$. Generalizes to continuous targets as weighted buckets.

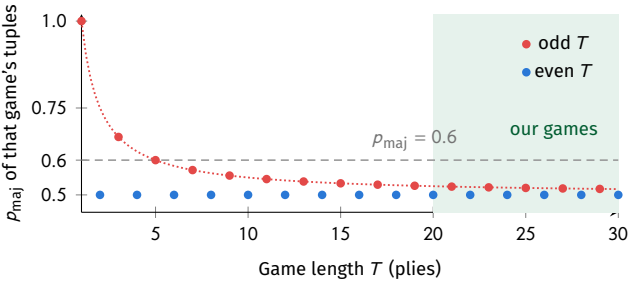
Architecture The original AlphaZero value head starts with a 1×1 convolution; ours does not, giving the first fully connected layer $256\times$ more input features. Reintroducing it is a plausible stability fix we did not need.

Retries Collapse only happens in the first few iterations, so restarting a collapsed seed is cheap. Stops working on large boards.

Zero-initialization What we settled on. Zero the weights of the value head's final linear layer, so it starts at $\tanh(0) = 0$ and cannot begin saturated. Eliminated collapse in every setting we tried.

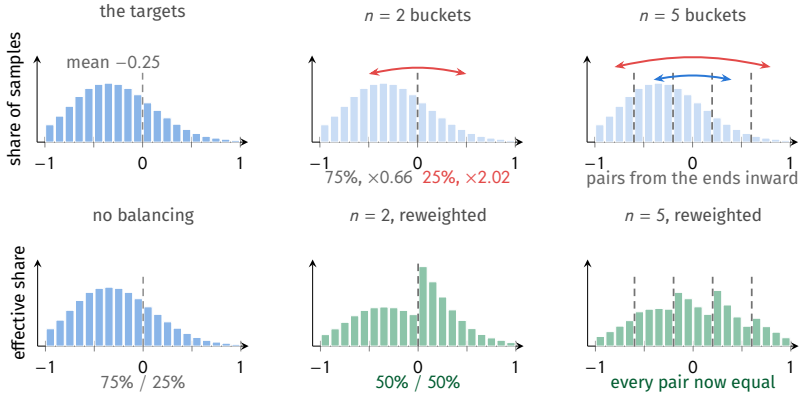
Other changes Tried alongside zero-initialization: more residual blocks, a separate backbone per head, gradient clipping, removing draws, and long games.

Appendix: p_{maj} against game length



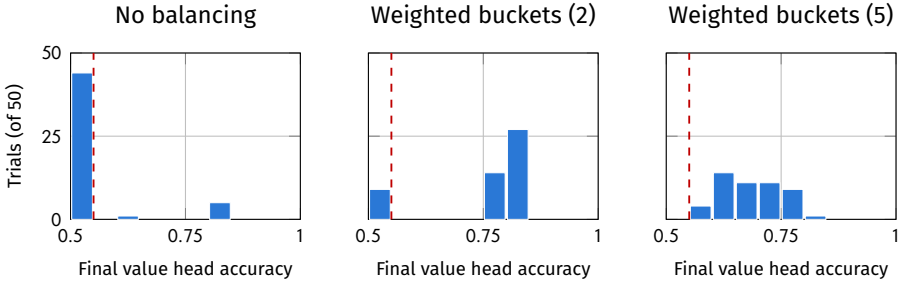
The majority-class fraction of the value targets in SP_i. Standard AlphaZero sits at $p_{\text{maj}} \approx 0.5$ whatever the game length, because one outcome is copied back onto alternating players. Ground truth targets start at ≈ 0.65 and settle at ≈ 0.55; collapse happens while $p_{\text{maj}} \approx 0.60$.

Appendix: weighted buckets



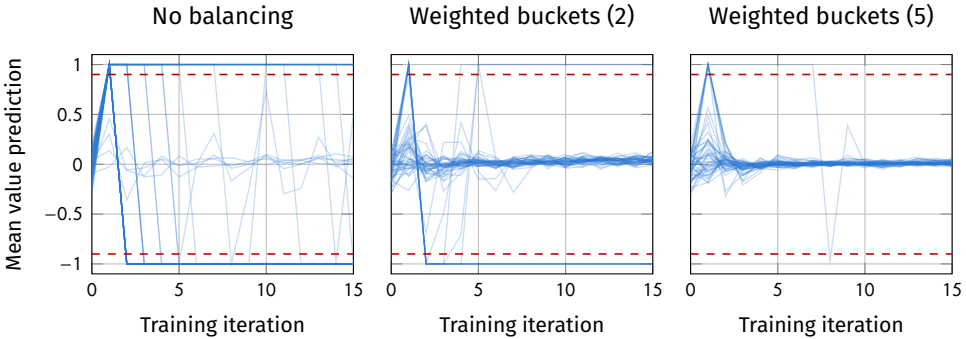
Class reweighting scales a sample's loss by the inverse of its class frequency, $w_j = N/(Cn_j)$. Weighted buckets generalizes it to a continuous target: bin, pair the bins from the ends inward, and weight bucket b, paired with b', by $w_b = (N_b + N_{b'})/(2N_b)$.

Appendix: weighted buckets help, but do not solve it



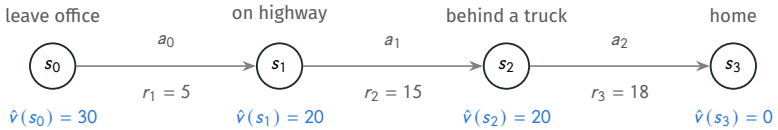
TD(0) on 25-6, 50 seeds each, at $n = 2$ and $n = 5$ buckets against an unbalanced control. 43/50 collapse with no balancing, 11/50 with 2 buckets, 9/50 with 5. Balancing slows learning. On 49-4, runs still collapse even with the balanced default targets, which is what sent us to the network instead.

Appendix: what collapse looks like while it happens



Mean value prediction per trial, $\lambda = 0$, 50 trials each. Lines pin to ± 1 within a few iterations; some flip sign, a few recover. In unrecoverable runs the pre-activation of tanh sits **saturated** – and TD(0)'s targets are the negation of the model's own prediction, a self-correcting signal it still cannot use.

Appendix: MC and TD(0) targets



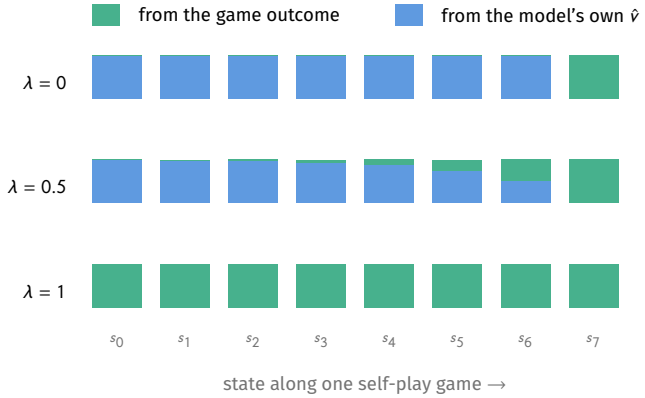
s_t where you are a_t what you do (here: keep driving) r_{t+1} minutes on that leg $\hat{v}(s_t)$ minutes still to go *(estimate)*

Monte Carlo: $G_0 = r_1 + r_2 + r_3 = 5 + 15 + 18 = 38$

TD(0): $r_1 + \hat{v}(s_1) = 5 + 20 = 25$

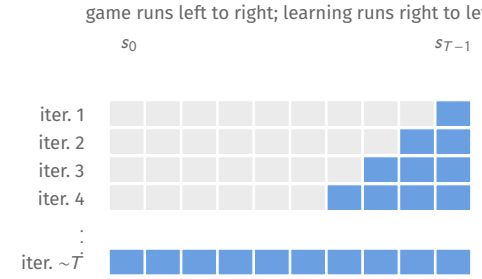
The driving-home example is Sutton & Barto, *Reinforcement Learning* (2018), Ch. 6.

TD(λ) Bootstrapping



Visualization of the amount of bootstrapping in the λ-return. The weight on the real outcome decays geometrically with distance from the end of the game.

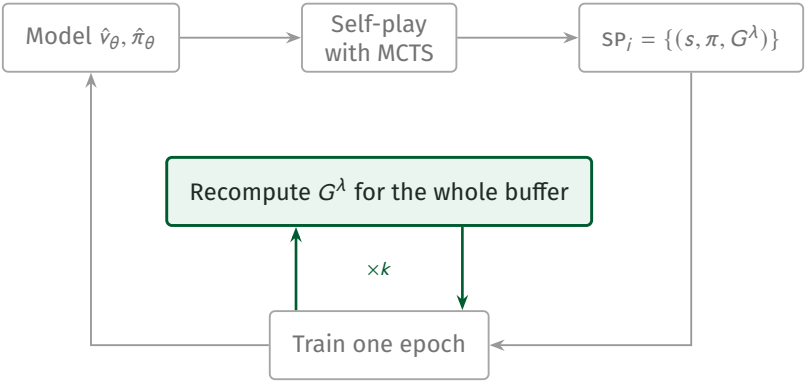
Appendix: why TD(0) converges slowly



One ply of credit per iteration, so $\sim T$ iterations to reach the opening.
Longer games (bigger boards) push this out further.

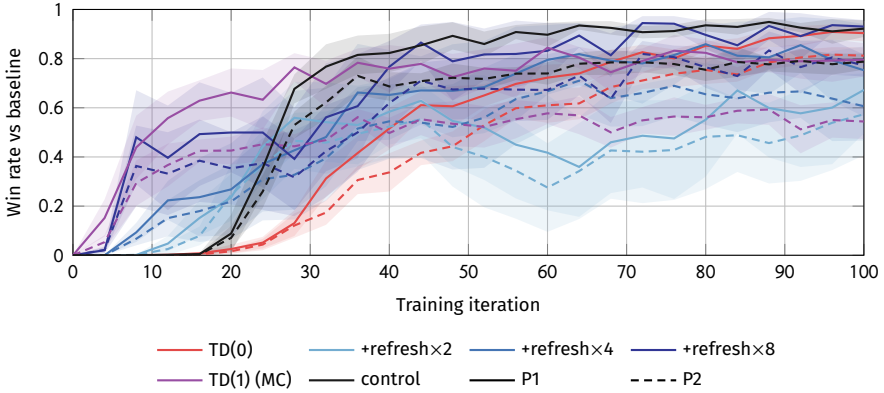
Each row is one training iteration; a filled cell is a state the value head has a meaningful target for. A state at distance d from the end is only learnable once $d - 1$ is, so the frontier moves one column per iteration and reaching the opening takes about T of them. Game length drives both halves of the tradeoff: it sets the credit propagation rate, and a TD(1) target's variance grows superlinearly with distance from T.

Appendix: TD(λ) with refreshes



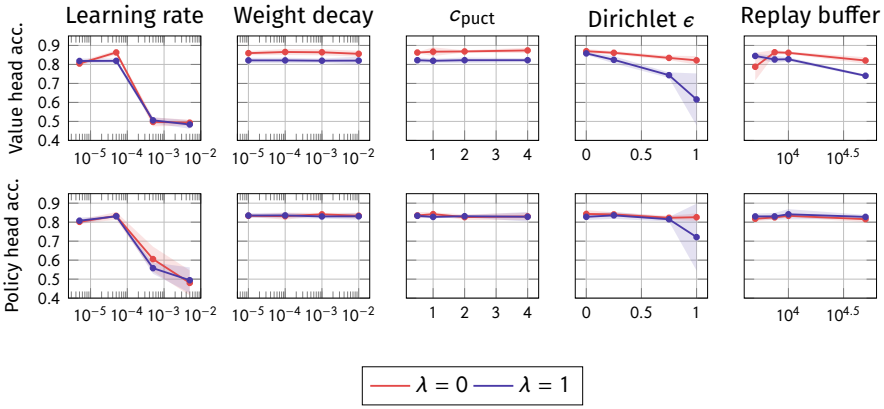
If slow convergence is slow credit propagation, do more than one propagation step per iteration: after each epoch, recompute G_t^λ for the whole replay buffer with the freshly trained model, k times.

Appendix: refreshes, win rate



36-6, P1 solid and P2 dashed. $k = 8$ converges about as fast as TD(1) while keeping TD(0)'s final win rate; the equal-compute control (8 passes, no refreshed targets) does not. **Empirical support for the credit propagation hypothesis.**

Appendix: the TD(λ) result across hyperparameters



Last-20% accuracy on the random set, value head above, policy head below. 50 iterations per configuration; bands are 95% CIs over 5 trials. The two λ settings track each other across all five sweeps, so the result is not an artefact of one configuration.