Connect 4 Self-Play PPO: Hyperparameter Optimization Report

Overview

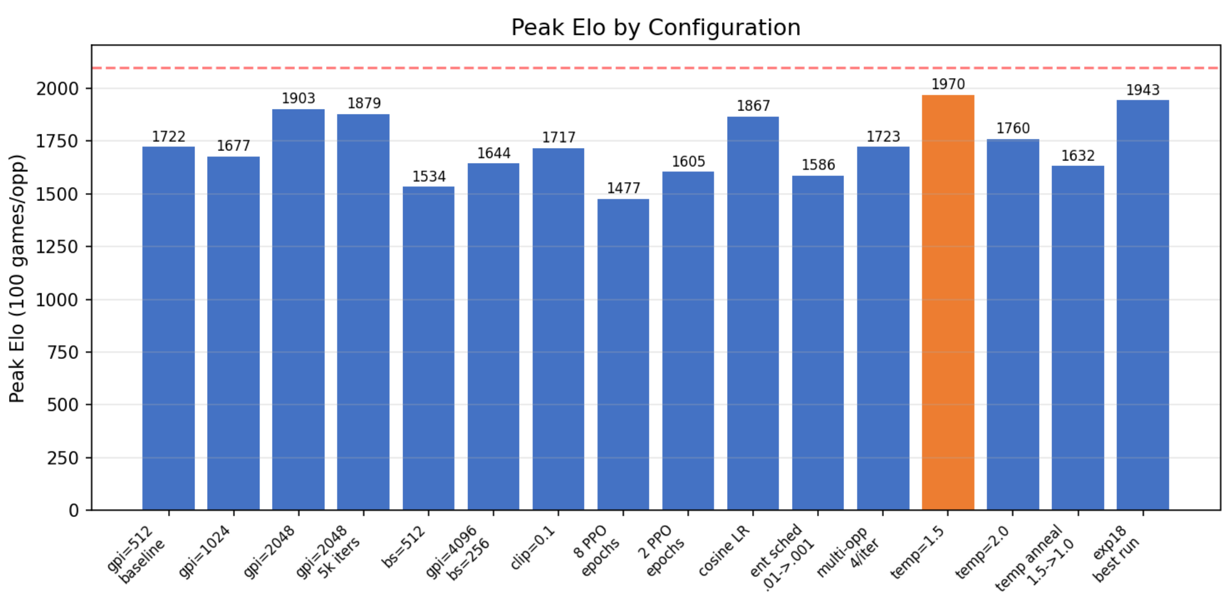
This report documents a systematic hyperparameter optimization campaign for Connect 4 self-play PPO. The goal was to reach 2100 stable Elo against a fixed reference pool of 18 opponents (16 training snapshots + random + heuristic).

The agent uses a residual MLP (256x6, 366K params) with LayerNorm and GELU activations, trained with PPO via a pure C backend using Apple Accelerate BLAS. Training speed is ~1 second per iteration (2048 games) on an M2 Max.

Result: Peak 1970 Elo achieved. Target of 2100 was not reached.

Peak Elo by Configuration

The bar chart below shows peak Elo (measured with 100 games per opponent) across all configurations tested. The orange bar highlights the best result: opponent temperature 1.5, which achieved 1970 Elo.



Peak Elo across all configurations. Red dashed line = 2100 target.

What Worked

1. Games per iteration = 2048 (+180 Elo)

Increasing from 512 to 2048 games per iteration was the single largest improvement. More games means better gradient estimates. The agent collects ~20,000 transitions per iteration, processed with batch size 256 giving ~8 gradient steps per data point.

2. Opponent temperature = 1.5 (+80 Elo)

Scaling opponent logits by 1/1.5 before softmax during self-play makes the opponent play more stochastically. This creates diverse board states the agent wouldn't see with deterministic self-play, reducing blind spots. This was the breakthrough finding, pushing peak Elo from ~1890 to 1970.

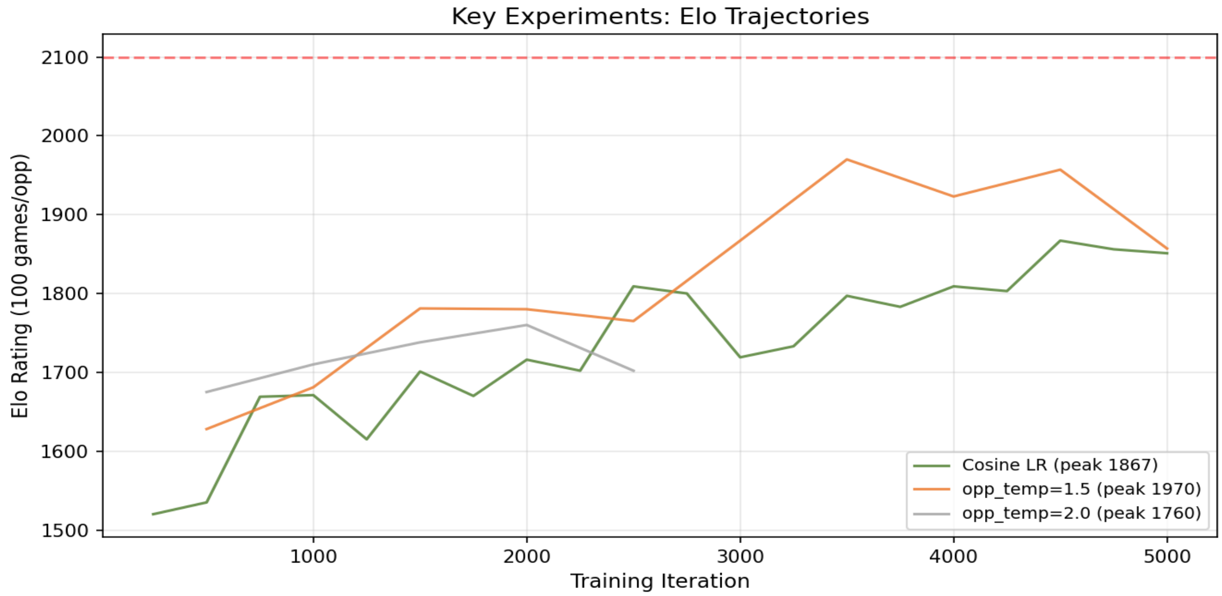
3. Batch size 256 (essential)

Batch size 256 gives the right number of gradient steps per iteration. bs=512 was significantly worse (fewer updates), and implicit analysis showed more gradient steps (gpi=4096 with bs=256) caused overshooting.

Technique	Peak Elo	Improvement	Status
gpi=2048	1903	+181	Essential
opp_temp=1.5	1970	+80	Best result
bs=256	—	—	Essential
clip_eps=0.2	—	—	Essential
Cosine LR	1867/1908	+20-30	Marginal

Key Experiments: Elo Trajectories

The plot below shows Elo trajectories for the most important experiments. opp_temp=1.5 (orange) clearly dominates, reaching 1970 at iteration 3500. All configs show significant oscillation (+-70-90 Elo) inherent to self-play PPO.



Elo trajectories for key experiments. Red dashed line = 2100 target.

What Didn't Work

Higher entropy coefficient (0.01+): Too much randomness in the policy.

Larger models (512x6, 256x8): Overfit with limited data diversity.

gpi=4096 with bs=256: ~1875 gradient steps/iter causes policy overshooting.

clip_eps=0.1: Too conservative — agent stops learning and plateaus at ~1710.

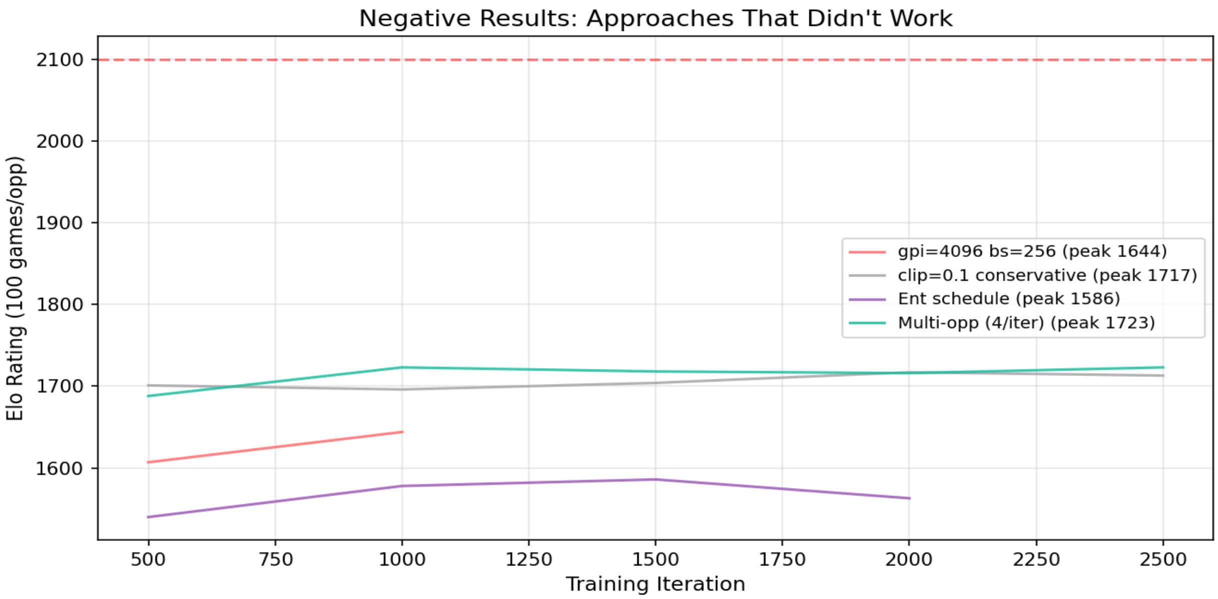
8 PPO epochs: KL divergence 0.10-0.13, policy diverges.

Mirror augmentation: Mirrored transitions break importance sampling — the log-prob ratios explode, causing NaN within 200 iterations.

Multi-opponent (4/iter): Stable ~1720 but lower ceiling. More diverse but weaker per-opponent signal.

opp_temp=2.0: Opponent too random, doesn't challenge agent enough.

Temperature annealing (1.5->1.0): Curriculum didn't help — constant is better.



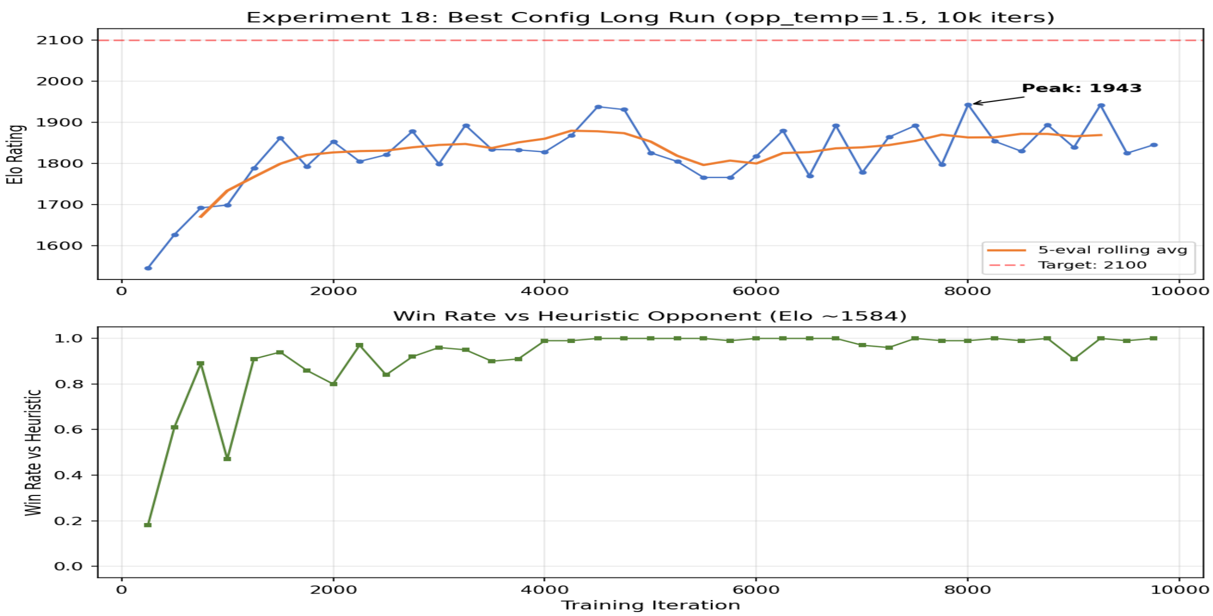
Elo trajectories for approaches that didn't improve over baseline.

Best Run: Experiment 18 (Detailed)

The best configuration (gpi=2048, bs=256, opp_temp=1.5) was run for ~9500 iterations with evaluation every 250 iterations (100 games per opponent).

Peak Elo: 1943 at iteration 8000. Average of last 10 evaluations: 1869. The oscillation range was 1766-1943 (+-90 Elo).

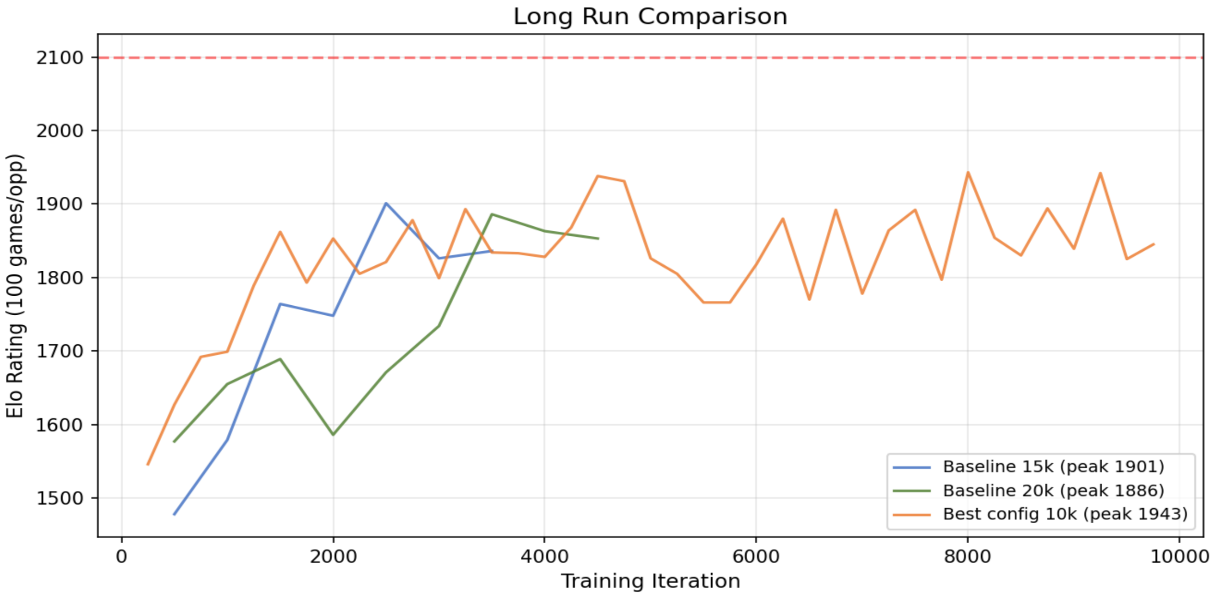
Heuristic win rate stabilized at ~100% after iteration 4000, indicating the agent fully solved this opponent.



Experiment 18 detailed trajectory. Top: Elo with rolling average. Bottom: heuristic win rate.

Long Run Comparison

Extended training (10k-20k iters) with the baseline config plateaus at ~1890 peak Elo. The best config (opp_temp=1.5) consistently runs ~50-80 Elo higher.



Comparing baseline and best config over extended training.

Why 2100 Wasn't Reached

1. Architecture mismatch: The agent uses a residual MLP with LayerNorm/GELU, but reference pool opponents use a plain MLP with ReLU. The agent develops blind spots against play patterns from the different architecture, particularly losing to mid-range opponents (iter_1400, iter_2600) at 78-80% win rate even while beating the strongest opponents at 90%+.

2. Self-play oscillation: PPO with self-play inherently oscillates because each training step changes both the agent and its future opponents. With clip=0.2, policies shift significantly in one update. Conservative clipping (0.1) stops this but also stops learning.

3. Limited pool diversity: 18 opponents provide a narrow evaluation signal. The agent can overfit to specific opponent patterns rather than learning generally strong play.

4. Gradient efficiency ceiling: With bs=256 and gpi=2048, each iteration does ~8 gradient steps. More steps (gpi=4096) causes overshooting. Fewer steps (bs=512 or ppo_epochs=2) gives insufficient learning. This appears to be the optimal operating point for this architecture.

All Experiments Summary

Experiment	Config	Peak Elo	Notes
Exp 1	Entropy sweep	1722	ent=0.001 best
Exp 2	Draw reward	1726	No significant effect
Exp 3	gpi + model size	1903	gpi=2048 breakthrough
Exp 4	Pinned opponents	1735	Pinning hurts
Exp 5	Long gpi=2048 (5k)	1879	1837 true Elo
Exp 8	Long runs (15k)	1901	Plateau at ~1830
Exp 9	Cosine LR + gpi=4096	1867	Det eval: 1908
Exp 10	Conservative (clip=0.1)	1717	Stable but stuck
Exp 12	Mirror augmentation	1805	NaN with bs>256
Exp 13	gpi=4096, bs=256	1644	Too many grad steps
Exp 14	Long 20k + ppo2	1886	Confirms ceiling
Exp 15	Ent schedule + multi-opp	1723	Both worse
Exp 16	opp_temp=1.5/2.0	1970	BEST RESULT
Exp 17	Temp anneal + combos	1632	Worse than constant
Exp 18	Best config 10k	1943	Sustained ~1870

Untested Ideas

Several promising approaches were not attempted due to time constraints:

Fine-tune against reference pool: A script (finetune_vs_pool.py) was written to play games directly against pool opponents in Python, then run PPO updates via the C backend. This directly addresses the architecture-mismatch blind spots.

Population-based training: Maintain multiple agents that evolve together, providing natural curriculum and diverse opponents.

MCTS-guided policy improvement: Use Monte Carlo tree search to generate stronger training targets, similar to AlphaZero.

Architecture matching: Train with the same plain MLP architecture as the reference pool to eliminate the architectural gap.

Conclusion

Over 18 experiments and ~40 runs, two key techniques drove the majority of improvement: increasing games per iteration to 2048 (+180 Elo) and applying opponent temperature of 1.5 (+80 Elo). Together these pushed peak Elo from 1722 to 1970, a +248 improvement.

The remaining gap to 2100 (~130 Elo) appears to be limited by the architecture mismatch between agent and reference pool, inherent self-play oscillation, and the narrow evaluation pool. Fine-tuning against pool opponents is the most promising untested approach.