Extended Abstract

1 Extended Abstract

We adapt AlphaZero to continuous control tasks and compare AlphaGo-style (human-initialized) versus AlphaZero-style (from-scratch) learning for real-world driving controls. Using a realistic driving simulator with GPT-based dynamics, we find that human-guided Monte Carlo Tree Search achieves 23% better performance than baseline PID control, while standard MCTS fails to converge due to exploration instability. Our results suggest human priors provide crucial safety constraints for learning in sensitive control environments.

Motivation While AlphaZero achieved superhuman performance in games by learning from scratch, real-world control tasks present different challenges. Human-centered environments like autonomous driving require safety constraints and behavioral preferences that are difficult to encode in reward functions alone. We investigate whether starting from human policies (like AlphaGo) outperforms learning from scratch (like AlphaZero) for continuous control tasks where exploration mistakes can cause system instability.

Method We compare three approaches on a realistic driving control task: a tuned PID controller as our human baseline, standard AlphaZero learning from scratch, and our AlphaGo-style method with human guidance. We use the Comma AI controls challenge, which simulates real vehicle dynamics with a GPT-based model trained on customer data. The task requires outputting steering torque to follow a desired trajectory while minimizing control effort. We adapt AlphaZero's Monte Carlo Tree Search to continuous actions using progressive widening and Gaussian action sampling.

Implementation Standard MCTS failed in the control environment due to action sensitivity in closed-loop systems. We developed two key modifications: guided exploration that samples actions around the PID policy using Gaussian noise, and guided rollouts that use PID for descendant nodes while exploring alternatives only at the root. These changes provide stable value estimates while preserving the ability to discover policy improvements. We train neural networks to distill the MCTS search results using cross-entropy loss for the policy and mean squared error for the value function.

Results Human-guided MCTS planning achieved 23% lower cost than PID baseline, while standard MCTS diverged with extremely high variance. After learning neural policies from MCTS exploration, AlphaGo Control significantly outperformed standard AlphaZero across 5,000 evaluation rollouts. Ablation studies confirmed that both guided action sampling and guided rollouts are essential for stable learning. Without action sampling, policies learned oscillatory behavior. Without guided rollouts, policies became overly conservative due to high-variance value estimates.

Discussion Our findings indicate that human demonstrations provide crucial constraints for safe exploration in sensitive control environments. Unlike discrete games with well-defined rules, continuous control systems require careful balance between exploration and safety. The exploration-exploitation dilemma is particularly challenging when deviating from stable policies leads to rapid error accumulation. Human priors naturally encode this safety knowledge, enabling stable learning where pure exploration fails. This suggests fundamental differences between game and control domains for reinforcement learning.

Conclusion We demonstrate that AlphaGo-style human initialization outperforms AlphaZero-style learning from scratch for real-world control tasks. Human priors provide essential safety constraints and behavioral guidance that enable stable learning in sensitive environments. As reinforcement learning moves from games to real-world applications, incorporating human knowledge may be necessary for safe and efficient learning in safety-critical domains like autonomous driving and robotics.

From Game-Playing to Self-Driving: Comparing AlphaGo vs AlphaZero Approaches for Driving Controls

Ellen Xu
Department of Computer Science
Stanford University
ellenjxu@stanford.edu

Abstract

In this work, we investigate approaches comparing AlphaGo-like methods (human initialization followed by reinforcement learning) with AlphaZero-like methods (learning from scratch) for control tasks in human-centered environments. While AlphaZero achieved superior performance in Go without human data, we hypothesize that for real-world control tasks, human policies can encode safety constraints and behavioral priors difficult to capture in reward functions alone. We evaluate these approaches on a realistic driving simulator, using a PID controller as our human-level baseline. Our results show that human-guided Monte Carlo Tree Search (MCTS) achieves 23% higher rewards than baseline PID control, while standard MCTS fails to converge due to exploration instability. We explore two key components for stable convergence: guided action sampling and guided rollouts. These findings suggest that human priors may provide crucial constraints for safe and efficient learning in real-world reinforcement learning applications.

2 Introduction

Deep reinforcement learning has achieved superhuman performance in games and robotics. In the game of Go, AlphaZero ultimately outperformed AlphaGo by learning completely from self-play, while AlphaGo first trained on human expert games before reinforcement learning. However, for real-world domains like autonomous driving or industrial robotics which are inherently human-centered, the optimal learning strategy remains largely unexplored.

Does bootstrapping from human policies outperform learning from scratch in human-centered control environments?

We hypothesize that real-world control tasks benefit from human initialization because human policies inherently encode safety constraints and preferred behaviors that are difficult to specify through reward functions alone. Unlike games which have well-defined rules and rewards, real-world control systems often operate in noisy, safety-critical environments. This leads to constraints on exploration which can hinder learning for reinforcement learning systems.

In this work, we investigate this hypothesis by comparing AlphaGo-style methods (human-guided approaches) with AlphaZero-style methods (from-scratch learning) on a realistic driving control task. Specifically, we evaluate steering control using the Comma AI controls challenge, which simulates real-world vehicle dynamics with a GPT-based model trained on customer data.

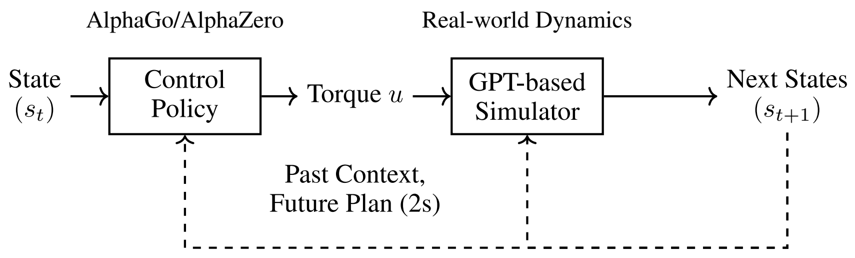


Figure 1: Control challenge architecture.

3 Related Work

3.1 AlphaZero and AlphaGo

AlphaZero has achieved state-of-the-art, super-human performance in Chess, Shogi, and the game of Go (Moerland et al., 2018; Schrittwieser et al., 2020; Silver et al., 2017). The key innovation of AlphaZero is the augmentation of Monte Carlo tree search (MCTS) with a neural network generated policy. The augmented policy can learn to generalize from self-play, and progressively bootstraps the network from random initialization up to superhuman play, without requiring extra human input. AlphaGo, in contrast, first learned from human expert games through supervised learning before improving via self-play reinforcement learning.

3.2 AlphaZero for Continuous Control

Recent work has extended AlphaZero beyond discrete games. AlphaZero Continuous (A0C) adapted the algorithm to continuous action spaces by replacing self-play with simulation and using progressive widening for action expansion (Moerland et al., 2018). Moss et al. (2024) further generalized these methods to partially observable environments.

In autonomous driving, Hoel et al. (2019) applied AlphaGo Zero for discrete highway lane changes. Cusumano-Towner et al. (2025) demonstrated large-scale self-play reinforcement learning for driving policies, by simulating copies of its own agent. These works illustrate the core ideas of AlphaZero – self-play, MCTS search, and neural network learning – can be applied to self-driving. However, the approaches focus on high-level planning rather than low-level continuous control, which presents unique challenges including noise sensitivity and stability requirements.

4 Method

We compare three approaches: (1) a tuned PID controller as our human baseline, (2) AlphaZero learning from scratch, and (3) AlphaGo with human guidance.

4.1 Baseline: PID Controller

We use a Proportional-Integral-Derivative (PID) controller as our human-level policy baseline. PID is typically used in industrial control systems, and is a useful heuristic for closed loop feedback control. The controller parameters ($K_p = 0.3, K_i = 0.05, K_d = -0.1$) were tuned for the environment task.

4.2 AlphaZero Implementation

We developed a parallelized AlphaZero Continuous (A0C) implementation combining MCTS with neural network learning. Our implementation follows the A0C framework by integrating progressive widening into MCTS to handle the continuous action space. In the base MCTS algorithm, the number of children m at each state s is determined by:

$$m = \lfloor k \cdot N_s^{\alpha} \rfloor \tag{1}$$

where N_s is the visit count for state s, k is a constant that controls the rate of expansion, and $\alpha \in (0, 1)$ determines how quickly the width grows with visits. This progressive widening technique prevents the tree from becoming too sparse in continuous domains. The PUCT formula for action selection at the root node remains

$$a^* = \arg\max_{a \in A(s)} \left(Q(s, a) + c_{puct} \cdot \frac{\sqrt{N_s}}{1 + N_{(s,a)}} \right) \tag{2}$$

where c_{puct} is the exploration weight, N_s is the visit count for state s, and $N_{(s,a)}$ is the visit count for state-action pair (s, a).

Our implementation differs from Moerland et al. (2018) since we use Q-value estimates rather than normalized visit counts to construct the target policy, and we employ a softmax temperature to control the sharpness of the distribution. The policy network learns from MCTS search results using cross-entropy loss:

$$\mathcal{L}_{\text{policy}} = \mathbb{E}_{s \sim \mathcal{D}} \left[-\sum_{a \in \mathcal{A}} \pi_{\text{MCTS}}(a|s) \log \pi_\theta(a|s) \right]$$

where $\pi_{\text{MCTS}}(a|s) = \text{softmax}(Q(s, a)/\tau)$ is the target policy derived from Q-values with temperature τ, and $\pi_\theta(a|s)$ is the neural network policy. The temperature parameter prevents overconfident target policies when Q-value estimates have high variance.

The value network is trained to predict expected returns minimizing mean squared error:

$$\mathcal{L}_{\text{value}} = \mathbb{E}_{s \sim \mathcal{D}} \left[(V_\phi(s) - z)^2 \right]$$

where z is the discounted return from MCTS rollouts and $V_\phi(s)$ is the value network prediction.

The policy and value network are distilled from the MCTS exploration. The networks enable generalization and improves MCTS efficiency by (1) reducing search breadth (policy network biases exploration toward promising actions) and (2) reducing search depth (truncating rollouts and using value network leaf node estimates).

4.3 AlphaGo Control

Standard MCTS planning performed poorly in the control environment due to action sensitivity in closed-loop feedback systems. Search depths beyond 2-3 steps caused rapid error accumulation, leading to extreme Q-value estimates and system divergence (Figure 5). We experimented with two key modifications to address the instability:

Guided Exploration: Instead of initializing with sampling from a random policy, we sample actions using Gaussian exploration around the PID policy output:

$$a_t \sim \mathcal{N}(\pi_{\text{PID}}(s_t), \sigma^2)$$

where π_{PID} is the PID controller policy and σ^2 is the action variance, controlling the amount of exploration around the selected action.

Rationale: While discrete AlphaZero uses policy priors $\pi_\theta(a|s)$ as probability weights for action selection, continuous action spaces have unbounded probability densities that cannot be directly used as weights. Our method achieves similar bias toward promising actions by sampling from a Gaussian centered on the expert policy, ensuring exploration remains within a reasonable action region while preserving the ability to discover improvements.

Guided Rollouts: We replace random simulation rollouts with PID-guided rollouts. At the root node, we explore alternative actions using the action sampling above, but for all descendant nodes, we follow the PID policy:

$$a_t \sim \begin{cases} \mathcal{N}(\pi_{\text{PID}}(s_t), \sigma^2) & \text{if root node} \\ \pi_{\text{PID}}(s_t) & \text{otherwise} \end{cases}$$

Rationale: This formulation is well-suited for policy improvement because: (1) it provides low-variance value estimates by leveraging the known PID policy for rollouts, in comparison with Monte Carlo rollouts which are high-variance, and (2) it enables direct policy improvement by comparing the value of alternative actions against the current policy, following the standard Q-learning paradigm where $\pi'(s) = \arg\max_a Q^{\pi}(s, a)$ yields policy improvement when $Q^{\pi}(s, a)$ estimates are accurate.

These modifications enable more stable policy improvement through $Q^{\pi}(s, a)$ estimates. The approach preserves MCTS's ability to discover improved policies, while avoiding the exploration instability in standard tree search.

5 Experimental Setup

5.1 Controls Challenge Environment

The Comma AI controls challenge[1] is a realistic driving controls environment for evaluating controls algorithms. The goal is to output torque to steer a car along a desired trajectory, while minimizing control effort. To simulate the car's steering responses, a GPT-based simulator trained on real-world noisy dynamics from customer vehicles is used.

The controller receives ego vehicle state $v_{\text{ego}}, a_{\text{ego}}, roll$, current and desired acceleration $a_{\text{current}}, a_{\text{desired}}$, and 2 seconds of future plan states and past context. The controller output u is continuous steering torque corresponding to a steering wheel input.

The controls challenge captures three critical aspects of real-world RL: (1) high-dimensional continuous state-action spaces, (2) sensitive dynamics where errors accumulate rapidly, and (3) realistic noise from a simulator trained on real vehicle data. We use this domain as a toy problem for evaluating exploration strategies in safety-critical control domains.

5.2 Problem Formulation

The Controls Challenge can be formulated as a finite-horizon MDP:

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, T, R, \gamma, T_h\},$$

where $\mathcal{S} \subseteq \mathbb{R}^{n_s}$ is the continuous state space, $\mathcal{A} \subseteq \mathbb{R}^{n_a}$ is the continuous action space, $T(s' \mid s, a)$ represents the GPT-based transition model, and $R(s, a)$ is the deterministic reward function:

$$R(s, a) = -\left|a_{\text{lateral}}(s, a) - a_{\text{desired}}\right| - \lambda_u \|a\|$$

The objective is to find the optimal policy for steer actions which maximizes expected cumulative reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{T_h - 1} \gamma^t R(s_t, a_t)\right].$$

5.3 Network Architecture

For all our experiments, actor and critic networks use multi-layer perceptrons with [256, 256] hidden layers and ReLU activation. The input dimension is 304 dimensions, consisting of the environment state as well as PID terms (error, integral, and previous error). The actor outputs a Gaussian policy with learnable log standard deviation. We use 30 training iterations with 10 epochs each and 200 steps, for a total of 60K training steps.

More implementation details and hyperparameters are provided in the Appendix Figure 3.

6 Results

Our results are evaluated on the Controls Challenge with a fixed set of 5,000 total evaluation trajectories. Our primary metric is the average reward, defined as a weighted sum of lateral acceleration error and jerk cost.

[1]Comma AI control challenge, `https://github.com/commaai/controls_challenge`

6.1 Planning Performance

We first evaluate the planning capabilities of different MCTS variants. Human-guided MCTS achieved 23% lower cost than the PID baseline, while standard MCTS failed to converge due to exploration instability (Table 1).

Table 1: Planning method comparison. Results averaged over 10 evaluation rollouts (200 steps each) with search depth $d = 3$ and simulations $n = 10$. MCTS planning requires 8× longer execution time per training step (0.627s vs 0.075s for PID baseline), highlighting the tradeoff between planning accuracy and computational efficiency.

Method	Total Cost	±Std	Runtime/step
PID (baseline)	80.17	95.04	**0.075s**
Standard MCTS	6443.13	4657.55	0.605s
Human-guided MCTS (ours)	**61.70**	**42.64**	0.627s

6.2 Policy Learning Results

After distilling MCTS exploration into neural networks, we evaluated the learned policies over 5,000 rollouts. AlphaGo Control significantly outperformed the standard AlphaZero (Table 2).

Table 2: Policy evaluation results averaged over 5,000 rollouts. AlphaZero training used 100 workers and 30 iterations.

Method	Total Cost	Lat Accel Cost	Jerk Cost
Standard AlphaZero	4887.31	97.22	**26.52**
AlphaGo Control (ours)	**197.73**	**3.38**	29.00

7 Quantitative Analysis

7.1 Learning Curves

Figure 2 shows that AlphaGo Control achieves stable learning with consistent reward improvement, while standard AlphaZero fails to converge. This may be due to the exploration-exploitation dilemma, where deviating from stable policies leads to error accumulation, preventing learning from random initialization.

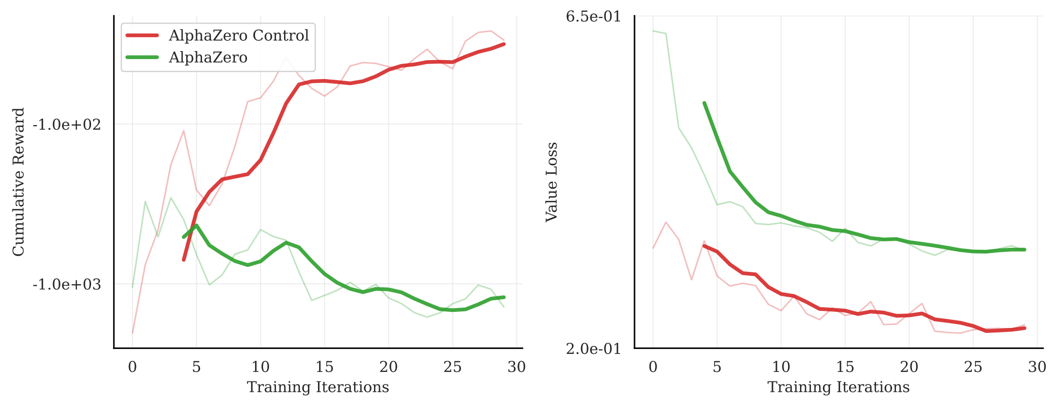


Figure 2: Learning curves comparing AlphaGo Control (human-guided) versus standard AlphaZero (from scratch). Left: episode rewards during training showing stable improvement for AlphaGo Control versus diverging returns for AlphaZero. Right: value network loss convergence.

7.2 Ablation Studies

We validated our design choices through ablation studies removing (A) human-guided action sampling and (B) guided rollouts.

Figure 3 shows that both components are essential to stable learning. Without human-guided exploration or rollouts, the network fails to learn higher rewards throughout training. The value network also does not converge to a low loss.

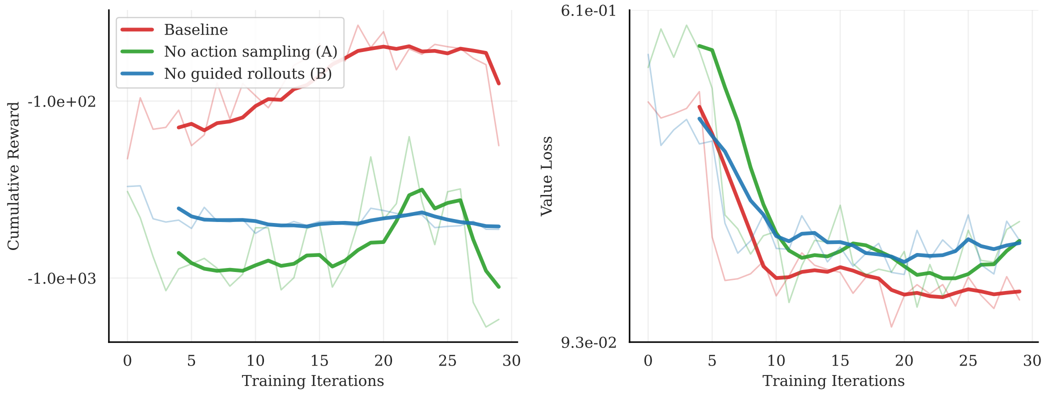
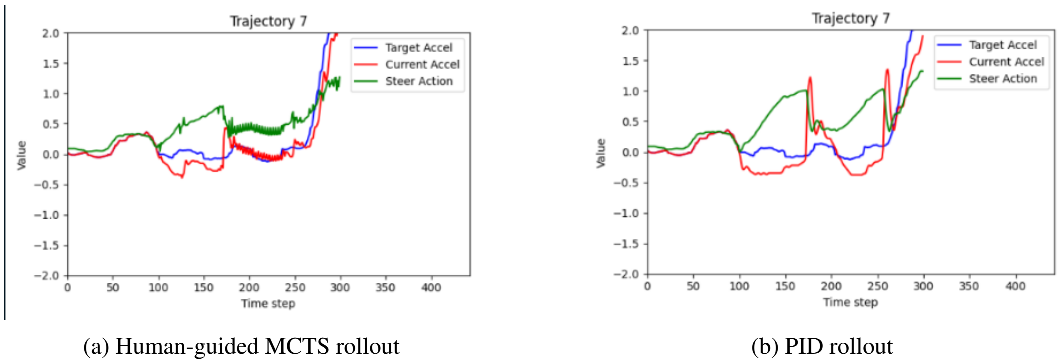


Figure 3: Learning curves for ablation studies. Without action sampling (red), random exploration fails to discover improvements in the large continuous action space. Without guided rollouts (blue), high-variance returns prevent effective value network learning, resulting in flat performance curves.

8 Qualitative Analysis

Analyzing the rollout trajectories, guided MCTS planning leads to improved trajectories compared to the PID baseline. Guided MCTS exploration around PID actions can lead to discovering better actions. This improvement is particularly evident during rapid acceleration changes, where PID tends to overshoot 4.



(a) Human-guided MCTS rollout

(b) PID rollout

Figure 4: Comparison of control trajectories. Control actions are in green, goal trajectory is in blue and actual trajectory in red.

Interestingly, ablation A (no action sampling) learns oscillatory control behavior, while ablation B (no guided rollouts) learns conservative policies that produce minimal control actions (Figure 6). The oscillatory behavior may be due to poor initial actions from a randomly initialized policy. The conservative behavior in ablation B suggests that high-variance value estimates may lead to similar Q-values across actions, causing the policy to default to low-magnitude control outputs to minimize immediate penalties. Furthermore, the difference in policy behaviors suggest multiple local optima exist, and human priors can help select for behaviorally optimal solutions.

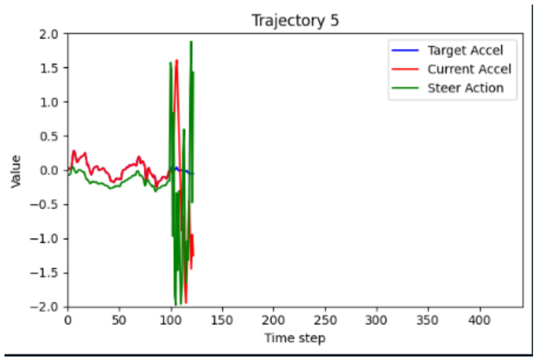


Figure 5: MCTS divergence without stability constraints. Initial deviations in control action (green) lead to outputs (red) and actions exploding after just a few steps.

9 Discussion

Our results suggest that human demonstrations can provide crucial constraints for safe exploration in sensitive control environments. While standard MCTS diverges rapidly in the control setting, human-guided search achieves stable improvement with 23% better performance than baseline PID control. We demonstrate in ablations two key components for stable convergence: (1) guided exploration with action sampling and (2) guided rollouts. These components were used in the AlphaGo training framework to train a control policy, which showed greater stability over standard AlphaZero learning from scratch.

This finding suggests that human priors can be effective for safe exploration in real-world control tasks. Unlike games with well-defined rules, control systems require careful balance between exploration and safety in order to achieve stable learning, which human priors may naturally provide. While we used a PID controller as our human policy, this could be replaced with other heuristics or initializing a policy learned through imitation learning from human data. These can extend beyond driving controls to other human-centered domain where safety and exploration in large continuous state spaces is important.

Our work provides preliminary evidence that the AlphaGo paradigm (human initialization followed by reinforcement learning improvement) may be more suitable for real-world control applications than pure self-play approaches. However, there are several limitations to our work. First, we focus on a single control task with limited scope and complexity. Second, computational constraints restricted our MCTS search depth, potentially limiting the full potential of from-scratch learning methods. Third, our comparison uses a relatively simple PID baseline rather than more sophisticated human policies. Future work aims to scale up search and generalize to other tasks to better evaluate from scratch and human-guided methods.

10 Conclusion

In this work, we investigated whether human-guided reinforcement learning approaches outperform learning from scratch in real-world control environments. Through experiments on a realistic driving control task, we demonstrated that AlphaGo-style methods (human initialization followed by reinforcement learning) achieve superior performance compared to AlphaZero-style approaches (learning from scratch). Our human-guided MCTS achieved 23% better performance than PID baseline control, while standard MCTS failed to converge due to exploration instability in the sensitive control environment. The AlphaGo Control approach leveraging human-guided MCTS learns a controller successfully whereas traditional AlphaZero from scratch fails to converge. We identified two critical components for stable learning: guided action sampling around human policies and guided rollouts that leverage human priors for low-variance value estimation.

11 Acknowledgements

I would like to acknowledge Robert Moss for his valuable guidance and feedback, as well as Mykel Kochenderfer and the Stanford Intelligent Systems Laboratory (SISL) for their support and providing the compute for this project.

References

Marco Cusumano-Towner, David Hafner, Alex Hertzberg, Brody Huval, Aleksei Petrenko, Eugene Vinitsky, Erik Wijmans, Taylor Killian, Stuart Bowers, Ozan Sener, Philipp Krähenbühl, and Vladlen Koltun. 2025. Robust Autonomy Emerges from Self-Play. arXiv:2502.03349 [cs.LG] `https://arxiv.org/abs/2502.03349`

Carl-Johan Hoel, Katherine Driggs-Campbell, Krister Wolff, Leo Laine, and Mykel J. Kochenderfer. 2019. Combining Planning and Deep Reinforcement Learning in Tactical Decision Making for Autonomous Driving. `https://doi.org/10.48550/arXiv.1905.02680` arXiv:1905.02680.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. 2018. A0C: Alpha Zero in Continuous Action Space. `http://arxiv.org/abs/1805.09613` arXiv:1805.09613 [stat].

Robert J. Moss, Anthony Corso, Jef Caers, and Mykel J. Kochenderfer. 2024. BetaZero: Belief-State Planning for Long-Horizon POMDPs using Learned Approximations. `http://arxiv.org/abs/2306.00249` arXiv:2306.00249 [cs].

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. 2020. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *Nature* 588, 7839 (Dec. 2020), 604–609. `https://doi.org/10.1038/s41586-020-03051-4` arXiv:1911.08265 [cs].

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. `http://arxiv.org/abs/1712.01815` arXiv:1712.01815 [cs].

A Additional Experiments

We provide additional trajectory visualizations for the ablation studies in Figure 6, and AlphaGo Control rollouts in Figure 7.

B Implementation Details

B.1 Training Hyperparameters

Key hyperparameters include: learning rate 1×10^{-3}, batch size 10,000, 100 parallel workers, 30 training iterations, MCTS depth 3, 10 simulations per action, action sampling variance 0.1, and discount factor $\gamma = 0.99$. Full details are provided in Table 3.

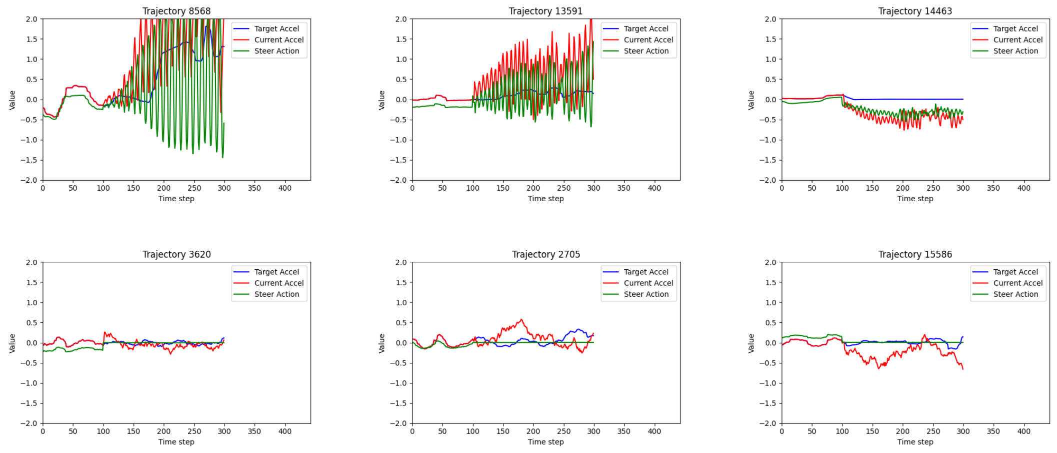


Figure 6: Sampled trajectories for ablations. Top row: ablation A (no action sampling) produces oscillatory behavior. Bottom row: ablation B (no guided rollouts) produces near-zero control output.

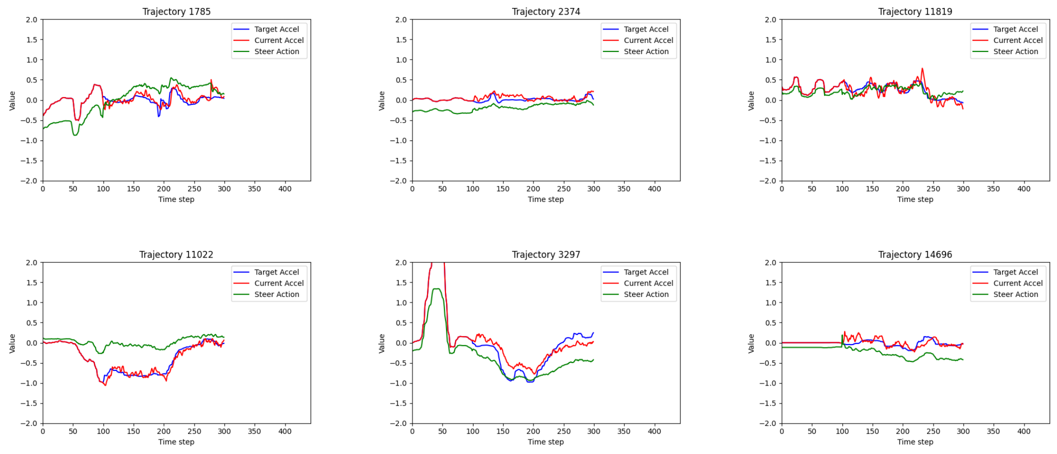


Figure 7: 6 sampled trajectories for AlphaGo Control learned policy. The controller is able to properly track lateral acceleration, as shown in the red and blue trajectories.

Table 3: Hyperparameters used in AlphaGo Control training

Parameter	**Default Value**	**Description**
Training Parameters		
Max iterations	30	Maximum training iterations
Episode steps	200	Steps per episode during rollout
Workers	100	Number of parallel tree workers
Learning rate	1×10^{-3}	Adam optimizer learning rate
Training epochs	10	Epochs per iteration
Batch size	10,000	Training batch size
Replay buffer size	100,000	Maximum replay buffer capacity
Network Architecture		
Hidden sizes	[256, 256]	Hidden layer dimensions for actor/critic
Log std	-2	Initial log standard deviation for actor
L2 regularization	1×10^{-4}	L2 penalty coefficient
Value loss weight	0.5	Weight for value function loss
MCTS Parameters		
Exploration weight	0.1	UCB exploration constant
Discount factor (γ)	0.99	Reward discount factor
MCTS depth	3	Search tree depth
MCTS simulations	10	Number of simulations per action
Action variance	0.1	Variance for action sampling
UCB parameter (k)	2	UCB formula parameter
Dirichlet alpha (α)	0.5	Dirichlet noise parameter
Policy Training		
Temperature (τ)	1	Softmax temperature for policy targets
Entropy coefficient	0.001	Entropy regularization weight
Evaluation		
Evaluation episodes	5	Episodes for evaluation during training
Evaluation steps	200	Steps per evaluation episode