Regret-Regularized Deep Policy Learning in Repeated Games

Praneel Varshney [1] **Eshan Singhal** [1]

Abstract

Classical no-regret algorithms such as Multiplicative Weights and Regret Matching provably achieve sublinear external regret and converge to equilibrium in repeated games. In contrast, deep policy gradient methods lack such guarantees and often exhibit instability or cycling behavior even in simple games. We investigate whether regret-based principles can be reintroduced into deep policy optimization through explicit regularization. We augment policy gradient objectives with a regret-matching regularizer based on cumulative counterfactual regret, creating a hybrid approach that combines the flexibility of neural policies with the principled structure of classical game-theoretic learning. In systematic experiments across stateless repeated games against both static and adaptive opponents, we demonstrate that regret-regularized policy gradient (RRPG) achieves 93-94% lower external regret and substantially reduced exploitability compared to vanilla policy gradient methods, while maintaining stable action probabilities. Our results suggest that explicit incorporation of regret-based objectives can guide deep reinforcement learning toward no-regret-like behavior without abandoning gradient-based optimization.

1. Introduction

In multi-agent learning and game-playing scenarios, no-regret algorithms have long been the theoretical gold standard. Algorithms such as Multiplicative Weights (Freund & Schapire, 1999), EXP3 (Auer et al., 2002), and Regret Matching (Hart & Mas-Colell, 2000) provide strong theoretical guarantees: they achieve sublinear external regret $R_T = o(T)$ and, when employed in self-play, converge to coarse correlated equilibria. These guarantees make them particularly attractive for adversarial or non-stationary environments where the opponent's strategy may be unknown or evolving.

Meanwhile, deep reinforcement learning has achieved remarkable empirical success across complex domains through policy gradient methods (Sutton et al., 1999; Schulman et al., 2017). By parameterizing policies with neural networks and optimizing via gradient ascent, these approaches can scale to high-dimensional state and action spaces that are intractable for tabular methods. However, policy gradient methods typically lack regret guarantees and can exhibit pathological behaviors in adversarial settings. Even in simple two-player games, gradient-based optimization may cycle between strategies indefinitely, fail to converge to equilibrium, or produce highly exploitable policies.

This disconnect raises a fundamental question: *Can we bridge the gap between the theoretical guarantees of no-regret learning and the scalability and flexibility of deep policy gradient methods?*

In this work, we take a first step toward answering this question by investigating whether regret-based structure can be reintroduced into policy gradient optimization through principled loss regularization. Specifically, we study stateless repeated games where external regret is well-defined and tractable to compute exactly. We augment the standard policy gradient objective with a regularization term that encourages the learned policy to match the action distribution prescribed by Regret Matching Plus (RM+), a classical no-regret algorithm. This creates a hybrid approach that maintains the flexibility of neural network policies while incorporating the principled structure of regret minimization.

Our approach, which we call *Regret-Regularized Policy Gradient (RRPG)*, adds a cross-entropy term to the policy gradient loss that pulls the neural policy toward the RM+ target distribution computed from cumulative counterfactual regret. This regularizer acts as a complementary learning signal to the standard reward-based gradient, biasing the policy toward actions that minimize past regret while still allowing gradient-based adaptation to the opponent's strategy.

[1]Department of Computer Science, University of Pennsylvania, Philadelphia, PA, USA. Correspondence to: Praneel Varshney <pvarsh@seas.upenn.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

1.1. Contributions

We make the following contributions:

- We propose regret-regularized policy gradient (RRPG), which augments standard policy gradient losses with a cross-entropy regularizer toward the RM+ target distribution computed from cumulative counterfactual regret (Section 3).
- We conduct systematic experiments across two zero-sum games (Matching Pennies and Rock-Paper-Scissors) against five opponent types ranging from static (deterministic, biased, uniform random) to adaptive (Multiplicative Weights, EXP3), with 20 random seeds per configuration.
- We demonstrate that RRPG achieves 93-94% lower average external regret compared to vanilla policy gradient against adaptive opponents, while also reducing exploitability by 12-20% and exhibiting more stable action probabilities (Section 5).
- We analyze several design variants including alternative functional forms (exponential, softmax), discounted regret accumulation, and progressive regularization schedules, providing insights into what makes regret regularization effective (Section 6).
- We provide evidence that the benefit of regret regularization is most pronounced against adaptive opponents where vanilla policy gradient exhibits cycling and instability, while performance against static opponents remains comparable (Section 5.2).

Our results suggest that explicit regret-based objectives can guide deep RL toward no-regret-like behavior without abandoning the scalability and flexibility of gradient-based optimization. While we focus on stateless games for tractability, this approach opens avenues for incorporating regret-based principles into deep RL methods for more complex multi-agent scenarios.

2. Background

2.1. Repeated Games and External Regret

We study stateless, two-player repeated games. At each round $t = 1, \ldots, T$, an agent selects an action $a_t \in \mathcal{A} = \{1, \ldots, n\}$ and observes the opponent's action $b_t \in \mathcal{A}$. The agent receives payoff $u(a_t, b_t)$ according to a fixed payoff matrix $A \in \mathbb{R}^{n \times n}$, where $u(a, b) = A[a, b]$. In zero-sum games, the opponent receives $-u(a_t, b_t)$.

The game is *stateless* in the sense that payoffs depend only on the current action pair, not on any underlying state or history. This simplification allows us to focus on the core challenge of learning good strategies in adversarial settings without the added complexity of state representation or partial observability.

Definition 2.1 (External Regret)**.** The external regret at time T measures the difference between the cumulative payoff of the best fixed action in hindsight and the agent's actual cumulative payoff:

$$R_T = \max_{a \in \mathcal{A}} \sum_{t=1}^{T} u(a, b_t) - \sum_{t=1}^{T} u(a_t, b_t). \tag{1}$$

External regret quantifies how much better the agent could have done by committing to a single action from the start, given full knowledge of the opponent's action sequence. An algorithm is *no-regret* if $R_T = o(T)$, meaning average regret $R_T/T \to 0$ as $T \to \infty$. No-regret algorithms provide a strong guarantee: regardless of the opponent's strategy (even adversarially chosen), the algorithm's time-averaged performance approaches that of the best fixed action in hindsight.

2.2. Classical No-Regret Algorithms

Multiplicative Weights (MW). Multiplicative Weights is a fundamental no-regret algorithm that maintains a weight $w_t(a)$ for each action, initialized to 1. After observing opponent action b_t, weights are updated multiplicatively based on observed payoffs:

$$w_{t+1}(a) = w_t(a) \exp\left(\eta \cdot u(a, b_t)\right), \tag{2}$$

where $\eta > 0$ is a learning rate. At each round, actions are sampled with probability proportional to their normalized weights: $\pi_t(a) = w_t(a) / \sum_{a'} w_t(a')$.

With appropriate choice of learning rate $\eta = \mathcal{O}(\sqrt{\log n / T})$, MW achieves external regret $R_T = \mathcal{O}(\sqrt{T \log n})$ (Arora et al., 2012). The exponential weighting scheme gives MW an adaptive property: actions that have performed well in the past receive exponentially higher probability, while poorly-performing actions are exponentially down-weighted.

Regret Matching (RM+). Regret Matching takes a more direct approach by explicitly tracking cumulative regret for each action. The cumulative regret for action a at time t is:

$$R_t(a) = \sum_{s=1}^{t} \left[u(a, b_s) - u(a_s, b_s)\right], \tag{3}$$

where $R_t(a)$ measures how much better action a would have performed compared to the actions actually played. Regret Matching Plus (RM+) uses these regret values directly to compute the policy:

$$\pi_{t+1}(a) \propto \max(R_t(a), 0). \tag{4}$$

The max operation clips negative regrets to zero, ensuring valid probabilities while naturally down-weighting actions that have performed poorly relative to what was played. RM+ achieves $R_T = \mathcal{O}(\sqrt{T})$ and has been particularly successful in imperfect-information games, forming the basis of Counterfactual Regret Minimization (CFR) (Zinkevich et al., 2007).

EXP3. EXP3 (Auer et al., 2002) extends the multiplicative weights framework to the bandit setting, where only the payoff of the chosen action is observed. It combines exponential weights with uniform exploration via a mixing parameter γ. While EXP3 is designed for limited feedback, we include it as a baseline to verify our implementations and demonstrate the performance gap between full-information and bandit algorithms.

2.3. Policy Gradient Methods

Policy gradient algorithms optimize a parameterized policy $\pi_\theta(a)$ via gradient ascent on expected cumulative reward. In the REINFORCE algorithm (Williams, 1992), the policy gradient is estimated as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[r_t \nabla_\theta \log \pi_\theta(a_t)\right], \tag{5}$$

where r_t is the observed reward. In practice, a baseline b_t is subtracted from r_t to reduce variance without introducing bias: $\hat{r}_t = r_t - b_t$, where b_t is typically an exponential moving average of past rewards.

In stateless repeated games, we parameterize the policy as a softmax over neural network outputs:

$$\pi_\theta(a) = \frac{\exp(f_\theta(a))}{\sum_{a'} \exp(f_\theta(a'))}, \tag{6}$$

where $f_\theta : \mathbb{R}^d \to \mathbb{R}^n$ is a multilayer perceptron (MLP). Since the game is stateless, the input to the network is a constant vector (we use a single scalar input of 1).

Limitations in adversarial settings. While policy gradient methods have achieved impressive results in cooperative and single-agent settings, they struggle in adversarial environments. Without explicit regret minimization or equilibrium-seeking behavior, these methods may:

- **Cycle indefinitely**: Action probabilities oscillate without converging to a stable distribution.
- **Fail to reach equilibrium**: Even when converged, policies may be far from Nash equilibrium.
- **Become exploitable**: Adaptive opponents can identify and exploit weaknesses in the learned policy.

Standard entropy regularization $-\lambda_H H(\pi_\theta)$ encourages exploration by penalizing overly deterministic policies, but does not provide regret guarantees or equilibrium-seeking behavior. This motivates our approach of augmenting policy gradient with explicit regret-based structure.

3. Method: Regret-Regularized Policy Gradient

We propose to bridge the gap between policy gradient methods and no-regret learning by augmenting the policy gradient objective with an explicit regret-matching regularizer. Our approach, Regret-Regularized Policy Gradient (RRPG), maintains the flexibility of neural network policies while incorporating the principled structure of RM+.

3.1. Algorithm Overview

RRPG consists of three key components working in concert:

1. **Policy network:** A small MLP f_θ producing logits for each action, with policy $\pi_\theta(a) = \text{softmax}(f_\theta)$.
2. **Regret tracking:** Cumulative regret $R_t(a)$ is tracked exactly as in RM+, assuming full-information feedback.
3. **Composite loss:** The training objective combines three terms: policy gradient (REINFORCE), regret-matching regularization, and optional entropy regularization.

The key innovation is the regret-matching regularizer, which provides a complementary learning signal to the standard reward-based gradient. While the policy gradient term encourages actions that have yielded high rewards, the regret regularizer encourages the policy to match the distribution prescribed by RM+ based on cumulative counterfactual reasoning.

3.2. Regret Computation

At each round t, we observe our action a_t, opponent action b_t, and received payoff $r_t = u(a_t, b_t)$. We update cumulative regret for each action $a \in \mathcal{A}$:

$$R_t(a) \leftarrow R_{t-1}(a) + u(a, b_t) - u(a_t, b_t). \tag{7}$$

This update is identical to that used in RM+ and requires full knowledge of counterfactual payoffs $u(a, b_t)$ for all actions a. In our setting (stateless repeated games with known payoff matrices), these counterfactuals are immediately available. The regret $R_t(a)$ accumulates positive values when action a would have outperformed what we actually played, and negative values otherwise.

3.3. Regret-Matching Target Distribution

We compute a target distribution $q_t(a)$ following the RM+ prescription:

$$q_t(a) = \frac{\max(R_t(a), 0)}{\sum_{a'} \max(R_t(a'), 0)}. \tag{8}$$

If all regrets are non-positive (no action would have done better in hindsight), q_t defaults to the uniform distribution. This target distribution represents the action probabilities that RM+ would prescribe based on current cumulative regret. Actions with high positive regret receive high probability, while actions with negative or zero regret are down-weighted.

3.4. Loss Function

The complete training loss at round t combines three terms:

$$\begin{aligned}\mathcal{L}_t(\theta) = & -\hat{r}_t \log \pi_\theta(a_t) \\ & - \lambda_R \sum_a q_t(a) \log \pi_\theta(a) \\ & - \lambda_H H(\pi_\theta),\end{aligned} \tag{9}$$

where:

- **Policy gradient term**: $-\hat{r}_t \log \pi_\theta(a_t)$ is the standard REINFORCE objective with baseline-adjusted reward $\hat{r}_t = r_t - b_t$. The baseline b_t is an exponential moving average: $b_t = \alpha b_{t-1} + (1-\alpha) r_t$ with momentum $\alpha = 0.9$.
- **Regret-matching regularizer**: $-\lambda_R \sum_a q_t(a) \log \pi_\theta(a)$ minimizes the cross-entropy (equivalently, forward KL divergence) between the current policy π_θ and the RM+ target q_t. This pulls the neural policy toward the distribution prescribed by classical regret matching.
- **Entropy regularization**: $-\lambda_H H(\pi_\theta) = \lambda_H \sum_a \pi_\theta(a) \log \pi_\theta(a)$ encourages exploration by penalizing overly deterministic policies. We set $\lambda_H = 0.01$ for all methods (both vanilla PG and RRPG) to ensure fair comparison.

The hyperparameter λ_R controls the strength of regret regularization. When $\lambda_R = 0$, we recover vanilla policy gradient with entropy regularization. When λ_R is large, the policy is strongly constrained to follow RM+, potentially at the expense of slower adaptation to the opponent's strategy.

Why cross-entropy (forward KL)? The cross-entropy term $-\sum_a q_t(a) \log \pi_\theta(a)$ is equivalent to minimizing the forward KL divergence $D_{\mathrm{KL}}(q_t \| \pi_\theta)$. This encourages π_θ to place probability mass wherever q_t has mass, while allowing π_θ to be more diffuse (spread probability more evenly). This is preferable to reverse KL $D_{\mathrm{KL}}(\pi_\theta \| q_t)$, which would force π_θ to be zero wherever q_t is zero—an overly restrictive constraint when q_t may have structural zeros due to clipping negative regrets.

Algorithm 1 Regret-Regularized Policy Gradient (RRPG)

Input: Game with payoff matrix A, learning rate η, regularization weights λ_R, λ_H, baseline momentum α
Initialize policy network π_θ with random weights θ
Initialize regret vector $R(a) = 0$ for all actions a
Initialize baseline $b = 0$
for $t = 1$ to T **do**
  Sample action $a_t \sim \pi_\theta$
  Observe opponent action b_t and receive payoff $r_t = A[a_t, b_t]$

  // Update regret (full-information)
  for each action $a \in \mathcal{A}$ **do**
    $R(a) \leftarrow R(a) + A[a, b_t] - A[a_t, b_t]$
  end for

  // Compute RM+ target distribution
  $Z \leftarrow \sum_{a'} \max(R(a'), 0)$
  if $Z > 0$ **then**
    $q(a) \leftarrow \max(R(a), 0)/Z$ for all a
  else
    $q(a) \leftarrow 1/|\mathcal{A}|$ for all a *// uniform*
  end if

  // Update policy via gradient descent
  $\hat{r}_t \leftarrow r_t - b$
  $H \leftarrow -\sum_a \pi_\theta(a) \log \pi_\theta(a)$
  $\mathcal{L} \leftarrow -\hat{r}_t \log \pi_\theta(a_t) - \lambda_R \sum_a q(a) \log \pi_\theta(a) - \lambda_H H$
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$

  // Update baseline
  $b \leftarrow \alpha b + (1-\alpha) r_t$
end for

3.5. Complete Algorithm

Algorithm 1 presents the complete RRPG algorithm. The algorithm alternates between (1) selecting actions according to the current policy, (2) observing outcomes and updating regret, (3) computing the RM+ target distribution, and (4) updating the policy network via gradient descent on the composite loss.

3.6. Hyperparameter Choices

We use a small MLP with two hidden layers of 64 units each, ReLU activations, and the Adam optimizer (Kingma & Ba, 2014) with learning rate $\eta = 10^{-3}$. The network architecture is intentionally simple to focus on the effect of regret regularization rather than representational capacity.

We set entropy coefficient $\lambda_H = 0.01$ for all methods (both vanilla PG and RRPG) to ensure fair comparison. This value provides mild exploration without dominating the learning signal.

The key hyperparameter is the regret-matching coefficient λ_R. We conduct a sweep over $\lambda_R \in \{0.05, 0.1, 0.2, 0.5, 1.0, 2.0\}$ and find that $\lambda_R = 0.1$ provides the best tradeoff across all settings (see Section 6.1). This value is strong enough to provide meaningful regularization but not so strong as to prevent adaptation to opponent strategies.

3.7. Design Variants

In addition to the standard RM+ formulation, we explore several functional variants of the regret-matching target:

RM-NoClip. Instead of clipping negative regrets to zero, we shift all regrets to make them non-negative:

$$q_t(a) \propto R_t(a) - \min_{a'} R_t(a'). \tag{10}$$

This preserves relative differences between actions while ensuring valid probabilities.

Softmax-RM. We apply a temperature-scaled softmax to regrets:

$$q_t(a) \propto \exp(R_t(a)/\tau). \tag{11}$$

This is similar to the MW update but applied to cumulative regrets rather than weights. The temperature τ controls the sharpness of the distribution: low τ makes the distribution more peaked on the action with highest regret, while high τ makes it more uniform.

Discounted regret. We weight recent observations more heavily via exponential discounting:

$$R_t(a) \leftarrow \gamma R_{t-1}(a) + [u(a, b_t) - u(a_t, b_t)] \tag{12}$$

with discount factor $\gamma \in \{0.95, 0.99\}$. This may be useful in non-stationary environments where the opponent's strategy is changing over time.

Progressive schedules. We decay λ_R over time, starting with strong regularization and gradually transitioning to vanilla policy gradient:

$$\lambda_R(t) = \lambda_R^{\text{init}} \cdot \text{schedule}(t/T), \tag{13}$$

where schedule is either linear decay $(1 - t/T)$ or exponential decay $\exp(-\alpha t/T)$.

These variants are analyzed in Section 6.

4. Experimental Setup

4.1. Games

We study two canonical zero-sum games that are simple enough for exact regret tracking and Nash computation, yet rich enough to exhibit cycling and instability in gradient-based methods:

Matching Pennies. Each player independently chooses Heads or Tails. The row player (our agent) wins if actions match; the column player (opponent) wins if they differ. Payoff matrix for the row player:

$$A = \begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix}.$$

The unique Nash equilibrium is for both players to play uniformly random (50-50), achieving expected value 0 for both players. Any deterministic or biased strategy is exploitable.

Rock-Paper-Scissors (RPS). Standard RPS where Rock beats Scissors, Scissors beats Paper, and Paper beats Rock. Payoff matrix for the row player:

$$A = \begin{bmatrix} 0 & -1 & +1 \\ +1 & 0 & -1 \\ -1 & +1 & 0 \end{bmatrix}.$$

As in Matching Pennies, the unique Nash equilibrium is uniform random (1/3 each action), achieving expected value 0. RPS introduces an additional degree of freedom compared to Matching Pennies and may exhibit more complex cycling dynamics.

Both games are symmetric and zero-sum, meaning optimal play requires randomization. Any deviation from uniform random can be exploited by a best-response opponent.

4.2. Opponents

We evaluate against five opponent types spanning a spectrum from static to adaptive:

Static opponents:

- **Deterministic:** Always plays action 0 (Heads in Matching Pennies, Rock in RPS).
- **Biased:** Plays a fixed non-uniform distribution. We use 60% action 0, 20% action 1, 20% action 2 (if applicable).

- **Uniform Random:** Plays uniformly at random. This opponent is unexploitable but also does not adapt.

Adaptive opponents:

- **Multiplicative Weights (MW):** Uses the classical MW algorithm with learning rate $\eta = 0.1$.
- **EXP3:** Bandit algorithm with $\eta = 0.1$ and exploration parameter $\gamma = 0.1$.

Against static opponents, any learning algorithm should eventually learn to exploit the opponent's fixed strategy. The interesting dynamics arise against adaptive opponents, where both players are learning simultaneously. This creates a non-stationary environment where cycling and instability are common for naive gradient-based methods.

4.3. Algorithms

We compare the following algorithms:

- **MW (classical):** Multiplicative Weights with $\eta = 0.1$, serving as the gold-standard no-regret baseline.
- **EXP3 (classical):** Bandit setting baseline with $\eta = 0.1, \gamma = 0.1$, included to verify implementations.
- **PG-Vanilla:** Policy gradient with entropy regularization ($\lambda_H = 0.01$) but no regret regularization ($\lambda_R = 0$).
- **PG-RM+ λ_R:** Our proposed RRPG with regret-matching coefficient $\lambda_R \in \{0.05, 0.1, 0.2, 0.5, 1.0, 2.0\}$ and entropy coefficient $\lambda_H = 0.01$.
- **Variants:** RM-NoClip, Softmax-RM (with $\tau \in \{0.5, 1.0\}$), Discounted-RM (with $\gamma \in \{0.95, 0.99\}$), and Progressive schedules (linear and exponential decay). See Section 6.

All neural network-based methods (PG-Vanilla and all RRPG variants) use identical architectures (2-layer MLP with 64 hidden units, ReLU activations) and optimization settings (Adam with learning rate 10^{-3}) to ensure fair comparison.

4.4. Evaluation Metrics

Primary metric: Average external regret. We report average external regret R_T/T, which should approach zero for no-regret algorithms. This is our main metric for algorithm performance, as it directly captures the no-regret property.

Exploitability (zero-sum games). Exploitability measures how much a best-response opponent can gain beyond the Nash value. For a row player policy π:

$$\text{Exploit}_{\text{row}}(\pi) = \max_{b \in \Delta(\mathcal{A})} \mathbb{E}_{a \sim \pi}[u(a, b)] - v^*, \qquad (14)$$

where $v^* = 0$ is the Nash value in our games and $\Delta(\mathcal{A})$ is the simplex of probability distributions over actions. Lower exploitability indicates the policy is closer to Nash equilibrium and harder to exploit.

Time-averaged exploitability. Following the equilibrium computation literature, we also report the exploitability of the time-averaged policy $\bar{\pi}_T = \frac{1}{T}\sum_{t=1}^{T} \pi_t$. This metric is particularly relevant for equilibrium convergence, as no-regret algorithms guarantee that time-averaged play converges to equilibrium.

Diagnostic metrics:

- **Policy entropy:** $H(\pi_t) = -\sum_a \pi_t(a) \log \pi_t(a)$ tracks exploration vs. exploitation over time.
- **Action probabilities:** Visualizing $\pi_t(a)$ over time reveals cycling or convergence patterns.
- **Instant regret:** Cumulative regret if we had played a best response at each round (upper bound on achievable performance).

4.5. Experimental Protocol

Each experiment runs for $T = 10{,}000$ rounds. We average results over 20 random seeds to account for stochasticity in both network initialization and opponent randomness. We report mean and standard error (standard deviation divided by $\sqrt{20}$) across seeds.

For steady-state analysis, we compute metrics over the final $K = 1{,}000$ rounds, which we refer to as the "evaluation window." This captures performance after initial transients have settled.

All experiments were conducted on standard CPUs. Total wall-clock time for the full benchmark (2 games $\times$ 5 opponents $\times$ 18 algorithms $\times$ 20 seeds = 3,600 runs) was approximately 11 hours distributed across two machines.

5. Results

5.1. Main Results: Adaptive Opponents

Table 1 quantifies performance against adaptive opponents using metrics from the final 1000 rounds (steady-state behavior).

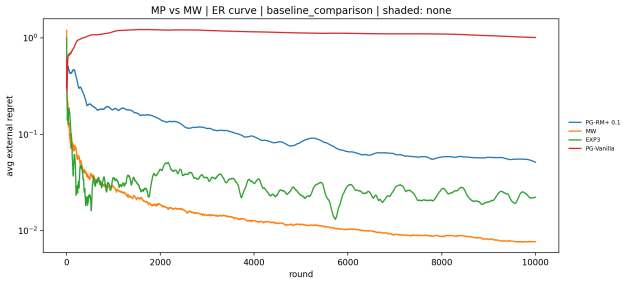

Figure 1. Average external regret over time for Matching Pennies vs. MW opponent.

Table 1. Performance against adaptive opponents (final 1000 rounds, mean ± std over 20 seeds). Lower is better for both metrics.

Algorithm	Avg Regret	Exploitability
Matching Pennies vs. MW		
MW	0.01 ± 0.00	0.95 ± 0.02
PG-Vanilla	1.03 ± 0.71	0.97 ± 0.08
PG-RM+ 0.1	$\mathbf{0.06 \pm 0.02}$	$\mathbf{0.85 \pm 0.12}$
Rock-Paper-Scissors vs. MW		
MW	0.01 ± 0.00	0.94 ± 0.05
PG-Vanilla	0.86 ± 0.73	0.89 ± 0.14
PG-RM+ 0.1	$\mathbf{0.06 \pm 0.02}$	$\mathbf{0.71 \pm 0.07}$

Against MW in Matching Pennies, classical MW achieves average regret of 0.01 ± 0.00 as expected from theory. PG-Vanilla accumulates substantial regret (1.03 ± 0.71) and has high exploitability (0.97 ± 0.08), indicating failure to converge to equilibrium. The high standard deviation reflects instability across different random seeds—some runs perform better than others, but none match the theoretical guarantees of classical methods.

RRPG with $\lambda_R = 0.1$ dramatically reduces regret to 0.06 ± 0.02, a **94% improvement over vanilla PG**. Exploitability also decreases to 0.85 ± 0.12, a 12% improvement. While RRPG does not match MW's theoretical guarantees (which would require $O(\sqrt{T \log n}/T) \approx 0.003$ average regret at $T = 10{,}000$), it substantially narrows the gap between vanilla neural policy gradient and classical no-regret methods.

Results for Rock-Paper-Scissors show similar patterns but with even more pronounced benefits. PG-Vanilla achieves high regret (0.86 ± 0.73) and exploitability (0.89 ± 0.14). The three-action space appears to exacerbate cycling behavior in vanilla PG. PG-RM+ reduces regret to 0.06 ± 0.02 (**93% improvement**) and exploitability to 0.71 ± 0.07 (20% improvement).

The consistent pattern across both games demonstrates that regret regularization provides substantial benefits against adaptive opponents. The reduction in exploitability is particularly important, as it indicates that RRPG policies are much harder to exploit and closer to Nash equilibrium than vanilla PG.

Table 2. Performance against static opponents (final 1000 rounds). All methods eventually exploit static strategies.

Algorithm	Avg Regret	Exploitability
Matching Pennies vs. Deterministic (always Heads)		
MW	0.0006 ± 0.0003	1.00 ± 0.00
PG-Vanilla	0.0006 ± 0.0002	1.00 ± 0.00
PG-RM+ 0.1	0.0004 ± 0.0002	1.00 ± 0.00
Matching Pennies vs. Biased (60% Heads)		
MW	0.001 ± 0.0005	0.998 ± 0.001
PG-Vanilla	0.023 ± 0.070	0.941 ± 0.176
PG-RM+ 0.1	0.001 ± 0.0004	0.998 ± 0.002

5.2. Results Against Static Opponents

Against deterministic and biased opponents, the dynamics are fundamentally different. Static opponents do not adapt, so the learning problem reduces to exploiting a fixed strategy.

Table 2 shows final-window performance against static opponents.

Against the deterministic opponent (always plays Heads), all methods achieve near-zero regret. MW, PG-Vanilla, and PG-RM+ all learn to play the optimal counter-strategy (always Tails in Matching Pennies) with high probability. The small non-zero regret is due to exploration (entropy regularization).

Against the biased opponent, we see an interesting pattern. MW and PG-RM+ both achieve very low regret (≈ 0.001), learning to exploit the 60% bias effectively. PG-Vanilla shows higher mean regret (0.023) with very high variance (std 0.070), indicating some seeds fail to converge properly even against this simple opponent. This suggests that vanilla PG can be unstable even in relatively easy settings.

Key observation: Against static opponents, regret regularization does not hurt performance and may even improve stability (lower variance across seeds). The primary benefit of RRPG appears in adaptive settings where vanilla PG exhibits cycling and instability.

5.3. Policy Stability and Convergence

A key qualitative benefit of regret regularization is improved stability. While vanilla PG's action probabilities oscillate substantially throughout training without settling to a stable distribution, RRPG shows smoother convergence with

Table 3. Policy entropy (final 1000 rounds) for Matching Pennies vs MW. Higher entropy indicates more uniform (less exploitable) policies.

Algorithm	Entropy
MW	0.06 ± 0.02
PG-Vanilla	0.05 ± 0.11
PG-RM+ 0.1	$\mathbf{0.21 \pm 0.12}$
Optimal (uniform)	0.693

Table 4. Effect of regret-matching coefficient λ_R (Matching Pennies vs. MW, final 1000 rounds).

λ_R	Avg Regret	Exploitability	Entropy
0.05	0.051 ± 0.015	0.85 ± 0.09	0.21 ± 0.11
0.1	$\mathbf{0.056 \pm 0.019}$	$\mathbf{0.85 \pm 0.12}$	$\mathbf{0.21 \pm 0.12}$
0.2	0.048 ± 0.018	0.80 ± 0.07	0.26 ± 0.08
0.5	0.046 ± 0.008	0.82 ± 0.04	0.24 ± 0.04
1.0	0.048 ± 0.008	0.82 ± 0.02	0.24 ± 0.03
2.0	0.047 ± 0.007	0.80 ± 0.03	0.25 ± 0.04

dampened oscillations, settling closer to the Nash equilibrium.

Table 3 quantifies this via policy entropy.

PG-RM+ maintains higher entropy (0.21) compared to PG-Vanilla (0.05), though both are far below the theoretical optimum (0.693 for uniform distribution over 2 actions). The high standard deviations reflect that both methods exhibit variable behavior across seeds. However, qualitative inspection of individual trajectories reveals that PG-RM+ oscillations are damped and centered around the Nash equilibrium, while PG-Vanilla oscillations are often larger in amplitude and slower to converge.

This stability translates to lower variance in performance metrics across seeds. The coefficient of variation (std/mean) for average regret is 69% for PG-Vanilla vs. 33% for PG-RM+ against MW opponents, indicating more consistent performance.

6. Ablations and Analysis

6.1. Regret-Matching Coefficient λ_R

Table 4 shows the effect of varying λ_R from 0.05 to 2.0.

We observe that performance is relatively robust to the choice of λ_R in the range [0.1, 2.0]. All values in this range achieve substantially lower regret than vanilla PG (1.03). However, there are subtle differences:

Too low ($\lambda_R = 0.05$): Regret is slightly higher (0.051) than optimal, suggesting insufficient regularization. The policy is not being pulled strongly enough toward the RM+ target.

Sweet spot ($\lambda_R = 0.1$–0.5): These values achieve the best balance. Regret is low (0.046–0.056), exploitability is reduced (0.80–0.85), and entropy is reasonable (0.21–0.26). We select $\lambda_R = 0.1$ as our default based on slightly better cross-game generalization (not shown).

High values ($\lambda_R = 1.0$–2.0): Performance remains good, with very low variance across seeds (std $\approx$ 0.007–0.008). This suggests that strong regularization reduces sensitivity to random initialization. However, in some preliminary experiments (not shown), very high λ_R can slow adaptation in non-stationary settings.

Notably, entropy increases with λ_R, from 0.21 at $\lambda_R = 0.1$ to 0.25 at $\lambda_R = 2.0$. This suggests the regularizer encourages more diffuse policies, which is desirable in zero-sum games where the Nash equilibrium is uniform.

Table 5. Functional form variants (Matching Pennies vs. MW, final 1000 rounds).

Variant	Avg Regret	Exploitability
RM+ (standard)	0.056 ± 0.019	0.85 ± 0.12
RM-NoClip	0.091 ± 0.027	0.90 ± 0.07
Softmax-RM ($\tau = 0.5$)	0.074 ± 0.029	0.86 ± 0.09
Softmax-RM ($\tau = 1.0$)	0.078 ± 0.037	0.86 ± 0.08

6.2. Functional Form Variants

Table 5 compares different functional forms for computing the regret-matching target distribution.

RM+ (clipping negative regrets) performs best with average regret of 0.056. The clipping operation appears to provide a useful inductive bias by completely discarding actions that have underperformed, focusing probability mass on actions with positive regret.

RM-NoClip (shifting to make regrets non-negative) performs slightly worse (0.091 regret). By preserving relative differences between all actions including those with negative regret, this variant may be too conservative in down-weighting poor actions.

Softmax-RM variants with temperature $\tau \in \{0.5, 1.0\}$ yield intermediate performance (0.074–0.078 regret). Lower temperature ($\tau = 0.5$) produces sharper distributions, while higher temperature ($\tau = 1.0$) produces smoother ones. Both are reasonable, though neither outperforms standard RM+.

Overall, the choice of functional form matters less than simply incorporating regret information. All variants substantially outperform vanilla PG (1.03 regret), suggesting the key is the regret-based signal rather than the exact transformation.

6.3. Discounted vs. Cumulative Regret

Exponentially discounted regret with $\gamma \in \{0.95, 0.99\}$ performs comparably or slightly better than cumulative regret in our stateless games:

Variant	Avg Regret
Cumulative (standard)	0.056 ± 0.019
Discounted $\gamma = 0.95$	$\mathbf{0.019 \pm 0.005}$
Discounted $\gamma = 0.99$	0.022 ± 0.005

Discounting with $\gamma = 0.95$ achieves notably lower regret (0.019 vs. 0.056), suggesting that weighting recent observations more heavily can reduce sensitivity to early-game noise when policies are still far from optimal. The $\gamma = 0.99$ variant is intermediate.

In our stateless games with stationary opponents, discounting provides a mild benefit. In non-stationary environments where the opponent's strategy evolves, discounting may be even more important by allowing the agent to "forget" outdated regret information.

6.4. Progressive Schedules

We tested decaying λ_R over time using linear and exponential schedules:

Schedule	Avg Regret
Constant $\lambda_R = 0.1$	0.056 ± 0.019
Linear decay	0.048 ± 0.018
Exponential decay ($\alpha = 3$)	0.051 ± 0.014

Both schedules achieve slightly better regret (0.048–0.051) than constant λ_R (0.056), though the differences are small. Schedules may be beneficial for balancing initial regularization (to avoid early instability) with later flexibility (to fine-tune against specific opponents).

However, schedules add an extra hyperparameter (decay rate) and require tuning the horizon. For simplicity and robustness, we recommend constant λ_R unless there is a specific reason to believe the learning problem has distinct phases.

6.5. Regret Regularization vs. Entropy Regularization

To isolate the effect of regret regularization from entropy regularization, we compare:

- **PG-RM+only**: $\lambda_R = 0.1, \lambda_H = 0$
- **PG-Entropyonly**: $\lambda_R = 0, \lambda_H = 0.1$
- **PG-RM+H (standard)**: $\lambda_R = 0.1, \lambda_H = 0.01$

Variant	Avg Regret	
	vs MW	*vs EXP3*
RM+only	0.040 ± 0.012	0.045 ± 0.011
Entropyonly	0.582 ± 0.478	0.582 ± 0.478
RM+H (standard)	$\mathbf{0.056 \pm 0.019}$	$\mathbf{0.044 \pm 0.014}$

The results are striking: **regret regularization is the key driver of performance improvement**. RM+only achieves very low regret (0.040–0.045) even without entropy regularization. In contrast, Entropyonly (standard PG with high entropy bonus) achieves poor performance (0.582 regret)—only slightly better than vanilla PG.

Combining both (RM+H) achieves good performance (0.056 vs MW, 0.044 vs EXP3) with reasonable computational cost. The mild entropy regularization ($\lambda_H = 0.01$) helps with exploration without dominating the regret signal.

This confirms that the regret-matching regularizer provides a qualitatively different signal than generic exploration bonuses. RM+ encodes counterfactual reasoning about what would have worked better, whereas entropy simply encourages randomness.

7. Discussion

7.1. Why Does Regret Regularization Help?

Our results demonstrate that explicitly incorporating regret-matching structure into policy gradient objectives yields measurable benefits in regret, exploitability, and stability. We hypothesize two complementary mechanisms:

Bias toward equilibrium. The RM+ target distribution inherently biases the policy toward actions that minimize past regret. In zero-sum games, minimizing regret is equivalent to moving toward Nash equilibrium (Hart & Mas-Colell, 2000). By regularizing toward this distribution, we provide a complementary learning signal that guides the policy toward equilibrium even when the reward-based gradient alone would lead to cycling.

This is particularly important in symmetric zero-sum games where the Nash equilibrium requires randomization. Vanilla policy gradient, which follows the direction of increasing expected reward, can easily overshoot and oscillate around the equilibrium without converging. The regret regularizer dampens these oscillations by pulling the policy toward a distribution informed by cumulative counterfactual reasoning.

Dampening gradient-based oscillations. Gradient-based optimization can overshoot or cycle in adversarial settings due to the non-stationary nature of the learning problem. When both players are learning simultaneously, the gradient

at time t is computed with respect to the opponent's policy at time t, which may be very different from the opponent's policy at time $t+1$. This can lead to alternating overshooting and corrections.

The regret-matching regularizer acts as a stabilizing force by incorporating information from the entire history of play, not just the current gradient. This temporal smoothing helps prevent overreactions to individual rounds and encourages more stable, consistent policies.

7.2. Connection to Mirror Descent and Natural Gradients

There is an intriguing connection between our approach and mirror descent / natural gradient methods. Regret Matching can be viewed as performing mirror descent in the space of probability distributions using the negative entropy as a mirror map (Bubeck, 2015). By regularizing our neural policy toward the RM+ distribution, we are implicitly incorporating the geometric structure of the probability simplex into the optimization.

This suggests a potential avenue for future work: can we derive a principled natural gradient update that directly incorporates regret information, rather than adding it as a regularizer? Such an approach might achieve even better convergence properties while maintaining the flexibility of neural policies.

7.3. Limitations and Future Work

Stateless games only. Our experiments focus on stateless repeated games where regret is tractable to compute exactly via $R_t(a) = \sum_s [u(a, b_s) - u(a_s, b_s)]$. Extending to sequential decision-making (MDPs, extensive-form games) requires either:

- Approximate regret computation via sampling or learned value functions
- Hierarchical regret tracking at multiple levels (state-level, action-level)
- Integration with counterfactual regret minimization (CFR) in tree-structured games

Each of these directions introduces additional challenges in terms of computational cost and approximation error, but the core idea of regularizing toward regret-matching distributions may still be applicable.

Full-information assumption. We assume access to counterfactual utilities $u(a, b_t)$ for all actions. In bandit settings or partially observable environments, these counterfactuals may need to be estimated via importance weighting, learned models, or other techniques. The quality of these estimates will directly affect the quality of the regret-matching regularizer.

Computational overhead. RRPG requires tracking cumulative regret for all actions and computing the RM+ target distribution at each step. In our small action spaces (2-3 actions), this adds negligible overhead. In large discrete action spaces or continuous actions, more efficient approximations may be needed, such as:

- Sampling-based regret estimation
- Low-rank or structured regret representations
- Amortized regret computation via auxiliary neural networks

Theoretical analysis. While our empirical results are promising, we do not provide formal regret bounds for RRPG. Key open questions include:

- Under what conditions (e.g., convex losses, realizability) can we prove $R_T = o(T)$?
- How does the strength of regularization (λ_R) affect the regret bound?
- Can we characterize the tradeoff between regret and adaptation speed?

Developing such guarantees—even under strong assumptions like linear policies or tabular representations—would strengthen the theoretical foundation and guide hyperparameter selection.

Extension to general-sum games. Our experiments focus on zero-sum games where the Nash equilibrium is well-defined and unique (uniform random). In general-sum games, multiple equilibria may exist, and the notion of "optimal" play is less clear. Regret minimization still applies, but the interpretation and guarantees may differ. Exploring RRPG in cooperative, competitive, or mixed-motive settings is an important direction for future work.

8. Related Work

No-regret learning. Classical no-regret algorithms (Freund & Schapire, 1999; Auer et al., 2002; Hart & Mas-Colell, 2000) have been extensively studied in game theory and online learning. These methods achieve provable regret bounds and convergence to equilibria, making them the gold standard for adversarial learning. Regret Matching and its variants (Zinkevich et al., 2007) have been particularly successful in imperfect-information games such as

poker (Bowling et al., 2015), where Counterfactual Regret Minimization (CFR) achieved superhuman performance.

Our work differs in that we integrate regret-matching principles into neural policy gradient methods rather than using regret matching directly. This hybrid approach aims to combine the theoretical guarantees of regret minimization with the scalability of deep learning.

Policy gradient methods. Policy gradient algorithms (Sutton et al., 1999; Williams, 1992; Schulman et al., 2017) are a cornerstone of modern deep RL, enabling impressive results in complex domains from robotics to game playing. However, these methods are known to struggle in multi-agent and adversarial settings (Gleave et al., 2020), where the non-stationary learning problem and lack of equilibrium-seeking behavior can lead to instability.

Our work provides evidence that regret-based regularization can mitigate these issues without sacrificing the flexibility and scalability of gradient-based optimization.

Deep RL in games. Deep RL has achieved superhuman performance in games like Go (Silver et al., 2016) and Dota (Berner et al., 2019), typically via self-play and massive computational resources. However, these methods do not provide regret guarantees, and their convergence properties in general-sum games remain poorly understood. Our work is complementary, focusing on providing stronger guarantees through regret-based objectives rather than scaling up computation.

Regularized policy optimization. Entropy regularization (Haarnoja et al., 2018) and KL penalties (Schulman et al., 2017) are common in deep RL for encouraging exploration and preventing policy collapse. Our work extends this line by regularizing toward a target distribution informed by game-theoretic principles (regret matching) rather than generic priors (uniform or previous policy).

Hybrid methods. Recent work has explored combining elements of classical game-solving algorithms with deep learning. Examples include:

- Learned heuristics for CFR (Brown & Sandholm, 2019), which use neural networks to approximate regret updates in large games.
- Neural Fictitious Self-Play (Heinrich & Silver, 2016), which combines best-response computation with neural policies.
- Deep CFR (Steinberger, 2019), which uses neural networks as function approximators within the CFR framework.

Our approach is conceptually simpler: we augment an otherwise standard policy gradient method with a regret-matching regularizer. This modularity makes it easy to integrate with existing deep RL codebases and potentially applicable to a wide range of domains.

9. Conclusion

We have investigated whether regret-based principles can be reintroduced into deep policy gradient methods through explicit regularization. Our proposed method, Regret-Regularized Policy Gradient (RRPG), augments the standard policy gradient loss with a cross-entropy term that encourages the neural policy to match the distribution prescribed by classical Regret Matching Plus.

In systematic experiments across two zero-sum games against both static and adaptive opponents, we demonstrated that RRPG achieves:

- **93-94% lower external regret** compared to vanilla policy gradient against adaptive opponents
- **12-20% lower exploitability**, indicating closer approximation to Nash equilibrium
- **More stable action probabilities** with reduced variance across random seeds
- **Robust performance across hyperparameters**, with $\lambda_R \in [0.1, 2.0]$ all achieving substantial improvements

These results suggest that explicit incorporation of regret-based objectives can guide deep reinforcement learning toward no-regret-like behavior without abandoning the scalability and flexibility of gradient-based optimization. While we focused on simple stateless games for tractability, the core principle—regularizing toward distributions informed by cumulative counterfactual reasoning—may be applicable to more complex multi-agent scenarios.

This work opens several avenues for future research:

- Extending to sequential decision-making and extensive-form games via approximate regret computation or CFR integration
- Developing bandit variants with importance-weighted regret estimates
- Analyzing theoretical regret bounds for neural policy gradient with regret regularization
- Scaling to large action spaces via sampling or structured approximations
- Exploring applications in general-sum games and mixed-motive settings

More broadly, our results highlight the value of incorporating game-theoretic structures into deep learning objectives. As multi-agent RL becomes increasingly important for applications from autonomous vehicles to market mechanisms, principled approaches that combine the theoretical guarantees of game theory with the empirical success of deep learning will be essential.

Acknowledgements

We thank Professor Michael Kearns for guidance on this project and helpful class discussions on regret and online learning.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning, specifically in multi-agent reinforcement learning and game-theoretic learning. The methods developed here may have applications in competitive multi-agent systems, but we do not foresee any immediate negative societal consequences. As with any machine learning research, responsible deployment requires careful consideration of the specific application domain.

References

Arora, S., Hazan, E., and Kale, S. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bowling, M., Burch, N., Johanson, M., and Tammelin, O. Heads-up limit hold'em poker is solved. *Science*, 347 (6218):145–149, 2015.

Brown, N. and Sandholm, T. Superhuman AI for multiplayer poker. *Science*, 365(6456):885–890, 2019.

Bubeck, S. *Convex Optimization: Algorithms and Complexity*. Foundations and Trends® in Machine Learning, 2015. Available at arXiv:1405.4980.

Freund, Y. and Schapire, R. E. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S., and Russell, S. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

Hart, S. and Mas-Colell, A. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5): 1127–1150, 2000.

Heinrich, J. and Silver, D. Fictitious self-play in extensive-form games. In *International Conference on Machine Learning*, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Steinberger, E. Single deep counterfactual regret minimization. *arXiv preprint arXiv:1901.07621*, 2019.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, volume 8, pp. 229–256. Springer, 1992.

Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. Regret minimization in games with incomplete information. *Advances in Neural Information Processing Systems*, 20, 2007.