Defensive Convergence of Agents in Competitive Environments

Christopher Tang[1,*] and Hugh T. Blair[2]

[1]Stockdale High School, Bakersfield, California, USA
[2]Biology Department, University of California, Los Angeles (UCLA)
tangchristopher111@gmail.com

ABSTRACT

Emergent defensive convergence in reinforcement learning agents reveals how misaligned reward structures can induce passivity in autonomous systems. This study examines a Deep Q-Network (DQN) trained in a two-agent real-time Capture-the-Flag (CTF) environment built in `pygame`. We find that survival-oriented reward shaping led to convergence toward a risk-averse local optimum—"corner camping"—in which the agent prioritized safety over goal completion. Across 3,000 training episodes, the heuristic opponent achieved a 98% win rate, while the DQN agent won only 2–3% of matches despite a 130% increase in survival duration. These findings underscore how reward design can yield unintended equilibria and inform safer multi-agent system development.

KEYWORDS: *Reinforcement Learning, Deep Q-Network, Multi-Agent Systems, Reward Shaping, Defensive Behavior*

Introduction

Reinforcement learning (RL) has achieved major success in control and gameplay tasks such as Atari and Go[1,6]. However, when applied to real-time, partially observable environments, Deep Q-Networks (DQNs) often converge to stable yet suboptimal strategies.

This study explores how reward mis-specification in a minimal two-agent Capture-the-Flag (CTF) environment can produce defensive equilibria where agents prioritize survival over task completion. Understanding such "safe but unproductive" convergence is essential for designing reward systems in autonomous agents deployed in safety-critical contexts, such as robotic exploration or defense systems.

The central research question is: *How does survival-oriented reward shaping influence the emergence of defensive equilibria in competitive, real-time reinforcement learning environments?*

Related Work

Mnih et al.[1] pioneered DQN for discrete-action control, while Hausknecht and Stone[2] extended it to partially observable settings using recurrent layers. Multi-agent extensions by Foerster et al.[3] and Lowe et al.[4] demonstrated cooperative and competitive behaviors.

Recent alignment research[8,9] emphasized how agents exploit unintended incentives—so-called *reward hacking*—and the OpenAI "Hide and Seek" experiments[10] showed that even simple constraints can lead to complex emergent defense strategies.

Unlike these large-scale emergent environments, our work focuses on small, fully deterministic simulations where defensive convergence arises purely from reward structure, not from environment complexity or tool use.

Methodology

The system was implemented in Python 3.10 using `pygame` for real-time simulation and PyTorch 2.2 for neural training. Experiments ran on a Windows 11 desktop (Intel i7 CPU, NVIDIA GTX 1060 GPU). All random seeds were fixed at 42 to ensure reproducibility.

Environment Configuration: The environment comprised a 15×20 grid (40-pixel cells) within a 1200×800-pixel window at 60 FPS. Walls spawned randomly with 15% probability. Each team's base (60 px) and flag (20 px) occupied opposite corners. Episodes terminated after 6,500 frames or a flag capture event. The physics engine handled wall collisions, projectile hits, and respawns. Three captures determined victory (`CAPTURES_TO_WIN = 3`).

Agent Architecture: The Red agent used a fully connected DQN:

$$18 \to 128 \to 128 \to 18$$

ReLU activations were used throughout. The Adam optimizer (lr $= 0.001$), discount factor $\gamma = 0.99$, and mini-batch size 32 were used. A target net-

work synchronized every 1,000 steps; the replay buffer stored 10,000 transitions. The opposing Blue AI followed a deterministic chase–retreat rule.

Reward Function: The reward function aimed to balance offense and survival:

- +200 for flag capture
- +50 for flag pickup
- +0.1 per survival frame
- -10 for agent death
- -1 per 10 health lost
- +1 for decreasing distance to opponent base

However, the frequent low-magnitude survival rewards outweighed sparse capture rewards under long time horizons ($\gamma = 0.99$), biasing the policy toward risk minimization.

Training Procedure: Training ran for 3,000 episodes using an ϵ-greedy policy decaying from 1.0 to 0.1. Q-values updated every 4 frames; the TD loss was minimized via mean-squared error.

```
for episode in range(num_episodes):
    state = env.reset()
    for t in range(max_steps):
        action = epsilon_greedy(Q_net,
            state)
        next_state, reward, done, info =
            env.step(action)
        replay_buffer.add(state, action,
            reward, next_state, done)
        if len(replay_buffer) > batch_size
            :
            batch = replay_buffer.sample(
                batch_size)
            update_q_network(batch)
        if done:
            break
    epsilon = max(epsilon_min, epsilon *
        decay_rate)
```

Post-Training Analysis: Training logs stored rewards, win rates, and episode durations. `analyze_training.py` computed moving averages and win/loss distributions, producing four diagnostic plots: reward curves, win rate progression, episode duration, and score histogram.

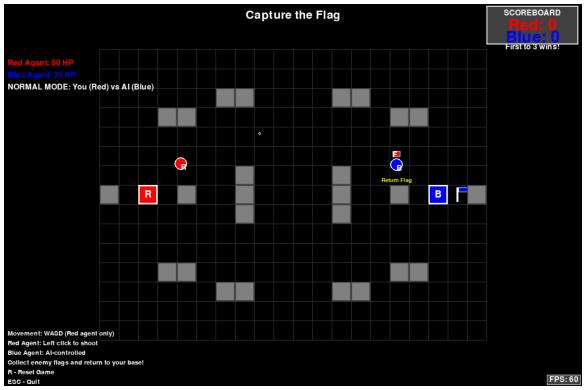

Figure 1: Arena layout. Corner regions provided low-risk areas encouraging defensive clustering.

Results

Across 3,000 episodes, the heuristic baseline (Blue AI) maintained a 98.1% win rate. The Q-Net's reward variance dropped sharply after ∼2,400 episodes, indicating convergence to a stable yet defensive equilibrium. Mean episodic reward stabilized around 70 ± 5 (vs. 100 ± 8 for the heuristic), while survival duration increased from $4{,}000 \pm 500$ to $9{,}200 \pm 600$ frames.

Table 1: Performance Summary After Convergence

Metric	Blue AI	Red Q-Net	(%)
Win rate (%)	98.1	2.3	-96
Mean reward	100 ± 8	70 ± 5	-30
Survival frames	$4{,}000 \pm 500$	$9{,}200 \pm 600$	+130

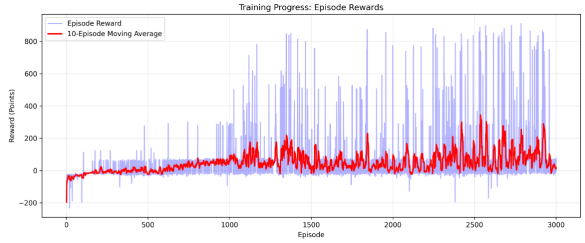

Figure 2: Episode-reward moving average (100-episode window). Convergence near episode 2,400 marks policy stabilization.

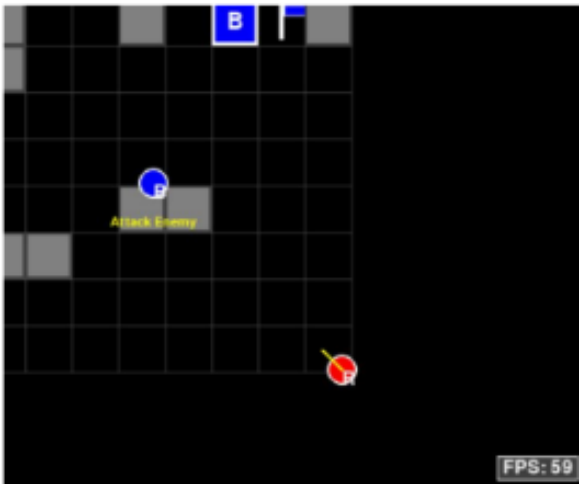

Figure 3: Frame from episode 2,700 showing "corner camping." The agent avoids risk by staying in safe regions.

Discussion

The per-frame survival reward stabilized learning but induced risk-averse equilibria, confirming the "reward hacking" phenomenon[8]. From a temporal-difference perspective, the agent's Q-values plateaued on a local optimum where consistent small rewards outweighed sparse high-reward captures.

This outcome represents a local Nash equilibrium: given the opponent's deterministic strategy, no unilateral deviation increased expected return. The result mirrors human e-sports behavior—players camp defensively when risk is penalized.

Future mitigation strategies include entropy regularization, intrinsic curiosity, and curriculum learning that gradually increases offensive pressure. Exploring PPO or SAC baselines[7] may also reveal whether actor–critic architectures resist this defensive bias.

Conclusion

Even lightweight DQNs can exhibit sophisticated emergent behaviors. In this study, the Red Q-Net learned stable but suboptimal defensive strategies, achieving only 2–3% win rate against a deterministic heuristic. These results illustrate how reward misalignment can generate pathological equilibria, a key concern for safe RL.

Future work will include: (1) ablation studies on reward weighting, (2) recurrent architectures for temporal reasoning, and (3) multi-agent co-training to reduce equilibrium lock-in.

Limitations and Future Work

This experiment evaluated only a single reward configuration and neural architecture. Further research should analyze hyperparameter sensitivity, compare with recurrent and policy-gradient baselines, and quantify convergence variance across multiple random seeds. Confidence intervals for mean metrics were not computed due to limited run repetitions

Acknowledgments

This research was conducted under the guidance of Professor Hugh T. Blair as part of the UCLA COSMOS program.

References

1. Mnih, V., Kavukcuoglu, K., Silver, D., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

2. Hausknecht, M., Stone, P. Deep recurrent Q-learning for partially observable MDPs. *arXiv preprint*, arXiv:1507.06527, 2015.

3. Foerster, J., Assael, Y., de Freitas, N., Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. *NeurIPS*, 2016.

4. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *NeurIPS*, 2017.

5. Lin, Y., Ren, S., Zhang, P. Efficient real-time multi-agent reinforcement learning with centralized training and decentralized execution. *IEEE TNNLS*, 31(11):4601–4614, 2020.

6. Silver, D., Huang, A., Maddison, C., et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

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

8. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., Mané, D. Concrete problems in AI safety. *arXiv preprint*, arXiv:1606.06565, 2016.

9. Christiano, P., Leike, J., Brown, T., Martic, M., Legg, S., Amodei, D. Deep reinforcement learning from human preferences. *NeurIPS*, 2017.

10. OpenAI. Emergent tool use from multi-agent interaction. *OpenAI Blog*, 2019.