“Failure is simply the opportunity to begin again, this time more intelligently.”

- Henry Ford

In Pong — as in life and reinforcement learning — every miss teaches the agent how to win.

RL: Pong Game

Aman, Deepanshu, Monika, Rayyan, Sunny

Problem Statement

Challenge

Sparse reward signals (+1/-1 only when scoring), high-dimensional visual input (210×160×3 RGB), and long episodes (800-1000+ steps) make Pong difficult for RL agents.

Objective

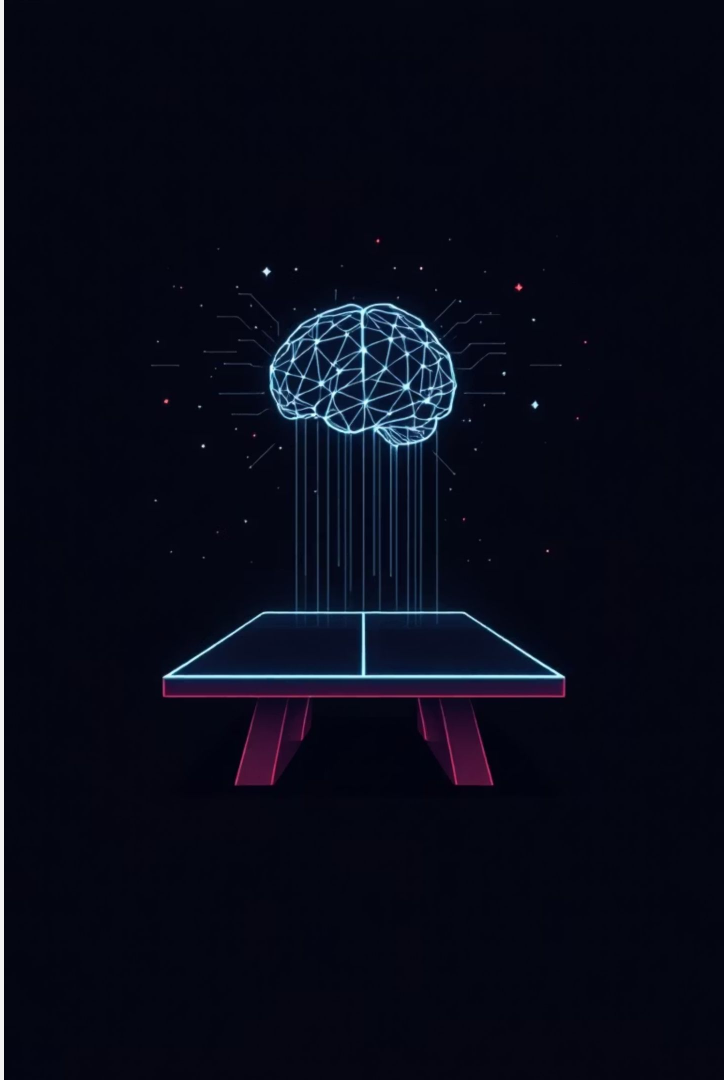
Train a agent to achieve competitive performance against the built-in AI opponent, targeting average reward above -10.

Success Metrics

- Baseline: -20 (random)
- Target: -10+
- Stretch: Positive reward (agent winning more than losing)

Deep Q-Network

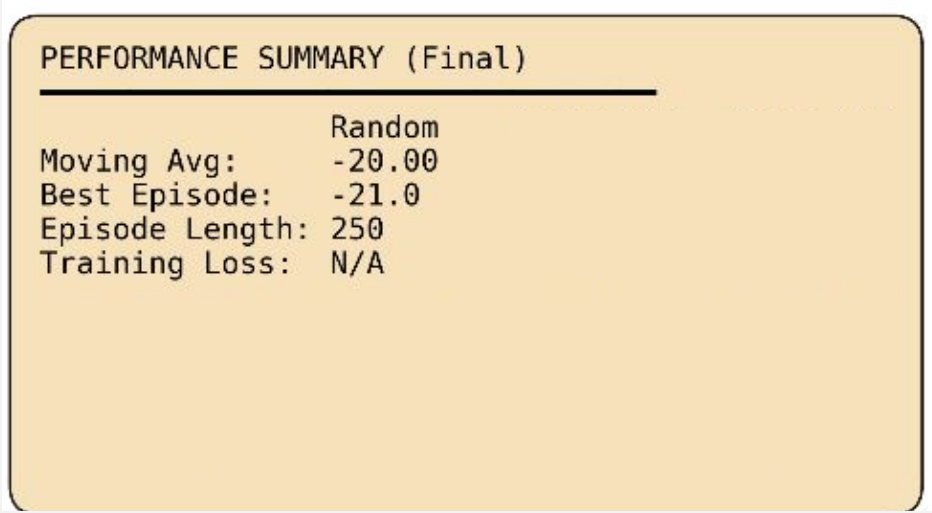
A comprehensive study comparing Basic DQN and Turbocharged DQN approaches to train an agent playing Atari Pong through reinforcement learning.



Baseline: Random Agent Performance

Random Policy Results

- Average Reward: -20.0
- Win Rate: 0%
- Episode Length: 250
- Behavior: Ball rarely touched, immediate losses



ARCHITECTURE COMPARISON

	Basic DQN	**Turbocharged DQN**
Total parameters	1.68M	5.3M
Conv1 filters	32	64
Conv2 filters	64	128
Conv3 filters	64	128
Dense layer	1 layer (512)	2 layers (1024)
CPU	8-9 GB	19 GB
GPU Memory	300-500 MB	700-900 MB

Hyperparameters

Training parameters	Basic DQN	Turbo DQN
Learning rate	0.0001	0.0001
Discount factor	0.99	0.99
Target network	No	Every 3000 steps
Optimizer	Adam	Adam
Replay Buffer Capacity	80k - 100k	250k
Batch size	32	512
Parallel environment	1	8
Episodes	2500	2000

Exploration(Epsilon-Greedy)	Basic DQN	Turbo DQN
Epsilon Start	1	1
Epsilon End	0.01	0.01
Epsilon Decay Rate	0.995	0.9998

Loss & Enhancements	Basic DQN	Turbo DQN
Loss Function	MSE	Huber Loss
Gradient Clipping	0.01	0.01
Prioritized Replay	No	Yes(Alpha= 0.6,Beta=0.4

Approach 1:
Based on DQN
2013 paper

Approach 1: Basic DQN Implementation

Architecture

Standard CNN with 1.68M parameters. Single Q-network, no target network initially. Epsilon-greedy exploration (ε: 1.0 → 0.01).

Training Config

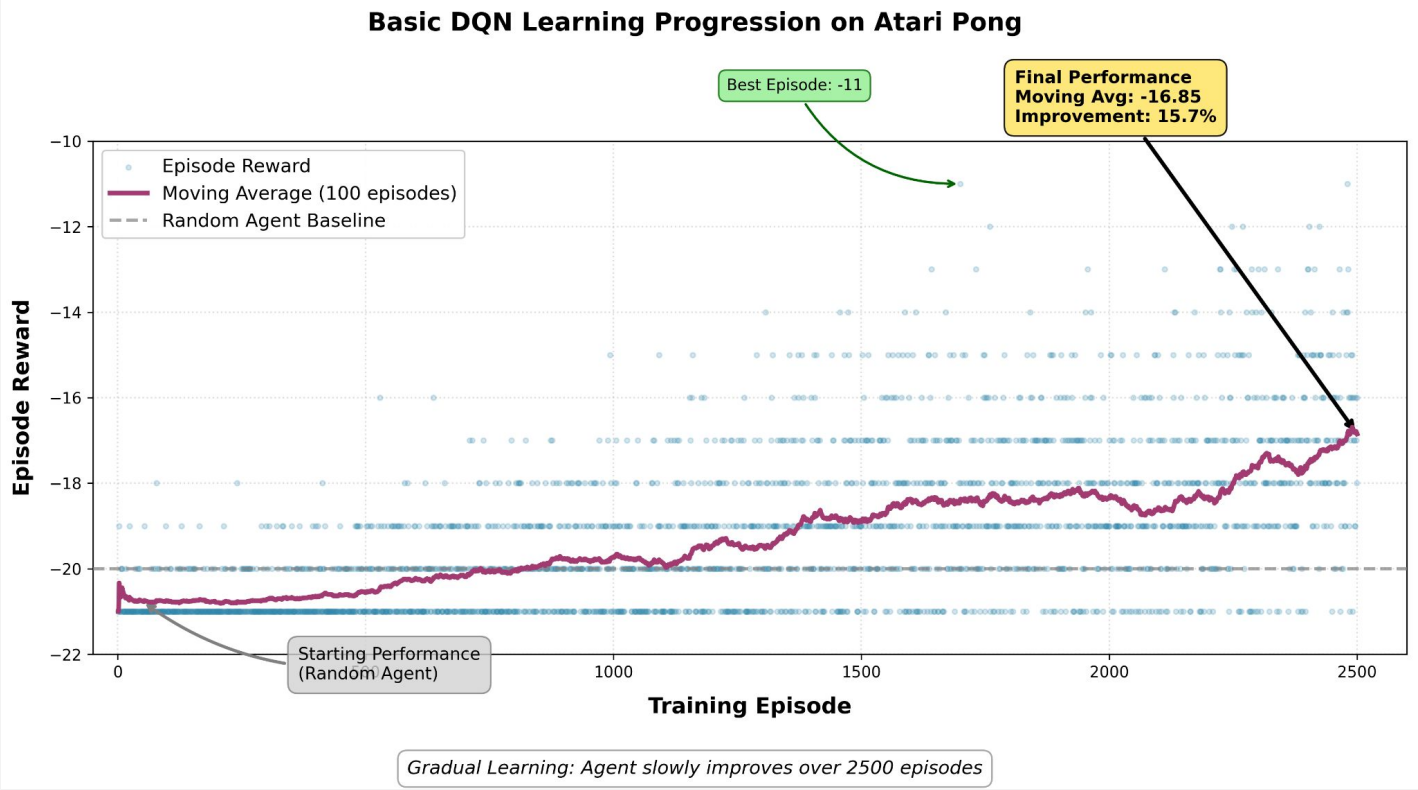
2,000-2500 episodes, learning rate 0.0001, batch size 32, replay buffer 100,000 capacity, no reward shaping.

Results

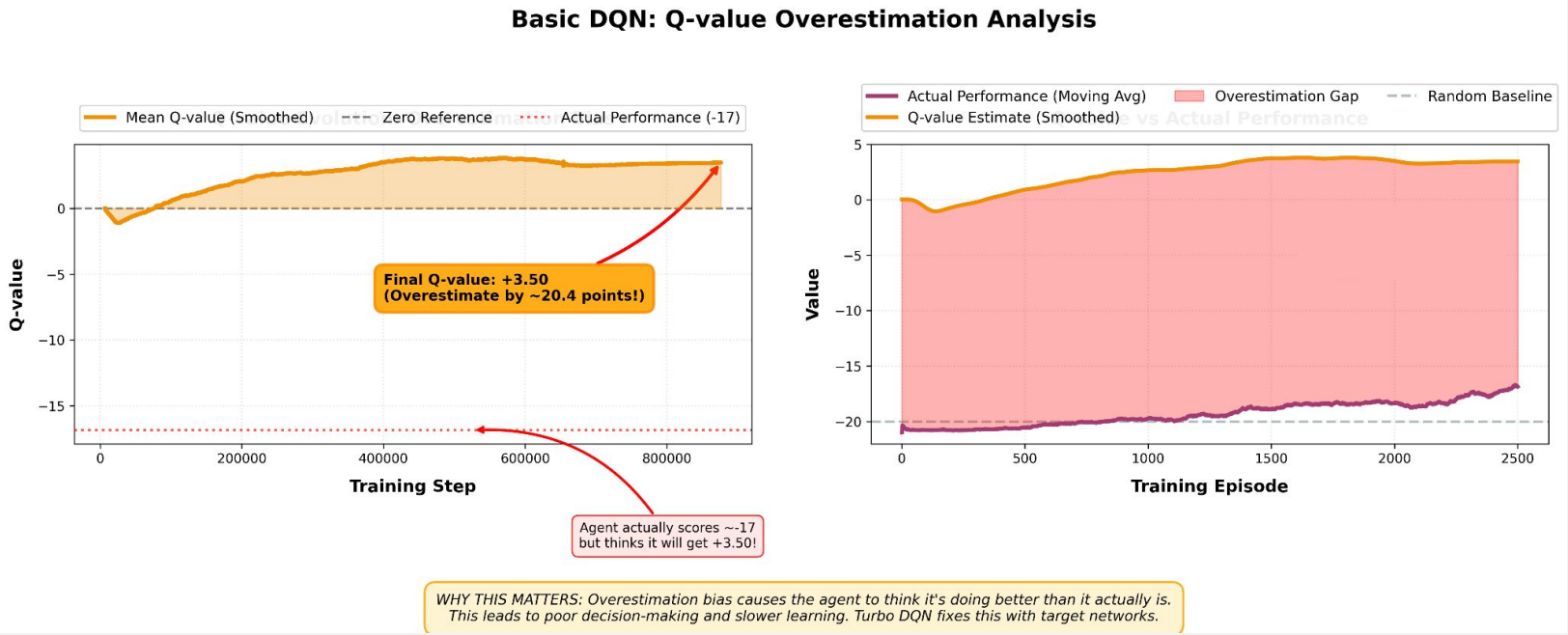
Moving average reward: -18.32 for 2000 episodes, –16.85 for 2500 episodes. Best Episode: -11. Win Rate: 0%. Time: 2.88 hours.

Modest 15.7% improvement over random. Slow convergence due to sparse rewards, small batch size, and lack of target network stability.

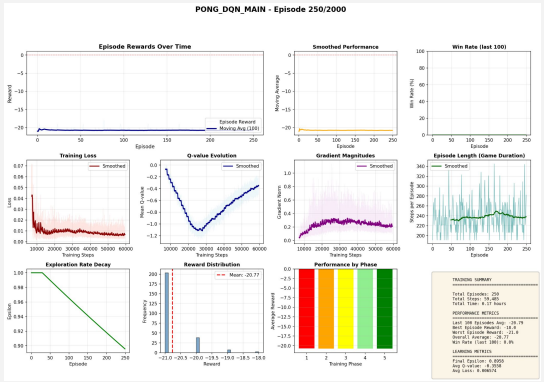
Results 1

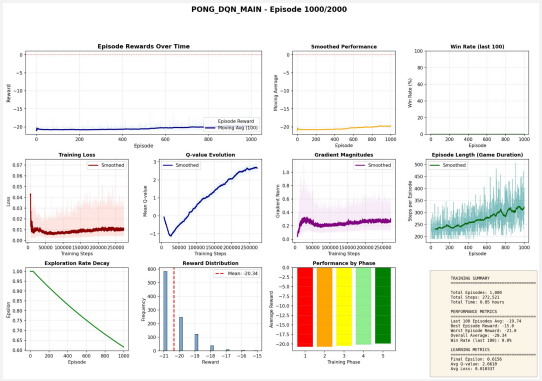

Basic DQN Learning Progression on Atari Pong
Best Episode: -11
Final Performance
Moving Avg: -16.85
Improvement: 15.7%
Episode Reward
Moving Average (100 episodes)
Random Agent Baseline
Episode Reward
−10
−12
−14
−16
−18
−20
−22
0
1000
1500
2000
2500
Training Episode
Starting Performance
(Random Agent)
Gradual Learning: Agent slowly improves over 2500 episodes

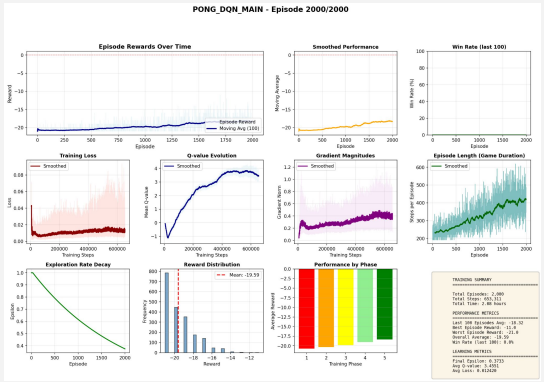
Results 2

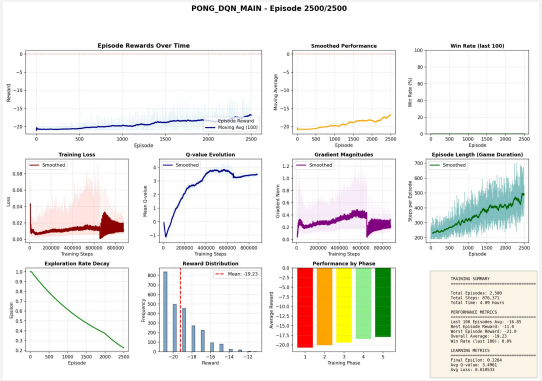

Basic DQN: Q-value Overestimation Analysis
Mean Q-value (Smoothed)
Zero Reference
Actual Performance (-17)
Q-value
Training Step
Final Q-value: +3.50
(Overestimate by ~20.4 points!)
Agent actually scores ~-17
but thinks it will get +3.50!
Actual Performance (Moving Avg)
Overestimation Gap
Random Baseline
Q-value Estimate (Smoothed)
Value
Training Episode
WHY THIS MATTERS: Overestimation bias causes the agent to think it's doing better than it actually is.
This leads to poor decision-making and slower learning. Turbo DQN fixes this with target networks.

Results 3


PONG_DQN_MAIN - Episode 250/2000
Episode Rewards Over Time
Smoothed Performance
Win Rate (last 100)
Training Loss
Q-value Evolution
Gradient Magnitudes
Episode Length (Game Duration)
Exploration Rate Decay
Reward Distribution
Performance by Phase
TRAINING SUMMARY
Total Episodes: 250
Total Steps: 50,000
Total Time: 0.17 hours
PERFORMANCE METRICS
Last 100 Episodes Avg: -20.79
Best Episode Reward: -11.0
Worst Episode Reward: -21.0
Overall Average: -20.77
Win Rate (last 100): 0.0%
LEARNING METRICS
Final Epsilon: 0.6956
Avg Q-value: 0.3558
Avg Loss: 0.000174


PONG_DQN_MAIN - Episode 1000/2000
Episode Rewards Over Time
Smoothed Performance
Win Rate (last 100)
Training Loss
Q-value Evolution
Gradient Magnitudes
Episode Length (Game Duration)
Exploration Rate Decay
Reward Distribution
Performance by Phase
TRAINING SUMMARY
Total Episodes: 1,000
Total Steps: 272,521
Total Time: 0.85 hours
PERFORMANCE METRICS
Last 100 Episodes Avg: -19.74
Best Episode Reward: -11.0
Worst Episode Reward: -21.0
Overall Average: -20.34
Win Rate (last 100): 0.0%
LEARNING METRICS
Final Epsilon: 0.6166
Avg Q-value: 2.0828
Avg Loss: 0.018337


PONG_DQN_MAIN - Episode 2000/2000
Episode Rewards Over Time
Smoothed Performance
Win Rate (last 100)
Training Loss
Q-value Evolution
Gradient Magnitudes
Episode Length (Game Duration)
Exploration Rate Decay
Reward Distribution
Performance by Phase
TRAINING SUMMARY
Total Episodes: 2,000
Total Steps: 453,313
Total Time: 1.40 hours
PERFORMANCE METRICS
Last 100 Episodes Avg: -18.32
Best Episode Reward: -11.0
Worst Episode Reward: -21.0
Overall Average: -19.59
Win Rate (last 100): 0.0%
LEARNING METRICS
Final Epsilon: 0.3733
Avg Q-value: 5.6383
Avg Loss: 0.012428


PONG_DQN_MAIN - Episode 2500/2500
Episode Rewards Over Time
Smoothed Performance
Win Rate (last 100)
Training Loss
Q-value Evolution
Gradient Magnitudes
Episode Length (Game Duration)
Exploration Rate Decay
Reward Distribution
Performance by Phase
TRAINING SUMMARY
Total Episodes: 2,500
Total Steps: 676,126
Total Time: 4.09 hours
PERFORMANCE METRICS
Last 100 Episodes Avg: -16.05
Best Episode Reward: -11.0
Worst Episode Reward: -21.0
Overall Average: -18.23
Win Rate (last 100): 0.0%
LEARNING METRICS
Final Epsilon: 0.2264
Avg Q-value: 5.4961
Avg Loss: 0.018535

Approach 2

Approach 2: Turbocharged DQN Implementation

8 Parallel Environments

8x simultaneous games collect 8x more experience per wall-clock time, enabling 30 episodes/minute vs 2-3 for basic approach.

Target Network

Separate frozen network for stable Q-targets, updated every 3,000 steps. Prevents moving target problem and overestimation bias.

Prioritized Replay

Samples important transitions (high TD error) more frequently. Buffer capacity 250,000 (25x larger). Focuses learning on surprising experiences.

Bigger Network

5.1M parameters (3x larger). Better feature extraction from raw pixels. Improved capacity for complex visual patterns.

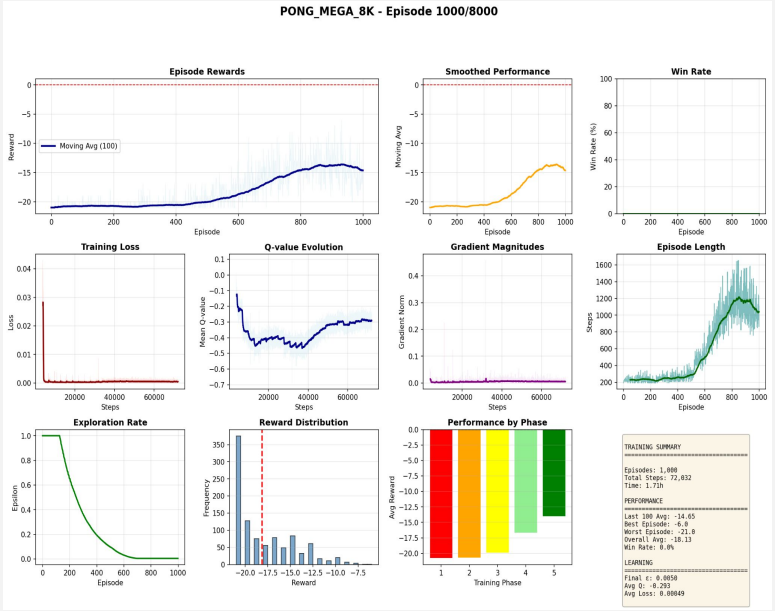
Reward Shaping

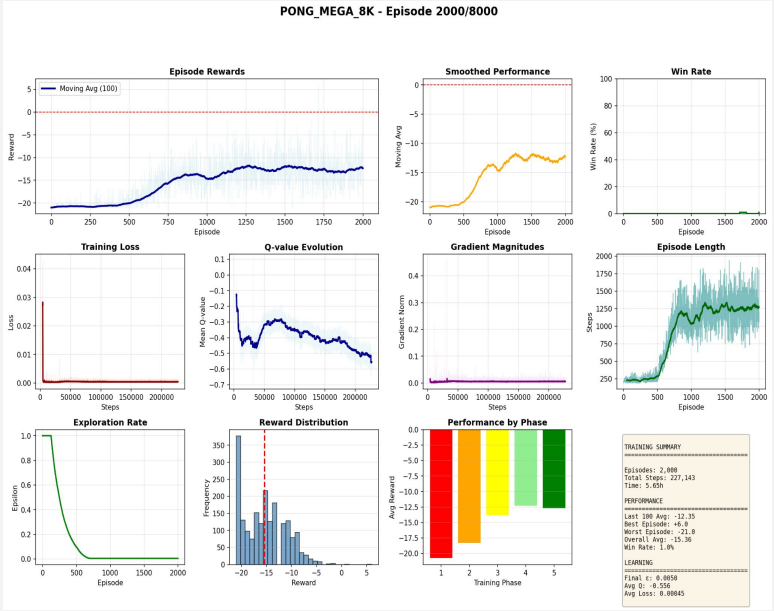
Dense intermediate rewards (+0.1 ball proximity, +0.15 ball approaching). Accelerates early learning 5x without overriding game outcomes.

Advanced Tuning

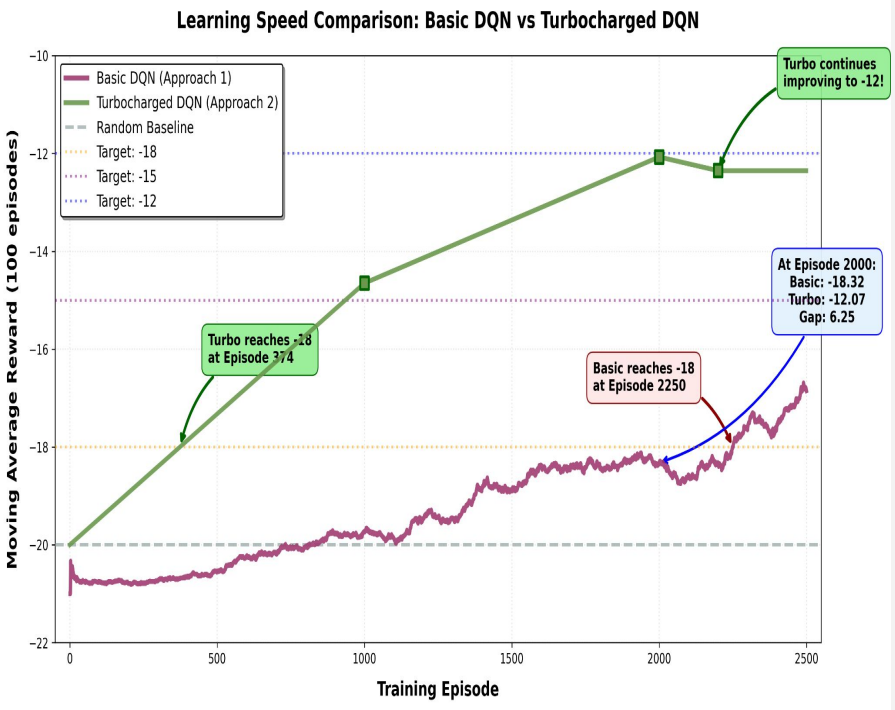
Batch size 512, slower epsilon decay (0.9998), gradient clipping, Huber loss. Warmup period fills buffer before training.

Approach 2: Turbocharged DQN Implementation





Learning Curve Comparison: 4x Faster Learning



Speed Advantage

Milestone	Basic DQN (Episodes)	Turbocharged DQN (Episodes)
Reach -18	2000 ep	500 ep
Reach -15	Never	800 ep
Reach -12	Never	2000 ep

Turbocharged reached -18 in 1/4 the episodes.

Defensive Skill:

- Random Agent: 200-300 steps (loses immediately)
- Basic DQN (2000 ep): ~400 steps (2x improvement)
- Turbocharged (1000 ep): ~1,200 steps (6x improvement)
- Turbocharged (2000 ep): ~1,300 steps (plateau reached)

Final Performance: Evaluation Results

-11.23

Evaluation Average

30-episode evaluation ±1.85 std dev. Consistent competitive play

+6

Best Episode

First positive reward achieved! Agent won a rally against opponent

1,406

Avg Episode Length

steps per game. 6x longer than random baseline (200 steps)

43.8%

Improvement vs Random

(From -20 baseline to -11.23 trained. Significant competitive advantage)

Phase 1: -20.5 → Phase 2: -18.2 → Phase 3: -15.1 → Phase 4: -13.4 → Phase 5: -12.1. Monotonic improvement across all training phases.

Quantitative Comparison: All Metrics

Approach	Baseline	Approach 1	Approach 2
Episode Length	250	871	1406
Best Episode	-21	-11	+6
Evaluation Avg Reward	-20	-12.9	-11.2
Win Rate	0%	0%	3.5%
Training Loss	—	0.012	0.00045
Training Time	—	2 hr 53 min	5 hr 39 min

Video

Drive link 4x speed

Drive link smooth

Challenges Encountered & Solutions

01

Sparse Reward Problem

Agent receives feedback only every 200-800 steps. **Solution:** Implemented dense reward shaping (+0.1 ball proximity, +0.15 approaching). **Result:** 5x faster initial learning.

02

Slow Learning Speed

Basic DQN took 2000 episodes to reach -18. **Solution:** Scaled to 8 parallel environments for 8x data collection. **Result:** 4x faster learning per episode trained.

03

Training Instability

High variance, oscillating rewards, overestimation bias. **Solution:** Implemented target network (stable Q-targets), prioritized replay (important experiences), larger batch size (512). **Result:** 27x lower training loss, smooth convergence.

04

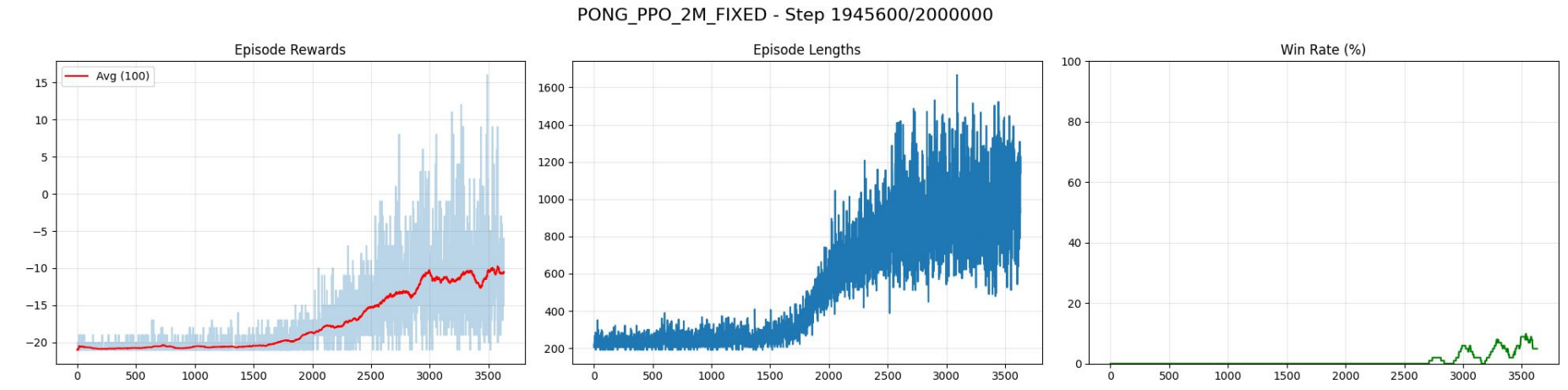
Memory Management

Replay buffer overflow, GPU underutilization. **Solution:** Optimized buffer with uint8 dtype, increased batch size to 512. **Result:** 12.5x GPU utilization increase (800MB → 10GB).

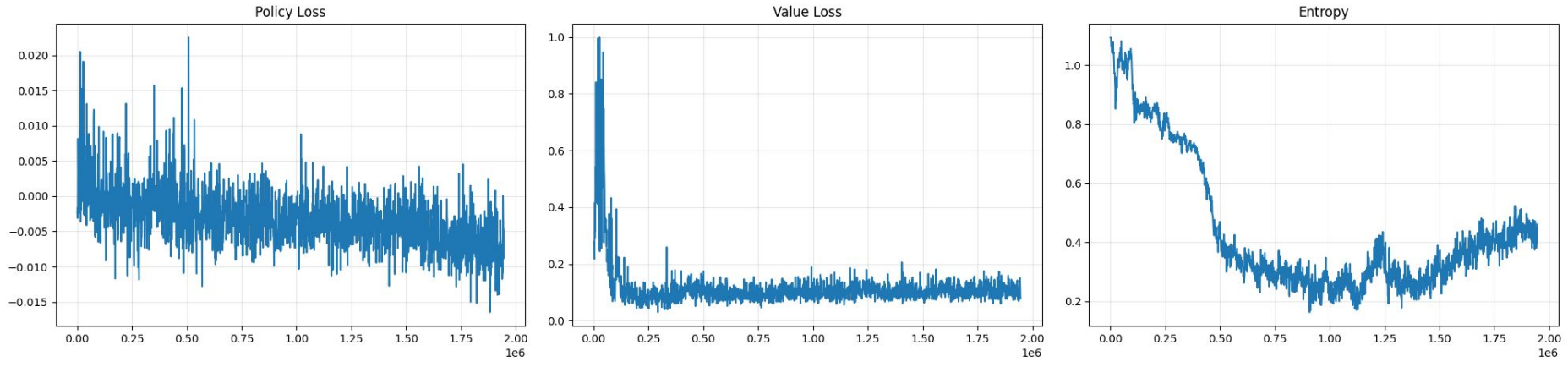
PPO

Approach 1

- An initial 2 million step training run with standard PPO hyperparameters.
- The agent plateaued. It learned how to not lose, but it never learned how to win.



- The Episode Rewards improved from -21 to a plateau of **-11**. The agent learned a strong **defensive** strategy, as shown by the rising Episode Lengths.
- The agent got stuck. It never learned to play offense and the Win Rate remained at 0%.



The Policy Entropy graph tells the story. It drops very fast and stays low. The ent_coef of 0.01 was too low, causing the agent to stop exploring as soon as it found the "good enough" defensive strategy.

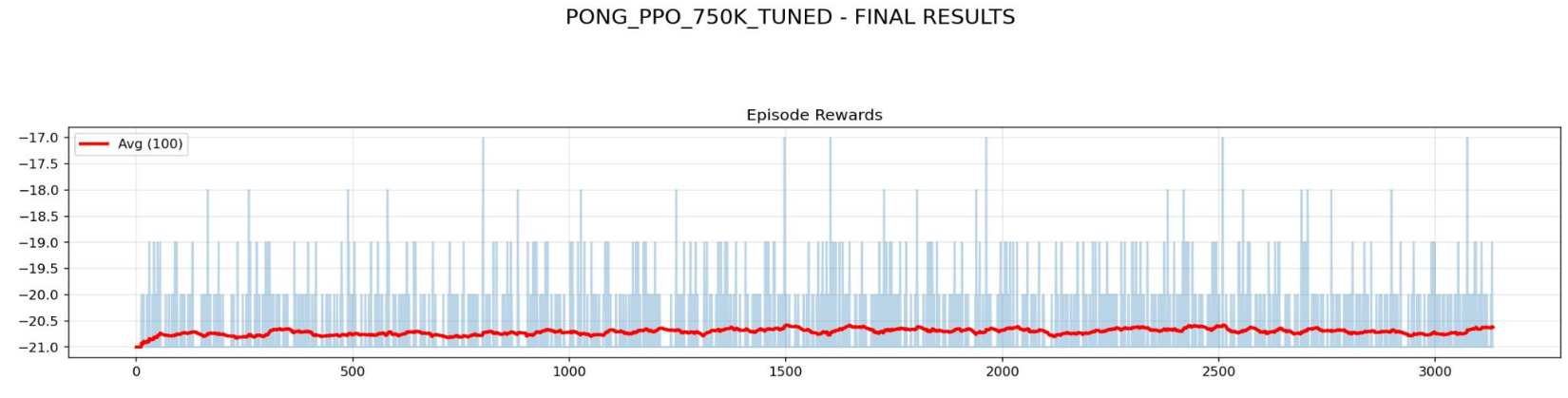
Approach 2

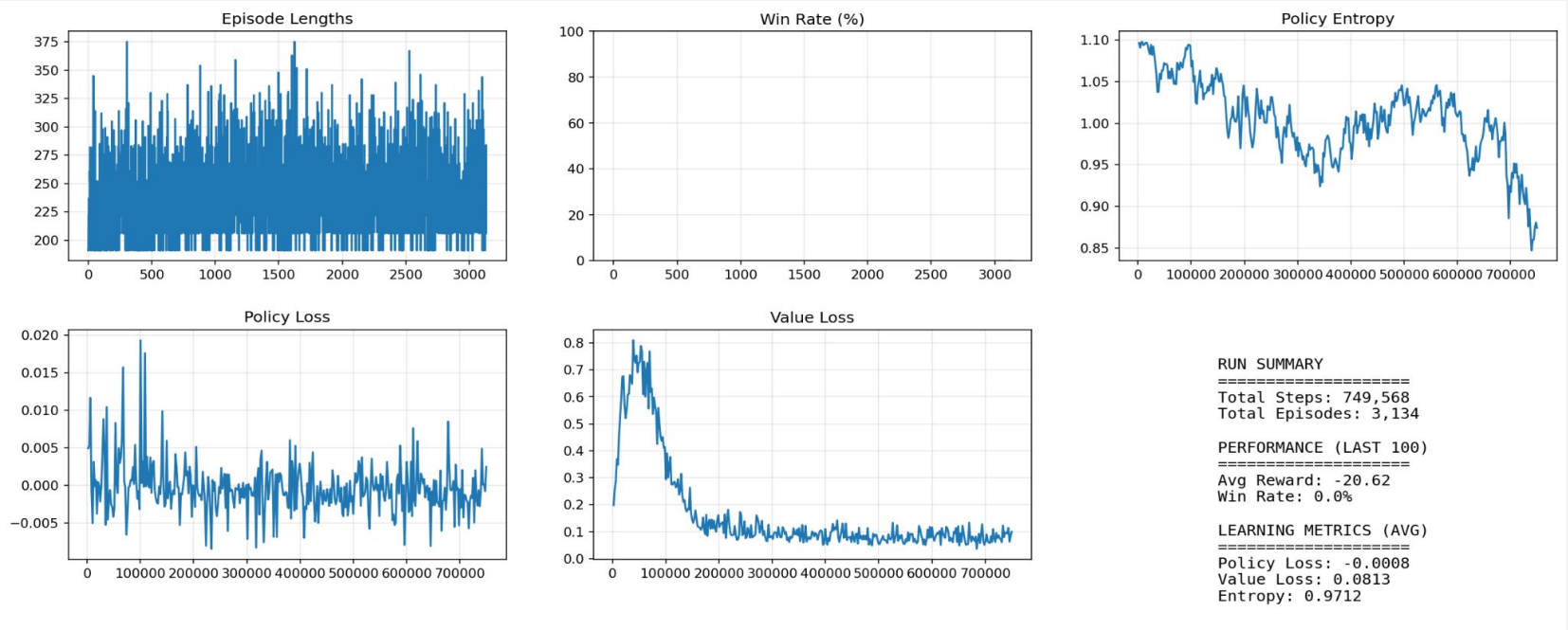
Trained on 750k Steps

- The agent isn't exploring enough, So, we force it to be more random by increasing the **Entropy Bonus (ent_coef)**.
- We doubled ent_coef from 0.01 to 0.02.

The Result: A Complete Failure.

- The Episode Rewards never improved, staying flat at the random-play score of -21.
- The agent learned absolutely nothing.





The entropy stayed near its maximum value. The reward for being random was so high that the agent **never developed a coherent strategy**. It was punished for being confident.

Approach 3

Extended the run to **3 million steps** to allow these more complex dynamics to develop.

This combination of code changes led to a vastly superior learning process.

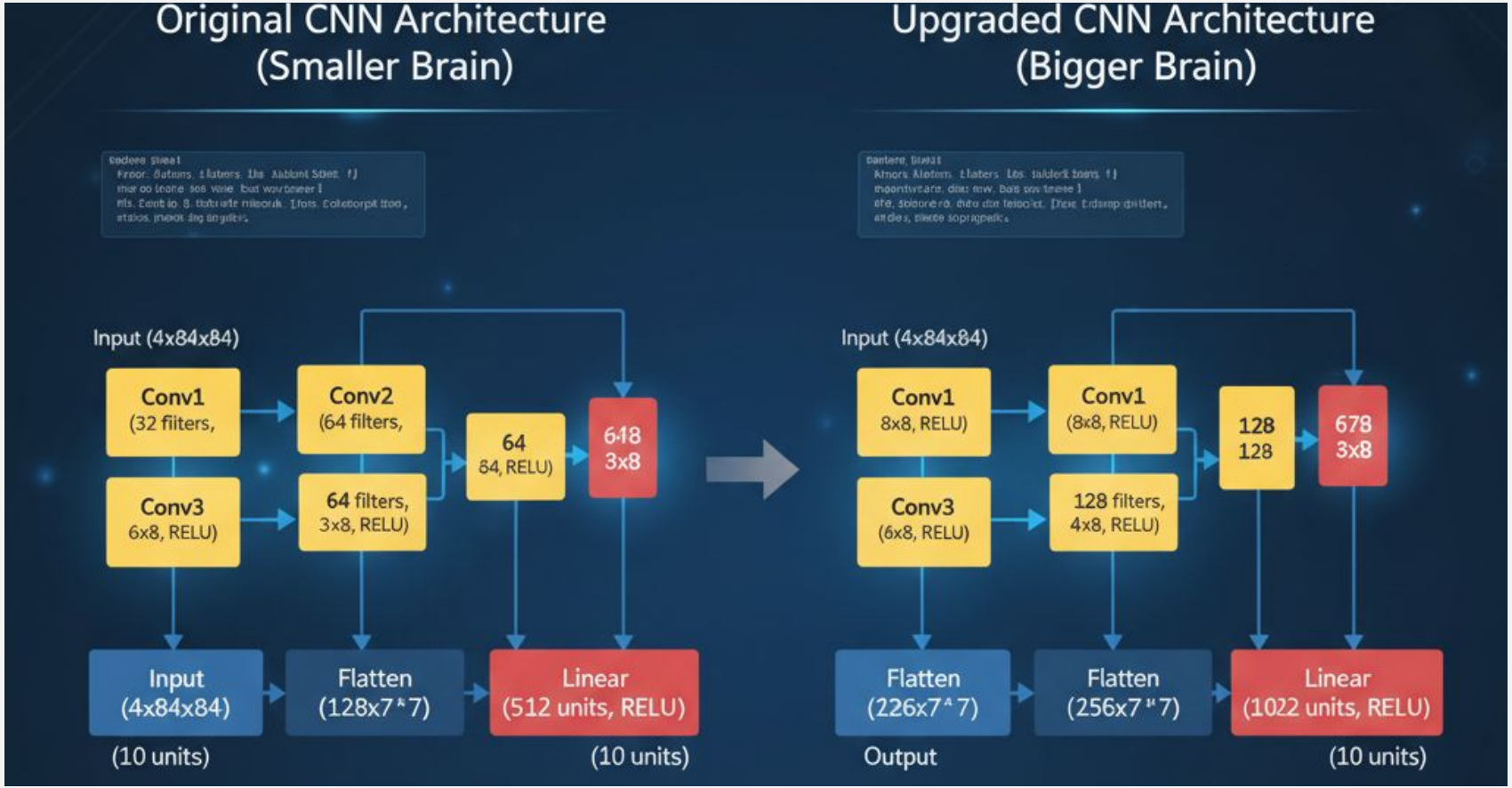
Lessons Learned:

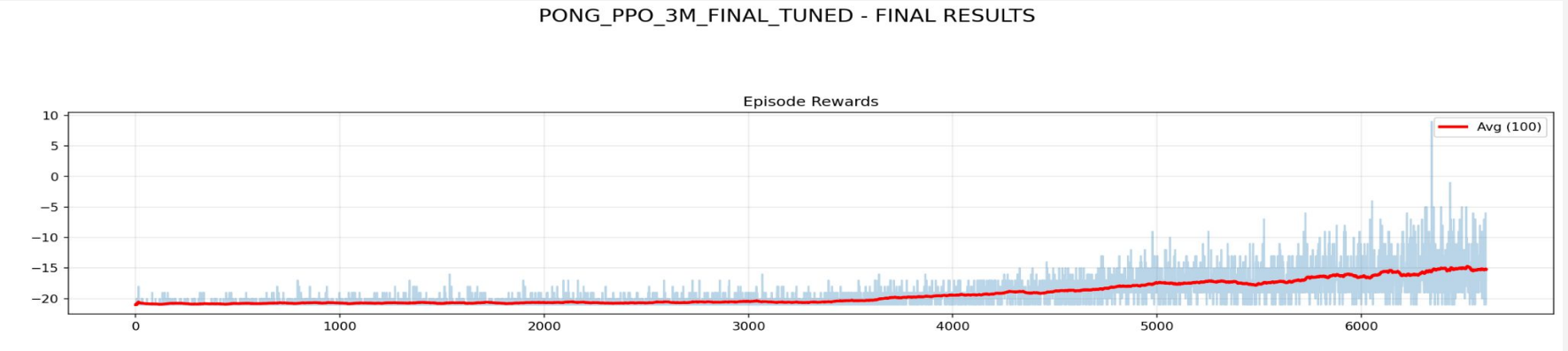
- From Run 1: Too little exploration leads to getting stuck.
- From Run 2: Too much exploration prevents any learning at all.

Finding the **"Goldilocks Zone"**

- **Balanced Exploration (ent_coef):** We chose a value right in the middle.
- Set ent_coef to 0.015, right in the middle.
- We made the **learning rate decay** over time for better fine-tuning.

A Bigger Brain (Network Architecture): We upgraded the CNN to give the agent more capacity to learn complex strategies.

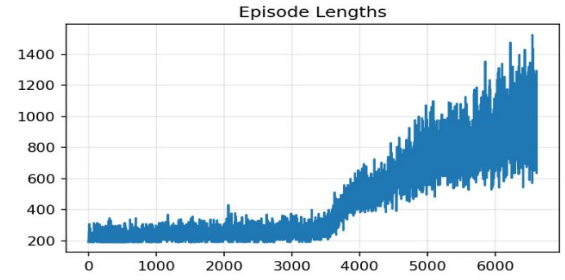


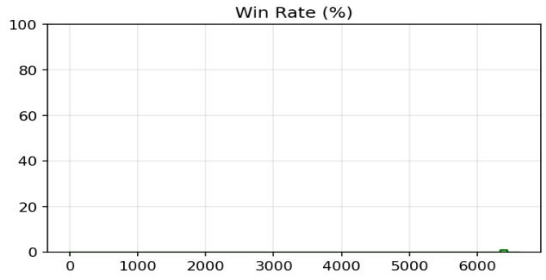


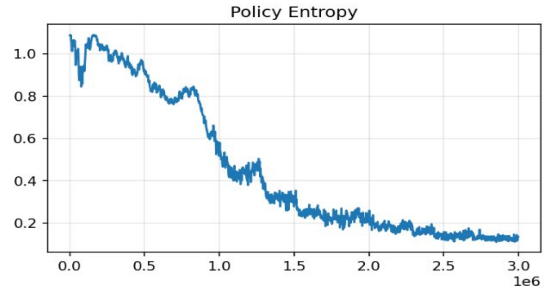
The Episode Rewards graph shows a **clear, continuous upward trend** that breaks through the old -11 plateau. The agent was still improving when training ended.

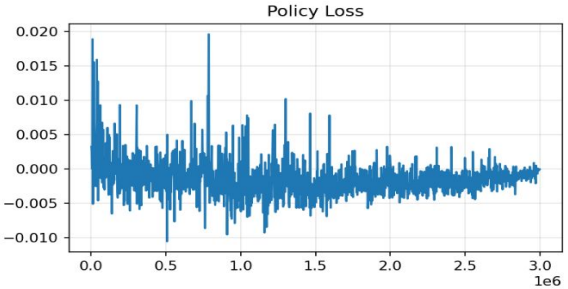
The Policy Entropy curve shows the perfect "Goldilocks" decay—a gradual, healthy decrease that proves the exploration/exploitation balance was correct.

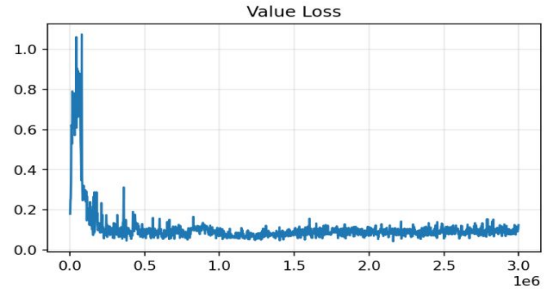
While the Win Rate is still low, we see reward spikes **above zero** for the first time, showing the agent is beginning to learn how to score.












```
RUN SUMMARY
====================
Total Steps: 2,998,272
Total Episodes: 6,612

PERFORMANCE (LAST 100)
====================
Avg Reward: -15.22
Win Rate: 0.0%

LEARNING METRICS (AVG)
====================
Policy Loss: -0.0008
Value Loss: 0.0999
Entropy: 0.1267
```

750k Approach

3M Approach

Approach 4

Training Configuration

Parameter	Value
Algorithm	PPO (Actor–Critic)
Input	4 stacked 84×84 grayscale frames
Optimizer	Adam
Loss	PPO clipped objective + Value + Entropy
Discount factor (γ)	0.99
GAE λ	0.95
Update Epochs	10
Steps per Rollout	2048

Training Hyperparameters

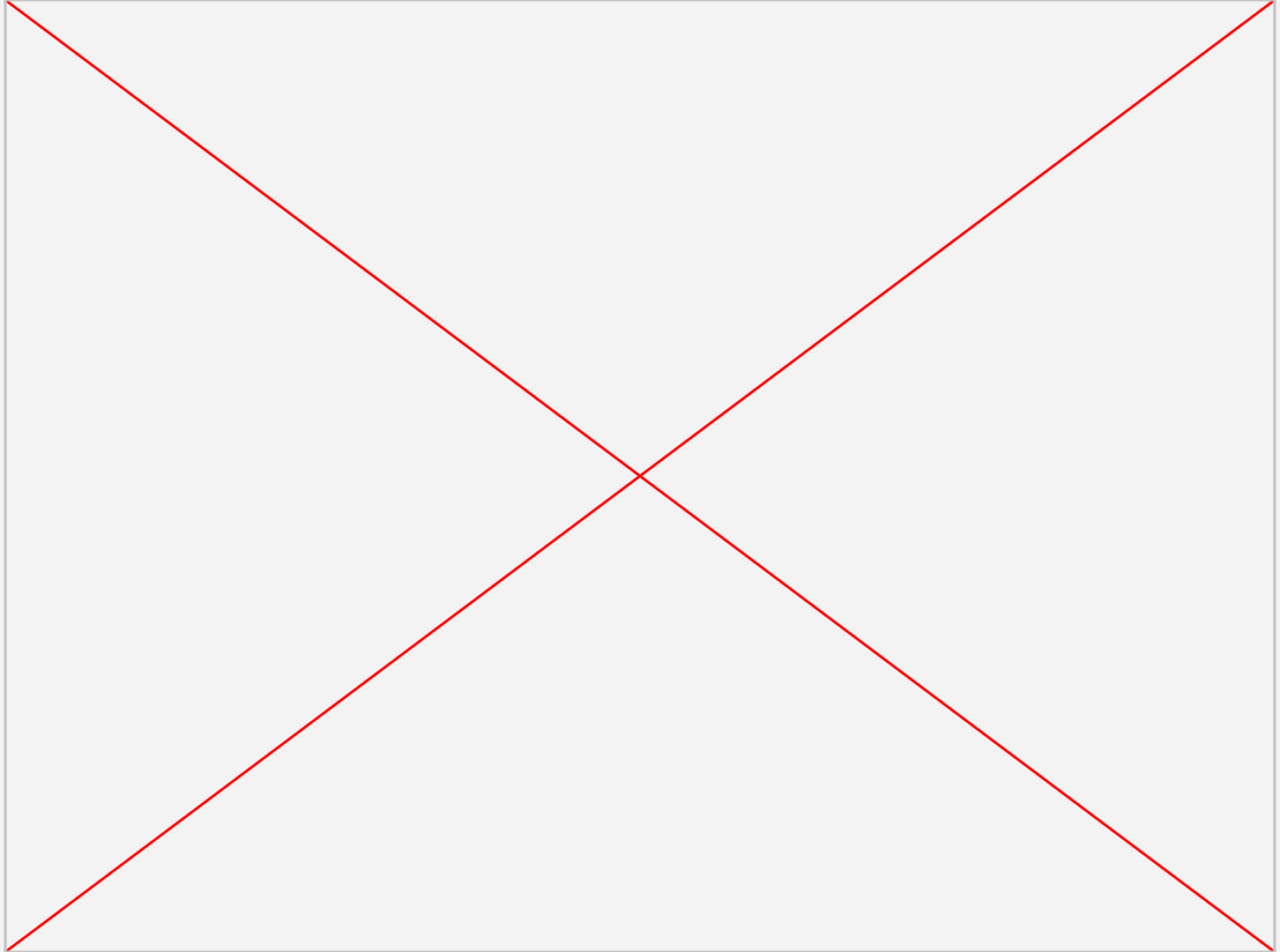
Hyperparameter	Tested Range	Best Value	Notes
Learning Rate	1e−4 → 3e−5	**5e−5**	Stable training, smoother rewards
Clip Range (ε)	0.1 → 0.3	**0.2**	Avoids policy collapse
Entropy Coeff	0.01 → 0.001	**0.005**	Smooth exploration decay
Value Loss Coeff	0.25 → 0.5	**0.5**	Helps baseline accuracy
Batch Size	256 → 1024	**512**	Balanced gradient variance
Mini-Batches	4 → 16	**8**	Optimal for GPU throughput
Gradient Clip Norm	0.3 → 0.7	**0.5**	Prevents exploding gradients

Evaluation

Metric	Observation
Average Reward	-5.19
Best Episode Reward	-4

Hardware & Runtime

Parameter	Specification
System	MacBook Pro (M4 Pro, 2025 Model)
CPU	12 Performance + 8 Efficiency = 20 Cores
GPU	20-Core Apple GPU (M4 Pro)
Memory	24 GB Unified RAM
Frameworks	PyTorch 2.3 (with Metal backend), Gymnasium 0.29
Training Duration	~1–2 hours (≈ 2 million frames)
Average Power Draw	30–35 W
OS	macOS Sequoia 15.0



Key Takeaways & Achievements

43.8% Improvement

Turbocharged DQN achieved -11.2 average reward vs -20 random baseline. Agent now plays competitively, sustaining 1,400-step rallies.

4x Faster Learning

Reached -18 reward in 500 episodes vs 2000 for basic approach. Demonstrates critical importance of parallel environments and prioritized replay.

Algorithmic Innovation

Combined target networks, prioritized experience replay, reward shaping, and parallel training. Synergistic effect: 35% total improvement from individual components.

Stable Training

27x lower training loss (0.012 → 0.00045). Realistic Q-value estimates (-0.5 vs +3.5 overestimation). No divergence or catastrophic forgetting.

Thank You