

Applied Reinforcement Learning for a 2D Shooter Game Environment

World Models, Reward Shaping, and COOM Benchmark

Can Karaçelebi

CENG 7822 - Reinforcement Learning

January 11, 2026

Outline

- 1 Introduction
- 2 Paper Showcase: COOM Benchmark
- 3 Model-Free RL Results
- 4 World Models Approach
- 5 Conclusion

Project Overview

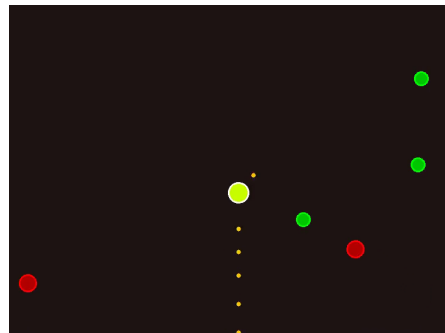
Goal: Train autonomous agents for a 2D top-down shooter game

Challenges:

- Multi-objective optimization (survive, collect, eliminate)
- Continuous state dynamics
- Reactive and strategic decision-making

Approaches:

- Model-free RL: DQN, PPO, SAC
- Model-based RL: World Models (VAE + MDN-RNN)



State Space:

- Agent position & velocity
- Health, cooldown status
- Nearest enemies (relative positions)
- Nearest prizes (relative positions)

Action Space:

- Movement: 5 options (stay, N, S, E, W)
- Shoot: 2 options (fire, hold)
- Direction: 8 options (8 angles)
- Total: $5 \times 2 \times 8 = 80$ combinations

Reward Configurations:

Event	Base	Surv	Aggr
Prize	+1.0	+0.5	+2.0
Kill	+1.0	+0.5	+2.0
Damage	-1.0	-3.0	-0.5
Death	-5.0	-10.0	-3.0

COOM: A Game Benchmark for Continual Reinforcement Learning

Tomilin, Fang, Zhang, Pechenizkiy

NeurIPS 2023 (Datasets and Benchmarks Track)

Key Contribution

First benchmark specifically designed for **continual reinforcement learning** in complex 3D image-based environments with varying objectives and visuals.

Built on: ViZDoom platform

Available:

<https://github.com/hyintell/COOM>

Why Continual RL Matters

Traditional RL: Train on one task, evaluate on same task

Continual RL: Train on sequence of tasks, retain knowledge

Key Challenges:

- ❶ **Catastrophic Forgetting:** Performance on task 1 degrades after training on task 2
- ❷ **Knowledge Transfer:** Can skills from task 1 accelerate learning task 2?
- ❸ **Sample Efficiency:** How quickly can the agent adapt to new scenarios?

Real-world motivation:

- Robots deployed in changing environments
- Game AI that adapts to new levels/mechanics
- Agents that should improve over time, not reset

COOM Scenarios (8 Total)

Combat Scenarios:

- **Run and Gun:** Eliminate enemies with ranged weapons
- **Chainsaw:** Melee combat, seek and chainsaw enemies

Evasion Scenarios:

- **Hide and Seek:** Escape and hide from pursuing enemies
- **Floor is Lava:** Platform navigation to safe zones

Each scenario has **default** and **hard** difficulty variants

Navigation Scenarios:

- **Pitfall:** Traverse tunnels, avoid pits
- **Arms Dealer:** Collect and deliver weapons to locations

Survival Scenarios:

- **Raise the Roof:** Find switches to prevent ceiling crush
- **Health Gathering:** Collect health kits to survive

COOM Task Sequence Types

Cross-Domain (CD): Same objective, different visuals

- Base: Run and Gun scenario
- Modifications: wall textures, enemy types, obstacles, view height
- Tests: visual generalization without goal change

Cross-Objective (CO): Different scenarios entirely

- Sequence: Run and Gun → Chainsaw → Hide and Seek → Floor is Lava → ...
- Goal changes drastically: eliminate → melee → evade → navigate
- Tests: skill transfer across fundamentally different tasks

Sequence Lengths: 4-task, 8-task (core), 16-task (extended)

1. Average Performance

$$\bar{P} = \frac{1}{N} \sum_{i=1}^N P_i^{\text{final}} \quad (1)$$

Success rate averaged over all N tasks at sequence end.

2. Forgetting

$$F_i = P_i^{\text{after_training_i}} - P_i^{\text{end_of_sequence}} \quad (2)$$

How much performance on task i degrades after subsequent training.

3. Forward Transfer

$$FT_i = \frac{\text{AUC}_{\text{continual}}}{\text{AUC}_{\text{scratch}}} - 1 \quad (3)$$

Training efficiency compared to learning from scratch.

Method	AP \uparrow	FGT \downarrow	FWT \uparrow
Fine-tuning	0.31	0.58	-0.12
EWC	0.35	0.51	-0.08
PackNet	0.42	0.24	-0.15
ClonEx-SAC	0.61	0.18	0.05

Table: Cross-Objective 8-task sequence

Key Observations:

- Even best method (ClonEx-SAC) shows **18% forgetting**
- Forward transfer is often **negative** (prior tasks hurt!)
- Cross-objective sequences are significantly harder than cross-domain

What Makes Continual RL Hard?

- ① **Catastrophic Forgetting:** Neural nets overwrite old knowledge when learning new tasks
- ② **Task Interference:** Skills for one objective may conflict with another
- ③ **No Clear Task Boundaries:** In deployment, agents don't know when tasks change
- ④ **Sample Efficiency:** Can't afford to retrain from scratch on each new task

Why World Models Might Help:

- Learned dynamics are more likely to transfer between tasks
- Replay in imagination doesn't require real samples
- Latent space can encode transferable features

Open Problem: No method achieves $\geq 65\%$ average performance with $\leq 15\%$ forgetting

Our 2D Shooter vs COOM:

- Multi-objective: survive + collect + eliminate
- Visual observations (\rightarrow World Model VAE)
- Continuous dynamics
- Potential for scenario variations
- 8 COOM scenarios test diverse skills
- ViZDoom visual observations
- 3D navigation
- Cross-domain and cross-objective sequences

Future Work: Apply COOM evaluation framework (AP, FGT, FWT) to test our agents' robustness to environment modifications

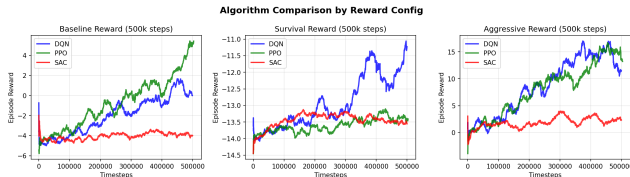
Algorithm Comparison

Key Findings:

- **PPO** achieves best performance (+38.74)
- **DQN** comparable but more variable
- **SAC** significantly underperforms

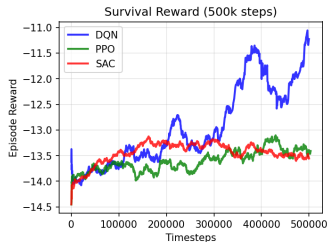
Why SAC Failed:

- Action space mismatch
- 12x slower training speed
- Entropy regularization interference



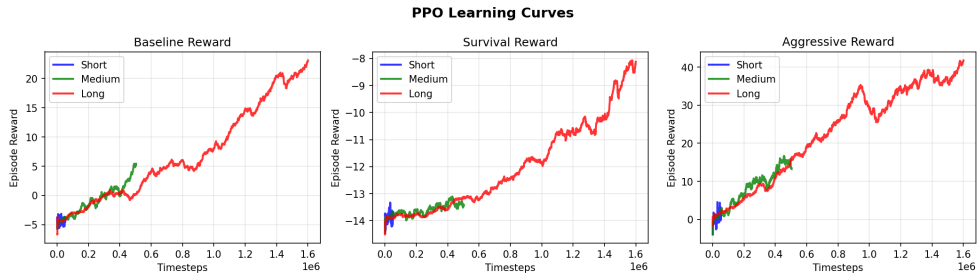
Algorithm Comparison (Detailed)

Algorithm Comparison by Reward Config



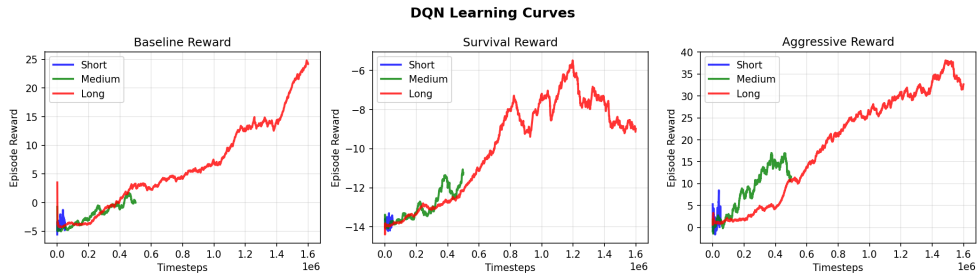
PPO with aggressive rewards achieves +38.74 — SAC underperforms due to action space mismatch

Learning Curves: PPO



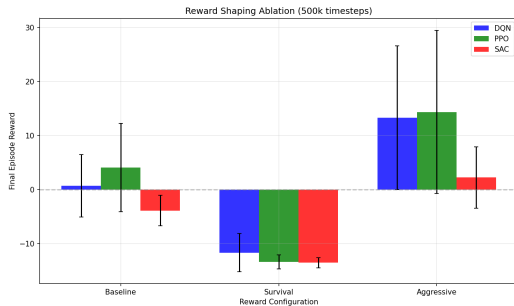
PPO shows stable, monotonic improvement across all configurations

Learning Curves: DQN



DQN shows reward decrease with survival config (sparse rewards)

Reward Shaping Impact



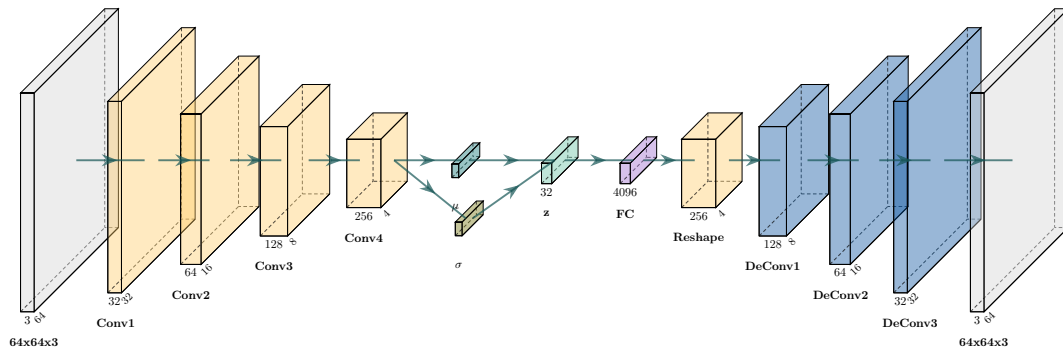
Reward design dominates algorithm choice!

Algo	Aggr	Base	Surv
PPO	+38.74	+21.96	-8.80
DQN	+33.95	+19.42	-8.77
SAC	+1.49	-5.02	-13.73

Why Aggressive Works:

- Stronger positive signals
- Reduced death penalty → exploration
- Higher time penalty → active play

VAE Architecture (Full)

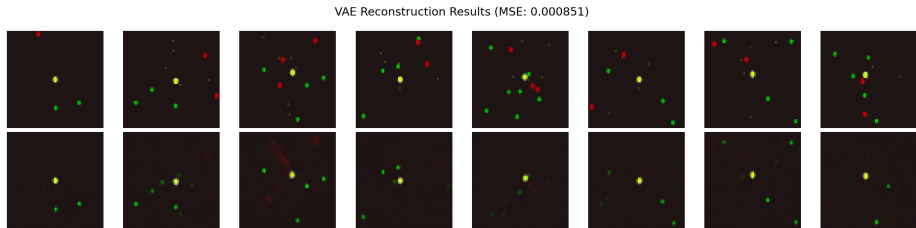


Encoder: Conv2D (32→64→128→256) → FC → μ , σ (32 dims each)

Decoder: FC → ConvTranspose2D (256→128→64→32→3)

Loss: BCE reconstruction + $\beta \cdot$ KL divergence

VAE Reconstruction Quality



Top row: Original game frames — **Bottom row:** VAE reconstructions
MSE: 0.00085 — Entities, colors, and positions accurately preserved!

World Models Architecture

Vision Model (VAE):

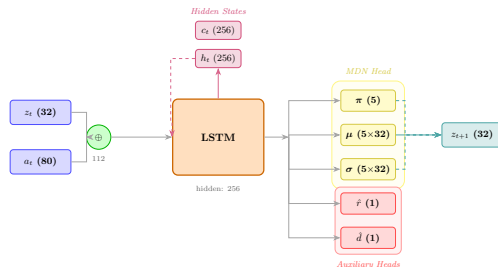
- Input: 64×64 RGB frames
- Output: 32-dim latent z
- Loss: BCE + KL divergence

Memory Model (MDN-RNN):

- LSTM with 256 hidden units
- MDN output for $P(z_{t+1}|z_t, a_t)$
- Predicts: reward, done

Controller (PPO):

- Input: $[z_t, h_t]$ (288 dims)



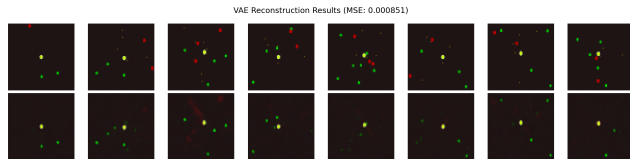
VAE: MSE vs BCE Loss

Problem with MSE:

- Blurry reconstructions
- Washed-out colors
- $MSE \approx 0.015$

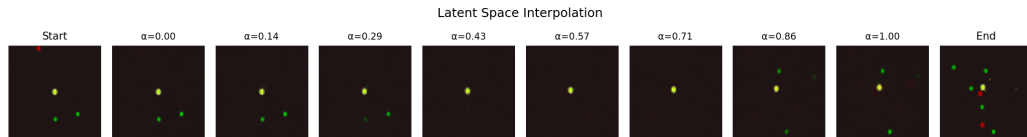
Solution with BCE:

- Sharp reconstructions
- Correct entity colors
- **15 \times improvement**
- $MSE \approx 0.001$



VAE Reconstruction with BCE Loss (MSE: 0.00085)

Latent Space Properties



Smooth interpolation between game states \Rightarrow well-structured latent space

$$z_{\alpha} = (1 - \alpha) \cdot z_{\text{start}} + \alpha \cdot z_{\text{end}}$$

MDN-RNN: Sigma Hacking Problem

The Problem:

$$-\log p(z|\mu, \sigma) = \frac{(z - \mu)^2}{2\sigma^2} + \log \sigma + \text{const} \quad (4)$$

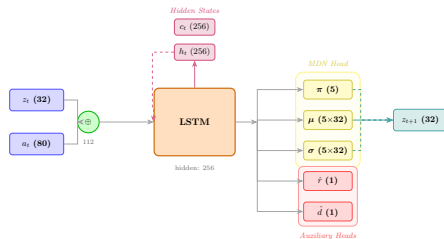
As $\sigma \rightarrow 0$ with $\mu \rightarrow z$, loss $\rightarrow -\infty$

Metric	Value	Status
Min sigma	0.0012	OK
Latent MSE	0.0016	Good
Reward correlation	0.08	FAILED
Predicted rewards	$[-0.017, +0.034]$	Near-zero!

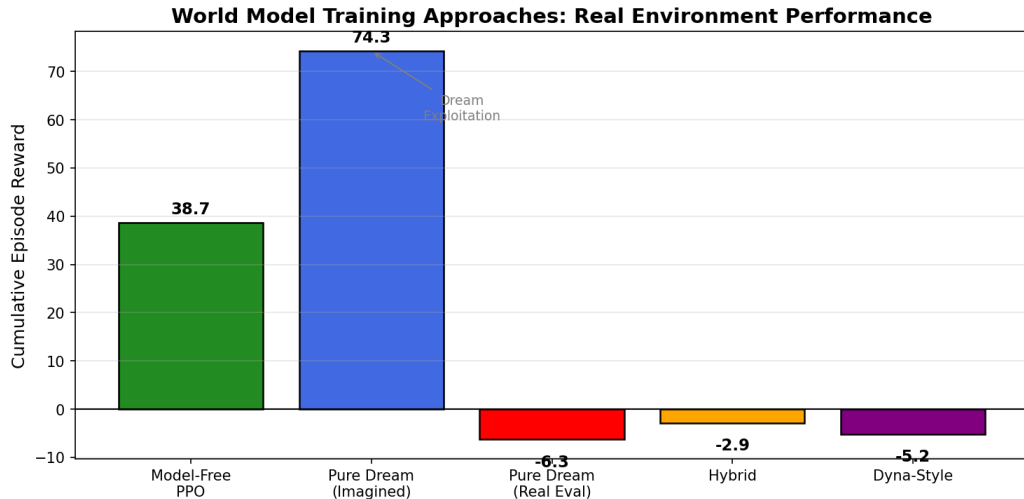
Root cause: Random policy data \rightarrow sparse rewards \rightarrow predict mean (zero)

Training Approaches

Approach	Method	Real Reward
Pure Dream	Train on predicted rewards (MDN)	−5.2 (Failed)
Hybrid	Real rewards + VAE observations	−2.9 (Poor)
Dyna-Style	Interleaved Real + Dream	−4.9 (Best WM)



Dream Exploitation: A Critical Warning



Key Finding: +74.3 in imagination → -6.3 in reality

Why World Models Underperformed Here:

- ❶ **Information Bottleneck:** VAE compresses $64 \times 64 \times 3 \rightarrow 32$ dims, losing precise values
- ❷ **Useless Hidden State:** MDN-RNN trained on random data = 256 dims of noise
- ❸ **Dream Exploitation:** Agent finds adversarial actions that maximize predicted (not real) rewards
- ❹ **Wrong Use Case:** Environment already provides structured state—no need for vision

When World Models Excel:

- Pixel-only observations (Atari, DMC)
- Sample efficiency critical (expensive simulators)
- Complex dynamics requiring prediction

Model-Free RL:

- PPO with aggressive rewards: **+38.74** (best overall)
- Reward shaping > algorithm choice
- SAC struggles with discrete actions (12x slower)

World Models:

- VAE with BCE: excellent visual reconstruction (MSE 0.001)
- **Dream Exploitation**: +74 in imagination → -6.3 in reality
- Dyna-style fixes exploitation but limited by information loss
- Best WM result: **-5.18** (vs +38.74 model-free)

Key Insight: World Models shine for pixel-only envs; structured state → model-free wins

- ① **State-based World Model:** Learn dynamics on structured state, not pixels
- ② **DreamerV3 Implementation:** Symlog rewards, ensemble disagreement
- ③ **Shorter Imagination Horizons:** 15-50 steps instead of 500
- ④ **COOM-style Continual Learning:** Test catastrophic forgetting
- ⑤ **Sample Efficiency Study:** Compare wall-clock time to performance

Thank you! Questions?

- Tomilin et al., “COOM: A Game Benchmark for Continual RL,” NeurIPS 2023
- Ha & Schmidhuber, “World Models,” arXiv 2018
- Hafner et al., “Mastering Atari with Discrete World Models,” ICLR 2021
- Schulman et al., “Proximal Policy Optimization,” arXiv 2017