RL for Hollow Knight: Silksong*

Puyuan Ye
Harvard University
puyuanye@college.harvard.edu

Dec 9, 2025

Abstract

This project trains a reinforcement learning agent to defeat the Lace boss in *Hollow Knight: Silksong* directly from screen pixels. A dueling Deep Q-Network with Noisy exploration and 10-step returns learns combat behavior through extensive reward engineering. We document three major failure modes: corner hiding, tool spamming, and passive play, and the reward modifications that resolved them. The final agent achieves a 40% win rate and deals an average of 79.2% damage per episode. Our main finding is that reward design, rather than algorithm choice, dominated development effort.

1 Introduction

Boss fights in action games present a challenging reinforcement learning setting requiring real-time perception, control, and resource management from raw visual input. Unlike standard benchmarks, real games introduce noisy observations, delayed rewards, and no access to internal state.

This project trains an RL agent to defeat the Lace boss in *Hollow Knight: Silksong*. The agent controls Hornet and must attack, dodge, heal using Silk, and deploy tools. Episodes terminate upon victory, defeat, or timeout.

2 Problem Formulation

2.1 Markov Decision Process

We model the task as an episodic MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$:

- State space $\mathcal{S}$: Raw visual observations
- Action space $\mathcal{A}$: 96 discrete action combinations
- Transitions P: Unknown and governed by the game
- Reward R: Engineered reward signal
- Discount factor $\gamma = 0.99$

*Code available at: https://github.com/AlvinYeeeeee/Silksong_RL

2.2 Observation Space

The agent observes 4 consecutive grayscale frames of size 192×192, normalized to $[-1, 1]$. Frame stacking provides short-term temporal information.

Since game memory is inaccessible, auxiliary state is extracted via pixel analysis (Table 1), including Hornet HP, Silk, and boss HP.

Feature	Range	Detection Method
Hornet HP	0–10	Brightness at fixed health bead positions
Silk	0–18	Filled segments of resource bar
Boss HP	0–100%	Ratio of health-colored pixels in HP bar

Table 1: State features extracted via pixel analysis.

Figure 1 shows a representative frame used for state extraction.

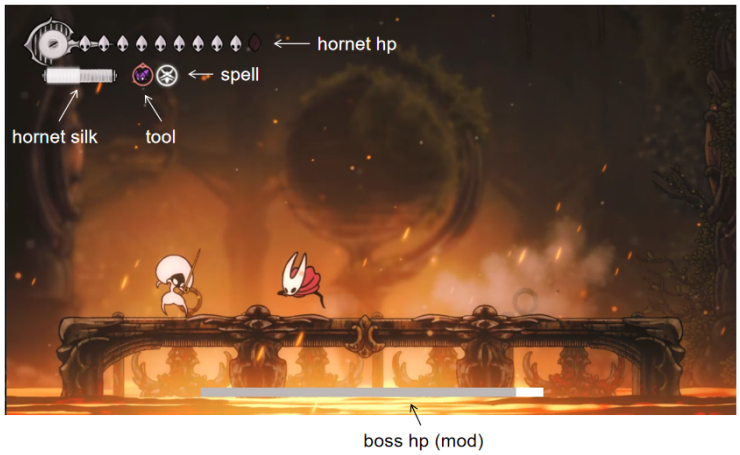

Figure 1: Representative game frame used for pixel-based state extraction. Health bars and resource indicators are parsed directly from the screen.

2.3 Action Space

The action space is factored into four independent categories:

$$|\mathcal{A}| = 3 \times 4 \times 4 \times 2 = 96 \tag{1}$$

Category	Options	Count
Move	No-op, Left, Right	3
Attack	No-op, Attack, Spell, Tool	4
Displacement	No-op, Short Jump, Long Jump, Dash	4
Heal	No-op, Heal	2

Table 2: Factored action space structure.

2.4 Episode Termination

Episodes terminate upon victory when boss HP is depleted, defeat when the death screen is detected, or timeout at 300 seconds. All termination signals are inferred from pixel observations.

2.5 Technical Implementation

The environment interfaces with the game through screen capture using `mss` and keyboard input simulation using `pydirectinput`. The system operates at approximately 6 FPS due to capture overhead.

3 Neural Network Architecture

The Q-network consists of a five-layer convolutional feature extractor followed by a dueling decision head. The CNN maps four stacked frames to an 11,520-dimensional latent representation. The dueling architecture separates state value and action advantage estimation. All fully connected layers use NoisyLinear modules to enable state-dependent exploration.

4 DQN Optimizations

We incorporate the following six improvements:

- Double DQN: Reduces Q-value overestimation by decoupling action selection and evaluation.
- Dueling DQN: Separates state value and action advantage to improve learning efficiency.
- Noisy Networks: Adds learnable noise to weights to enable state-dependent exploration.
- 10-step Returns: Propagates rewards faster across long time horizons.
- DrQ Augmentation: Applies random image augmentations to improve visual generalization.
- Mixed Precision (FP16): Increases training throughput with minimal numerical precision loss.

5 Training Process

Training begins with a short random exploration phase followed by DQN optimization using replay buffer sampling. The network updates every 4 environment steps, and the target network updates every 8,000 steps.

5.1 Hyperparameters

Parameter	Value
Replay buffer size	180,000
Batch size	32
Learning rate	8×10^{-5}
Discount γ	0.99
N-step returns	10
Target update interval	8,000 steps
Optimizer	Adam

Table 3: Training hyperparameters.

6 Reward Engineering

The primary challenge of this project was reward design rather than algorithm selection.

6.1 Stage 1: Corner Hiding

Initially, the agent learned to hide in the corner and avoid combat because damage penalties outweighed hit rewards. This was fixed by introducing a large terminal defeat penalty, inactivity penalties, and a substantial victory bonus.

6.2 Stage 2: Tool Spamming

The agent learned to deploy all tools immediately due to front-loaded damage rewards. A 5-second cooldown and penalty for invalid tool usage resolved this issue.

6.3 Stage 3: Final Tuning

Final tuning added survival rewards, improved hit rewards, heal bonuses, and movement penalties to produce sustained aggressive combat behavior.

6.4 Final Reward Structure

Event	Reward
Survival (per step)	$+0.02$
Hit boss	$+0.85 + 2.0 \times \Delta\text{HP}_{\text{boss}}$
Take damage	-0.35
Combo damage within 1.5s	-0.30
Successful heal	$+0.70$
Critical heal	$+1.50$
Tool on cooldown	-0.30
Inactivity	-0.30

Table 4: Per-step reward structure.

Outcome	Formula	Range
Victory	$18 + \frac{\text{HP}}{9} \times 8 + \frac{90}{\max(t,30)}$	+18 to +29
Defeat	Scaled by boss HP remaining	−3 to −20

Table 5: Episode reward structure.

6.5 Key Insight

All observed failure modes were rational under the current reward structure, which reinforces that reward design dominated learning behavior.

7 Results

7.1 Quantitative Results

Metric	Value
Win rate	4/10 (40%)
Average boss HP remaining	20.8%
Average damage dealt	79.2%

Table 6: Evaluation results.

7.2 Behavioral Evolution

Stage	Observed Behavior
Random policy	Dies quickly with minimal damage
After Stage 1	Actively engages boss
After Stage 2	Tools spaced across fight
Final agent	Dodges, heals, and attacks strategically; no tool usage

Table 7: Behavioral evolution.

7.3 Qualitative Observations

From video analysis[1], the trained agent exhibits active engagement, selective healing when low on HP, and consistent use of dashing and jumping during combat.

7.4 Limitations

- Win rate remains at 40 percent; reward function can be further optimized
- Training speed is limited by approximately 6 FPS capture
- No access to audio telegraphs
- Reaction time is limited by the 0.17 second action gap

[1]Evaluation video available at `https://www.bilibili.com/video/BV1jU2eBrETx/`

References