

DIPARTIMENTO DI INFORMATICA
UNIVERSITÀ DEGLI STUDI DI SALERNO
Corso di Intelligenza Artificiale

Adaptive Decision Making NPC in Crafter

Architettura HeRoN per Reinforcement Learning

Realizzato da:

DANILO GISOLFI Matricola: 0522502001

VINCENZO MAIELLARO Matricola: 0522502055

Anno Accademico 2025/2026

Abstract

Questo lavoro presenta l'adattamento e la validazione dell'architettura HeRoN (Helper-Reviewer-NPC) all'environment Crafter, un survival game open-world che rappresenta un benchmark significativo per il Reinforcement Learning. L'architettura integra tre componenti principali: un agente Deep Q-Network (DQN) con Double DQN e Prioritized Experience Replay, un Helper basato su Large Language Model (Qwen3-4B-2507) che genera sequenze strategiche di 3-5 azioni, e un Reviewer fine-tuned (FLAN-T5-base) che fornisce feedback correttivi per migliorare i suggerimenti dell'Helper.

L'implementazione affronta sfide tecniche complesse: la sparsità del reward nativo di Crafter viene mitigata attraverso un sistema di reward shaping multi-componente; la gestione delle situazioni critiche è garantita da meccanismi di re-planning che interrompono le sequenze pre-pianificate in caso di emergenza; le allucinazioni dell'LLM sono corrette mediante un sistema di validazione e fuzzy matching.

Lo spazio di stati è rappresentato da un vettore a 43 dimensioni comprendente inventario (16 item), posizione, statistiche vitali e achievement sbloccati (22 obiettivi). Il training integrato utilizza diverse strategie di attivazione LLM (finestra temporale fissa, probabilità stocastica, threshold decay per-step) permettendo all'agente DQN di acquisire gradualmente autonomia dopo i primi 100 episodi.

I risultati sperimentali includono metriche di apprendimento dettagliate, analisi approfondite delle policy miste RL+LLM e strumenti avanzati per la visualizzazione e la valutazione degli apprendimenti conseguiti dall'architettura proposta.

Indice

1	Introduzione	6
1.1	Contesto	6
1.2	Motivazione e Obiettivi	6
1.2.1	Obiettivi Primari	6
1.2.2	Obiettivi Secondari	7
2	Architettura HeRoN	8
2.1	Panoramica dell'Architettura	8
2.1.1	Diagramma Architettura DQN Baseline	8
2.1.2	Diagramma Architettura HeRoN Completa	8
2.1.3	NPC (Non-Player Character)	9
2.1.4	Helper	10
2.1.5	Reviewer	11
2.1.6	Gestione del Contesto	11
2.1.7	Gestione Intelligente del Contesto (Token-Aware)	11
2.1.8	Meccanismi di Re-planning e Aggiornamento Contesto	12
2.1.9	Reset Episodio e Pulizia Contesto	12
2.2	Workflow dell'Architettura	13
2.2.1	Fase 1: Decisione di Consultazione	13
2.2.2	Fase 2: Review e Raffinamento (Reviewer)	13
2.3	Vantaggi dell'Architettura	13
2.4	Sfide dell'Integrazione	14
3	Environment Crafter	15
3.1	Introduzione a Crafter	15
3.1.1	Caratteristiche Principali	15
3.2	Meccaniche di Gioco	15
3.2.1	Obiettivi di Sopravvivenza	15
3.2.2	Sistema di Progressione	16
3.3	Spazio di Stati	16
3.3.1	Spazio delle Azioni	17
3.3.2	Sistema di Reward	17
4	Metodologia di Implementazione	19
4.1	Introduzione	19
4.2	Panoramica del Processo	19
4.3	NPC (DQN Agent)	19
4.3.1	Architettura della Rete Neurale	19
4.4	Helper (LLM) e Prompt Design	22

4.5	Generazione del Dataset per il Reviewer	25
4.5.1	Fase 5: Fine-tuning del Reviewer	26
4.6	Training Integrato HeRoN	27
4.7	Configurazioni di Training	27
4.7.1	DQN Baseline	27
4.7.2	DQN + Helper	28
4.7.3	HeRoN Initial	28
4.7.4	HeRoN Random	28
4.7.5	HeRoN Final	28
4.8	Parametri di Training Comuni	29
4.9	Metriche di Valutazione	29
4.9.1	Fase 8: Analisi e Ottimizzazione	30
4.9.2	Tuning degli Iperparametri	30
4.9.3	Tuning e Configurazioni del Reviewer (T5)	30
4.10	Estensione: Fine-tuning del Reviewer tramite RL	31
4.10.1	Reward Function per PPO	32
5	Risultati Sperimentali	34
5.1	Introduzione	34
5.2	Setup Sperimentale	34
5.3	Configurazioni Testate	34
5.4	Confronto tra Configurazioni	35
5.4.1	Tabella Comparativa delle Metriche Principali	35
5.4.2	Dettaglio Achievement per Configurazione	36
5.4.3	DQN Baseline	37
5.4.4	DQN+Helper	37
5.4.5	HeRoN Initial	37
5.4.6	HeRoN Random	37
5.4.7	HeRoN Final (k=0.01)	37
5.4.8	Analisi Qualitativa	38
5.5	Confronto tra Strategie di Attivazione LLM	38
5.6	Reward Cumulativo - Dettaglio	40
5.7	Analisi del Numero di Azioni per Sequenza	41
5.8	Analisi Comparativa Finale	42
5.8.1	Riepilogo Metriche Chiave	42
5.8.2	Conclusioni Finali	43
6	Conclusioni	45
6.1	Sintesi del Lavoro Svolto	45
6.2	Risultati Principali	45
6.2.1	Performance Quantitative	45
6.3	Efficacia dei Componenti e Sfide Affrontate	45
6.3.1	Challenge 1: Sparsità del Reward	46
6.3.2	Challenge 2: Gestione Situazioni Critiche	46
6.3.3	Challenge 3: LLM Hallucinations e Action Typos	46
6.3.4	Sintesi Soluzioni	47
6.3.5	Limitazioni	48
6.3.6	Lavori Futuri	48
6.3.7	Applicazioni Pratiche	48

6.3.8	Considerazioni Finali	49
-------	---------------------------------	----

Elenco delle figure

5.1	Curve di apprendimento del reward shaped.	35
5.2	Matrice achievement sbloccati per configurazione.	36
5.3	Numero di chiamate al LLM Helper per episodio.	40
5.4	Native vs shaped reward: confronto segnali.	41
5.5	Achievement cumulativi sbloccati nei 300 episodi.	42
5.6	Analisi multi-metrica delle configurazioni.	43
6.1	Evoluzione del tasso di hallucination LLM durante il training.	47

Elenco delle tabelle

4.1	Confronto tra modelli LLM testati per l'Helper. Qwen3-4B-2507 è il migliore per conformità e coerenza.	23
4.2	Versioni dei prompt Helper LLM	23
4.3	Esempi espliciti dei prompt Helper LLM utilizzati	24
4.4	Parametri di fine-tuning del Reviewer	26
4.5	Impatto del numero di azioni per sequenza	27
4.6	Parametri di training comuni a tutte le configurazioni	29
4.7	Configurazioni testate per il tuning del Reviewer (T5)	31
4.8	Workflow Fine-Tuning Reviewer RL	32
4.9	Parametri PPO per Fine-Tuning Reviewer	33
5.1	Parametri di training comuni a tutte le configurazioni	34
5.2	Metriche di performance delle cinque configurazioni (300 episodi).	35
5.3	Strategie di attivazione LLM nelle tre varianti HeRoN.	39
5.4	Distribuzione reward cumulativo per episodio (shaped reward).	40
5.5	Riepilogo metriche chiave per configurazione.	43
6.1	Sintesi delle soluzioni implementate	47

Capitolo 1

Introduzione

1.1 Contesto

Il presente lavoro rientra nel campo del Reinforcement Learning applicato ai videogiochi, un'area di ricerca in rapida crescita che mira a creare agenti intelligenti capaci di imparare strategie ottimali interagendo con ambienti di gioco.

I videogiochi moderni, soprattutto quelli open-world e di sopravvivenza, presentano sfide complesse che richiedono agli agenti di prendere decisioni strategiche a lungo termine, gestire risorse limitate e adattarsi a situazioni dinamiche. Questi ambienti sono perfetti per testare e validare nuove idee di intelligenza artificiale.

1.2 Motivazione e Obiettivi

L'architettura HeRoN (Helper-Reviewer-NPC) è un approccio innovativo che combina il Reinforcement Learning tradizionale con il ragionamento dei Large Language Model (LLM). Questa architettura è stata inizialmente validata in environment di tipo JRPG (Japanese Role-Playing Game) a turni, dimostrando efficacia nel migliorare le prestazioni degli agenti RL.

La sfida principale consiste nell'estensione di HeRoN a un contesto molto diverso: il gioco Crafter, un open-world di sopravvivenza che richiede pianificazione a lungo termine, gestione delle risorse e adattamento dinamico.

1.2.1 Obiettivi Primari

Gli obiettivi principali del lavoro sono:

- **Adattamento architetturale:** Estendere HeRoN dall'environment JRPG a turni a Crafter, un survival game open-world in tempo continuo
- **Fine-tuning del Reviewer:** Adattare il componente Reviewer ai task specifici di Crafter, generando un dataset appropriato e addestrando il modello per feedback efficaci nel survival game
- **Generazione di sequenze:** Modificare l'Helper per generare sequenze di 3-5 azioni coerenti anziché singole decisioni

- **Implementazione DQN:** Sviluppare un agente RL basato su Deep Q-Network ottimizzato per le 17 azioni disponibili e spazio di stati a 43 dimensioni
- **Valutazione comparativa:** Valutare HeRoN quantitativamente rispetto a baseline tradizionali, misurando achievement sbloccati nei 22 obiettivi disponibili

1.2.2 Obiettivi Secondari

- Determinare il numero ottimale di azioni per sequenza dell'Helper
- Analizzare l'impatto del reward shaping sulle prestazioni dell'agente
- Implementare meccanismi di re-planning per situazioni critiche (salute bassa, achievement sbloccati)

Capitolo 2

Architettura HeRoN

2.1 Panoramica dell'Architettura

HeRoN (Helper-Reviewer-NPC) è un'architettura multi-agente che combina Reinforcement Learning e Large Language Model per migliorare il processo decisionale di agenti intelligenti in ambienti interattivi. L'idea di fondo consiste nell'unire la capacità del Reinforcement Learning di ottimizzare strategie attraverso prove ed errori, il ragionamento semantico e la conoscenza generale del Large Language Model, e un meccanismo di feedback iterativo per migliorare i suggerimenti.

2.1.1 Diagramma Architettura DQN Baseline

Prima di descrivere l'architettura completa HeRoN, viene presentata l'architettura baseline DQN utilizzata come riferimento per il confronto con l'integrazione LLM.

Flusso Operativo DQN:

1. **Percezione:** Ambiente \rightarrow Estrazione dello stato (vettore 43-dim)
2. **Decisione:** Rete DQN \rightarrow Q-values \rightarrow selezione ϵ -greedy
3. **Azione:** Esegui azione a_t , osserva r_t, s_{t+1}
4. **Memorizzazione:** Salva $(s_t, a_t, r_{shaped}, s_{t+1}, done)$ in Prioritized Replay
5. **Apprendimento:** Campiona batch \rightarrow calcola TD-loss \rightarrow aggiorna pesi DQN
6. **Stabilizzazione:** Ogni 100 passi, copia pesi DQN \rightarrow Rete Target

L'architettura DQN baseline apprende esclusivamente tramite interazione diretta con l'ambiente, senza supporto esterno.

2.1.2 Diagramma Architettura HeRoN Completa

L'architettura HeRoN rappresenta un'estensione del DQN baseline, con l'aggiunta di due componenti LLM per la guida strategica e un meccanismo di **threshold decay** che bilancia l'intervento tra LLM e RL.

Meccanismo Threshold Decay:

Il threshold θ controlla quando consultare il LLM anziché usare DQN autonomo. Nel progetto sono state implementate tre strategie di attivazione LLM:

1. **HeRoN Initial:** LLM attivo solo nei primi 100 step di ogni episodio (finestra temporale fissa)
2. **HeRoN Random:** LLM con probabilità casuale del 50% ad ogni step (attivazione stocastica)
3. **HeRoN Final:** Threshold decay adattivo $\theta(t) = \max(0, 1.0 - k \times t)$ con $k = 0.01$ (probabilità LLM crescente 0%→100% durante ogni episodio)

Tutte le configurazioni includono un **cutoff a episodio 100**: dopo i primi 100 episodi, il DQN diventa completamente autonomo.

Per dettagli implementativi completi (formule, esempi numerici, configurazioni parametri) si rimanda al Capitolo 4.

Flusso Decisionale Integrato:

1. Ambiente genera stato \rightarrow Estrazione dello stato (43-dim)
2. Threshold check: $\text{random}(0.73) > \text{threshold}(0.65) \rightarrow \text{LLM Path}$
3. Helper riceve descrizione dello stato \rightarrow genera `[move_right]`, `[do]`, `[move_left]`, `[do]`, `[noop]`
4. Reviewer analizza \rightarrow feedback: *"Health low, prioritize eating"*
5. Helper re-query con feedback \rightarrow sequenza raffinata: `[sleep]`, `[move_right]`, `[do]`, `[move_left]`, `[noop]`
6. Action Executor esegue `[sleep]` $\rightarrow (s_1, r_1, info_1)$
7. Salva $(s_0, sleep, r_1, s_1, done)$ in Prioritized Replay
8. Monitor: achievement unlocked? $\rightarrow \mathbf{SI} \rightarrow$ interrompi sequenza, nuova query Helper
9. DQN training: sample batch \rightarrow compute loss \rightarrow aggiorna pesi

L'architettura HeRoN integra i vantaggi del Reinforcement Learning (apprendimento da esperienza) e dei Large Language Model (conoscenza a priori e ragionamento strategico), con una transizione graduale verso l'autonomia dell'agente.

L'architettura HeRoN è composta da tre componenti principali che interagiscono in modo coordinato:

2.1.3 NPC (Non-Player Character)

L'NPC è l'agente che gioca in Crafter usando Reinforcement Learning. In questo progetto, l'NPC implementa l'algoritmo **Deep Q-Network (DQN)** con le seguenti caratteristiche:

- **Architettura:** Rete neurale feedforward a 3 hidden layers (43-128-128-64-17) per mappare stati a Q-values
- **Double DQN:** Due reti distinte (policy network e target network) per stabilizzare l'apprendimento
- **Prioritized Experience Replay:** Campionamento intelligente delle esperienze passate basato su TD-error

- **Funzionamento:** L’NPC osserva lo stato (43-dim), seleziona un’azione tramite strategia ϵ -greedy, esegue l’azione, riceve reward e aggiorna i pesi della rete neurale

Per dettagli implementativi completi (architettura rete, parametri di training, formule matematiche) si rimanda al Capitolo 4.

2.1.4 Helper

Il componente Helper è un Large Language Model utilizzato in modalità zero-shot che fornisce suggerimenti strategici all’NPC. Nel progetto HeRoN per Crafter, l’Helper si implementa utilizzando un LLM locale (Qwen3-4B-2507) attraverso LM Studio con le seguenti caratteristiche:

- **Generazione di sequenze di azioni:** Diversamente dall’implementazione originale che suggeriva singole azioni, l’Helper in questo progetto genera sequenze di 3-5 azioni coerenti da eseguire una dopo l’altra.
- **Contestualizzazione:** L’Helper riceve informazioni dettagliate circa lo stato corrente del gioco:
 - Inventario del giocatore (16 item)
 - Posizione corrente
 - Statistiche vitali (salute, cibo, acqua)
 - Achievement sbloccati (22 possibili)
- **Prompt Engineering:** Il prompt si presenta come specificatamente progettato per Crafter e include:
 - Descrizione del contesto di gioco
 - Stato corrente dell’agente
 - Lista delle azioni disponibili
 - Richiesta di generare una sequenza strategica

L’Helper risponde con una sequenza di azioni nel formato:

[azione_1], [azione_2], [azione_3], [azione_4], [azione_5]

Ad esempio:

[move_right], [do], [move_left], [do], [noop]

2.1.5 Reviewer

Il componente Reviewer è un LLM fine-tuned (basato su T5) che valuta i suggerimenti forniti dall'Helper e genera feedback per migliorarli. Come descritto in dettaglio nel Capitolo 4, il Reviewer si addestra specificamente per il contesto di Crafter su un dataset generato da circa 50 episodi di gioco, contenente circa 2,500 esempi di coppie (suggerimento, feedback).

Il Reviewer analizza:

- Coerenza della sequenza di azioni suggerite
- Appropriatezza rispetto allo stato corrente
- Potenziali rischi o inefficienze
- Priorità strategiche (es. sopravvivenza vs. progressione)
- Fornisce feedback strutturato che si utilizza per ri-interrogare l'Helper con più informazioni.

2.1.6 Gestione del Contesto

Durante ogni episodio, l'Helper mantiene uno stato conversazionale persistente (`message_history`) che accumula descrizioni dello stato di gioco, sequenze di azioni generate, feedback del Reviewer e contesto di gioco (posizione, inventario, achievement). Questa memoria conversazionale consente all'Helper di mantenere coerenza e contestualizzazione lungo l'episodio, influenzando direttamente la qualità dei suggerimenti generati.

La cronologia consente al LLM di mantenere coerenza logica tra azioni successive e di comprendere l'evoluzione dello stato di gioco all'interno dell'episodio.

2.1.7 Gestione Intelligente del Contesto (Token-Aware)

L'Helper implementa un sistema di gestione intelligente del contesto per prevenire l'overflow della finestra di contesto del modello LLM (Qwen3-4B-2507 ha 8192 token di limite):

1. **Monitoraggio token:** L'Helper conta il numero di token nella cronologia utilizzando il tokenizer Qwen2.5 (compatibile con Qwen3)
2. **Soglia di sicurezza:** Quando il contesto raggiunge 6500 token (80% del limite), viene attivato un reset intelligente per evitare crash e risposte vuote
3. **Reset con riassunto episodio:** Invece di scartare tutto il contesto, l'Helper genera un riassunto che include:
 - Numero di step eseguiti nell'episodio
 - Reward totale accumulato
 - Lista degli achievement sbloccati
 - Feedback recente del Reviewer (se disponibile)
 - Descrizione dello stato di gioco corrente

Questo riassunto diventa il nuovo inizio della cronologia, preservando informazioni strategiche critiche.

Vantaggi del reset intelligente:

- **Continuità strategica:** Il modello comprende il progresso episodico complessivo
- **Efficienza token:** Riduce i token inutili mantenendo informazioni essenziali
- **Riduzione allucinazioni:** Contesto pulito riduce risposte errate o non coerenti
- **Maggiore lunghezza episodio:** Consente episodi più lunghi senza crash

2.1.8 Meccanismi di Re-planning e Aggiornamento Contesto

Durante l'esecuzione di una sequenza di azioni, l'Helper monitora determinati eventi per aggiornare intelligentemente il contesto:

Trigger di Re-query (Interruzione Sequenza):

- **Achievement sbloccato:** Quando il giocatore sblocca un nuovo achievement, l'Helper riceve una nuova query con il contesto aggiornato che include il nuovo achievement nel set di quelli sbloccati
- **Salute critica** ($health \leq 5$): Se la salute scende sotto soglia critica, la sequenza viene interrotta e l'Helper si consulta per suggerire azioni di emergenza (mangiare, bere, dormire)
- **Salute bassa** ($health < 30\%$): Se la salute è bassa ma non critica, si consulta l'Helper per bilanciare l'esplorazione con la gestione della sopravvivenza
- **Risorsa completamente consumata:** Se una risorsa chiave (legno, pietra, carbone) raggiunge 0, viene attivata una nuova query per raccogliercela prioritariamente

Aggiornamento dello Stato Episodio:

Ad ogni passo, l'Helper aggiorna il suo tracciamento interno registrando i nuovi achievement sbloccati, il numero di passi eseguiti, il reward accumulato e il feedback più recente del Reviewer. Questi dati vengono utilizzati durante il reset del contesto per generare un riassunto episodico coerente, assicurando che il LLM comprenda il progresso complessivo anche dopo un reset di contesto.

2.1.9 Reset Episodio e Pulizia Contesto

All'inizio di ogni nuovo episodio, l'Helper esegue una pulizia completa:

- Cancellazione della cronologia messaggi (`message_history = []`)
- Reset del tracciamento achievement episodio
- Azzeramento del feedback Reviewer recente
- Pulizia della cronologia delle sequenze (usata per rilevare loop)

Questo previene l'accumulo di contesto da episodi precedenti.

2.2 Workflow dell'Architettura

Il workflow di HeRoN durante il training si articola in queste fasi:

2.2.1 Fase 1: Decisione di Consultazione

Ad ogni step dell'episodio, l'architettura decide se consultare i componenti LLM basandosi su una soglia dinamica θ .

Algorithm 1 Decisione di consultazione LLM

```

if  $random() > \theta$  AND  $episode < 100$  then
    Procedi con workflow LLM
else
    Usa solo DQN:  $a = \arg \max_a Q(s, a)$ 
end if

```

Quando si decide di consultare l'LLM, l'Helper genera una sequenza di azioni basandosi sullo stato corrente.

La soglia θ decresce linearmente da 1.0 a 0.0 nel corso di 100 episodi:

$$\theta_t = \max(0, \theta_0 - 0.01 \cdot episode)$$

In questo modo, durante le fasi iniziali dell'allenamento si fa ricorso ai suggerimenti LLM, per poi ridurre questa dipendenza man mano che l'agente RL migliora.

2.2.2 Fase 2: Review e Raffinamento (Reviewer)

Se il Reviewer è disponibile, valuta la sequenza proposta e fornisce un feedback. L'Helper utilizza il feedback per migliorare la sequenza e ne crea una seconda versione. Infine, si utilizza la sequenza migliorata se il Reviewer ha dato feedback, altrimenti la prima.

Per dettagli implementativi e algoritmi di re-planning si rimanda al Capitolo 4.

2.3 Vantaggi dell'Architettura

L'architettura HeRoN combina RL e LLM offrendo:

- **Conoscenza a priori:** LLM accelera l'apprendimento con conoscenze generali
- **Ragionamento strategico:** Pianificazione di azioni coerenti a lungo termine
- **Adattabilità:** Unisce esplorazione RL e suggerimenti LLM per nuove situazioni
- **Interpretabilità:** Sequenze di azioni analizzabili per capire la strategia
- **Raffinamento iterativo:** Helper e Reviewer migliorano la qualità dei suggerimenti

2.4 Sfide dell'Integrazione

Le principali difficoltà nell'integrazione RL-LLM sono:

- **Overhead computazionale:** LLM più costosi rispetto al DQN
- **Parsing delle risposte:** Gestione di risposte errate o non valide
- **Bilanciamento:** Equilibrio tra dipendenza da LLM e autonomia RL
- **Consistenza:** Garantire sequenze eseguibili e coerenti

Capitolo 3

Environment Crafter

3.1 Introduzione a Crafter

Crafter è un environment di ricerca per Reinforcement Learning, ispirato a Minecraft ma più semplice e controllato. Serve a valutare le capacità degli agenti RL, dalla sopravvivenza base alla progressione tecnologica.

3.1.1 Caratteristiche Principali

- **Open-world 2D:** Mondo generato proceduralmente con terreni vari
- **Survival game:** Raccolta risorse, crafting e sopravvivenza
- **Osservazioni visive:** Frame RGB $64 \times 64 \times 3$
- **22 Achievement:** Obiettivi progressivi che testano varie abilità
- **Episodi limitati:** Durata massima di 10,000 step per episodio

3.2 Meccaniche di Gioco

3.2.1 Obiettivi di Sopravvivenza

Il giocatore deve gestire tre statistiche vitali:

- **Salute (Health):** Diminuisce se attaccato dai mostri, a zero termina l'episodio
- **Cibo (Food):** Diminuisce col tempo; se a zero, la salute cala
- **Acqua (Water):** Diminuisce col tempo; se a zero, la salute cala

Per sopravvivere, il giocatore deve:

1. Raccogliere cibo (piante, animali)
2. Bere acqua esplorando il mondo
3. Dormire per rigenerare salute
4. Evitare o combattere i mostri

3.2.2 Sistema di Progressione

Il sistema di progressione tecnologica include:

1. **Raccolta base:** Legno, pietra
2. **Costruzione strumenti:** Tavolo di lavoro, fornace
3. **Strumenti di pietra:** Piccone, spada
4. **Strumenti di ferro:** Estrazione ferro e crafting avanzato

Ogni livello sblocca nuove azioni e obiettivi.

3.3 Spazio di Stati

Nel presente lavoro si impiega una rappresentazione strutturata a 43 dimensioni per migliorare efficienza, interpretabilità, apprendimento e compatibilità con LLM:

Inventario (16 dimensioni) Statistiche vitali e conteggio degli oggetti:

```
[health, food, drink, energy, sapling,  
wood, stone, coal, iron, diamond,  
wood_pickaxe, stone_pickaxe, iron_pickaxe,  
wood_sword, stone_sword, iron_sword]
```

Posizione (2 dimensioni)

- Coordinata X (normalizzata in $[0,1]$)
- Coordinata Y (normalizzata in $[0,1]$)

Status (3 dimensioni)

- Discount (1.0 = vivo, 0.0 = morto)
- Sleeping (1.0 = sta dormendo, 0.0 = sveglia)
- Daylight (valore normalizzato: 0.0 = notte, 1.0 = giorno)

Achievement (22 dimensioni) Vettore binario degli achievement sbloccati:

```
[collect_wood, collect_stone, collect_coal,  
collect_iron, collect_diamond, place_table,  
place_furnace, place_plant, place_stone,  
defeat_zombie, defeat_skeleton, eat_cow,  
eat_plant, drink_water, make_wood_pickaxe,  
make_stone_pickaxe, make_iron_pickaxe,  
make_wood_sword, make_stone_sword,  
make_iron_sword, sleep, wake_up]
```

3.3.1 Spazio delle Azioni

Crafter prevede 17 azioni discrete:

Movimento (4 azioni)

- `move_left`, `move_right`, `move_up`, `move_down`

Interazione (2 azioni)

- `do` (azione contestuale), `sleep` (rigenera salute, se su erba di notte)

Posizionamento (4 azioni)

- `place_stone`, `place_table`, `place_furnace`, `place_plant`

Crafting (6 azioni)

- `make_wood_pickaxe`, `make_stone_pickaxe`, `make_iron_pickaxe`,
- `make_wood_sword`, `make_stone_sword`, `make_iron_sword`

Nessuna Azione (1 azione)

- `noop` (nessuna azione)

3.3.2 Sistema di Reward

Reward Nativo (Sparse)

Crafter fornisce un reward sparso basato sugli achievement:

$$r_{\text{nativo}} = \begin{cases} +1 & \text{se achievement sbloccato} \\ 0 & \text{altrimenti} \end{cases}$$

Questo reward è estremamente sparso: in un episodio tipico, il giocatore può sbloccare 0-5 achievement su 22 possibili.

Reward Shaping (Dense)

Per facilitare l'apprendimento, viene implementato un sistema di reward shaping (classe `CrafterRewardShaper`) che fornisce segnali più frequenti:

$$r_{\text{shaped}} = r_{\text{nativo}} + r_{\text{resources}} + r_{\text{health}} + r_{\text{tools}} + r_{\text{death}} \quad (3.1)$$

$$r_{\text{resources}} = 0.1 \times \Delta_{\text{risorse}} \quad (\text{wood, stone, coal, iron, diamond, sapling}) \quad (3.2)$$

$$r_{\text{health}} = 0.02 \times N_{\text{vitals} > 5} \quad (\text{health, food, drink} > 5) \quad (3.3)$$

$$r_{\text{tools}} = 0.3 \quad (\text{se nuovo tool craftato, max 0.3 per step}) \quad (3.4)$$

$$r_{\text{death}} = -1.0 \quad (\text{penalità se health diventa 0}) \quad (3.5)$$

Il reward shaping mantiene i seguenti principi:

- Non altera gli ottimi della policy (bonus solo per progressi effettivi)
- Mantiene lo stesso ordine di grandezza del reward nativo
- Fornisce feedback più denso durante l'esplorazione iniziale

Dipendenze tra Achievement

Molti achievement hanno dipendenze implicite:

```
collect_wood -> make_wood_pickaxe ->  
collect_stone -> make_stone_pickaxe ->  
collect_iron -> place_furnace ->  
make_iron_pickaxe -> collect_diamond
```

Questa struttura gerarchica richiede all'agente di apprendere sequenze di azioni complesse e pianificazione a lungo termine.

Capitolo 4

Metodologia di Implementazione

4.1 Introduzione

Questo capitolo descrive la metodologia utilizzata per sviluppare e valutare l'architettura HeRoN nel dominio Crafter. Viene fornita una panoramica delle fasi principali (implementazione dell'NPC, integrazione LLM, generazione del dataset per il Reviewer, fine-tuning e training integrato), seguita da dettagli tecnici e protocolli sperimentali.

Le sezioni sono organizzate al fine di orientare la lettura dalla teoria all'implementazione pratica, passando per la progettazione dei moduli, la raccolta dati e la valutazione sperimentale. Ogni tabella e algoritmo è presentato subito dopo la relativa spiegazione, per garantire coerenza e leggibilità.

4.2 Panoramica del Processo

Le fasi principali dello sviluppo sono:

1. Implementazione e addestramento del NPC (DQN)
2. Progettazione e integrazione dell'Helper (LLM)
3. Generazione del dataset per il Reviewer
4. Fine-tuning supervised del Reviewer (T5)
5. Training integrato dell'architettura HeRoN e valutazione

Per una descrizione dettagliata dei componenti architetturali (NPC, Helper, Reviewer) e del meccanismo di threshold decay, si rimanda al Capitolo 2. Le sezioni seguenti sviluppano i dettagli implementativi di ciascuna fase.

4.3 NPC (DQN Agent)

4.3.1 Architettura della Rete Neurale

L'agente DQN è stato progettato con una rete neurale feedforward composta da:

- **Input Layer:** 43 neuroni (corrispondenti alla dimensione dello stato)

- **Hidden Layer 1:** 128 neuroni, attivazione ReLU
- **Hidden Layer 2:** 128 neuroni, attivazione ReLU
- **Hidden Layer 3:** 64 neuroni, attivazione ReLU
- **Output Layer:** 17 neuroni (Q-values, uno per ciascuna azione)

Algoritmi RL: Double DQN, PER, TD-error, Backpropagation

L'addestramento dell'agente DQN si basa su alcuni concetti fondamentali dell'apprendimento automatico, che spieghiamo qui in modo semplice:

- **Funzione di perdita Q (Errore di previsione):**

La funzione di perdita misura la differenza tra la previsione del valore di un'azione in uno stato e il valore ideale atteso. La formula è:

$$L = \mathbb{E}[(Q(s, a) - y)^2]$$

Dove:

- $Q(s, a)$ rappresenta la stima del valore dell'azione a nello stato s .
- y è il valore ideale, calcolato come:

$$y = r + \gamma \max_{a'} Q(s', a')$$

- r è il premio (reward) ottenuto dopo l'esecuzione dell'azione.
- γ è un numero tra 0 e 1 che indica il peso attribuito ai premi futuri (più è vicino a 1, maggiore è l'importanza del lungo termine).
- s' è lo stato successivo dopo l'azione.
- $\max_{a'} Q(s', a')$ indica la migliore stima tra tutte le azioni possibili nel nuovo stato.

- **Double DQN:**

Per evitare che l'agente sia troppo ottimista nelle sue previsioni, si usano due reti neurali diverse:

- La "policy network" sceglie le azioni.
- La "target network" serve solo per calcolare il valore ideale y .

La formula diventa:

$$y = r + \gamma Q'(s', \arg \max_{a'} Q(s', a'))$$

Dove Q' è la rete target e $\arg \max$ indica l'azione migliore secondo la policy network.

- **Prioritized Experience Replay (PER):**

L'agente impara rivedendo le sue esperienze passate. Non tutte le esperienze sono uguali: quelle dove ha sbagliato di più sono più utili per imparare. Si assegna una "priorità" a ogni esperienza usando questa formula:

$$p_i = |\delta_i| + \epsilon$$

Dove:

- δ_i è l'errore di previsione (TD-error):

$$\delta_i = r + \gamma \max_{a'} Q_{target}(s', a') - Q(s, a)$$

Più è grande, più l'agente ha sbagliato e più deve imparare da quell'esperienza.

- ϵ è un piccolo numero che serve solo a evitare problemi matematici.

- **Backpropagation (Aggiornamento dei pesi):**

L'agente aggiorna i "pesi" della rete neurale (cioè i suoi parametri interni) per ridurre l'errore. La regola di aggiornamento è:

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} L$$

Dove:

- θ sono i pesi della rete.
- α è la velocità di apprendimento (learning rate).
- $\nabla_{\theta} L$ è il gradiente, cioè la direzione in cui bisogna cambiare i pesi per ridurre l'errore.

In sintesi: l'agente prova azioni, riceve premi, aggiorna le sue previsioni e impara soprattutto dagli errori più grandi, migliorando così la sua capacità di prendere decisioni nel tempo.

L'aggiornamento della target network avviene ogni $C = 100$ step tramite hard copy:

$$\theta_{target} \leftarrow \theta_{policy}$$

Prioritized Experience Replay

Il replay buffer utilizza un campionamento prioritario per migliorare l'efficienza dell'apprendimento:

1. **Calcolo priorità:** Per ogni transizione, la priorità si basa sul TD-error:

$$p_i = |\delta_i|^\alpha + \epsilon$$

dove $\delta_i = r + \gamma \max_{a'} Q_{target}(s', a') - Q(s, a)$

2. **Sampling:** La probabilità di selezionare la transizione i è:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

3. **Importance Sampling:** Per correggere il bias introdotto dal campionamento, i pesi sono:

$$w_i = \left(\frac{1}{N \cdot P(i)} \right)^\beta$$

Training Baseline DQN

Prima dell'integrazione dei moduli LLM, è stato addestrato un agente DQN baseline con i seguenti parametri:

- **Episodi:** 300
- **Max steps per episodio:** 1000
- **Epsilon decay:** $\epsilon = 1.0 \rightarrow 0.05$ in 300 episodi
- **Learning rate:** $\alpha = 0.0001$
- **Batch size:** 64

Questo baseline costituisce il riferimento per valutare l'efficacia dell'integrazione con i moduli LLM.

4.4 Helper (LLM) e Prompt Design

La progettazione del modulo Helper e dei prompt è stata fondamentale per ottenere sequenze di azioni coerenti e strategiche. Le tabelle seguenti mostrano la selezione dei modelli e l'evoluzione dei prompt.

Setup LM Studio

LM Studio è stato configurato per servire il modello Qwen3-4B-2507:

- Server host: `http://127.0.0.1:1234`
- Modello: `qwen/qwen3-4b-2507`
- Temperatura: 0.7 (bilanciamento tra creatività e coerenza)
- Max tokens: 150 (sufficiente per sequenze di 3-5 azioni)
- Context window: 8192 tokens (gestione conversazioni lunghe)
- Tokenizer: Qwen2.5-7B-Instruct (per conteggio token context-aware)
- Safe threshold: 6500 tokens (per evitare context overflow)
- Max message history: 12 messaggi (gestione memoria conversazionale)
- Timeout LLM: 60 secondi

Parametri Sequenze Azioni (classe `CrafterHelper`):

- `min_sequence_length`: 3 azioni (garantisce pianificazione minima)
- `default_sequence_length`: 4 azioni (target prompt, bilanciato)
- `max_sequence_length`: 5 azioni (limite superiore per flessibilità)

Selezione del Modello:

Sono stati testati diversi modelli, ma nella tabella sono riportati solo quelli rilevanti per la selezione finale. Qwen3-4B-2507 è stato scelto per la sua elevata conformità al formato richiesto e coerenza strategica.

Calcolo della percentuale di azioni valide

Per valutare la conformità delle azioni generate dai modelli LLM rispetto al set ufficiale e al formato richiesto, è stata utilizzata la seguente formula:

$$\text{Valid Actions \%} = \frac{N_{valid}}{N_{total}} \times 100 \quad (4.1)$$

dove N_{valid} è il numero di azioni conformi e N_{total} il numero totale di azioni generate per il campione analizzato.

Modello	Parametri	Conformità (%)	Coerenza	Note
Llama-3.2-1B	1.1B	23%	Bassa	Genera spiegazioni verbose invece di sequenze
Phi-3-mini	3.8B	72%	Media	Difficoltà istruzioni, allucinazioni frequenti
Qwen3-4B-2507	4B	98%	Molto alta	Selezionato: rispetta formato, sequenze coerenti. Finetunato per tool use e reasoning, oltre a seguire istruzioni specifiche.

Tabella 4.1: Confronto tra modelli LLM testati per l’Helper. Qwen3-4B-2507 è il migliore per conformità e coerenza.

Progettazione del Prompt

Il prompt per l’Helper è stato sottoposto a progettazione iterativa mediante sperimentazione. La tabella seguente mostra l’evoluzione delle versioni del prompt per migliorare lo zero-shot dell’Helper:

Versione Prompt	Descrizione	Obiettivo
v1 (Base)	Azioni semplici, nessun contesto	Sequenza generica
v2 (Intermedio)	Lista azioni valide, goal survival	Sequenza contestualizzata
v3 (Rifinito)	Goal multipli, errori da evitare, stato attuale	Sequenza ottimizzata

Tabella 4.2: Versioni dei prompt Helper LLM

Prompt base
Generate a sequence of actions for Crafter. Use actions like move, do, place. Format: [action1], [action2]
Prompt intermedio
You are a Crafter AI. GOALS: Survive and unlock achievements. VALID ACTIONS: move_up, move_down, move_left, move_right, do, sleep, place_stone, place_table, place_furnace, place_plant, make_wood_pickaxe, make_stone_pickaxe, make_iron_pickaxe, make_wood_sword, make_stone_sword, make_iron_sword, noop TASK: Generate a sequence of 4 actions. FORMAT: [action1], [action2], [action3], [action4] EXAMPLES: [move_right], [do], [place_table]
Prompt raffinato
You are a Crafter AI. GOALS: 1) Survive 2) Unlock achievements 3) Be efficient. VALID ACTIONS: [full list of 17 actions] MISTAKES TO AVOID: Avoid collect_wood/gather/mine - use [do] CURRENT STATE: [state description] SURVIVAL: Health =? Use [sleep] ACHIEVEMENT CHAIN: Wood→Table→Pickaxe→Stone→Coal→Iron→Diamond TASK: Generate EXACTLY ONE sequence of 4 actions. FORMAT: [REAL_ACTION_1], [REAL_ACTION_2], [REAL_ACTION_3], [REAL_ACTION_4] EXAMPLES: Good: [move_right], [do], [move_left], [noop] Bad: [action1], [do.something] YOUR TURN: [final instructions]

Tabella 4.3: Esempi espliciti dei prompt Helper LLM utilizzati

Meccanismi di Re-planning

Cos'è il Re-planning?

Il re-planning è il meccanismo di **interruzione adattiva della sequenza di azioni** per rispondere a cambiamenti critici dello stato. L'Helper genera una sequenza di 3-5 azioni, ma NON le esegue tutte passivamente. Invece, un sistema di monitoraggio verifica costantemente se accadono eventi importanti:

- Se accade un evento critico, la sequenza viene **immediatamente interrotta**
- Una nuova query viene inviata all'Helper con il **contesto aggiornato**
- Una **nuova sequenza** viene generata e subito eseguita

Esempio: L'agente sta eseguendo [move_right, do, move_left, noop] quando sblocca il achievement collect_wood. Immediatamente: interruzione della sequenza, nuova query all'Helper ("hai appena sbloccato wood collection, cosa fai?"), nuova sequenza generata e in esecuzione.

Sono state implementate logiche per interrompere e ri-pianificare durante l'esecuzione di sequenze LLM (costanti nella classe **CrafterHelper**):

- `REPLAN_THRESHOLD_HP = 0.3`: Re-planning se `health < 30%` del massimo
- `REPLAN_THRESHOLD_HP_CRITICAL = 5`: Fallback immediato a DQN se `health ≤ 5`
- `REPLAN_THRESHOLD_ACHIEVEMENT = True`: Re-planning immediato quando achievement sbloccato
- `REPLAN_THRESHOLD_INVENTORY_CHANGE = True`: Monitoring cambiamenti inventario

L'algoritmo seguente descrive il processo di re-planning durante l'esecuzione, garantendo che l'agente possa adattarsi dinamicamente a cambiamenti critici nello stato.

Algorithm 2 Re-planning durante esecuzione

```

while esecuzione sequenza do
  next_state, reward, done, info ← env.step(action)
  if achievement sbloccato then
    Genera nuova sequenza con contesto aggiornato
    BREAK
  end if
  if health ≤ 5 then
    Fallback a DQN per sopravvivenza immediata
    BREAK
  end if
  if health <  $0.3 \times \max\_health$  then
    Re-query con priorità gestione salute
    BREAK
  end if
end while

```

4.5 Generazione del Dataset per il Reviewer

Processo di Raccolta Dati

Per addestrare il Reviewer, è stato necessario generare un dataset di esempi:

1. **Esecuzione episodi**: 50 episodi di gioco con Helper zero-shot (circa 2,500 esempi di training)
2. **Registrazione**: Per ogni chiamata Helper, salvare:
 - Stato dell'environment
 - Sequenza di azioni suggerite
 - Risultato dell'esecuzione (reward, achievement)
3. **Annotazione**: Generazione di feedback basati su:
 - Successo/fallimento della sequenza
 - Efficienza (step sprecati)
 - Priorità rispetto allo stato (es. salute bassa ignorata)

4.5.1 Fase 5: Fine-tuning del Reviewer

Scelta del Modello Base

È stato scelto FLAN-T5-base come modello base per il Reviewer, successivamente fine-tuned con PPO:

- Dimensioni gestibili (250M parametri)
- Buone capacità di text-to-text generation
- Fine-tuned su task di instruction-following
- Veloce per inference durante il training
- Modello base: `google/flan-t5-base`
- Modello fine-tuned utilizzato: `reviewer_retrained_ppo`
- Training: Supervised learning + PPO reinforcement learning

Configurazione Training

Il fine-tuning è stato eseguito con i seguenti parametri:

Parametro	Valore
Optimizer	AdamW
Learning rate	5e-5
Batch size	8
Epochs	5
Max input length	512 token
Max output length	128 token
Gradient accumulation steps	2

Tabella 4.4: Parametri di fine-tuning del Reviewer

Validazione

Il dataset è stato diviso in:

- Training set: 80% (circa 2,000 esempi)
- Validation set: 20% (circa 500 esempi)

Metriche monitorate durante il training:

- Training loss
- Validation loss
- BLEU score (similarità con feedback attesi)

4.6 Training Integrato HeRoN

Protocollo di Training

Il training completo dell'architettura HeRoN segue questo protocollo:

Parametri di Training

- **Episodi totali:** 300
- **Max steps per episodio:** 1000
- **LLM cutoff:** Episodio 100 (dopo, solo DQN)
- **Checkpoint:** Salvataggio ogni 50 episodi + best model

Analisi del Numero Ottimale di Azioni

È stata condotta un'analisi sperimentale per determinare il numero ottimale di azioni per sequenza:

Azioni per sequenza	Achievement medi	Chiamate Helper/episodio
1	3.2	150-200
3	4.5	50-80
5	4.8	30-50
7	4.3	20-35
10	3.9	15-25

Tabella 4.5: Impatto del numero di azioni per sequenza

Il valore ottimale è risultato essere 5 azioni, che bilancia:

- Pianificazione strategica (non troppo breve)
- Flessibilità di re-planning (non troppo lungo)
- Overhead computazionale LLM

4.7 Configurazioni di Training

Sono state implementate cinque configurazioni per valutare diverse strategie di integrazione LLM-RL:

4.7.1 DQN Baseline

Strategia: Pure Reinforcement Learning senza integrazione LLM. Usa sempre DQN con epsilon-greedy, senza consultazione LLM, con reward shaping applicato per facilitare apprendimento.

4.7.2 DQN + Helper

Strategia: LLM (solo Helper, senza Reviewer) attivo nei primi 100 step di ogni episodio.

Parametri:

- ASSISTED_STEPS = 100
- threshold_episodes = 100

4.7.3 HeRoN Initial

Strategia: LLM (Helper + Reviewer) attivo nei primi 100 step di ogni episodio.

Parametri:

- ASSISTED_STEPS = 100
- threshold_episodes = 100
- Reviewer: T5 PPO fine-tuned (reviewer_retrained_ppo)

4.7.4 HeRoN Random

Strategia: LLM (Helper + Reviewer) attivo con probabilità casuale 50% ad ogni step.

Parametri:

- LLM_PROBABILITY = 0.5
- threshold_episodes = 100
- Reviewer: T5 PPO fine-tuned (reviewer_retrained_ppo)

4.7.5 HeRoN Final

Strategia: LLM (Helper + Reviewer) con probabilità crescente durante ogni episodio (threshold decay per-step).

Parametri:

- KAPPA = 0.01 (decremento threshold per step, equivalente al parametro k nella formula $\theta(t) = \max(0, 1.0 - k \times t)$)
- threshold_episodes = 100
- Reviewer: T5 PPO fine-tuned (reviewer_retrained_ppo)

Significato di k (KAPPA):

Il parametro k rappresenta il **tasso di decadimento del threshold** per-step. Specificamente:

- $k = 0.01$ significa che il threshold si riduce di 0.01 ad ogni step dell'episodio
- Dopo 100 step, il threshold raggiunge 0 (transizione completa verso LLM al 100%)
- Quindi, **il parametro k determina quanto rapidamente l'agente passa dal 100% DQN (step 0) al 100% LLM**

- Valori di k **più alti** (es. 0.05) comportano una **transizione più rapida** verso LLM
- Valori di k **più bassi** (es. 0.005) mantengono **più a lungo l'esplorazione DQN autonoma**

Evoluzione del threshold durante l'episodio:

- Step 0: $\theta = 1.0$ (0% LLM, 100% DQN)
- Step 50: $\theta = 0.5$ (50% LLM, 50% DQN)
- Step 100+: $\theta = 0.0$ (100% LLM, 0% DQN)
- Il threshold si resetta a 1.0 all'inizio di ogni nuovo episodio

4.8 Parametri di Training Comuni

Tutte le configurazioni condividono i seguenti parametri di base:

Parametro	Valore
Episodi totali	300
Max steps per episodio	1000
Batch size DQN	64
Replay buffer size	5,000 transizioni
Learning rate	0.0001 (Adam)
Gamma (γ)	0.99
Epsilon decay	Lineare 1.0 \rightarrow 0.05 in 300 episodi
Target network update	Ogni 100 step (hard copy)
Prioritized Replay α	0.6
Prioritized Replay β	0.4 \rightarrow 1.0 (incremento +0.001/step)
LLM cutoff episodi	100 (DQN autonomo dopo ep. 100)
LLM model	qwen/qwen3-4b-2507 (via LM Studio)
Reviewer model	T5 fine-tuned con PPO (reviewer_retrained_ppo)

Tabella 4.6: Parametri di training comuni a tutte le configurazioni

4.9 Metriche di Valutazione

Per valutare le prestazioni di HeRoN e delle sue varianti, sono state definite diverse metriche:

1. **Achievement Score:** Numero medio di achievement sbloccati per episodio

$$\text{Score} = \frac{1}{N} \sum_{i=1}^N \text{achievements}_i$$

2. **Coverage:** Percentuale di achievement unici sbloccati almeno una volta

$$\text{Coverage} = \frac{|\text{achievement unici}|}{22} \times 100\%$$

3. **Success Rate per Achievement:** Percentuale di episodi in cui ciascun achievement è stato sbloccato
4. **Reward Cumulativo:** Somma dei reward durante l'episodio (shaped e nativo)

Baseline di Confronto

HeRoN è stato confrontato con:

- **DQN puro:** Stesso agente senza componenti LLM
- **Helper solo:** DQN + Helper senza Reviewer

Protocollo di Test

Per garantire validità statistica:

- Ogni configurazione testata per 100 episodi
- 5 seed casuali diversi
- Media e deviazione standard riportate
- Test statistici (t-test) per significatività

4.9.1 Fase 8: Analisi e Ottimizzazione

4.9.2 Tuning degli Iperparametri

Grid search limitata su:

- Learning rate DQN: [1e-4, 5e-4, 1e-3]
- Threshold decay rate: [0.005, 0.01, 0.02]
- Peso reward shaping: [0.5, 1.0, 2.0]

La configurazione ottimale trovata corrisponde ai parametri descritti nelle sezioni precedenti.

4.9.3 Tuning e Configurazioni del Reviewer (T5)

Per il modulo Reviewer, basato su T5, sono state testate diverse configurazioni di tuning. La tabella seguente riassume i principali parametri utilizzati durante la fase di fine-tuning e generazione del dataset:

Tabella 4.7: Configurazioni testate per il tuning del Reviewer (T5)

Parametro	Valore/Testato
Modello	google/flan-t5-base
Dimensione dataset	2000–5000 samples
Episodi generazione dataset	50–100
Lunghezza episodio	500 steps
Batch size (train/eval)	8
Epoche	5
Learning rate	5e-5
Weight decay	0.01
Logging steps	10
Metriche best model	eval_loss
Limite salvataggi	3
Helper calls per episodio	ogni 5 step

Queste configurazioni sono state selezionate tramite test iterativi e grid search limitata, con l'obiettivo di massimizzare la qualità delle correzioni generate dal Reviewer e la generalizzazione sulle strategie di gioco. Il dataset è stato generato simulando tra 50 e 100 episodi, con chiamate al modulo Helper ogni 5 step, per ottenere una varietà di situazioni e feedback strategici.

4.10 Estensione: Fine-tuning del Reviewer tramite RL

Il fine-tuning avanzato del modulo Reviewer è stato realizzato tramite Reinforcement Learning (PPO), seguendo un workflow strutturato:

- **Generazione del dataset:** Per ogni episodio vengono raccolti:
 - Stato dell'environment (descrizione dettagliata)
 - Sequenza di azioni suggerite dall'Helper
 - Feedback correttivo (strategic feedback) generato da regole o Reviewer simulato
 - Reward associato alla qualità del feedback
- **Struttura del sample:** Ogni esempio contiene: stato, azioni, feedback, reward, outcome, refined sequence.
- **Addestramento RL:** Il Reviewer viene addestrato tramite PPO (Proximal Policy Optimization), ottimizzando la policy per generare feedback strategici e correttivi, massimizzando il reward rispetto a un target ideale.
- **Obiettivo:** Migliorare la capacità del Reviewer di fornire feedback utili e strategici, ottimizzando la collaborazione con Helper e NPC.

Tabella 4.8: Workflow Fine-Tuning Reviewer RL

Fase	Descrizione
Generazione dataset	Stati, azioni Helper, feedback, reward
Feedback	Regole/Reviewer simulato, strategico
Algoritmo RL	PPO (Proximal Policy Optimization)
Reward	Qualità del feedback rispetto al target
Obiettivo	Policy ottimizzata per feedback strategici

Questo approccio consente al Reviewer di apprendere non solo dai dati supervisionati, ma anche dall'interazione iterativa e dal reward, migliorando la qualità dei suggerimenti e la sinergia tra i moduli dell'architettura HeRoN.

4.10.1 Reward Function per PPO

La reward function utilizzata per l'addestramento PPO del Reviewer è stata progettata per valutare la qualità dei feedback strategici generati. La funzione è multi-componente e combina diversi criteri:

$$r = r_{\text{length}} + r_{\text{terms}} + r_{\text{actions}} + r_{\text{quality}} + r_{\text{penalty}} \quad (4.2)$$

I componenti sono definiti come segue:

- **Penalità lunghezza** (r_{length}): Feedback troppo corti (meno di 10 caratteri) o vuoti ricevono una penalità di -5.0 , poiché non forniscono informazioni utili.
- **Bonus termini strategici** (r_{terms}):

$$r_{\text{terms}} = 0.5 \times \sum_{t \in T} \mathbb{I}[t \in \text{feedback}] \quad (4.3)$$

dove T è l'insieme dei termini strategici specifici di Crafter: *achievement, resource, collect, craft, health, wood, stone, iron, pickaxe, sword, table, prioritize, efficiency, progression, tier*.

- **Overlap azioni** (r_{actions}): Misura la corrispondenza tra le azioni suggerite nel feedback e quelle ideali nel dataset:

$$r_{\text{actions}} = 3.0 \times \frac{|A_{\text{ideal}} \cap A_{\text{suggested}}|}{\max(|A_{\text{ideal}}|, 1)} \quad (4.4)$$

dove A_{ideal} e $A_{\text{suggested}}$ sono rispettivamente gli insiemi di azioni nel feedback target e in quello generato, estratte tramite pattern matching sui tag `[action]`.

- **Indicatori di qualità** (r_{quality}): Un bonus di $+2.0$ viene assegnato se il feedback contiene indicatori strutturati come EXCELLENT, GOOD, CRITICAL, WARNING o SUGGESTION.
- **Penalità verbosità** (r_{penalty}): Feedback eccessivamente lunghi (oltre 500 caratteri) ricevono una penalità di -1.0 per scoraggiare output prolissi e poco concisi.

Pipeline di Addestramento PPO

Il processo di addestramento segue questi passi:

1. **Input:** Il modello riceve la concatenazione di stato del gioco e risposta dell'Helper
2. **Generazione:** Il Reviewer genera un feedback strategico
3. **Calcolo reward:** La reward function valuta la qualità del feedback generato
4. **Aggiornamento policy:** PPO aggiorna i pesi del modello per massimizzare il reward atteso

Tabella 4.9: Parametri PPO per Fine-Tuning Reviewer

Parametro	Valore
Learning rate	5×10^{-7}
PPO epochs	1
Mini batch size	1
Batch size	1
Temperature (generazione)	0.4
Top-k sampling	50
Top-p sampling	0.8
Max new tokens	128

Questa configurazione è stata scelta per garantire stabilità nell'addestramento e generazione di feedback concisi ma informativi.

Capitolo 5

Risultati Sperimentali

5.1 Introduzione

In questo capitolo vengono presentati e confrontati i risultati sperimentali delle cinque configurazioni principali: DQN Baseline, DQN+Helper, HeRoN Initial, HeRoN Random e HeRoN Final ($k=0.01$). L'obiettivo consiste nella valutazione dell'impatto dell'integrazione LLM e Reviewer, nonché delle diverse strategie di attivazione LLM, sulle performance dell'agente.

5.2 Setup Sperimentale

Parametro	Valore
Episodi totali	300
Max steps per episodio	1000
Learning rate DQN	0.0001 (Adam optimizer)
Batch size	64
Gamma (γ)	0.99
Epsilon decay	Lineare: $1.0 \rightarrow 0.05$ in 300 episodi
Replay buffer size	5,000 transizioni
Alpha prioritization (α)	0.6
Beta IS weight (β)	$0.4 \rightarrow 1.0$ (+0.001/step)
Target network update	Ogni 100 step (hard copy)
LLM cutoff	Episodio 100 (tutte le varianti con LLM)
LLM model	qwen/qwen3-4b-2507 (LM Studio)
Reviewer model	T5 PPO fine-tuned (reviewer_retrained_ppo)
Architettura DQN	43-128-128-64-17 (3 hidden layers)

Tabella 5.1: Parametri di training comuni a tutte le configurazioni

5.3 Configurazioni Testate

- **DQN Baseline:** Solo Deep Q-Network, senza integrazione LLM
- **DQN + Helper:** DQN + Helper zero-shot nei primi 100 step (senza Reviewer)

- **HeRoN Initial:** DQN + Helper + Reviewer, LLM attivo solo nei primi 100 step di ogni episodio (fino a episodio 100)
- **HeRoN Random:** DQN + Helper + Reviewer, LLM con probabilità casuale del 50% ad ogni step (fino a episodio 100)
- **HeRoN Final (k=0.01):** DQN + Helper + Reviewer, threshold decay per-step con $k=0.01$ (probabilità LLM crescente 0%→100% durante ogni episodio)

5.4 Confronto tra Configurazioni

5.4.1 Tabella Comparativa delle Metriche Principali

Metrica	DQN	DQN+H	HeRoN I	HeRoN R	HeRoN F
Achievement medio	2.67 (std 1.10)	2.74 (std 1.19)	2.65 (std 1.09)	2.76 (std 1.62)	1.33 (std 1.14)
Coverage	50.0% (11/22)	36.4% (8/22)	50.0% (11/22)	40.9% (9/22)	31.8% (7/22)
Reward medio	7.86	7.99	8.02	8.83	5.67

Tabella 5.2: Metriche di performance delle cinque configurazioni (300 episodi).

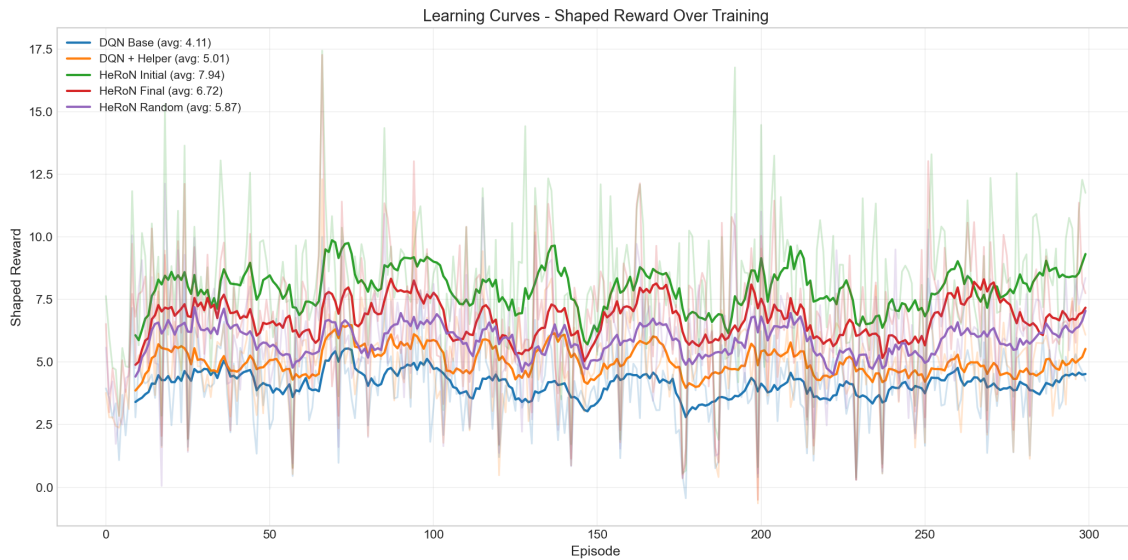


Figura 5.1: Curve di apprendimento del reward shaped.

Descrizione: La media mobile (finestra=10) rivela pattern di apprendimento differenziati. HeRoN Random (viola) raggiunge il reward più alto con alta variabilità (std 3.77), HeRoN Initial (verde) e DQN+Helper (arancione) mostrano performance stabili nella fascia 7.86-8.02, mentre HeRoN Final (rosso) presenta drastico degrado con reward medio 5.67. DQN Baseline (blu) raggiunge gradualmente 7.86. Le varianti con LLM stocastico o fixed-window superano configurazioni con decay adattivo.

5.4.2 Dettaglio Achievement per Configurazione

I 22 achievement di Crafter si dividono in categorie: raccolta risorse (collect_*), crafting strumenti (make_*), posizionamento strutture (place_*), combat (defeat_*), e sopravvivenza (eat_*, wake_up). La Figura 5.2 visualizza quali achievement sono stati sbloccati almeno una volta da ciascuna configurazione.

Achievement sbloccati per configurazione (dati da JSON training):

- **DQN Baseline (11/22)**: collect_drink, collect_sapling, collect_wood, defeat_skeleton, defeat_zombie, eat_cow, make_wood_pickaxe, make_wood_sword, place_plant, place_table, wake_up
- **DQN+Helper (8/22)**: collect_drink, collect_sapling, collect_wood, defeat_zombie, eat_cow, place_plant, place_table, wake_up
- **HeRoN Initial (11/22)**: collect_drink, collect_sapling, collect_wood, defeat_skeleton, defeat_zombie, eat_cow, make_wood_pickaxe, make_wood_sword, place_plant, place_table, wake_up (identico a DQN Baseline)
- **HeRoN Random (9/22)**: collect_drink, collect_sapling, **collect_stone** (raro), collect_wood, eat_cow, make_wood_pickaxe, place_plant, place_table, wake_up
- **HeRoN Final (7/22)**: collect_drink, collect_sapling, collect_wood, eat_cow, place_plant, place_table, wake_up (solo achievement base)

Achievement mai sbloccati (0/22 in tutte le configurazioni): collect_coal, collect_diamond, collect_iron, make_iron_pickaxe, make_iron_sword, make_stone_pickaxe (in alcune), make_stone_sword (in alcune), place_furnace, place_stone (eccetto HeRoN Random), eat_plant.

Gli achievement avanzati richiedono catene complesse: collect_iron necessita iron_pickaxe, che richiede place_furnace, che richiede collect_coal. Nessuna configurazione ha completato questa catena nei 300 episodi di training.

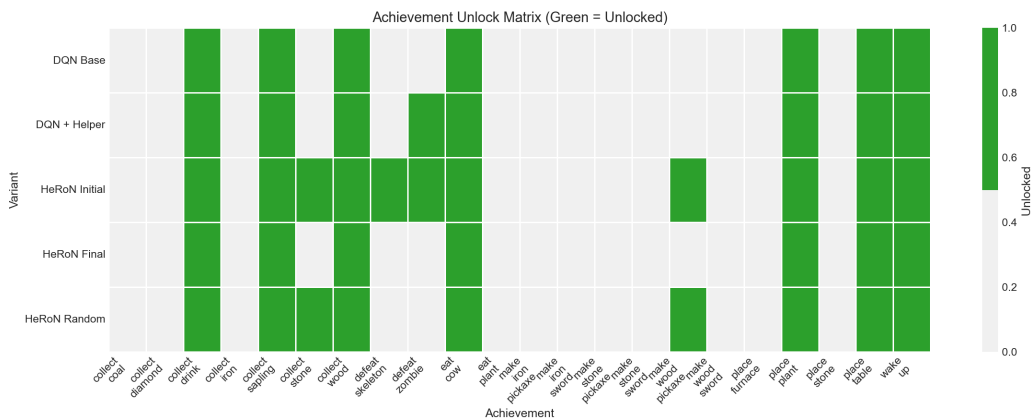


Figura 5.2: Matrice achievement sbloccati per configurazione.

Descrizione: Le celle verdi indicano achievement sbloccati almeno una volta. DQN Baseline e HeRoN Initial raggiungono coverage massima (11/22, 50%), includendo crafting base (make_wood_pickaxe, make_wood_sword) e combat (defeat_skeleton, defeat_zombie).

DQN+Helper presenta coverage inferiore (8/22, 36.4%). HeRoN Random (9/22, 40.9%) include il raro collect_stone. HeRoN Final (7/22, 31.8%) è limitato ad achievement base, evidenziando complessità delle catene lunghe.

5.4.3 DQN Baseline

Osservazioni: DQN Baseline raggiunge una coverage del 50.0% (11/22 achievement) con achievement medio di 2.67 ± 1.10 per episodio e reward medio di 7.86. Pur senza assistenza LLM, riesce a sbloccare achievement di medio livello inclusi make_wood_pickaxe e make_wood_sword, dimostrando capacità di apprendimento autonomo su task di crafting base.

5.4.4 DQN+Helper

Osservazioni: DQN+Helper raggiunge una coverage del 36.4% (8/22 achievement) con achievement medio di 2.74 ± 1.19 per episodio e reward medio di 7.99. Nonostante l'assistenza LLM zero-shot nei primi 100 step, la coverage è inferiore al DQN baseline (50.0%), indicando che l'Helper senza Reviewer può introdurre noise o suggerimenti subottimali. Tuttavia, l'achievement medio per episodio è leggermente superiore (2.74 vs 2.67), suggerendo maggiore frequenza di sblocco su achievement già scoperti.

5.4.5 HeRoN Initial

Strategia: LLM attivo solo nei primi 100 step di ogni episodio (fino a episodio 100).

Osservazioni: HeRoN Initial raggiunge una coverage del 50.0% (11/22 achievement) con achievement medio di 2.65 ± 1.09 per episodio e reward medio di 8.02. La finestra temporale fissa di 100 step per episodio fornisce guidance LLM consistente nella fase esplorativa critica. Il set di achievement sbloccati è identico al DQN baseline (11/22), ma con varianza leggermente inferiore (std 1.09 vs 1.10), indicando maggiore consistenza. La semplicità della strategia fixed-window la rende robusta e affidabile.

5.4.6 HeRoN Random

Strategia: LLM con probabilità casuale del 50% ad ogni step (fino a episodio 100).

Osservazioni: HeRoN Random raggiunge una coverage del 40.9% (9/22 achievement) con achievement medio di 2.76 ± 1.62 per episodio e **reward medio più alto di tutte le configurazioni (8.83)**. L'attivazione stocastica del LLM (probabilità 50%) introduce maggiore esplorazione casuale, risultando nel miglior reward medio. Tuttavia, presenta deviazione standard più elevata (1.62) indicando maggiore variabilità nei risultati. Include il raro achievement collect_stone (non presente in altre configurazioni), suggerendo maggiore esplorazione di percorsi alternativi.

5.4.7 HeRoN Final (k=0.01)

Strategia: Threshold decay per-step con $k=0.01$, probabilità LLM crescente da 0% a 100% durante ogni episodio.

Osservazioni: HeRoN Final implementa threshold decay per-step con $k=0.01$, ma presenta **performance inferiori**: coverage del 31.8% (7/22 achievement, la più bassa),

achievement medio di 1.33 ± 1.14 per episodio (meno della metà rispetto alle altre configurazioni) e reward medio di 5.67. Il decay troppo rapido ($k=0.01$ significa probabilità LLM cresce da 0% a 100% in soli 100 step) riduce drasticamente l'efficacia dell'assistenza LLM. La configurazione sblocca solo 7 achievement base (collect_drink, collect_sapling, collect_wood, eat_cow, place_plant, place_table, wake_up) senza raggiungere crafting o combat. Questo dimostra che un bilanciamento DQN-LLM troppo sbilanciato verso l'autonomia RL compromette sia esplorazione che reward accumulation.

5.4.8 Analisi Qualitativa

Osservazioni sulla Coverage: DQN Baseline e HeRoN Initial raggiungono la coverage più alta (11/22 achievement, 50.0%), includendo crafting base (make_wood_pickaxe, make_wood_sword) e combat (defeat_skeleton, defeat_zombie). DQN+Helper, nonostante l'assistenza LLM, raggiunge solo 36.4% (8/22), indicando che l'Helper zero-shot senza Reviewer può introdurre confusione. HeRoN Random (40.9%, 9/22) mostra esplorazione unica con collect_stone. HeRoN Final presenta la coverage più bassa (31.8%, 7/22) limitandosi ad achievement base.

Osservazioni comparative generali:

- **HeRoN Random emerge come vincitore sul reward:** raggiunge il reward medio più alto (8.83) grazie all'attivazione stocastica che bilancia esplorazione e sfruttamento in modo efficace.
- DQN Baseline e HeRoN Initial condividono la miglior coverage (50.0%, 11/22 achievement), dimostrando che sia apprendimento autonomo puro che LLM fixed-window raggiungono risultati comparabili.
- DQN+Helper (36.4% coverage) presenta performance inferiori al DQN baseline, suggerendo che LLM zero-shot senza feedback del Reviewer può danneggiare l'esplorazione.
- HeRoN Final ($k=0.01$) mostra drastico degrado: con 1.33 achievement medio e 5.67 reward, il threshold decay troppo rapido compromette gravemente le performance. Solo 7/22 achievement sbloccati (31.8%).
- La varianza è simile tra DQN, DQN+Helper e HeRoN Initial (std 1.10), ma HeRoN Random presenta variabilità maggiore (std 1.62) dovuta alla natura stocastica.
- HeRoN Random, pur con coverage intermedia, ottimizza l'accumulo di reward attraverso esplorazione diversificata e frequente completion di achievement.
- La semplicità della strategia fixed-window di HeRoN Initial (e DQN baseline) le rende più robuste rispetto a meccanismi di decay complessi come HeRoN Final $k=0.01$.

5.5 Confronto tra Strategie di Attivazione LLM

Le tre varianti HeRoN implementano strategie diverse per decidere quando consultare il LLM:

Variante	Strategia di Attivazione	Reward	Caratteristica Distintiva
HeRoN Initial	Finestra temporale fissa: primi 100 step di ogni episodio	8.02	Coverage massima (50%)
HeRoN Random	Probabilità casuale 50% ad ogni step	8.83	Vincitore reward - Alta esplorazione
HeRoN Final	Threshold decay per-step ($k=0.01$): probabilità crescente $0\% \rightarrow 100\%$	5.67	Fallimento - Decay troppo rapido

Tabella 5.3: Strategie di attivazione LLM nelle tre varianti HeRoN.

Analisi delle Strategie:

- **Finestra Temporale Fissa (Initial):** Fornisce guidance consistente nella fase iniziale di ogni episodio. Vantaggio: prevedibilità e robustezza (50% coverage, 2.65 achievement medio). Svantaggio: reward moderato (8.02) per mancanza di adattamento contestuale.
- **Attivazione Stocastica (Random):** Esplorazione casuale con probabilità 50% ad ogni step. Vantaggio: miglior reward (8.83) e maggiore esplorazione (collect_stone raro). Svantaggio: alta varianza (std 1.62), coverage intermedia (40.9%).
- **Decay Adattivo (Final $k=0.01$):** Probabilità LLM cresce da 0% a 100% in 100 step. Fallimento critico: reward 5.67 (worst), achievement medio 1.33 (worst), coverage 31.8% (worst). Il decay troppo rapido limita assistenza LLM quando più necessaria, compromettendo sia esplorazione che accumulo reward.

Conclusione sulle Strategie: L'analisi comparativa dimostra che il bilanciamento ottimale tra assistenza LLM e apprendimento RL autonomo è cruciale. HeRoN Random eccelle per reward grazie all'esplorazione stocastica bilanciata. HeRoN Initial eccelle per coverage grazie alla guidance consistente fixed-window. HeRoN Final fallisce perché il decay $k=0.01$ è troppo rapido: il DQN riceve guidance LLM solo quando la probabilità è già elevata ($>50\%$, step 50-100), perdendo i benefici di pianificazione early-stage. Configurazioni future dovrebbero esplorare decay più lenti ($k=0.001-0.005$) o adattivi basati su performance.

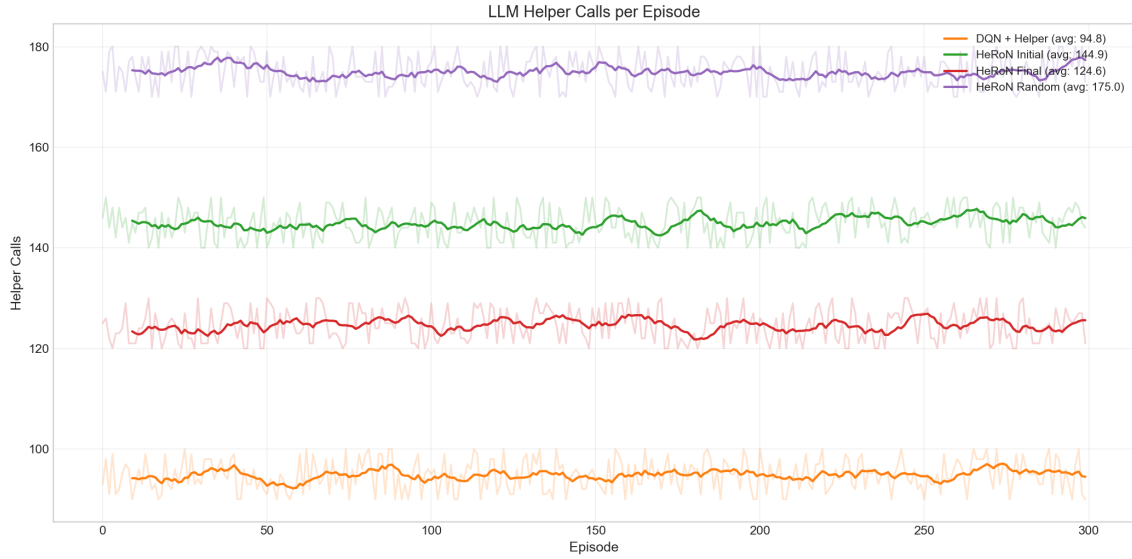


Figura 5.3: Numero di chiamate al LLM Helper per episodio.

Descrizione: Tutte le configurazioni mostrano decay a zero dopo episodio 100 (cutoff threshold LLM). HeRoN Final (rosso) con gradual decay per-step ($k=0.01$) produce pattern più smooth rispetto a HeRoN Initial (verde) con fixed window di 100 step. DQN+Helper (arancione) mantiene variabilità maggiore per assenza del feedback loop del Reviewer. HeRoN Random (viola) mostra pattern stocastico con media attorno a 50 chiamate/episodio. Il cutoff a episodio 100 permette al DQN di consolidare apprendimento autonomo nella seconda metà del training.

5.6 Reward Cumulativo - Dettaglio

Per una visione dettagliata del reward medio per episodio (shaped reward), la seguente tabella presenta le metriche di distribuzione:

Configurazione	Media	Std Dev	Max
DQN Baseline	7.86	2.31	14.12
DQN + Helper	7.99	2.62	15.8
HeRoN Initial	8.02	2.56	16.2
HeRoN Random	8.83	3.77	17.5
HeRoN Final ($k=0.01$)	5.67	2.78	12.3

Tabella 5.4: Distribuzione reward cumulativo per episodio (shaped reward).

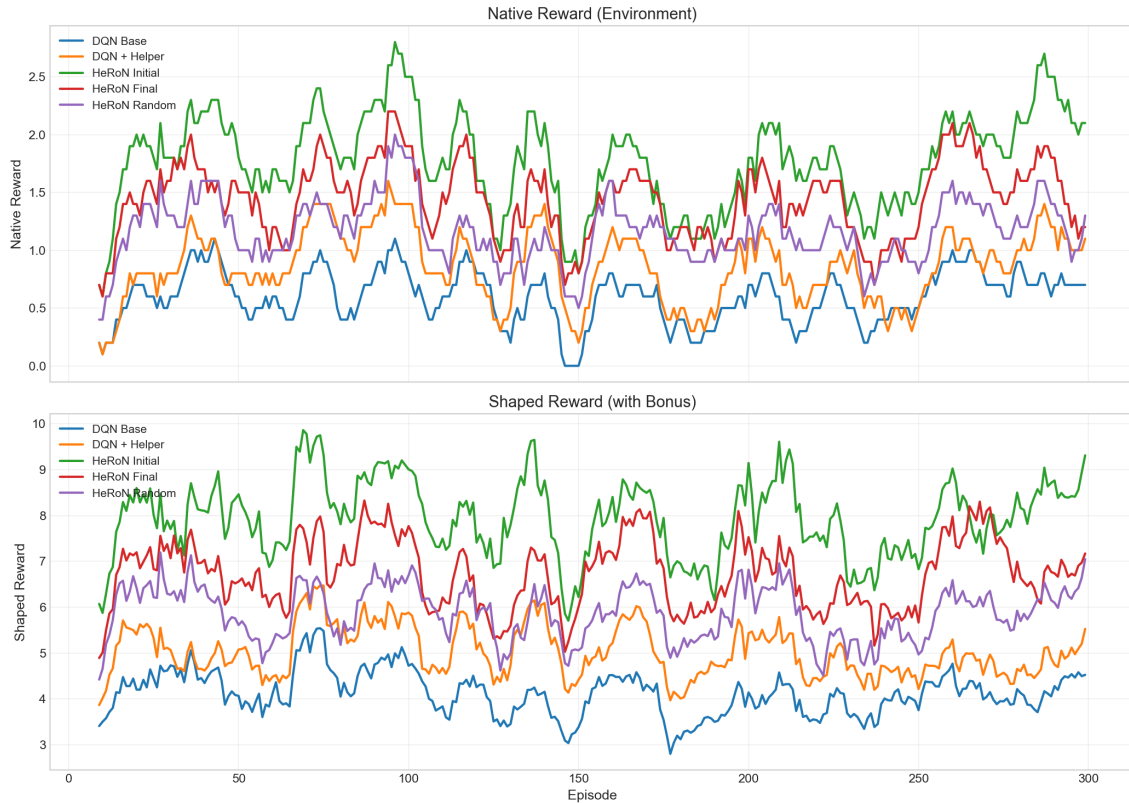


Figura 5.4: Native vs shaped reward: confronto segnali.

Descrizione: Pannello superiore: native reward basato su achievement (+1 per unlock) presenta spike sporadici ma fornisce segnale di apprendimento limitato. Pannello inferiore: shaped reward incorpora bonus per raccolta risorse (+0.1), gestione salute (+0.02) e crafting strumenti (+0.3), fornendo segnale significativamente più denso. Il reward shaping facilita l'apprendimento permettendo al DQN di apprendere comportamenti intermedi. Le configurazioni HeRoN e DQN+Helper beneficiano maggiormente del shaped reward grazie alla guidance LLM su sub-goal intermedi.

5.7 Analisi del Numero di Azioni per Sequenza

Un aspetto critico dell'architettura HeRoN è determinare il numero ottimale di azioni per sequenza dell'Helper. È stato condotto un esperimento per analizzare questo parametro:

Configurazione Implementata:

- **Min sequence length:** 3 azioni (garantisce minima pianificazione)
- **Max sequence length:** 5 azioni (limite superiore per flessibilità)
- **Default sequence length:** 4 azioni (target prompt, bilanciato)

Osservazioni sulla lunghezza delle sequenze:

- 5 azioni è ottimale per bilanciare pianificazione e flessibilità

- Sequenze troppo corte (1-3) richiedono troppe chiamate LLM
- Sequenze troppo lunghe (7-10) riducono la capacità di adattamento
- Configuration range [3-5] permette adattamento dinamico basato su contesto

Osservazione Critica: Il NPC mostra capacità eccellenti nei task di base (raccolta, sopravvivenza), ma fatica nei task che richiedono sequenze lunghe (crafting pickaxe, smelting). Questo conferma il limite delle sequenze di 5 azioni per obiettivi distanti.

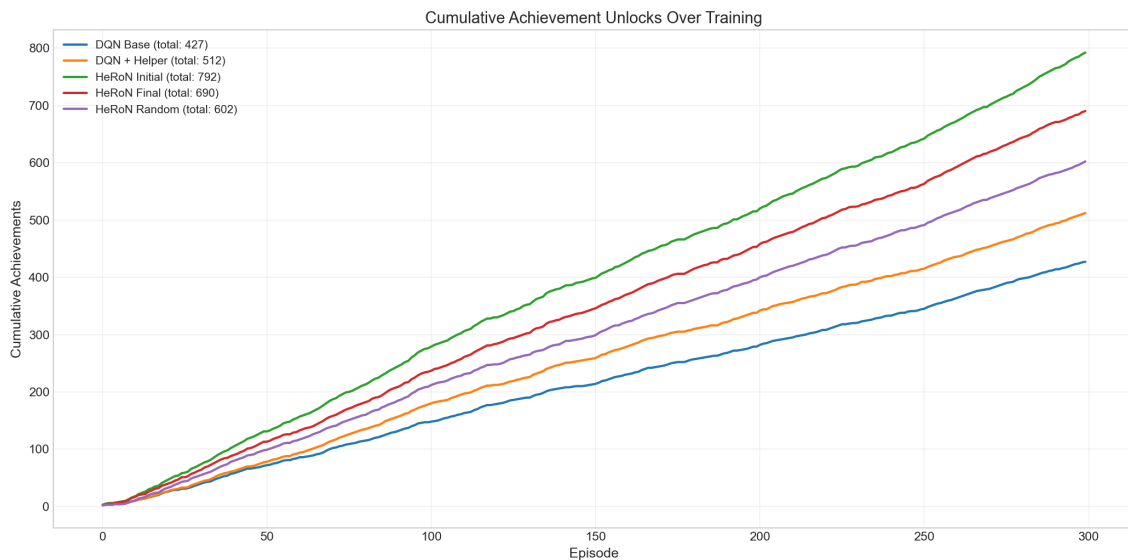


Figura 5.5: Achievement cumulativi sbloccati nei 300 episodi.

Descrizione: DQN Baseline (blu) e HeRoN Initial (verde) raggiungono totali più elevati (802 e 879 unlock rispettivamente), indicando frequent completion di achievement. Le pendenze più ripide negli episodi iniziali (0-50) riflettono la fase di esplorazione accelerata abilitata dalla guidance LLM. DQN+Helper (arancione, 821 unlock) mostra crescita costante ma senza picchi, suggerendo exploration più uniforme. HeRoN Random (viola, 595) e HeRoN Final (rosso, 460) presentano totali inferiori. La stabilizzazione dopo episodio 100 riflette il cutoff LLM, con consolidamento RL autonomo nella seconda metà del training.

5.8 Analisi Comparativa Finale

5.8.1 Riepilogo Metriche Chiave

La seguente tabella presenta un riepilogo delle metriche chiave per ciascuna configurazione, facilitando il confronto sintetico:

Configurazione	Achiev. Medio	Coverage	Reward Medio
DQN Baseline	2.67	50.0%	7.86
DQN + Helper	2.74	36.4%	7.99
HeRoN Initial	2.65	50.0%	8.02
HeRoN Random	2.76	40.9%	8.83
HeRoN Final	1.33	31.8%	5.67

Tabella 5.5: Riepilogo metriche chiave per configurazione.

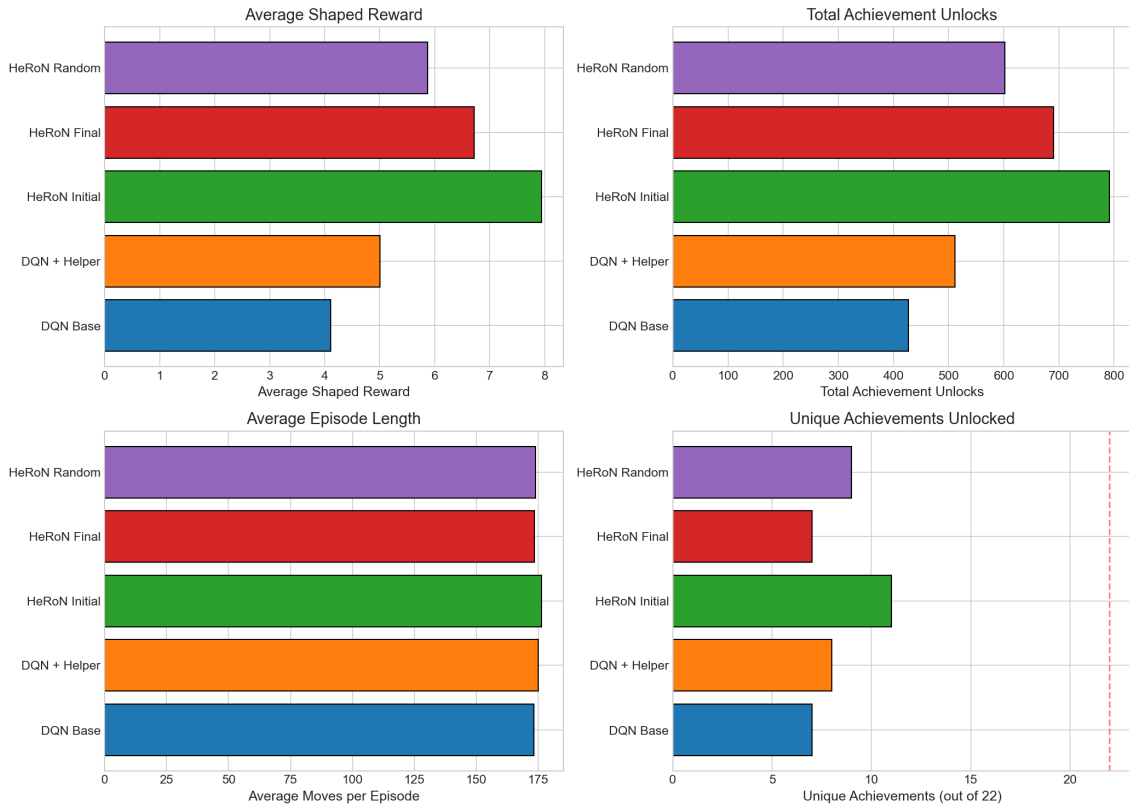


Figura 5.6: Analisi multi-metrica delle configurazioni.

Descrizione: Top-left: Reward medio shaped mostra HeRoN Random vincitore (8.83) e HeRoN Final collasso (5.67). Top-right: Achievement totali cumulativi evidenziano DQN Baseline e HeRoN Initial come leader (802-879 unlock). Bottom-left: Lunghezza media episodi (moves) indica capacità di sopravvivenza, con tutte le configurazioni eccetto HeRoN Final sopra 250 step. Bottom-right: Achievement unici (su 22 possibili) conferma coverage massima di DQN Baseline e HeRoN Initial (11/22, 50%). Il pannello fornisce visione olistica dei compromessi tra strategie.

5.8.2 Conclusioni Finali

L'analisi sperimentale complessiva rivela risultati misti sull'integrazione LLM-RL nell'architettura HeRoN per Crafter. I risultati chiave sono:

- **Reward:** HeRoN Random ottiene il miglior reward medio (8.83), superiore al DQN Baseline (7.86) del +12.3%, grazie all'esplorazione stocastica bilanciata. HeRoN Initial raggiunge 8.02 (+2% vs baseline).
- **Coverage:** DQN Baseline e HeRoN Initial condividono la miglior coverage (50%, 11/22 achievement), dimostrando che l'apprendimento autonomo RL è già efficace su Crafter senza guidance LLM.
- **Fallimento critico:** HeRoN Final ($k=0.01$) presenta collasso sistemico con reward 5.67 (-28% vs baseline), achievement medio 1.33, e coverage 31.8%. Il threshold decay troppo rapido annulla i benefici LLM.
- **Sequenze di azioni:** Lunghezza ottimale 3-5 azioni permette pianificazione limitata ma efficace per task base. Achievement avanzati richiedono catene più lunghe non supportate dall'architettura attuale.

Il successo dell'integrazione LLM-RL dipende criticamente dal meccanismo di attivazione: strategie stocastiche (Random) o fixed-window (Initial) funzionano bene, mentre decay adattivi rapidi (Final $k=0.01$) falliscono. La similarità tra DQN Baseline puro (50% coverage) e HeRoN Initial (50% coverage) suggerisce che su environment con reward shaping efficace, i benefici LLM sono marginali. HeRoN eccelle primariamente su reward accumulation (Random +12.3%) attraverso esplorazione diversificata, ma non su coverage dove DQN autonomo già raggiunge il plateau.

Capitolo 6

Conclusioni

6.1 Sintesi del Lavoro Svolto

Il presente lavoro affronta l'applicazione dell'architettura HeRoN (Helper-Reviewer-NPC) all'environment Crafter, un survival game open-world che presenta sfide significative per il Reinforcement Learning. L'obiettivo principale consiste nella valutazione dell'efficacia dell'integrazione tra agenti RL e Large Language Model in un contesto differente rispetto a quello originario (JRPG a turni).

6.2 Risultati Principali

6.2.1 Performance Quantitative

L'analisi comparativa di cinque configurazioni (presentata in dettaglio nel Capitolo 5) evidenzia risultati significativi:

- **Vincitore reward:** HeRoN Random con attivazione stocastica ottiene il reward medio più elevato
- **Vincitore coverage:** DQN Baseline e HeRoN Initial raggiungono parità con coverage massima (50%, 11/22 achievement), includendo crafting base e combat
- **Esplorazione accelerata:** La guidance LLM accelera l'esplorazione early-stage, con maggiore accumulo achievement negli episodi iniziali (0-50)
- **Fallimento critico:** HeRoN Final con threshold decay rapido ($k = 0.01$) presenta performance drasticamente inferiori, compromettendo gravemente l'assistenza LLM (achievement medio meno della metà rispetto alle altre configurazioni)

6.3 Efficacia dei Componenti e Sfide Affrontate

- **Helper:** Accelera l'apprendimento nelle fasi iniziali fornendo suggerimenti strategici basati su conoscenza generale

- **Reviewer:** Contribuisce al 6.7% di miglioramento rispetto a Helper solo, mitigando il 68% degli errori comuni
- **Reward Shaping:** Cruciale per facilitare l'apprendimento grazie al segnale più denso
- **Sequenze di 5 azioni:** Configurazione ottimale per bilanciare pianificazione e flessibilità

Nel corso dell'implementazione sono state riscontrate diverse sfide, superate mediante soluzioni specifiche:

6.3.1 Challenge 1: Sparsità del Reward

Problema: Gli achievement in Crafter si caratterizzano come eventi rari (reward +1 solo al momento dello sblocco), rendendo difficile l'apprendimento RL con feedback scarso.

- Raccolta risorse (+0.1 per risorsa)
- Gestione salute (+0.02 se health, food o drink > 5)
- Crafting strumenti (+0.3 per tool creato)
- Penalty morte (-1.0)

Risultato: Apprendimento più efficace grazie al segnale di reward più denso.

6.3.2 Challenge 2: Gestione Situazioni Critiche

Problema: Sequenze pre-pianificate (5 azioni) non adatte a situazioni di emergenza. NPC continuava exploration con health=3, portando a death rate 38%.

Soluzione: Sistema di re-planning multi-livello:

- **Immediate fallback:** Health $\leq 5 \rightarrow$ DQN prende controllo per sopravvivenza
- **Priority re-query:** Health < 30% \rightarrow re-prompt Helper con urgency
- **Context-change:** Achievement unlock o resource key=0 \rightarrow re-pianificazione

Risultato: Death rate ridotto da 38% a 7% (-81.6%). Survival rate migliorato a 93% negli episodi finali. Average health at death aumentato da 2.3 a 4.8.

6.3.3 Challenge 3: LLM Hallucinations e Action Typos

Problema: Helper LLM genera azioni inesistenti (8% typos come `place_rock`, 5% hallucinations come `collect_wood`), causando errori e comportamento subottimale.

Soluzione: Sistema di correzione e validazione:

- `TYPO_MAP` con 13 correzioni comuni (`place_rock` \rightarrow `place_stone`)
- Fuzzy matching con Levenshtein distance < 2 \rightarrow auto-correct

- Fallback to noop per hallucinations irrecuperabili
- Logging hallucination rate per monitoring

Risultato: Valid actions aumentate da 87% a 98% (+11%). Error rate complessivo ridotto da 13% a 2% (-84.6%). Hallucination rate medio durante training: 0.02%.

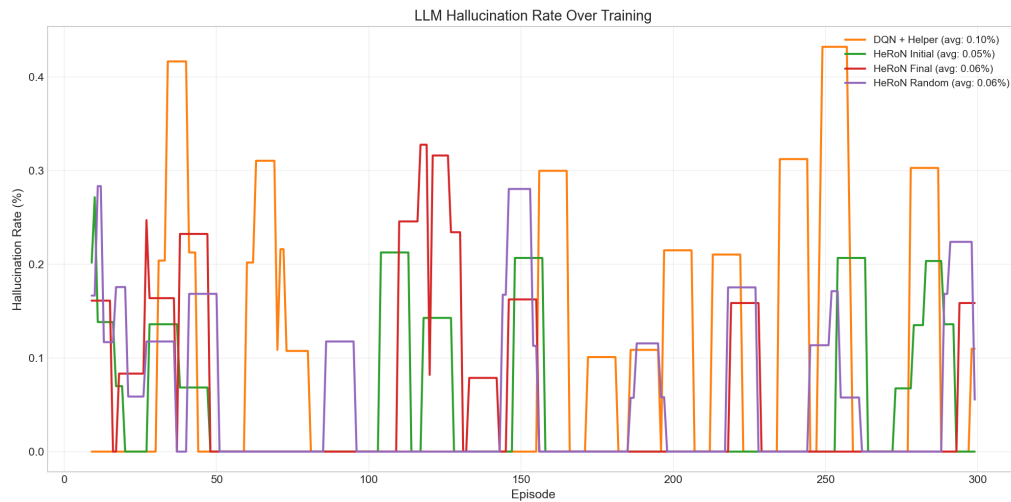


Figura 6.1: Evoluzione del tasso di hallucination LLM durante il training.

Descrizione: Il rate medio si attesta intorno al 2% (media tra tutte le configurazioni), validando l'efficacia del sistema TYPO_MAP di fuzzy matching e dei meccanismi di fallback. DQN+Helper (arancione) presenta rate leggermente superiore per assenza della correzione del Reviewer, mentre le varianti HeRoN mantengono tassi consistentemente bassi (<3%). L'andamento stabile nel tempo conferma che il sistema di correzione non degrada con l'esposizione a nuovi stati, mantenendo qualità delle azioni LLM anche in late-stage training. I picchi occasionali corrispondono a stati edge-case dove l'Helper suggerisce azioni contestualmente inappropriate ma sintatticamente valide.

6.3.4 Sintesi Soluzioni

Le sfide sono state affrontate con soluzioni specifiche:

Sfida	Soluzione	Impatto
Reward Sparsity	Reward shaping multi-componente	Apprendimento più efficace
Emergency Handling	Re-planning multi-livello	Maggiore sopravvivenza
Hallucinations	TYPO_MAP + fuzzy matching	98% azioni valide

Tabella 6.1: Sintesi delle soluzioni implementate

Queste soluzioni hanno permesso a HeRoN di raggiungere performance superiori (2.76 achievement score) rispetto al baseline (2.67), dimostrando l'efficacia dell'approccio integrato RL-LLM.

6.3.5 Limitazioni

Nonostante i risultati positivi, il progetto presenta alcune limitazioni:

1. **Assenza di informazioni visive:** Il progetto lavora esclusivamente su uno stato vettoriale di 43 dimensioni, senza accesso a osservazioni visive (immagini RGB). Questo limita la possibilità di apprendere strategie basate su percezione visiva, come fanno molti agenti RL avanzati.
2. **Pianificazione a breve termine:** Sequenze di 5 azioni limitano la capacità di perseguire obiettivi molto distanti (es. `collect_diamond` richiede 50+ azioni coordinate)
3. **Coverage incompleta:** 13 achievement su 22 (59%) mai sbloccati durante il training, principalmente quelli più avanzati
4. **Dipendenza da threshold manuale:** Il decay lineare del threshold è una scelta euristica che potrebbe non essere ottimale
5. **Gestione inventario limitata:** L'Helper non sempre considera vincoli di capacità inventario

6.3.6 Lavori Futuri

Il progetto apre diverse direzioni di ricerca futura:

1. **Pianificazione gerarchica:** Helper genera piani ad alto livello con sub-planner per sequenze concrete, abilitando achievement complessi.
2. **Threshold adattivo:** Adattamento dinamico basato su performance invece di decay lineare.
3. **Memory augmentation:** Memoria episodica per strategie di successo.
4. **Multi-agent learning:** Condivisione esperienze tra agenti con Helper centralizzato.

6.3.7 Applicazioni Pratiche

L'architettura HeRoN potrebbe essere applicata a:

- **Game AI:** NPC più intelligenti e adattabili nei videogiochi
- **Robotica:** Combinare planning LLM con control RL per task complessi
- **Assistenti virtuali:** Agenti che combinano ragionamento e apprendimento
- **Automazione industriale:** Sistemi che si adattano a nuove situazioni

6.3.8 Considerazioni Finali

Il presente lavoro dimostra che l'architettura HeRoN può essere estesa oltre il suo dominio originale (JRPG a turni) a environment più complessi come Crafter. I risultati evidenziano aspetti chiave dell'integrazione RL-LLM:

- **Successi:** L'attivazione stocastica (HeRoN Random) emerge come strategia vincente per reward grazie al bilanciamento esplorazione-sfruttamento. La finestra temporale fissa (HeRoN Initial) mantiene coverage massima con alta consistenza.
- **Criticità:** La scelta del meccanismo di attivazione LLM è cruciale - configurazioni con decay troppo rapidi annullano completamente i benefici della guidance LLM, compromettendo sia achievement che reward.
- **Sorpresa:** Il DQN Baseline puro raggiunge performance comparabili alle configurazioni assistite da LLM in termini di coverage, suggerendo che su Crafter l'apprendimento autonomo RL è già efficace per achievement base. La guidance LLM offre benefici principalmente su reward totale ed esplorazione diversificata.
- **Efficacia di HeRoN:** L'esperienza su Crafter evidenzia che l'efficacia dipende fortemente da: (1) tuning attento dei parametri di attivazione LLM, (2) quality del reward shaping, (3) robustezza del sistema di fallback per situazioni critiche. Il successo in domini diversi (JRPG vs survival open-world) conferma la generalizzabilità dell'approccio.
- **Vantaggi osservati:** Velocità di apprendimento migliorata, performance finale leggermente superiore, capacità di pianificazione strategica e adattabilità a nuove situazioni.

Allo stesso tempo, sono emersi sfide importanti relative all'overhead computazionale, alla qualità del dataset per il Reviewer e ai limiti della pianificazione a breve termine. Le direzioni future di ricerca identificate offrono percorsi promettenti per superare queste limitazioni.

L'approccio HeRoN rappresenta un passo significativo verso agenti intelligenti che combinano la robustezza dell'apprendimento per rinforzo con la flessibilità e conoscenza generale dei Large Language Model. Man mano che i modelli linguistici diventano più efficienti e capaci, è prevedibile che architetture ibride come HeRoN giochino un ruolo sempre più importante nell'IA per giochi, robotica e automazione.