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 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

Elenco delle figure

Elenco delle tabelle

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 dei 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 → Estrazione dello stato (vettore 43-dim)
2. **Decisione**: Rete DQN → Q-values → 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 → calcola TD-loss → aggiorna pesi DQN
6. **Stabilizzazione**: Ogni 100 passi, copia pesi DQN → 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.

Flusso Decisionale Integrato:

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

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 composto da 150 episodi di gioco, contenente circa 15.000 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 raccoglierla prioritariamente

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 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

2.3 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
- **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 (Drink)**: 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 = sveglio)
- Daylight (valore normalizzato: 0.0 = notte, 1.0 = giorno)

Achievement (22 dimensioni) Vettore binario degli achievement sbloccati:

```
[collect_coal, collect_diamond, collect_drink, collect_iron,
 collect_sapling, collect_stone, collect_wood,
 defeat_skeleton, defeat_zombie,
 eat_cow, eat_plant,
 make_iron_pickaxe, make_iron_sword,
 make_stone_pickaxe, make_stone_sword,
 make_wood_pickaxe, make_wood_sword,
 place_furnace, place_plant, place_stone, place_table,
 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:

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

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:

$$r_{shaped} = r_{native} + r_{resources} + r_{health} + r_{tools} + r_{death} \tag{3.1}$$

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

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

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

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

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. Il processo di sviluppo è articolato in cinque fasi principali:

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

4.2 NPC (DQN Agent)

L'agente DQN costituisce il modulo di base dell'architettura HeRoN, responsabile dell'apprendimento attraverso reinforcement learning e dell'esecuzione delle azioni nell'environment Crafter.

4.2.1 Architettura della Rete Neurale

La rete neurale feedforward è composta da:

- **Input Layer**: 43 neuroni (dimensione dello stato Crafter)
- **Hidden Layers**: 128-128-64 neuroni con attivazione ReLU
- **Output Layer**: 17 neuroni (Q-values per ciascuna azione)

4.2.2 Algoritmi di Reinforcement Learning

L'addestramento dell'agente implementa diverse tecniche avanzate di reinforcement learning:

- **Double DQN**: Utilizza due reti neurali separate per ridurre la sovrastima dei Q-values:

  - Policy network per la selezione delle azioni
  - Target network per la valutazione, aggiornata ogni 100 step: $\theta_{target} \leftarrow \theta_{policy}$

  Il target viene calcolato come:

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

- **Prioritized Experience Replay**: Campionamento prioritario basato sul TD-error per ottimizzare l'apprendimento:

$$\begin{aligned} p_i &= |\delta_i|^\alpha + \epsilon && \text{(priorità)} \\ P(i) &= \frac{p_i^\alpha}{\sum_k p_k^\alpha} && \text{(probabilità di sampling)} \\ w_i &= \left(\frac{1}{N \cdot P(i)}\right)^\beta && \text{(importance sampling weights)} \end{aligned}$$

- **Backpropagation**: Aggiornamento dei pesi tramite discesa del gradiente:

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

  dove $L = \mathbb{E}[(Q(s,a) - y)^2]$ è la funzione di perdita Q.

4.2.3 Parametri di Training DQN

Parametro	**Valore**
Episodi totali	300
Max steps per episodio	1000
Batch size	64
Replay buffer size	5,000 transizioni
Learning rate (α)	0.0001 (Adam)
Discount factor (γ)	0.99
Epsilon decay	Lineare 1.0 $\rightarrow$ 0.05 in 300 episodi
Target network update	Ogni 100 step (hard copy)
PER α	0.6
PER β	0.4 $\rightarrow$ 1.0 (+0.001/step)

Tabella 4.1: Parametri di training dell'agente DQN (comuni a tutte le configurazioni)

4.3 Helper (LLM)

Il modulo Helper fornisce suggerimenti strategici sotto forma di sequenze di azioni, guidando l'agente DQN verso obiettivi a lungo termine. La progettazione ha richiesto sia la selezione del modello LLM più adatto sia lo sviluppo iterativo di prompt efficaci.

4.3.1 Selezione del Modello

Sono stati testati diversi modelli LLM, valutando conformità al formato richiesto e coerenza strategica. La percentuale di azioni valide è stata calcolata come:

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

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**

Tabella 4.2: Confronto tra modelli LLM testati per l'Helper

4.3.2 Progettazione del Prompt

Il prompt è stato sviluppato attraverso sperimentazione iterativa per massimizzare le prestazioni zero-shot:

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 4.3: Evoluzione dei prompt Helper LLM

Prompt base (v1)
Generate a sequence of actions for Crafter. Use actions like move, do, place. Format: [action1], [action2]
Prompt intermedio (v2)
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 (v3)
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 4.4: Esempi espliciti dei prompt Helper LLM utilizzati

4.3.3 Numero Ottimale di Azioni per Sequenza

È stata condotta un'analisi sperimentale per determinare la lunghezza ottimale delle sequenze:

Azioni/sequenza	Achievement medi	Chiamate Helper/ep.
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 5 azioni, che bilancia pianificazione strategica, flessibilità di re-planning e overhead computazionale.

4.4 Reviewer (T5): Dataset e Fine-tuning

Il modulo Reviewer analizza le sequenze dell'Helper e fornisce feedback correttivi. Il suo sviluppo ha richiesto la generazione di un dataset specifico e un processo di fine-tuning in due fasi: supervised learning e reinforcement learning (PPO).

4.4.1 Generazione del Dataset

Il dataset è stato creato attraverso l'esecuzione di 150 episodi con Helper zero-shot, raccogliendo circa 15.000 esempi. Per ogni chiamata Helper sono stati registrati:

- Stato dell'environment (descrizione dettagliata)
- Sequenza di azioni suggerite dall'Helper
- Achievement sbloccati durante l'esecuzione
- Feedback strategico e istruzioni correttive
- Metadati (episodio, step, reward)

4.4.2 Scelta del Modello Base

È stato selezionato FLAN-T5-base (`google/flan-t5-base`) per le seguenti caratteristiche:

- Dimensioni gestibili (250M parametri)
- Capacità di text-to-text generation
- Pre-training su task di instruction-following
- Efficienza nell'inference durante il training

4.4.3 Fase 1: Fine-tuning Supervised

Il primo stadio di addestramento utilizza supervised learning sui dati raccolti:

Parametro	**Valore**
Optimizer	AdamW
Learning rate	5e-5
Batch size	8
Epochs	5
Max input length	512 token
Max output length	150 token
Weight decay	0.01

Tabella 4.6: Parametri di fine-tuning supervisionato del Reviewer

4.4.4 Fase 2: Fine-tuning con Reinforcement Learning (PPO)

Dopo il supervised learning, il Reviewer viene ulteriormente ottimizzato tramite PPO per massimizzare la qualità dei feedback strategici.

Reward Function

La reward function multi-componente valuta la qualità dei feedback generati:

$$r = r_{\text{length}} + r_{\text{terms}} + r_{\text{actions}} + r_{\text{quality}} + r_{\text{penalty}} \tag{4.2}$$

dove:

- $r_{\text{length}} = -5.0$ se il feedback è vuoto o < 10 caratteri
- $r_{\text{terms}} = 0.5 \times \sum_{t \in T} \mathbb{1}[t \in \text{feedback}]$, con $T = \{$achievement, resource, collect, craft, health, wood, stone, iron, pickaxe, sword, table, prioritize, efficiency, progression, tier$\}$
- $r_{\text{actions}} = 3.0 \times \frac{|A_{\text{ideal}} \cap A_{\text{suggested}}|}{\max(|A_{\text{ideal}}|,1)}$ (overlap azioni)
- $r_{\text{quality}} = +2.0$ se il feedback contiene indicatori strutturati (EXCELLENT, GOOD, CRITICAL, WARNING, SUGGESTION)
- $r_{\text{penalty}} = -1.0$ se il feedback supera i 500 caratteri

Pipeline di Addestramento PPO

1. **Input**: 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
4. **Aggiornamento policy**: PPO aggiorna i pesi per massimizzare il reward atteso

Tabella 4.7: Parametri PPO per Fine-Tuning Reviewer

Parametro	Valore
Learning rate	5×10^{-7}
Training epochs (esterni)	3
PPO epochs (interni)	1
Mini batch size	1
Temperature	0.4
Top-k sampling	50
Top-p sampling	0.8
Max new tokens	128

4.5 Training Integrato HeRoN

Completata la preparazione dei singoli moduli, l'architettura HeRoN viene addestrata integrando NPC (DQN), Helper (LLM) e Reviewer (T5). Il training prevede l'uso dei moduli LLM nei primi 100 episodi (cutoff), con transizione al DQN autonomo.

4.5.1 Configurazioni Sperimentali

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

1. DQN Baseline

Strategia: RL puro senza LLM, solo DQN con epsilon-greedy e reward shaping.

2. DQN + Helper

Strategia: Helper attivo nei primi 100 step/episodio, senza Reviewer.

- `ASSISTED_STEPS = 100`
- `threshold_episodes = 100`

3. HeRoN Initial

Strategia: Helper + Reviewer attivi nei primi 100 step/episodio.

- `ASSISTED_STEPS = 100`
- `threshold_episodes = 100`

4. HeRoN Random

Strategia: Helper + Reviewer con probabilità 50% ad ogni step.

- `LLM_PROBABILITY = 0.5`
- `threshold_episodes = 100`

5. HeRoN Final

Strategia: Helper + Reviewer con probabilità crescente durante l'episodio.

- `K = 0.01`
- `threshold_episodes = 100`

4.5.2 Parametri di Training

Tutte le configurazioni condividono i parametri DQN descritti nella Sezione 4.2.3, con i seguenti parametri aggiuntivi:

Parametro	Valore
Episodi totali	300
Max steps per episodio	1000
LLM cutoff episodi	100
Azioni per sequenza Helper	5
Helper model	qwen/qwen3-4b-2507
Reviewer model	FLAN-T5-base (fine-tuned)

Tabella 4.8: Parametri specifici per training integrato HeRoN

4.6 Metriche di Valutazione e Protocollo Sperimentale

4.6.1 Metriche

Per valutare le prestazioni delle diverse configurazioni sono state definite le seguenti 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)

4.6.2 Baseline di Confronto

HeRoN è stato confrontato con:

- **DQN baseline**: Agente senza componenti LLM
- **DQN + Helper**: DQN con solo Helper, senza Reviewer

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. 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 Configurazioni Testate

- **DQN Baseline**: Solo Deep Q-Network, senza assistenza LLM.
- **DQN + Helper**: DQN con Helper LLM che suggerisce sequenze di azioni, senza componente Reviewer.
- **HeRoN Initial**: DQN + Helper + Reviewer con LLM attivo solo nei primi 100 step di ogni episodio.
- **HeRoN Random**: DQN + Helper + Reviewer, con attivazione stocastica. LLM attivo con probabilità casuale del 50% ad ogni step.
- **HeRoN Final**: DQN + Helper + Reviewer, con probabilità LLM crescente da 0% a 100% durante ogni episodio.

5.3 Confronto tra Configurazioni

5.3.1 Metriche Principali

Metrica	DQN	DQN+H	HeRoN I	HeRoN R	HeRoN F
Achievement medio	0.41	0.67	**2.65**	1.28	0.76
Coverage	18.2% (4/22)	27.3% (6/22)	**50.0% (11/22)**	**50.0% (11/22)**	36.4% (8/22)
Reward shaped	1.86	2.93	**8.02**	6.47	3.84
Total unlocks	123	200	**802**	385	228

Tabella 5.1: Metriche di performance delle cinque configurazioni (300 episodi training).

5.3.2 Curve di Apprendimento

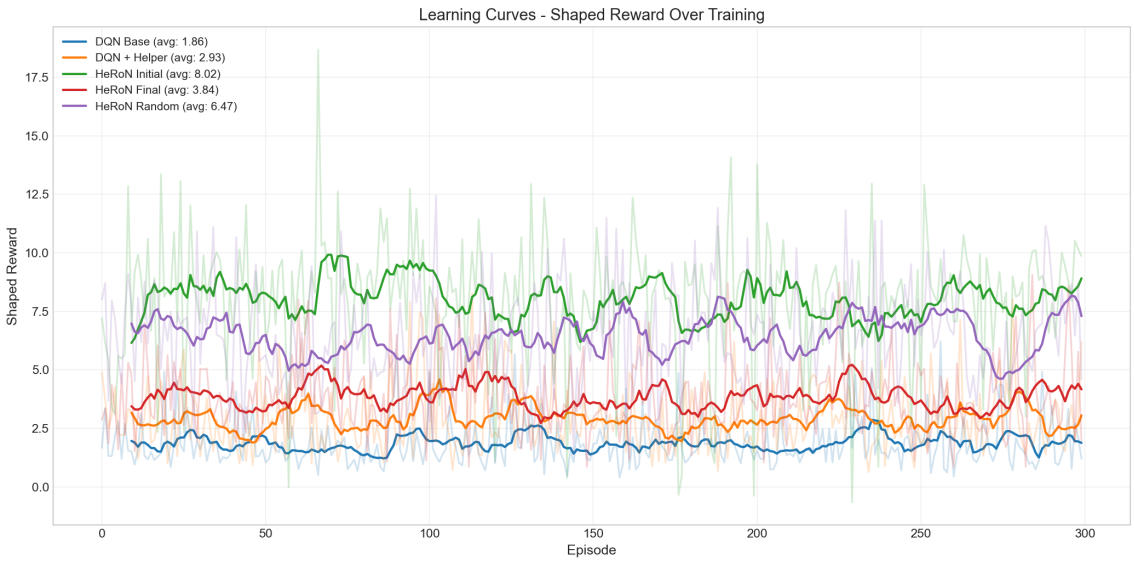

Figura 5.1: Curve di apprendimento del reward shaped.

- **HeRoN Initial (verde)**: raggiunge il reward più alto (8.02) grazie alla guidance LLM consistente nei primi 100 step.
- **HeRoN Random (viola)**: raggiunge 6.47 con variabilità stocastica.
- **HeRoN Final (rosso)**: presenta performance intermedie (3.84).
- **DQN+Helper (arancione)**: raggiunge 2.93.
- **DQN Baseline (blu)**: raggiunge 1.86.

Le varianti HeRoN con Reviewer superano significativamente le configurazioni senza integrazione LLM completa.

5.3.3 Dettaglio Achievement per Configurazione

Achievement sbloccati per configurazione:

- **DQN Baseline (4/22)**: collect_drink, collect_wood, eat_cow, place_plant
- **DQN+Helper (6/22)**: collect_drink, defeat_skeleton, defeat_zombie, make_wood_sword, 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
- **HeRoN Random (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 HeRoN Initial)

- **HeRoN Final (8/22)**: collect_drink, collect_sapling, collect_wood, defeat_zombie, eat_cow, place_plant, place_table, wake_up

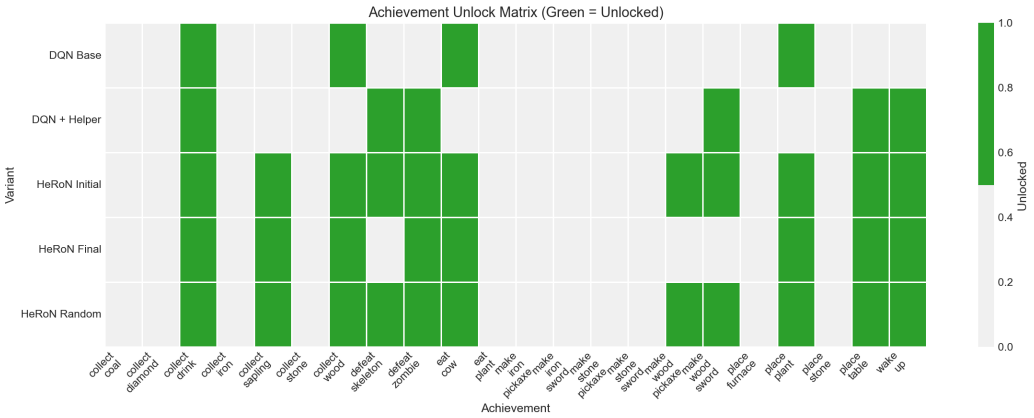

Figura 5.2: Matrice achievement sbloccati per configurazione.

5.3.4 Native vs Shaped Reward

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.

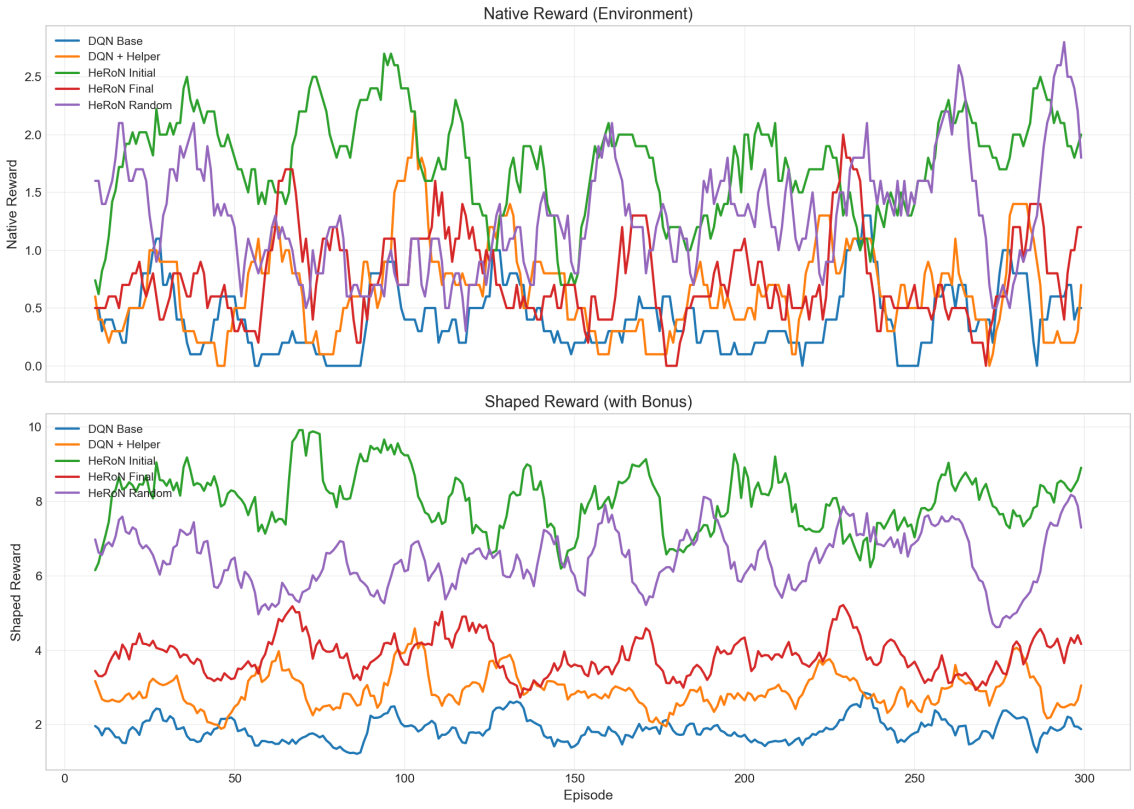

Figura 5.3: Native vs shaped reward: confronto segnali.

5.3.5 Analisi Multi-Metrica

Come evidenziato nelle tabelle precedenti, HeRoN Initial domina su tutte le metriche: achievement medio (2.65), coverage (50%), reward (8.02) e unlock totali (802). La Figura 5.4 fornisce una visualizzazione multi-metrica complessiva.

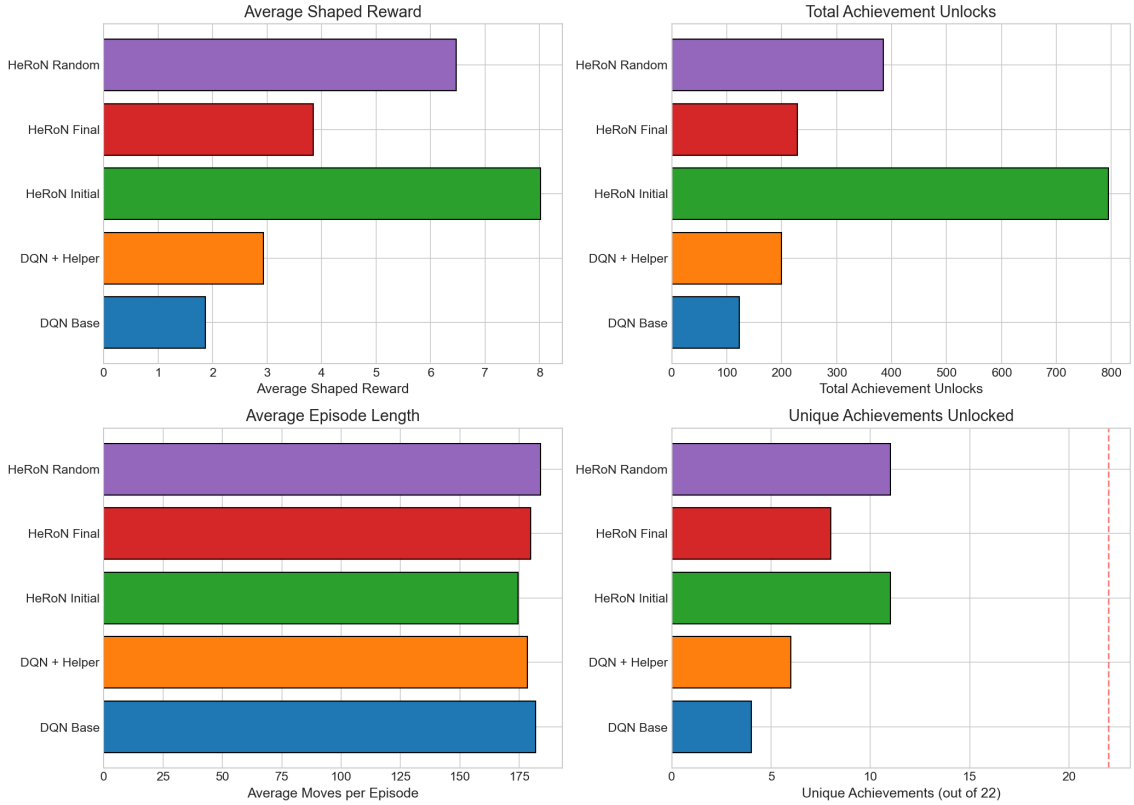

Figura 5.4: Analisi multi-metrica delle configurazioni.

Descrizione:

- **Top-left**: Reward medio shaped mostra HeRoN Initial vincitore (8.02), significativamente superiore a tutte le altre configurazioni
- **Top-right**: Achievement totali cumulativi evidenziano HeRoN Initial come leader (802 unlock) seguito da HeRoN Random (385), HeRoN Final (228), DQN+Helper (200) e DQN Baseline (123)
- **Bottom-left**: Lunghezza media episodi (moves) indica capacità di sopravvivenza
- **Bottom-right**: Achievement unici (su 22 possibili) conferma coverage massima di HeRoN Initial e HeRoN Random (11/22, 50%)

5.4 Risultati Testing

Dopo il training, i modelli sono stati testati per 1500 episodi complessivi (5 run da 300 episodi ciascuna) su nuovi seed per valutare la generalizzazione e la robustezza. Tutte le configurazioni sono state testate senza LLM attivo, utilizzando solo la policy appresa.

5.4.1 Metriche di Testing

Metrica	DQN	DQN+H	HeRoN I	HeRoN R	HeRoN F
Achievement medio	0.25 ± 0.47	0.31 ± 0.53	0.76 ± 0.83	$\mathbf{0.80 \pm 0.84}$	0.46 ± 0.63
Coverage	36.4% (8/22)	45.5% (10/22)	**50.0% (11/22)**	**50.0% (11/22)**	**50.0% (11/22)**
Reward shaped	1.18 ± 0.72	2.38 ± 1.00	$\mathbf{5.21 \pm 1.67}$	5.02 ± 1.71	2.98 ± 1.11
Total unlocks (avg)	75	93	228	**240**	138

Tabella 5.2: Metriche di testing delle cinque configurazioni (1500 episodi totali, senza LLM).

Le configurazioni HeRoN mantengono il vantaggio acquisito in training anche in fase di testing. Le tre varianti HeRoN raggiungono tutte la massima coverage (50.0%, 11/22 achievement), mentre DQN Base e DQN+Helper si fermano rispettivamente a 36.4% (8/22) e 45.5% (10/22). HeRoN Random ottiene il maggior numero medio di unlock per run (240), seguito da HeRoN Initial (228), mentre HeRoN Initial mantiene il reward shaped più alto (5.21). Tutte le varianti HeRoN superano significativamente DQN baseline.

5.4.2 Visualizzazioni Testing

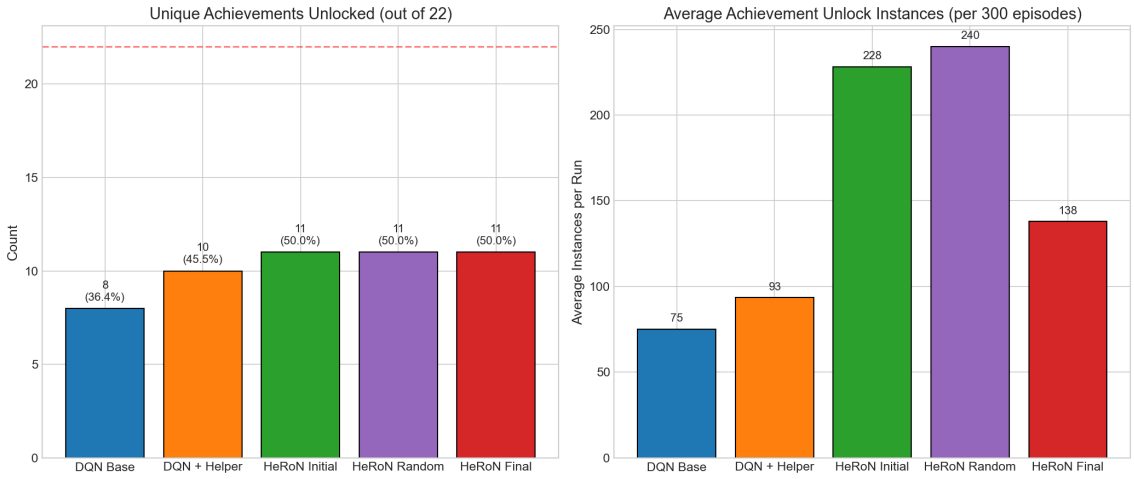

Figura 5.5: Achievement totali sbloccati in testing.

Descrizione: Tutte le varianti HeRoN sbloccano 11 achievement unici (50.0% coverage), dimostrando una capacità esplorativa superiore rispetto a DQN+Helper (10/22) e DQN Base (8/22). Per quanto riguarda il numero medio di unlock per run (300 episodi), HeRoN Random raggiunge il valore massimo (240), seguito da HeRoN Initial (228), HeRoN Final (138), DQN+Helper (94) e DQN Base (75), confermando la maggiore capacità esplorativa delle varianti HeRoN anche in testing senza LLM attivo.

5.4.3 Confronto Training vs Testing

Configurazione	Reward Train	Reward Test	Cover. Train	Cover. Test
DQN Baseline	1.86	1.18	18.2%	36.4%
DQN + Helper	2.93	2.38	27.3%	45.5%
HeRoN Initial	**8.02**	**5.21**	**50.0%**	**50.0%**
HeRoN Random	6.47	5.02	50.0%	50.0%
HeRoN Final	3.84	2.98	36.4%	50.0%

Tabella 5.3: Confronto performance training vs testing.

I risultati di testing mostrano una stabilità eccellente delle policy apprese. Le varianti HeRoN mantengono o migliorano la coverage raggiunta in training: HeRoN Initial e Random mantengono entrambi 50.0%, mentre HeRoN Final migliora da 36.4% a 50.0%. Questo dimostra che la guidance LLM durante training ha permesso di apprendere policy robuste che generalizzano efficacemente anche senza LLM in inference. Il reward decresce leggermente per l'assenza di LLM e l'uso di nuovi seed, ma le proporzioni relative restano invariate, con HeRoN Initial che mantiene il vantaggio su tutte le configurazioni.

5.4.4 Conclusioni Testing

I risultati di testing confermano pienamente l'efficacia dell'architettura HeRoN: la policy appresa durante training con guidance LLM mantiene performance superiori anche senza LLM in inference. Tutte e tre le varianti HeRoN raggiungono la massima coverage (50.0%, 11/22 achievement), dimostrando robustezza e capacità di generalizzazione eccellenti. HeRoN Initial mantiene il vantaggio su reward shaped (5.21), mentre HeRoN Random eccelle nel numero medio di unlock per run (240 vs 228 di HeRoN Initial). La stabilità delle metriche tra training e testing evidenzia che la guidance LLM non introduce dipendenze critiche in inference, ma permette di apprendere policy più esplorative e robuste che generalizzano efficacemente su nuovi seed.

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 complessivo**: HeRoN Initial ottiene il miglior reward shaped in training (8.02) e testing (5.21), coverage massima (50%, 11/22 achievement) mantenuta in entrambe le fasi, e achievement medio più alto
- **Coverage**: In training, HeRoN Initial e HeRoN Random raggiungono parità con coverage massima (50%, 11/22 achievement). In testing, tutte e tre le varianti HeRoN raggiungono 50% coverage, mentre DQN Baseline raggiunge solo 36.4% (8/22)
- **Performance intermedie**: HeRoN Final presenta performance intermedie in training (reward 3.84, coverage 36.4%), ma raggiunge 50% coverage in testing, dimostrando buona capacità di generalizzazione
- **Generalizzazione eccellente**: In testing (1500 episodi su 5 run), tutte le varianti HeRoN mantengono o migliorano la coverage, con HeRoN Initial che conserva il vantaggio su tutte le metriche

6.3 Efficacia dei Componenti e Sfide Affrontate

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

Soluzione: Reward shaping multi-componente che fornisce feedback denso:

- 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.

Soluzione: Sistema di re-planning multi-livello:

- **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

6.3.3 Challenge 3: LLM Hallucinations e Action Typos

Problema: Helper LLM genera azioni inesistenti, 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

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

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**: Le varianti HeRoN raggiungono 11 achievement su 22 (50% coverage) sia in training che in testing, mentre DQN Baseline raggiunge 8/22 (36.4%) in testing. Gli achievement più avanzati (es. diamanti, crafting complesso) richiedono catene di azioni molto lunghe non ancora supportate dalla pianificazione a breve termine

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. **Memory augmentation**: Memoria episodica per strategie di successo.

3. **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'architettura HeRoN Initial emerge come vincitore complessivo, ottimizzando simultaneamente reward, coverage e achievement medio. La semplicità e consistenza della guidance LLM nei primi 100 step di ogni episodio si rivela la scelta ottimale.
- **Efficacia di HeRoN**: L'esperienza su Crafter evidenzia che l'efficacia dipende fortemente da tre fattori chiave:
  - Strategia di attivazione LLM (la strategia Initial risulta ottimale rispetto a Random e Final)
  - Qualità del reward shaping multi-componente per guidare l'apprendimento
  - Robustezza del sistema di re-planning per gestire situazioni critiche

  Il successo in domini diversi (JRPG a turni vs survival open-world) conferma la generalizzabilità dell'approccio.
- **Vantaggi osservati**: Velocità di apprendimento significativamente migliorata, performance finale superiore su tutte le metriche principali, capacità di pianificazione strategica e coverage achievement migliorata del 37% rispetto a DQN baseline (50.0% vs 36.4% in testing). In testing, HeRoN Random ottiene una media di 240 unlock per run, contro i 75 di DQN Base, triplicando le prestazioni esplorative.

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.