Esplorazione di Ambienti Complessi tramite Approcci basati sul Guadagno Informativo

Sistemi Complessi: Modelli e Simulazione

Emanuele Masiero(@Emamasi-uni)

Settembre 2024

Indice

Capitolo 1

Introduzione

Nell'agricoltura di precisione, è spesso fondamentale ispezionare da vicino le piante cambiando il punto di vista per rilevare caratteristiche nascoste, come la presenza di parassiti o danni ai frutti. La visione attiva [1] diventa quindi un elemento chiave per i sistemi robotici che operano in ambienti complessi e non strutturati, come i campi agricoli. In tali contesti, la presenza di ostacoli, le variazioni di scala e la distribuzione imprevedibile degli elementi possono rappresentare sfide significative per l'acquisizione di informazioni cruciali.
Uno dei problemi principali riguarda il monitoraggio e la mappatura delle infestanti nei campi coltivati. Tra le soluzioni proposte, l'utilizzo di sciami di veicoli aerei senza pilota (UAV) si è dimostrato promettente, con gli UAV che raccolgono dati prevalentemente sotto forma di immagini. La classificazione delle caratteristiche rilevate viene eseguita tramite tecniche di deep learning, in particolare usando reti neurali convoluzionali (CNN) applicate a immagini RGB [2].
Tuttavia, il tempo di volo limitato di questi dispositivi riduce la risoluzione delle immagini acquisite, necessaria per una classificazione accurata. Di conseguenza, sono necessari numerosi voli per coprire uniformemente un campo esteso. Questo approccio può risultare inefficiente, poiché non tutte le aree del campo contengono caratteristiche rilevanti e alcune zone richiedono meno attenzione rispetto ad altre.

La strategia proposta in questa trattazione si basa su un approccio adattivo che focalizza l'attenzione solo sulle aree effettivamente rilevanti, utilizzando il guadagno informativo atteso e sfruttando tecnologie di deep learning e di l'apprendimento per rinforzo.
L'obiettivo è sviluppare un modello in grado di controllare un gruppo di agenti che esplorano un ambiente in modo efficiente, verificando lo stato delle diverse celle.

Il task non consiste nel raggiungere una posizione specifica, ma nell'ottenere una rappresentazione accurata dello stato dell'ambiente.

1.1 Definizione del problema

Consideriamo una griglia quadrata di dimensione $N \times N$ dove ogni cella contiene scatole di cartone con un bersaglio (marker) su un lato.
Le scatole sono posizionate nelle celle in modo da rendere il bersaglio visibile da certi punti di vista ma non da altri.
I bersagli sono posizionati solo su un lato per garantire che l'agente osservi la cella da punti di vista specifici (POV) per un rilevamento accurato.
Ogni cella può contenere al massimo 9 scatole ognuna delle quali può contenere un solo bersaglio. Perciò ogni cella può essere osservata da 9 punti di vista diversi.
L'agente (UAV) può vedere più celle contemporaneamente e si muove nell'ambiente cercando di massimizzare il guadagno informativo.

$$IG(W) = H(W) - H(W|O)$$

In questo modo, quando un UAV osserva il campo, deve scegliere il POV migliore per ottenere una visione chiara dei bersagli e contarli.
L'obiettivo è campionare l'ambiente in modo economico, questo significa che dopo aver osservato un insieme di punti di vista, il modello dovrà scegliere un punto di vista non ancora osservato tale da offrire il massimo aumento di accuratezza della stima risultante.
Ciò si traduce alla riduzione dell'entropia di Shannon definita come:

$$H(X) = -\sum_i P(x_i) log_2 P(x_i)$$

Questo ragionamento può essere esteso a qualsiasi misura di incertezza e accuratezza diversa dall'entropia di Shannon.

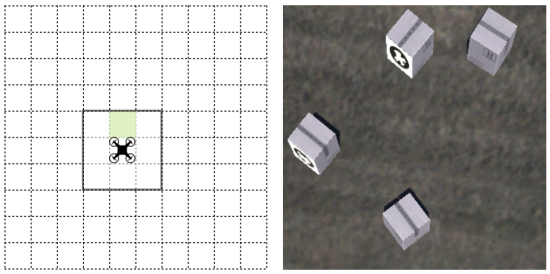


Figura 1.1: Cella osservata dal POV 2

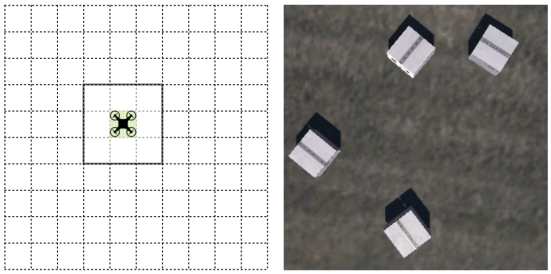


Figura 1.2: Cella osservata dal POV 5

1.2 Struttura dei dati

Le immagini catturate da ciascun punto di vista (POV) vengono pre-elaborate tramite una rete neurale convoluzionale (CNN), che genera una distribuzione di probabilità relativa al numero di bersagli presenti.
I risultati prodotti dalla CNN sono stati raggruppati in un dataset, utilizzato sia per la costruzione dell'ambiente che come input per l'addestramento dei modelli.
Il dataset contiene un totale di 49.500 osservazioni, corrispondenti a 5.500 celle.

IMAGE˙ID	POV˙ID	BOX˙COUNT	MARKER˙COUNT	P0	P1	...	P6	P7
601	7	0	0	9.53E-01	3.81E-02	...	6.79E-06	3.67E-07
601	1	0	0	9.52E-01	3.90E-02	...	6.48E-06	3.60E-07
601	4	0	0	9.53E-01	3.77E-02	...	7.35E-06	4.26E-07
601	9	0	0	9.55E-01	3.65E-02	...	7.35E-06	4.09E-07
601	8	0	0	9.54E-01	3.76E-02	...	6.32E-06	3.57E-07
601	5	0	0	9.54E-01	3.77E-02	...	6.78E-06	3.81E-07
601	6	0	0	9.53E-01	3.78E-02	...	6.68E-06	3.79E-07
601	2	0	0	9.52E-01	3.88E-02	...	6.47E-06	3.64E-07
601	3	0	0	9.52E-01	3.87E-02	...	6.47E-06	3.55E-07
...	...	...	...	...	...	...	...	...

Tabella 1.1: Struttura del dataset

1.3 Significato dei dati

- $IMAGE\_ID$: indica l'ID dell'immagine che è stata esaminata per creare la sequenza.
- POV_ID: indica il punto di vista della cella
- BOX_COUNT: indica il numero effettivo di scatole nell'immagine
- $MARKER\_COUNT$: indica il numero effettivo di bersagli nell'immagine
- $P_0, ..., P_7$: distribuzione di probabilità dove ogni valore P_i indica la probabilità che ci siano i marker nella cella. P_7 indica la probabilità che siano presenti un numero di marker ≥ 7

IMAGE_ID, BOX_COUNT, MARKER_COUNT identificano in maniera univoca una cella.

Capitolo 2

Descrizione del modello

Per affrontare il problema, introduciamo due modelli: il primo, chiamato modello base, ha il compito di stimare lo stato del mondo, mentre il secondo, denominato IG model (dove IG sta per "information gain", ossia guadagno informativo), stima l'incertezza relativa a dove effettuare il campionamento successivo. Entrambi i modelli ricevono in input la stessa sequenza variabile di osservazioni di celle.
Questi modelli sono utilizzati per definire una policy che guida l'esplorazione dell'agente nell'ambiente. Tuttavia, il modello base può soffrire di rumore nelle previsioni, e poiché l'output dell'IG model dipende dal modello base, esso eredita questo rumore. Di conseguenza, IG model può ottenere buone prestazioni solo se anche il modello base funziona in modo efficace.

2.1 Modelli neurali predittivi

Per l'implemetazione di entrambi i modelli sono state utilizzate reti neurali ricorrendi (RNN), in particolare un singolo livello LSTM (Long short-term memory) seguito da livelli lineari completamente connessi.
Il livello ricorrente ha lo scopo di gestire input di lunghezza variabili che possono essere ordinate arbitrariamente. Ogni modello è stato addestrato su 10 fold per rendere i risultati più stabili.
L'input è lo stesso per entrambi i modelli ed è composto dal punto di vista della cella concatenato con la distribuzione di probabilità pre-elaborata dalla CNN, cioè quella presente nel set di dati.

2.1.1 Modello base

Il modello base prende come input una sequenza variabile di POV osservati per una cella e produce una distribuzione di probabilità sullo stato reale del conteggio dei bersagli.
Il modello è stato addestrato con cross-entropy loss (CE) per calcolare l'errore residuo.

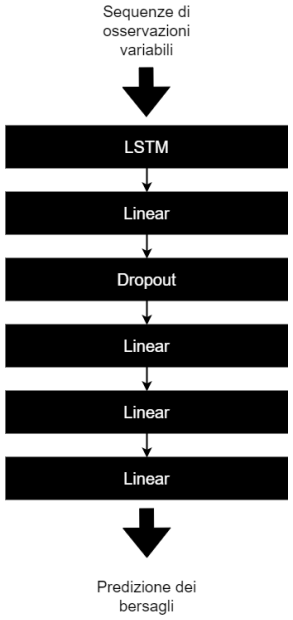


Figura 2.1: Architettura del modello base

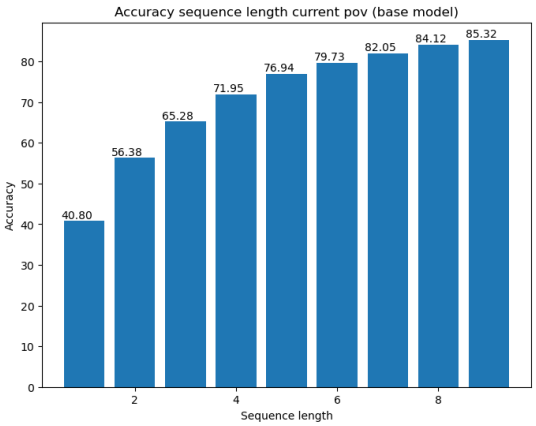


Figura 2.2: Accuratezza dei bersagli divisa per lunghezze di sequenza

Come mostrato in figura 2.2 l'accuratezza dei bersagli aumenta più osservazioni vengono effettuate.

2.1.2 IG model

IG model stima la riduzione dell'incertezza da osservazioni aggiuntive.
Consideriamo due diverse misure di incertezza: l'entropia di Shannon e la perdita (loss) del modello base. In particolare viene stimata:

- l'entropia di Shannon per ogni punto di vista
- la perdita (loss) attesa per ogni punto di vista del modello base
- stima dei bersagli come per base model

Il modello è stato addestrato sia con cross-entropy-loss (CE) che con lo scarto quadratico medio (MSE). Nel caso di CE, il modello cerca di prevedere il miglior punto di vista non ancora osservato. Al contrario, con la MSE, il modello stima i valori di incertezza per tutti i punti di vista, inclusi quelli già osservati. I due obiettivi sono diversi, uno di classificazione e l'altro di regressione.
IG model prevede l'entropia e la perdita per ogni POV, quindi è possibile scegliere quello che fornisce la maggiore riduzione dell'incertezza.
Inoltre il modello è forzato ad apprendere più task contemporaneamente perchè potrebbe migliorare le prestazioni, poichè la conoscenza è condivisa tra i livelli neurali.
L'obiettivo principale dell'IG model è fornire un percorso di campionamento che massimizzi l'accuratezza con il minor numero possibile di osservazioni. In altre parole, si cerca di ottimizzare l'incremento marginale di accuratezza a ogni fase del campionamento.
Sfruttando le previsioni di IG model il successivi punto di vista viene scelto con 3 differenti strategie:

- POV con loss minore (maggiore nel caso di CE) cioè quello con minimo errore.
- POV con entropia minima (massima nel caso di CE) cioè quello con più guadagno informativo.
- POV scelto randomicamente

Sia il modello base che IG model vengono eseguiti su una sequenza di osservazioni lunga N.

Tramite IG model viene scelto il successivo punto di vista migliore a seconda della strategia scelta. Il nuovo punto di vista viene aggiunto alla sequenza di input iniziale, che avrà quindi $N + 1$ osservazioni, e viene nuovamente valutata dal modello base.

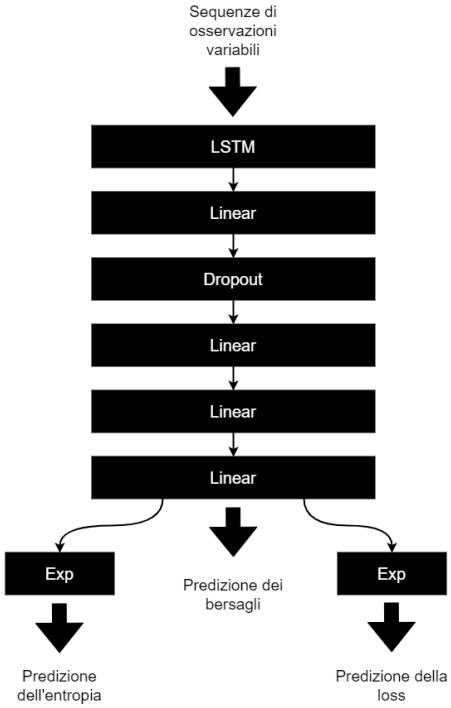
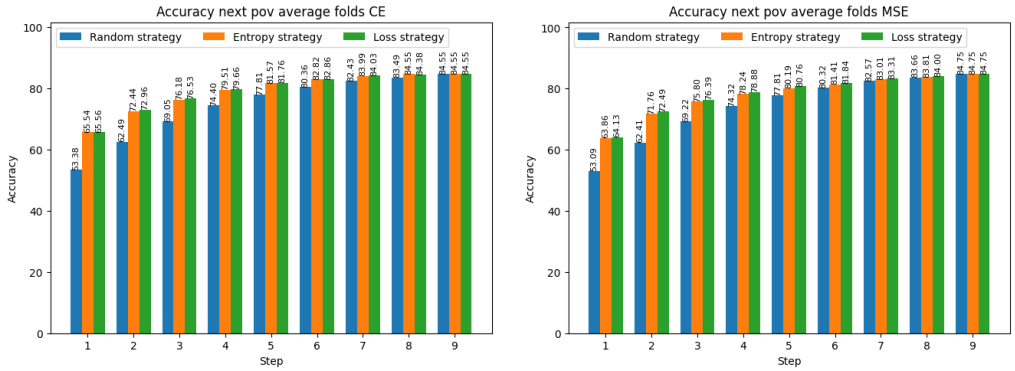


Figura 2.3: Architettura di IG model



Come si può vedere dai risultati la strategia di minimizzazione dell'entropia e della loss superano di gran lunga la strategia casuale.

2.2 Ambiente

L'ambiente è stato creato utilizzando le librerie Gymnasium e PyGame e rappresenta una griglia quadrata di dimensioni $N \times N$, composta da celle.
A ogni cella è assegnato un ID univoco, generato casualmente combinando tre elementi: image_id, marker_count e box_count, selezionati dal dataset disponibile.
Inoltre, ogni cella ha associati tre valori: il primo indica se la stima dei marker è corretta (calcolato tramite il base model), il secondo suggerisce il miglior punto di vista successivo da visitare (calcolato tramite l'IG model), e il terzo registra le osservazioni effettuate dall'agente per quella cella da ciascun punto di vista.
L'agente, rappresentato da un cerchio rosso, può muoversi nell'ambiente in quattro direzioni: su, giù, destra e sinistra. La sua visione copre un'area di 3x3 celle, permettendogli di osservare contemporaneamente fino a 9 celle.
Le celle vengono osservate da diverse angolazioni a seconda della posizione dell'agente.
Quando l'agente si trova al centro della griglia 3x3, lo schema dei punti di vista può essere rappresentato come segue:

1	2	3
4	5	6
7	8	9

Tabella 2.1: Schema di distribuzione dei vari punti di vista

La figura 2.4 mostra un esempio di ambiente simulato con una griglia 20x20. A ciascuna cella è associato un colore che diventa più scuro man mano che vengono osservati più punti di vista. Su ogni cella è inoltre presente un contatore che indica il numero di punti di vista osservati per quella specifica cella. Il contatore presente su ogni cella è di colore nero se la stima dei bersagli è errata, mentre è blu se la stima è corretta.
L'ambiente viene resettato quando si verifica una delle seguenti condizioni: raggiungimento del numero massimo di step, tutte le celle sono state osservate da tutti e 9 i punti di vista o se tutte le celle hanno una stima corretta dei marker. Ad ogni reset gli ID di ogni cella vengono riassegnati casualmente.

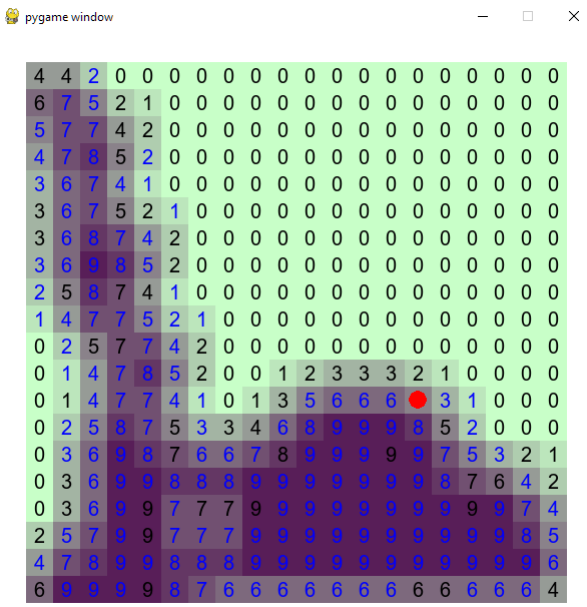


Figura 2.4: Ambiente di 20x20 celle

2.2.1 Algoritmo DQN

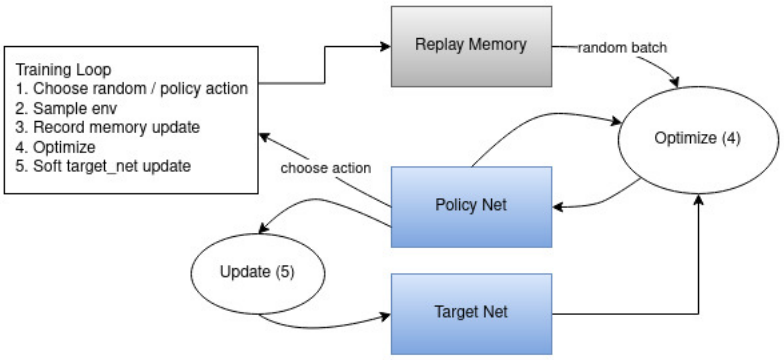
L'algoritmo utilizzato per l'addestramento del modello è il DQN [3] (Deep Q-Network), implementato tramite la libreria Stable Baselines3. Si tratta di una tecnica di reinforcement learning che combina il Q-learning tradizionale con le reti neurali per stimare il valore delle azioni, ed è particolarmente adatta a problemi con spazi di azione discreti.
Il DQN approssima la funzione di azione-valore $Q(s, a)$, che indica quanto sia vantaggioso eseguire un'azione a in uno stato s, con l'obiettivo di massimizzare il reward cumulativo a lungo termine. La rete neurale stima questa funzione per ogni possibile azione, permettendo all'agente di prendere decisioni ottimali.
Durante l'interazione con l'ambiente, le transizioni vengono memorizzate in un buffer chiamato replay buffer.
L'algoritmo utilizza due reti neurali:

- **Q-Network** (rete principale): quella che viene allenata e utilizzata per prendere decisioni.
- **Target Network**: una copia della rete principale, aggiornata periodicamente, usata per stabilizzare il processo di aggiornamento della rete principale.

DQN usa la Bellman Equation per aggiornare la funzione Q:

$$y = r_t + \gamma \max_{a'} Q'(s_{t+1}, a')$$

dove y è il target, r_t è la ricompensa istantanea e γ è il fattore di sconto per il futuro. L'obiettivo è minimizzare la differenza tra il valore stimato dalla rete principale $Q(s_t, a_t)$ e il target. La perdita usata da DQN è tipicamente la mean-squared error (MSE).
Durante l'addestramento, l'agente utilizza una strategia di esplorazione epsilon-greedy (esplorazione casuale o selezione basata sul massimo valore Q predetto) per bilanciare esplorazione. L'epsilon diminuisce nel tempo, passando da un valore iniziale alto (per incentivare l'esplorazione) a un valore finale basso (per favorire lo sfruttamento delle conoscenze apprese).



Capitolo 3

Politiche di esplorazione

Sono state implementate e confrontate due politiche di esplorazione:

- **Scelta del punto di vista migliore**: l'agente segue il miglior punto di vista da visitare per ogni cella, come indicato da modello IG.
- **Massimizzazione del guadagno informativo**: l'agente esplora l'ambiente cercando di massimizzare il guadagno informativo (information gain) di ciascuna cella che attraversa.

Ciascun agente addestrato è stato testato in un ambiente composto da una griglia di 20x20 celle con un numero massimo di step di 1000, su un totale di 20 esecuzioni. In ogni esecuzione, l'assegnazione degli ID alle celle varia, ma questa variazione è mantenuta costante tra tutti gli agenti, garantendo condizioni di confronto uniformi.

3.1 Scelta del punto di vista migliore

A ogni cella è associato un valore che indica il miglior punto di vista da visitare, secondo il modello IG. Questo valore viene aggiornato a ogni passo dell'agente in base alle osservazioni effettuate.
Il punto di vista migliore viene assegnato alla cella utilizzando le previsioni del modello IG, secondo due strategie:

- Punto di vista con entropia minore: ovvero quello che massimizza il guadagno informativo.
- Punto di vista con loss minore: ovvero quello che minimizza la perdita.

Sono stati quindi addestrati due agenti: uno che segue il punto di vista che minimizza l'entropia e l'altro che minimizza la perdita.

L'agente riceve i seguenti reward:

- +8 per ogni cella visitata dal punto di vista consigliato
- +1 per ogni cella con stima errata osservata.
- -2 se l'agente rimane fermo.
- +30 se l'intero ambiente viene mappato.

Ogni agente è stato confrontato con un agente che seleziona le azioni in modo completamente casuale.
I risultati ottenuti evidenziano chiaramente che l'agente che minimizza l'entropia (Entropy agent) è il più efficace sia nell'esplorazione dei punti di vista delle celle che nel miglioramento delle previsioni dei bersagli. Questo approccio consente all'agente di raccogliere rapidamente informazioni utili, portandolo a ottenere previsioni più accurate in tempi ridotti rispetto agli altri agenti.
Anche il Loss agent mostra prestazioni solide, seppur leggermente inferiori rispetto all'Entropy agent. Questo conferma che la strategia di minimizzazione della perdita è valida, ma meno efficiente nel guidare l'agente verso una copertura ottimale dell'ambiente e nel migliorare la qualità delle previsioni.
Come previsto, l'approccio casuale del Random agent si dimostra nettamente meno efficace. L'assenza di una politica guidata rallenta significativamente l'esplorazione e la raccolta di informazioni, risultando in un miglioramento marginale delle previsioni e una copertura meno omogenea dell'ambiente.
Come mostrato nelle figure 3.3 e 3.2, il numero di celle con previsioni dei bersagli corrette è superiore rispetto a quello delle celle viste da tutti e 9 i punti di vista. Questo suggerisce che non è sempre necessario osservare ogni cella da tutte le angolazioni possibili per ottenere una stima accurata dei bersagli, e che alcune celle possono essere correttamente stimate anche con un numero inferiore di osservazioni.
Il grafico in figura 3.4 mostra in modo chiaro la capacità dell'Entropy agent di migliorare le previsioni nel tempo. La sua curva di crescita è la più ripida, dimostrando che l'agente riesce a correggere più rapidamente le previsioni sui bersagli rispetto agli altri agenti, richiedendo un numero di step inferiore per ottenere un alto numero di celle con previsioni corrette.
Al contrario, il Random agent mostra una crescita più lenta e costante.
Questo comportamento conferma che, senza una strategia di esplorazione strutturata, l'agente fatica a migliorare significativamente le previsioni sui bersagli, poiché le osservazioni fatte non sono sufficientemente mirate a ridurre l'incertezza in modo efficace.

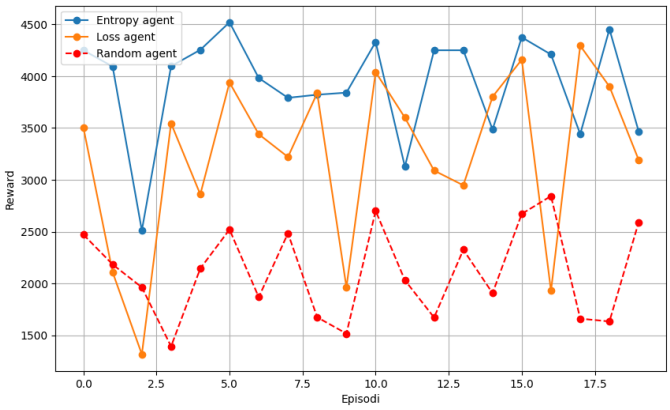


Figura 3.1: Reward cumulativi

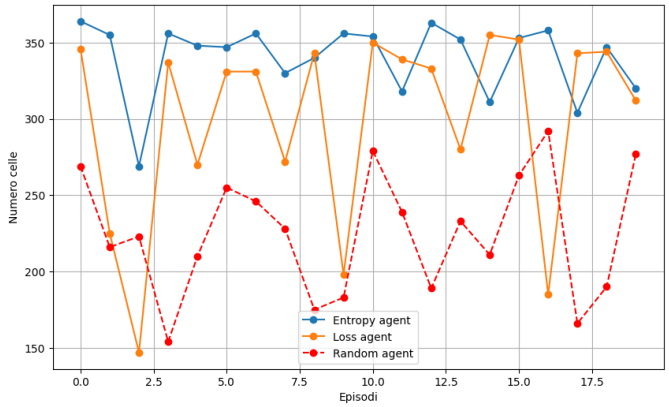


Figura 3.2: Celle con previsione dei bersagli corretta

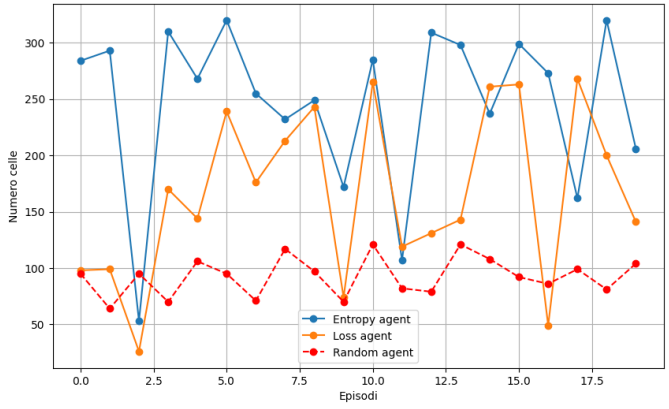


Figura 3.3: Celle con 9 POV visitati

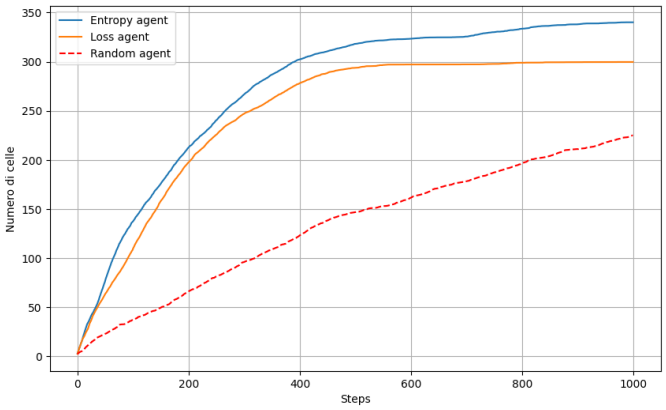


Figura 3.4: Celle con previsione dei bersagli corretta ad ogni step (media su 20 esecuzioni)

3.2 Massimizzazione del guadagno informativo

Per ogni cella attraversata dall'agente viene calcolato il guadagno informativo ottenuto. In particolare, l'agente visita le celle da specifici punti di vista e, per ogni cella osservata, il modello base effettua una previsione sui bersagli. L'entropia di Shannon viene calcolata sulla distribuzione di probabilità prodotta dal modello, riflettendo il grado di incertezza della previsione.
Il guadagno informativo per una cella è calcolato come la differenza tra l'entropia massima (ossia l'entropia della distribuzione con massima incertezza) e l'entropia calcolata sulle osservazioni effettuate dall'agente.
Quindi, se H_{max} rappresenta l'entropia massima e H_{obs} l'entropia delle osservazioni dell'agente, il guadagno informativo IG è dato da:

$$IG = H_{max} - H_{obs}$$

Il reward totale dell'agente è la somma del guadagno informativo di tutte le celle osservate:

$$r = \sum_{c \in C} IG_c$$

dove IG_c è il guadagno informativo della cella c e C è l'insieme delle celle osservate. Se il punto di vista di una cella è già stato osservato, il guadagno informativo per quella cella è pari a 0. Inoltre, l'agente subisce una penalizzazione di -2 se rimane fermo.

I risultati evidenziano un chiaro miglioramento rispetto all'agente che si muove in modo casuale, ma mostrano un lieve peggioramento sia in termini di

velocità che di qualità dell'esplorazione, se confrontati con la politica precedente.
In particolare, il numero di celle con bersagli correttamente identificati e il numero di celle osservate da tutti e 9 i punti di vista risultano inferiori rispetto alla politica che sceglie il punto di vista ottimale per ogni cella. Questa differenza è ancora più evidente quando si analizzano i dati riportati nella figura 3.8, che mette a confronto le due politiche.
Questo calo di performance può essere attribuito al fatto che la politica basata sul guadagno informativo, pur essendo efficace, non riesce a identificare con altrettanta precisione il miglior punto di vista da cui osservare ogni cella.

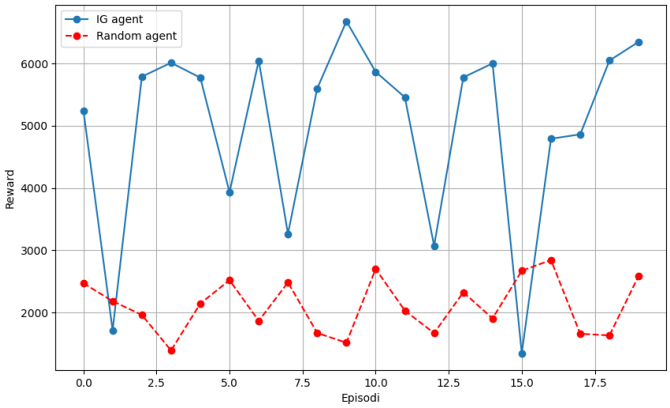


Figura 3.5: Reward cumulativi

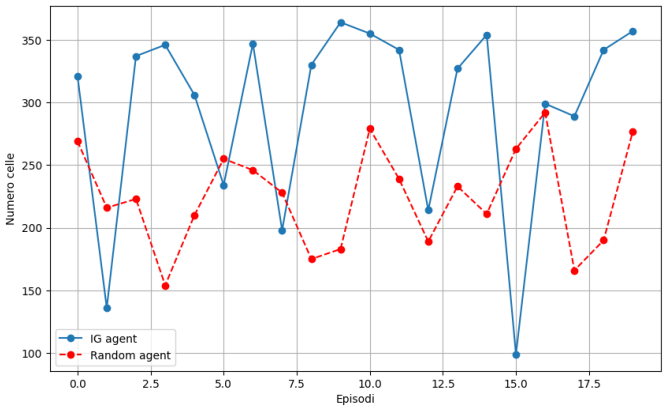


Figura 3.6: Celle con previsione dei bersagli corretta

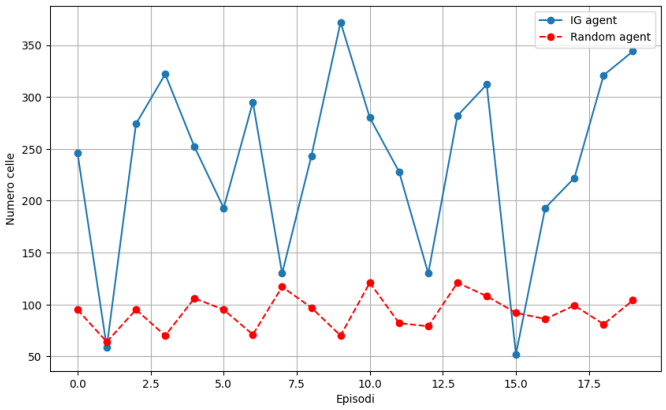


Figura 3.7: Celle con 9 POV visitati

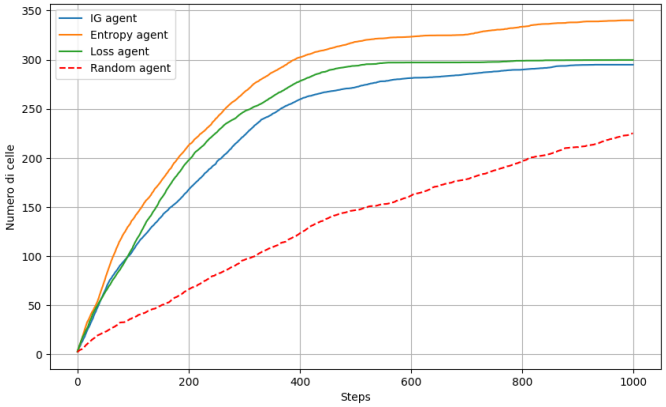


Figura 3.8: Celle con previsione dei bersagli corretta ad ogni step (media su 20 esecuzioni)

3.2.1 Curriculum learning

Questo esperimento ha l'obiettivo di migliorare le performance della politica che massimizza il guadagno informativo, attraverso un processo di addestramento su livelli di difficoltà crescente. L'agente viene allenato su tre livelli, con ambienti di dimensioni progressivamente più ampie:

- Livello 1: 5x5 celle con un massimo di 250 step per episodio,
- Livello 2: 10x10 celle con 500 step massimi,
- Livello 3: 20x20 celle con 1000 step massimi.

Il livello 3 corrisponde alle dimensioni dell'ambiente utilizzato per addestrare gli altri agenti, ed è quello impiegato per i test.

La politica di esplorazione rimane invariata, concentrandosi sulla massimizzazione del guadagno informativo delle celle attraversate.

I risultati mostrano un chiaro miglioramento rispetto all'agente che non ha seguito questo percorso di addestramento, evidenziando benefici in tutti i contesti testati. In particolare, l'agente addestrato su più livelli esplora l'ambiente in modo più preciso e sistematico rispetto a quello non allenato con questa tecnica.
La stabilità delle performance è superiore, il che indica che l'agente riesce a mantenere un comportamento di esplorazione più coerente.
Un aspetto interessante dei risultati è che, sebbene la qualità dell'esplorazione sia migliorata, superando persino quella della politica che seleziona il miglior punto di vista per cella, la velocità di esplorazione è inaspettatamente peggiorata. Questo significa che, pur riuscendo a coprire l'ambiente in maniera più accurata, l'agente impiega più tempo per farlo, come evidenziato dal confronto mostrato in figura 3.12.
Questo peggioramento della velocità potrebbe derivare dalla maggiore attenzione dedicata all'acquisizione di informazioni dettagliate da ciascuna cella, che rallenta il progresso generale dell'esplorazione. L'addestramento su livelli di difficoltà crescente ha permesso di ottenere un agente più robusto e preciso nelle sue azioni, ma a scapito della velocità. Questo compromesso suggerisce che, in alcuni casi, la qualità dell'esplorazione può prevalere sulla rapidità, a seconda degli obiettivi specifici del task. Tuttavia, per applicazioni in cui la velocità è cruciale, potrebbe essere necessario affinare ulteriormente la politica per trovare un equilibrio ottimale tra accuratezza e tempo di esplorazione.

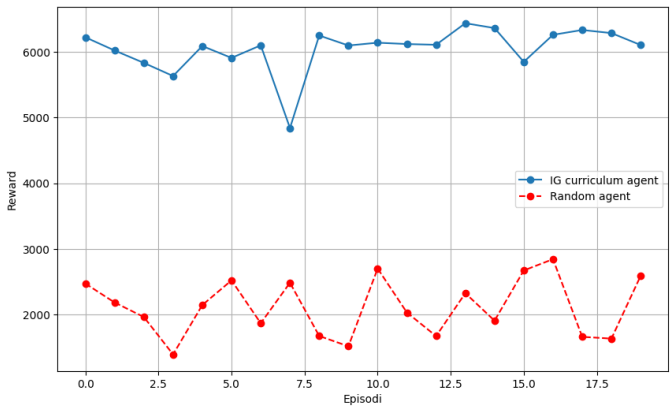


Figura 3.9: Reward cumulativi

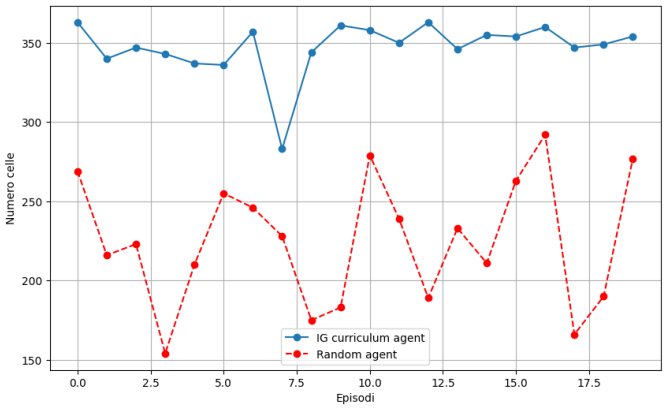


Figura 3.10: Celle con previsione dei marker corretta

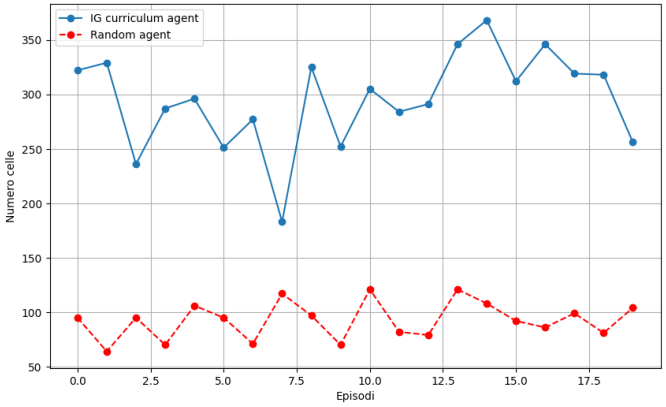


Figura 3.11: Celle con 9 POV visitati

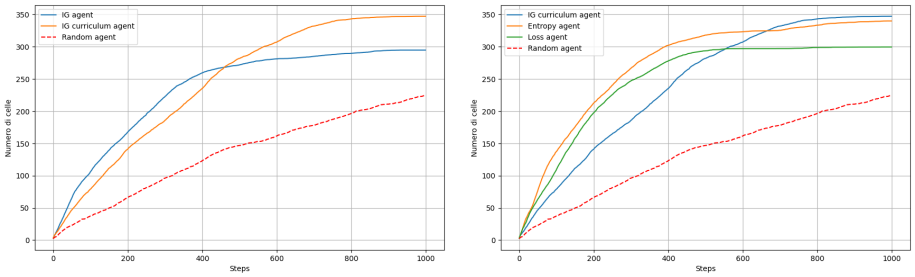


Figura 3.12: Celle con previsione dei bersagli corretta ad ogni step (media su 20 esecuzioni)

Capitolo 4

Conclusione

Sfruttando i modelli predittivi sviluppati, sono state implementate e confrontate due politiche di esplorazione distinte: una che seleziona il miglior punto di vista da esplorare in base alle previsioni dell'IG model, e l'altra che si concentra sulla massimizzazione del guadagno informativo per ogni cella attraversata.
Dai risultati emerge che la politica basata sulla scelta del miglior punto di vista si distingue per la sua efficienza in termini di velocità di esplorazione. Con un numero minore di step, questa politica riesce a ottenere risultati migliori, evidenziando un approccio più rapido nell'acquisire una conoscenza utile dell'ambiente. L'agente che segue questa strategia riesce quindi a coprire l'ambiente in modo più rapido, riducendo il numero di step necessari per ottenere stime accurate.
D'altro canto, la politica che massimizza il guadagno informativo risulta essere superiore in termini di accuratezza. Questo approccio, pur richiedendo più tempo per l'esplorazione, consente all'agente di acquisire una comprensione più approfondita dell'ambiente, migliorando la qualità complessiva delle previsioni dei marker e delle stime.
Entrambe le politiche, tuttavia, dimostrano chiaramente di superare le performance dell'agente che esplora l'ambiente in modo casuale. L'approccio casuale, come prevedibile, risulta inefficiente sia in termini di tempo che di accuratezza, evidenziando la necessità di una strategia guidata per ottenere risultati ottimali.

4.1 Sviluppi futuri

Nonostante i buoni risultati ottenuti, le performance del sistema potrebbero essere ulteriormente migliorate adottando un approccio più personalizzato

per quanto riguarda l'algoritmo di apprendimento. Attualmente, l'algoritmo fornito dalla libreria Stable Baselines3 non consente un elevato grado di personalizzazione per adattare il task in modo ottimale. Sviluppare un algoritmo di apprendimento ad hoc potrebbe permettere un migliore adattamento al problema specifico, ottimizzando le performance dell'agente.
Un'altra possibile area di miglioramento riguarda la valutazione del modello di base e di model IG su diversi sottoinsiemi di dati, variando la dimensione di questi set. Ciò potrebbe fornire ulteriori informazioni su come cambiano le performance al crescere dei dati.
Inoltre, sarebbe opportuno ottimizzare ulteriormente il sistema di calcolo dei reward. Durante i test, è stato osservato che l'agente talvolta si mostra indeciso nelle sue azioni. Questa indecisione è probabilmente dovuta alla visione limitata e locale che l'agente ha dell'ambiente circostante. Poiché l'agente può osservare solo una porzione 3x3 di celle alla volta, non dispone di una visione completa dello stato globale dell'ambiente, il che diventa un problema più marcato man mano che le dimensioni dell'ambiente aumentano.
Una soluzione potenziale a questo limite potrebbe essere l'estensione del problema a un'architettura multi-agente. Implementare un sistema di comunicazione tra agenti permetterebbe loro di condividere informazioni, migliorando la visione globale dell'ambiente e ottimizzando ulteriormente l'efficacia dell'esplorazione. Un simile approccio non solo aumenterebbe la rapidità e la precisione dell'esplorazione, ma ridurrebbe anche l'incertezza locale degli agenti, consentendo una migliore gestione degli ambienti di dimensioni maggiori e complessità più elevate.

Bibliografia

[1] L Marchegiani D Ognibene T Foulsham e GM Farinella. «Active Vision and Perception in Human-Robot Collaboration». In: (2022).

[2] D. Albani C. Carbone et al. «Monitoring and mapping of crop fields with uav swarms based on information gain». In: Autonomous Robotic Systems: 15th International Symposium. Springer (2022), pp. 306–319.

[3] David Silver Volodymyr Mnih Koray Kavukcuoglu, Alex Graves e Martin Riedmiller Ioannis Antonoglou Daan Wierstra. «Playing Atari with Deep Reinforcement Learning». In: (2013).