

Deep Reinforcement Learning per la Simulazione Pedonale: Analisi di Sensitività e Iperparametri

Relatore: Professore Giuseppe Vizzari

Co-relatore: Dottoressa Daniela Briola

Relazione della prova finale di:

Ruben Tenderini

Matricola 879290

Anno Accademico 2023/2024

Indice

Introduzione	1
1 Stato dell'arte	3
1.1 Modellazione e simulazione di pedoni	3
1.2 Agenti Intelligenti	4
1.2.1 Classi di agenti intelligenti	5
1.3 Interazione con ambiente	7
1.4 Reinforcement Learning	9
1.4.1 Introduzione	9
1.4.2 Deep Reinforcement Learning	10
1.4.3 Curriculum Learning	12
1.4.4 PPO	13
1.5 Valutazione modelli e validazione	14
2 Un modello di DRL per la simulazione di pedoni	16
2.1 Strumenti	16
2.2 Glossario	17
2.3 Ambiente	19
2.4 Agente	20
2.4.1 Raggi	21
2.4.2 Archi	22
2.4.3 Spazio delle azioni	22
2.4.4 Osservazioni	23
2.5 Rewards	25
2.6 Curriculum	28
2.7 Addestramento	34
2.7.1 Early Fail	34
2.7.2 Fasi di Addestramento	35
2.7.3 Risultati addestramento	35
2.8 Test	37
2.8.1 Ambienti di Test	37
2.8.2 Analisi dei risultati	40
3 Descrizione del progetto software	43
3.1 Componenti Godot	43

3.1.1	Training Scene	43
3.1.2	Level Batch	43
3.1.3	Sync	44
3.1.4	Level Manager	44
3.1.5	Pedestrian	44
3.2	Componenti Python	45
3.2.1	Runner	45
3.2.2	Config	45
3.2.3	Callbacks	47
3.3	Flusso di esecuzione	48
3.3.1	Training Scene	48
3.3.2	Level Batch	49
3.3.3	Level Manager	50
4	Analisi di Sensitività e di Iperparametri	52
4.1	Introduzione	52
4.1.1	Modifiche alla fase di Test	52
4.2	Analisi di Sensitività e degli Iperparametri	53
4.2.1	Studio sui principali iperparametri	53
4.2.2	Studio sull'architettura della rete neurale con strati nascosti di egual dimensione	58
4.2.3	Studio sul tempo di re-training	66
4.2.4	Studio sull'architettura della rete neurale con strati nascosti di diversa dimensione	71
4.3	Analisi Ablativa	73
4.3.1	Curriculum senza Porta Stretta	74
4.3.2	Curriculum senza Corridoio 3vs3	75
4.3.3	Curriculum senza Giunzione a T	77
4.3.4	Curriculum senza Intersezione 4	78
4.3.5	Curriculum senza Doppio Porta Stretta	80
4.3.6	Conclusioni	82
4.4	Analisi ad Alta Densità	82
4.4.1	Ambienti del Modello Esteso	83
4.4.2	Risultati ottenuti	84
4.4.3	Conclusioni	91
5	Conclusione e sviluppi futuri	93
Bibliografia		96

Introduzione

Le *simulazioni di pedoni e folle* stanno acquisendo un crescente interesse in molteplici settori, dalla pianificazione urbanistica alla gestione delle emergenze. Queste simulazioni sono fondamentali per supportare la progettazione di spazi pubblici, prevedere le dinamiche dei pedoni in luoghi affollati e simulare evacuazioni in situazioni di emergenza. La realizzazione di sistemi per la simulazione pedonale è complessa e richiede uno studio approfondito delle dinamiche pedonali e dei vari approcci modellistici disponibili.

Tradizionalmente, i modelli di simulazione pedonale si basano su *algoritmi definiti manualmente* sulla base di teorie in merito al movimento delle persone. Tuttavia, questa relazione esplora un approccio innovativo e ancora poco esplorato: il *Reinforcement Learning* (RL), una branca del *Machine Learning* (ML) per l'addestramento di agenti autonomi.

L'obiettivo principale della relazione è sviluppare *agenti intelligenti* utilizzando il Reinforcement Learning per simulare realisticamente il comportamento dei pedoni, replicando fedelmente le loro azioni con limitazioni simili a quelle umane, come la visione limitata e capacità decisionali ristrette.

Per la creazione degli ambienti virtuali è stato utilizzato *Godot Engine*, con la gestione degli agenti di Reinforcement Learning tramite *Godot-RL-Agents* e l'implementazione degli algoritmi di RL con la libreria *Stable Baselines 3* in PyTorch. Inoltre, è stata impiegata la libreria *PedPy* per l'analisi del movimento pedonale.

Il modello ottenuto offre prestazioni comparabili a quelle dei modelli definiti manualmente e dimostra la capacità di generalizzare quanto appreso durante l'addestramento per risolvere scenari inediti.

In seguito al conseguimento dei comportamenti desiderati e di risultati realistici, si è deciso di effettuare un *analisi di sensitività* e di iperparametri sul modello creato per ottimizzare e velocizzare l'addestramento del modello. Inoltre, è stato svolto uno studio ablativo per verificare quali componenti dell'architettura fossero di maggior rilevanza e utilità.

La relazione è organizzata come segue:

- Nel Capitolo 1 viene presentata una panoramica attuale sulla ricerca nel campo della simulazione pedonale e del Reinforcement Learning.
- Nel Capitolo 2 viene presentato un modello di Reinforcement Learning per la simulazione di pedoni.
- Nel Capitolo 3 viene descritta l'architettura software, in particolare i metodi e le tecniche utilizzate per la realizzazione del progetto.
- Nel Capitolo 4 viene esteso il modello base effettuando un'analisi di sensitività e uno studio ablativo.
- Nel Capitolo 5 viene effettuata una considerazione finale sul lavoro eseguito. La relazione si conclude con alcuni possibili sviluppi futuri.

Capitolo 1

Stato dell'arte

1.1 Modellazione e simulazione di pedoni

Gli approcci alla simulazione dei pedoni e delle folle hanno storicamente adottato un paradigma basato su *agenti* [2], ma ciò non implica necessariamente una rilevanza diretta per il settore dell'intelligenza artificiale (IA). I modelli di pedoni e folle sono stati prevalentemente fondati su approcci fisici, come il modello delle forze sociali, sugli automi cellulari, che variano da quelli più semplici, come [5], a quelli più complessi, come [3] e su modelli più propriamente definiti da agenti autonomi [7]. Gli obiettivi di tali simulazioni sono spesso orientati verso il *realismo visivo*, senza necessariamente perseguire una validazione rigorosa basata su misurazioni di flussi, densità e velocità [15] in una vasta gamma di situazioni.

Un primo tentativo di integrare tecniche di apprendimento automatico nel modellare il comportamento dei pedoni ha utilizzato metodi di apprendimento per rinforzo (Q-learning) e approcci di classificazione [8], come gli alberi decisionali, per scegliere tra un insieme limitato di azioni di movimento disponibili, basandosi sulla percezione della situazione corrente.

Recentemente, diversi studi hanno esplorato l'uso di tecniche di *regressione* per prevedere il vettore di velocità del pedone [1] o prevedere sia la velocità del pedone che la direzione [9]. Ad esempio, alcuni lavori hanno utilizzato reti neurali profonde addestrate con dati di tracciamento dei pedoni ottenuti da filmati di esperimenti , dove la velocità adottata nel fotogramma successivo del video era considerata come la verità di riferimento. Tuttavia, questo approccio presenta limitazioni simili a quelle delle tecniche di previsione delle traiettorie, come un orizzonte di previsione ridotto e una generalizzazione limitata dei modelli ottenuti, che possono riprodurre fedelmente situazioni simili a quelle del set di addestramento, ma risultare meno rappresentativi in contesti diversi.

Più di recente, l'approccio di apprendimento per rinforzo [10] è stato ulteriormente sviluppato, come dimostrato da studi in cui gli autori hanno

definito un modello di percezione che fornisce all’agente informazioni su un insieme finito di agenti vicini, sull’ostacolo più vicino e sull’obiettivo finale. Il modello di azione regola il vettore di velocità dell’agente in termini di variazione dell’angolo e accelerazione/decelerazione.

Il modello discusso in questo contesto è stato significativamente ispirato da tali approcci, ma con la differenza che, invece di eseguire un processo di addestramento specifico per ciascun esperimento, è stato definito un unico processo di addestramento basato su un approccio curriculare. Questa metodologia presenta situazioni che dovrebbero insegnare all’agente competenze di base, garantendo così la possibilità di ottenere risultati plausibili in un’ampia gamma di situazioni. L’*apprendimento curriculare*, infatti, è un approccio generale ma, nel contesto dell’apprendimento per rinforzo, è considerato un promettente metodo di *transfer learning* per ottenere un alto livello di generalizzazione [6] nei modelli comportamentali degli agenti.

1.2 Agenti Intelligenti

Per studiare il comportamento dei pedoni, viene utilizzato il modello dell’agente intelligente. Questo modello è definito come un’entità capace di percepire l’ambiente circostante attraverso sensori e di eseguire azioni tramite attuatori [14]. La Figura 1.1 illustra schematicamente le componenti principali di un agente e le interazioni di base con l’ambiente, rappresentato come l’insieme di tutto ciò che l’agente può percepire.

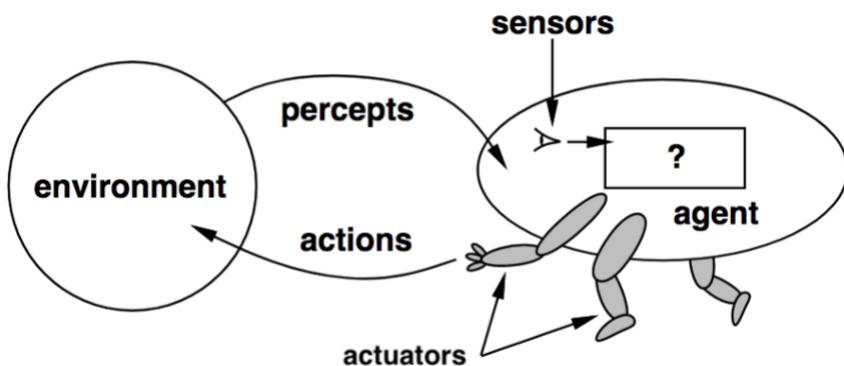


Figura 1.1: Diagramma di un semplice agente intelligente

Gli agenti, nonostante condividano una struttura e un funzionamento di base comune, possono differire enormemente per complessità e capacità. Una classificazione possibile li suddivide in 5 classi in base al grado di intelligenza e capacità di percezione.

1.2.1 Classi di agenti intelligenti

Di seguito viene fornita una breve descrizione di ciascuna classe di agente secondo la categorizzazione di Russell e Norvig, utilizzando gli schemi presenti nel libro [14].

Agenti a riflessi semplici Questi sono gli agenti più semplici e agiscono esclusivamente sulla base della percezione corrente. La condizione necessaria per il funzionamento di questi agenti è la completa osservabilità dell'ambiente.

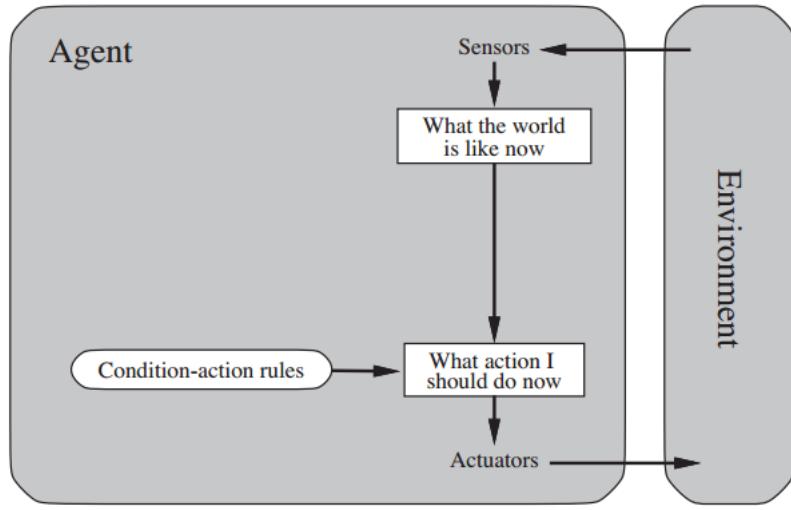


Figura 1.2: Schema di un agente a riflessi semplici tratta da [14]

Agenti a riflessi basati su modello A differenza degli agenti a riflessi semplici, questi possono operare in ambienti parzialmente osservabili. La caratteristica distintiva è la presenza di un modello interno che include uno stato corrente. Questo stato dipende da uno storico delle percezioni e da una serie di regole interne.

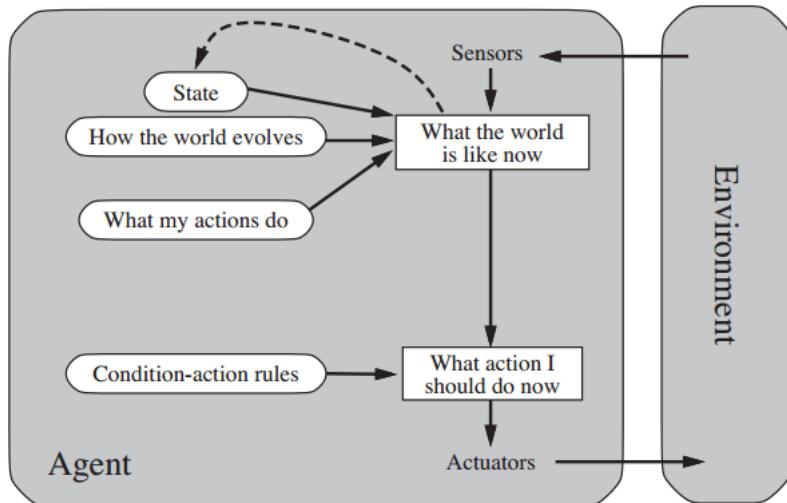


Figura 1.3: Schema di un agente a riflessi basato su modello tratta da [14]

Agenti guidati da obiettivo Gli agenti guidati da obiettivo si basano su un modello e includono una descrizione degli stati obiettivo e di quelli da evitare. Il modello distingue quindi gli stati desiderabili da quelli non desiderabili.

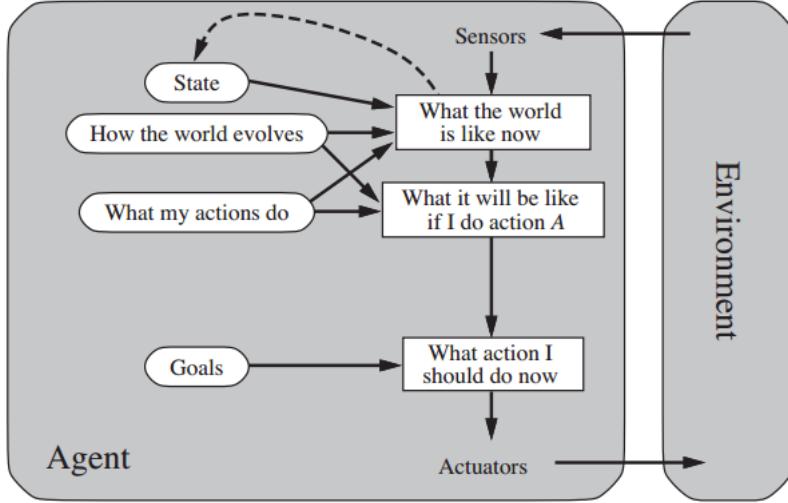


Figura 1.4: Schema di un agente guidato a obiettivi tratta da [14]

Agenti guidati da utility L’evoluzione naturale del modello guidato da obiettivo è il modello basato su utility, che supera la visione binaria della desiderabilità introducendo una metrica. Un agente basato sulla misura di utility ha come obiettivo la massimizzazione di questo valore.

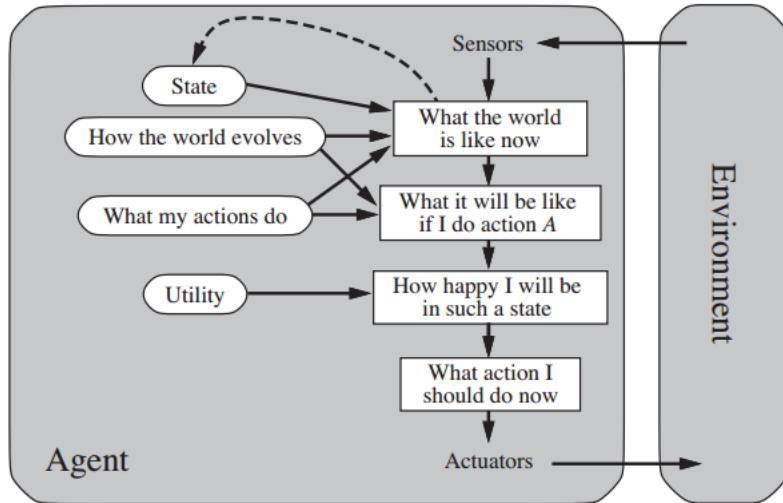


Figura 1.5: Schema di un agente guidato da utility tratta da [14]

Agenti capaci di apprendimento La massima espressione degli agenti intelligenti è rappresentata dagli agenti capaci di apprendere. Questi agenti iniziano da un ambiente completamente sconosciuto e, nel tempo, apprendono i comportamenti migliori in base alle risposte dell’ambiente e alla stima della performance attuale dell’agente.

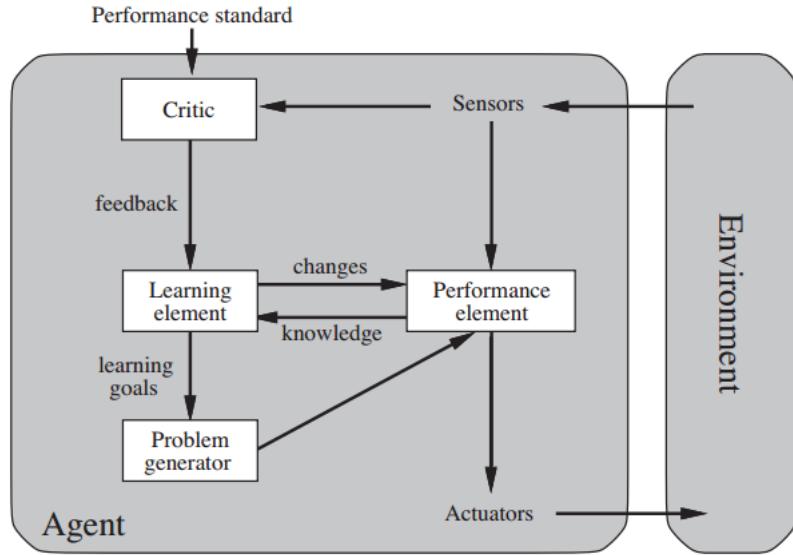


Figura 1.6: Schema di un agente capace di apprendimento tratta da [14]

1.3 Interazione con ambiente

Le interazioni tra agente e ambiente possono essere di diversa natura e rappresentano un elemento cruciale per la suddivisione delle classi di agenti. Una prima importante distinzione è tra modelli continui e discreti. Nei modelli continui, le variabili di spazio, tempo e stato sono continue, il che aumenta inevitabilmente la complessità di gestione rispetto ai modelli discreti, rendendo talvolta impossibile l'implementazione. I modelli discreti, invece, accettano una certa perdita di informazioni in cambio di miglioramenti in termini di prestazioni e semplicità del modello. La discretizzazione può essere applicata a una o più variabili, con quella dello spazio tra le più utilizzate, poiché semplifica notevolmente i processi decisionali dell'agente e l'evoluzione dell'ambiente. Inoltre, facilita il calcolo delle statistiche sugli agenti nei modelli microscopici. In Figura 1.7, possiamo osservare un ambiente con spazio e azioni discreto.¹

¹<https://www.chess.com/>



Figura 1.7: Scacchi è un esempio di ambiente con spazio discretizzato

Quando si elabora un modello, è fondamentale definire come l’agente percepisce l’ambiente. Integrare tutte le percezioni sensoriali umane è spesso troppo complesso a livello computazionale, poiché oltre ai cinque sensi, bisogna considerare anche le componenti cognitive e di pianificazione. Nei modelli odierni, la *percezione* dell’agente si basa spesso sulla vista e sulle informazioni che può estrapolare dagli oggetti presenti nell’ambiente, come marker o indicazioni che possono contenere informazioni, o anche da altri agenti. Ad esempio, in [12], viene presentato un modello in cui la percezione dell’ambiente da parte dell’agente è basata esclusivamente sulla vista e tutte le altre informazioni derivano da interpretazioni di ciò che è stato osservato.

L’ambiente, quindi, diventa un fornitore di informazioni che ogni agente può interpretare in modo diverso, proprio come avviene nella realtà. Anche il tempo è una variabile che può essere discretizzata e utilizzata per estrarre informazioni o scandire la rapidità con cui l’agente compie determinate azioni e prende decisioni. L’attivazione degli agenti e la scelta dell’azione successiva possono essere gestite temporalmente in modi diversi: alcuni modelli potrebbero richiedere sincronismo tra gli agenti, attivando decisioni e azioni a determinati momenti temporali, mentre altri potrebbero permettere maggiore indipendenza, con agenti che prendono decisioni autonomamente ogni delta di tempo. In generale, discretizzare il tempo, solitamente tramite l’unità base dei *timesteps*, semplifica notevolmente le operazioni e rende il modello più snello.

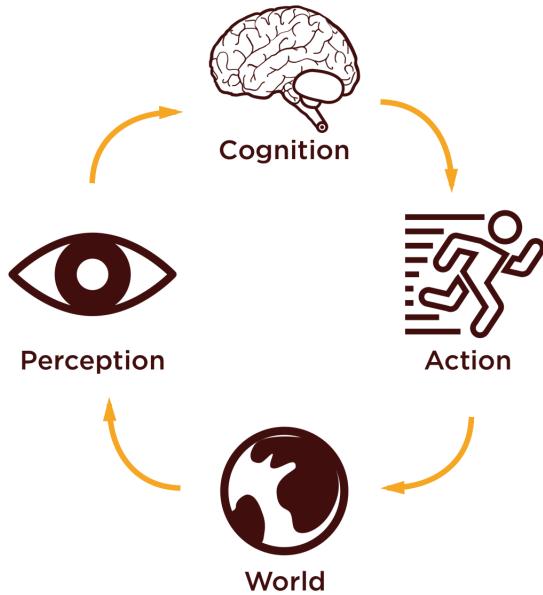


Figura 1.8: Ciclo di percezione, decisione e azione

1.4 Reinforcement Learning

1.4.1 Introduzione

L'*intelligenza artificiale* (IA) è un ramo dell'informatica che si occupa della creazione di programmi informatici capaci di dimostrare intelligenza. Tradizionalmente, qualsiasi software che mostri abilità cognitive come percezione, ricerca, pianificazione e apprendimento è considerato parte dell'IA.

Tutti i programmi informatici che dimostrano intelligenza sono considerati IA, ma non tutti gli esempi di IA possono apprendere. Il *machine learning* (ML) è l'area dell'IA che si occupa della creazione di programmi informatici in grado di risolvere problemi che richiedono intelligenza, apprendendo dai dati. Ci sono tre principali rami del ML: apprendimento supervisionato, apprendimento non supervisionato e apprendimento per rinforzo.

- L'*apprendimento supervisionato* (SL) ha il compito di apprendere da dati etichettati. In SL, un umano decide quali dati raccogliere e come etichettarli. L'obiettivo del SL è generalizzare.
- L'*apprendimento non supervisionato* (UL) ha il compito di apprendere da dati non etichettati. Anche se i dati non necessitano più di essere etichettati, i metodi utilizzati dal computer per raccogliere i dati devono comunque essere progettati da un umano. L'obiettivo in UL è comprimere.

- L'apprendimento per rinforzo (RL) ha il compito di apprendere attraverso il *trial-and-error*. In questo tipo di compito, nessun umano etichetta i dati e nessun umano raccoglie o progetta esplicitamente la raccolta dei dati. L'obiettivo in RL è agire.

Un recente e potente approccio all'apprendimento automatico, denominato deep learning (DL), si basa sull'uso di approssimazioni di funzioni non lineari multi-strato, comunemente note come reti neurali. Il deep learning rappresenta un insieme di tecniche e metodologie avanzate che impiegano reti neurali per affrontare e risolvere una vasta gamma di compiti di machine learning, inclusi l'apprendimento supervisionato (SL), l'apprendimento non supervisionato (UL) e l'apprendimento per rinforzo (RL).

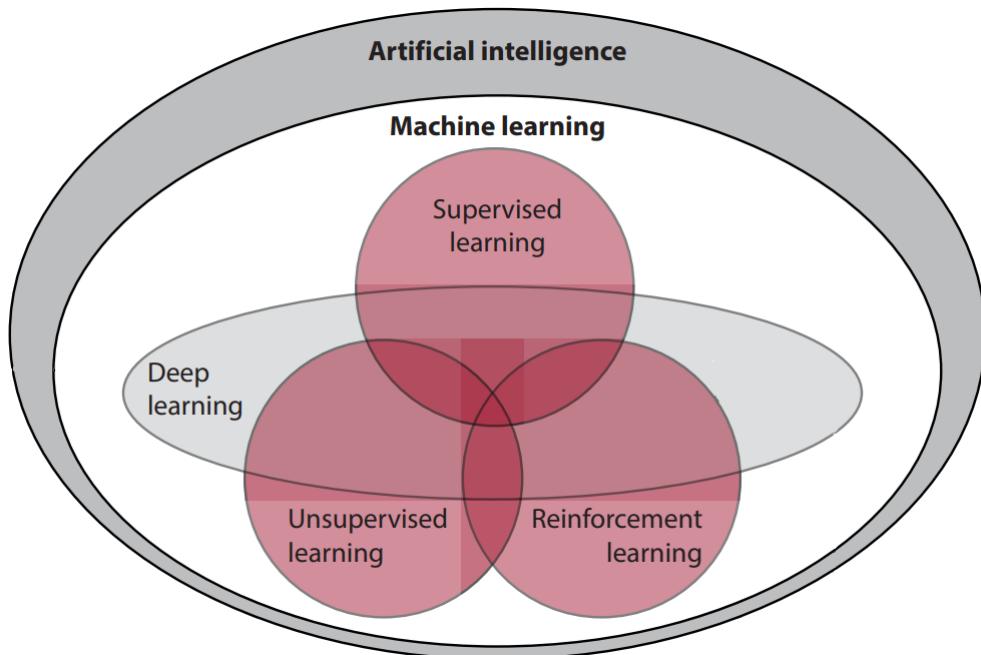


Figura 1.9: Principali ramificazioni del machine learning tratta da [11]

1.4.2 Deep Reinforcement Learning

Il Deep Reinforcement Learning (DRL) è semplicemente l'uso del Deep Learning per risolvere compiti di Reinforcement Learning.

Una definizione più accurata è fornita in [11]: il Deep Reinforcement Learning è un approccio di machine learning all'intelligenza artificiale che si occupa di sviluppare programmi informatici capaci di risolvere problemi che richiedono intelligenza. La caratteristica distintiva dei programmi di apprendimento profondo per rinforzo è quella di imparare attraverso trial-and-error, partendo da un feedback che è al tempo stesso sequenziale, valutativo e campionario, sfruttando potenti approssimazioni di funzioni non lineari.

La Figura 1.10 mostra il ciclo di interazione fondamentale nel processo di apprendimento del Reinforcement Learning. Questo può essere descritto in diverse fasi interconnesse. In primo luogo, l'agente osserva lo stato corrente dell'ambiente, un passaggio cruciale che fornisce le informazioni necessarie per decidere la prossima azione da intraprendere. Successivamente, l'agente sceglie un'azione da un insieme di possibilità, guidato dall'obiettivo di massimizzare le ricompense future. La selezione di questa azione è determinata da una policy, che può essere predefinita o sviluppata nel corso dell'apprendimento. L'azione scelta dall'agente provoca una transizione verso un nuovo stato ambientale, accompagnata da una ricompensa. Questa ricompensa può essere positiva, se l'azione è stata vantaggiosa, o negativa, se l'azione ha avuto esiti sfavorevoli.

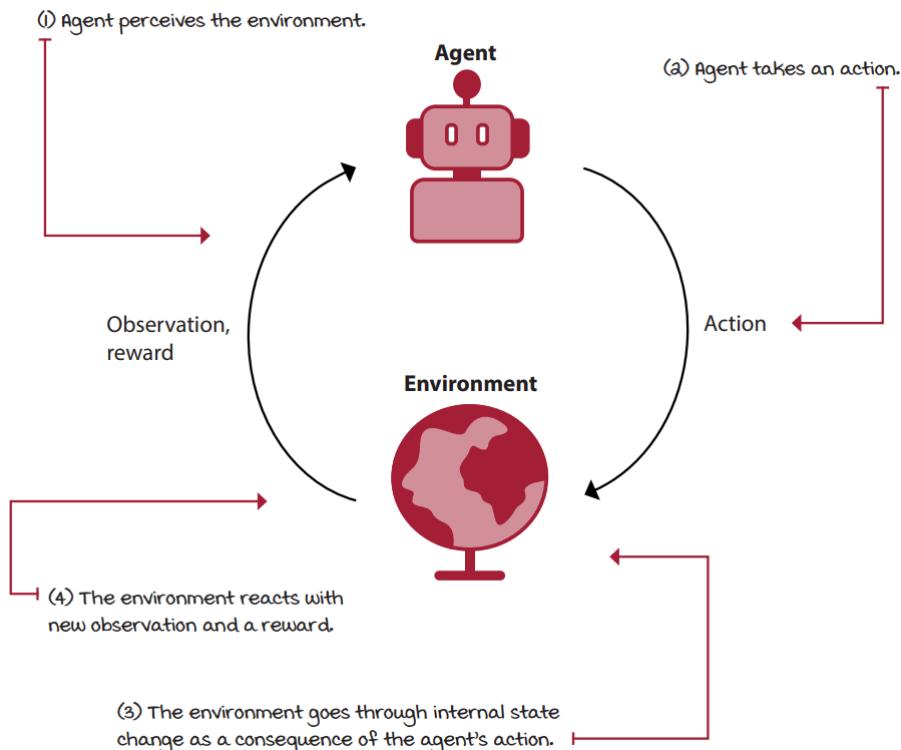


Figura 1.10: Illustrazione del ciclo di interazione nel Reinforcement Learning tratta da [11]

Successivamente, l'ambiente fornisce un feedback all'agente, presentandogli il nuovo stato e la ricompensa correlata. Questo feedback è essenziale per l'apprendimento dell'agente, il quale utilizza queste informazioni per aggiornare e perfezionare la propria policy di selezione delle azioni. Un episodio di apprendimento consiste in una serie di stati, azioni e ricompense che iniziano da uno stato iniziale e terminano in uno stato finale. Attraverso la sperimentazione e l'esperienza accumulate in ogni episodio, l'agente affina progressivamente la propria policy, migliorando le decisioni future.

1.4.3 Curriculum Learning

Spesso l'apprendimento per rinforzo viene applicato direttamente al problema da risolvere, svolgendo un processo di apprendimento specifico, di volta in volta. Viceversa, un obiettivo centrale del nostro lavoro era l'ottenimento di un modello generale, ottenuto da un processo di apprendimento anche elaborato, ma capace di portare allo sviluppo di competenze utili per risolvere problemi di movimento pedonale in una classe di ambienti abbastanza ampia.

Il *Transfer Learning* [20] rappresenta un miglioramento nell'apprendimento di un nuovo compito grazie al trasferimento di conoscenze da un compito correlato che è già stato appreso.

Negli ultimi anni, il Transfer Learning combinato con il Reinforcement Learning ha suscitato un notevole interesse nella comunità scientifica, rappresentando una direzione promettente per affrontare diverse sfide e risolvere problemi aperti nel campo del Reinforcement Learning. Studi approfonditi sul Transfer Learning applicato al Multi-Agent Reinforcement Learning sono stati condotti sia in [18] che in [19].

Il Transfer Learning viene suddiviso in due categorie principali: Intra-Agent Transfer e Inter-Agent Transfer. La prima categoria riguarda il riutilizzo della conoscenza appresa da un agente per risolvere nuovi compiti. La seconda categoria, invece, implica il trasferimento di conoscenza tramite la comunicazione tra agenti differenti.

All'interno della categoria dell'Intra-Agent Transfer, una metodologia rilevante è il Curriculum Learning. L'obiettivo del Curriculum Learning è migliorare sia la velocità di apprendimento sia le prestazioni finali degli agenti. Il curriculum consiste in una sequenza di compiti o ambienti con difficoltà incrementale in cui l'agente deve allenarsi. Gli agenti si allenano su un compito o in un ambiente fino a quando non acquisiscono i comportamenti necessari per superarlo, passando successivamente al compito seguente.

Il principio del Curriculum Learning è ispirato al modo naturale in cui esseri umani apprendono, ottenendo risultati migliori quando i concetti sono presentati in un ordine crescente di complessità. Secondo [4], il Curriculum Learning, nel contesto dell'apprendimento automatico, aiuta a migliorare notevolmente la generalizzazione dei modelli, accelerando la convergenza durante l'addestramento e migliorando la qualità dei minimi locali raggiunti.

1.4.4 PPO

Proximal Policy Optimization (PPO) [17] è un algoritmo di reinforcement learning basato su un’architettura di tipo *actor-critic*. Questa si compone di due reti neurali principali:

1. **Actor**: rete responsabile di selezionare le azioni da compiere data una certa osservazione dello stato dell’ambiente. L’attore apprende una politica, cioè una mappatura tra stati e azioni, che massimizza la ricompensa cumulativa attesa.
2. **Critic**: rete che valuta la qualità delle azioni suggerite dall’attore, stimando la funzione valore. In particolare, il critico calcola il *vantaggio* delle azioni eseguite, che misura quanto un’azione è migliore rispetto alla media delle azioni possibili in un dato stato.

Una delle principali innovazioni di PPO è l’introduzione di una funzione obiettivo surrogata che permette a un algoritmo on-policy di eseguire più step di gradiente sullo stesso mini-batch di esperienze. Questo approccio contrasta con i metodi on-policy tradizionali, come Advantage Actor Critic (A2C), che devono scartare immediatamente le esperienze dopo ogni passo di ottimizzazione.

La funzione obiettivo limitata ("clipped function", in inglese) di PPO permette di evitare che la politica si discosti troppo dopo ogni passo di ottimizzazione. Questa limitazione previene il collasso delle prestazioni dovuto all’elevata varianza intrinseca dei metodi di gradiente di politica on-policy, consentendo al contempo il riutilizzo dei mini-batch di esperienze per eseguire più passi di ottimizzazione.

PPO applica una strategia di limitazione simile anche agli aggiornamenti della funzione valore. Questa tecnica mantiene le variazioni dei Q-values entro limiti specifici, garantendo che le modifiche in termini di parametri siano regolari. Ciò è particolarmente utile per mantenere stabili le prestazioni del modello, anche quando le variazioni nello spazio dei parametri sono significative.

Nel contesto della presente relazione di stage, si ritiene non necessario fornire una descrizione dettagliata degli algoritmi relativi alla funzione valore e alla funzione obiettivo. Questo approccio è giustificato dal fatto che il lavoro di stage non ha apportato modifiche sostanziali a tali algoritmi. Pertanto, si è scelto di concentrarsi sugli aspetti innovativi e sui risultati ottenuti durante

il periodo di stage, riservando eventuali approfondimenti tecnici a fonti già esistenti e ben documentate in letteratura.

1.5 Valutazione modelli e validazione

La valutazione e l'eventuale validazione dei modelli ottenuti rappresentano il passo finale di ogni ciclo di esperimenti. Questo processo avviene attraverso test e metodologie condivise, permettendo il confronto con risultati esterni. Sebbene in campo di simulazione pedonale non esista uno standard universalmente riconosciuto, sono stati proposti schemi per test specifici, come quello descritto in [13], che fornisce una struttura per i test di evacuazione da edifici.

Struttura	Descrizione
Geometry	La configurazione del test
Scenario	Lo scenario di evacuazione da simulare
Expected result	Il risultato atteso che il modello di evacuazione deve produrre
Test method	Il metodo impiegato per il confronto tra il risultato atteso e i risultati simulati
User's actions	Le azioni richieste al tester durante l'esecuzione e la presentazione dei test

Tabella 1.1: Struttura proposta per i test in [13]

Nella maggior parte degli studi in questo ambito, vengono condotte analisi tramite grafici di misure considerate fondamentali. Tra queste, una delle più importanti è la relazione tra velocità media e la densità dei pedoni all'interno di uno spazio definito. Questo tipo di grafico fornisce numerose informazioni cruciali su come si comporta il flusso pedonale e identifica le quantità critiche che possono mettere in crisi il sistema.

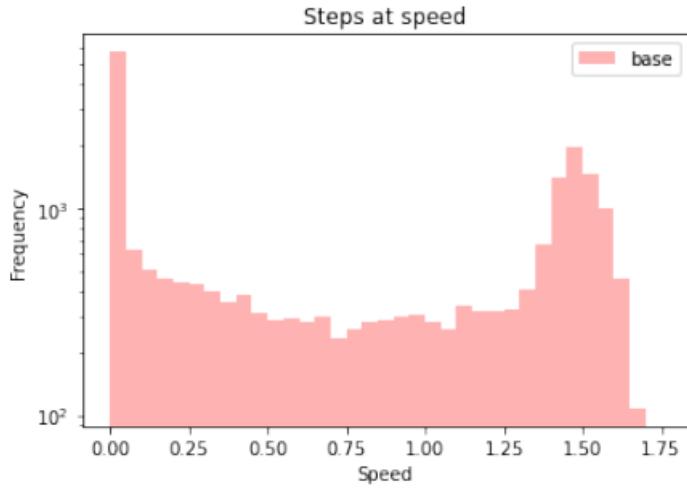


Figura 1.11: Istogramma di distribuzione della velocità di un pedone

La velocità può essere analizzata sotto diversi punti di vista, come il tempo e lo spazio. L'analisi temporale calcola per quanto tempo l'agente mantiene una determinata velocità, rivelando anche i periodi in cui i pedoni sono fermi e fornendo indicazioni utili per la modellazione. L'analisi spaziale mostra graficamente in quali sezioni dell'ambiente l'agente ha velocità maggiori, un ottimo strumento per individuare colli di bottiglia o sbilanciamenti.

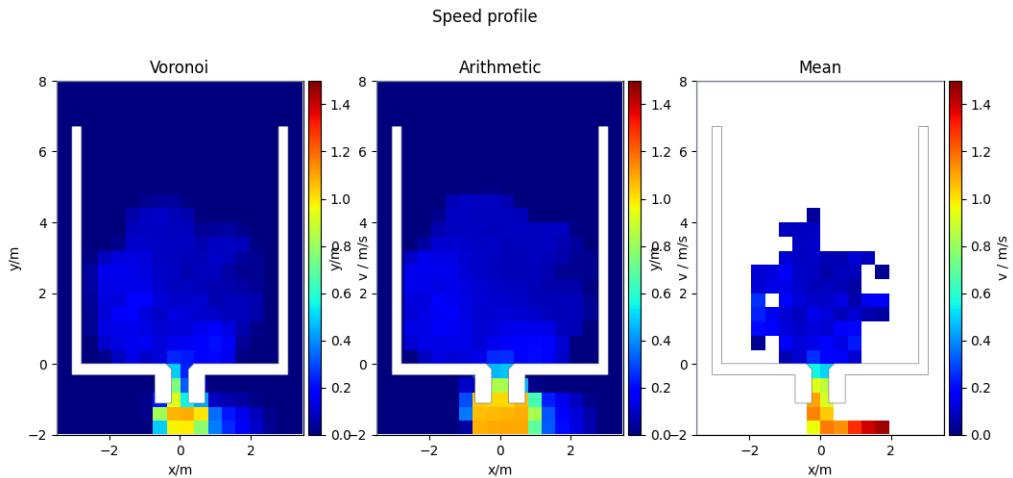


Figura 1.12: Esempio di profilo di velocità su un luogo da [16]

Capitolo 2

Un modello di DRL per la simulazione di pedoni

La prima parte della tesi si è concentrata sulla creazione di un sistema base in grado di sviluppare ambienti, monitorare i movimenti degli agenti e valutare i risultati conseguiti. Questo sistema è stato fortemente influenzato da ricerche precedentemente condotte, come descritto in [21], sebbene con l'adozione di strumenti e tecnologie differenti rispetto al modello attuale. Nella sezione successiva, verranno esaminati nel dettaglio tali strumenti e tecnologie impiegate per la realizzazione del sistema e per il conseguimento dei risultati descritti.

2.1 Strumenti

Gli strumenti e le metodologie principali utilizzati per condurre questo progetto sono descritti di seguito:

Godot Engine

Godot Engine¹ è stato scelto poiché permette di realizzare ambienti in maniera semplice, ma al tempo stesso fornisce una grande libertà nella personalizzazione e configurazione di tutti gli elementi al suo interno. Ad esempio, consente di gestire la fisica dei vari oggetti e realizzare script per il loro controllo utilizzando GDScript, un linguaggio di scripting integrato simile a Python.

Godot-RL-Agents

Godot-RL-Agents² è un pacchetto completamente open source che offre ai creatori di videogiochi e ai ricercatori nel campo dell'intelligenza artificiale, la possibilità di sviluppare comportamenti complessi per i loro personaggi non giocanti (NPC) o agenti. Questa libreria si integra perfettamente con Godot

¹<https://godotengine.org/>

²https://github.com/edbeeching/godot_rl_agents

Engine, permettendo di implementare facilmente algoritmi di apprendimento automatico.

Stable Baselines 3

Stable Baselines 3³ è una raccolta di implementazioni affidabili di algoritmi di apprendimento per rinforzo realizzati in PyTorch. Rappresenta la versione successiva e più avanzata di Stable Baselines, offrendo miglioramenti significativi in termini di prestazioni e facilità d'uso. È stato utilizzato per addestrare gli agenti nel nostro progetto, garantendo risultati stabili e replicabili.

PedPy

PedPy⁴ è una libreria Python open source, con licenza MIT, per l'analisi del movimento pedonale. Tale libreria è stata impiegata per analizzare i movimenti degli agenti, permettendo un'accurata valutazione delle loro dinamiche all'interno degli ambienti simulati.

2.2 Glossario

In questa sezione, si presenta un glossario dei termini rilevanti, utile per una comprensione approfondita dei concetti trattati nel capitolo.

L'**overfitting**, nel contesto del machine learning, si verifica quando un modello apprende troppo bene i dettagli e il rumore del set di dati di addestramento, al punto che le sue prestazioni su nuovi dati di test peggiorano. In altre parole, il modello diventa troppo complesso e specifico per i dati di addestramento, perdendo la capacità di generalizzare bene su dati non visti.

L'**underfitting**, nel contesto del machine learning, si verifica quando un modello è troppo semplice per catturare le tendenze e le relazioni sottostanti nei dati di addestramento. Di conseguenza, il modello ha scarse prestazioni sia sui dati di addestramento che sui dati di test, non riuscendo a rappresentare adeguatamente la complessità del problema.

³<https://github.com/DLR-RM/stable-baselines3>

⁴<https://github.com/PedestrianDynamics/PedPy>

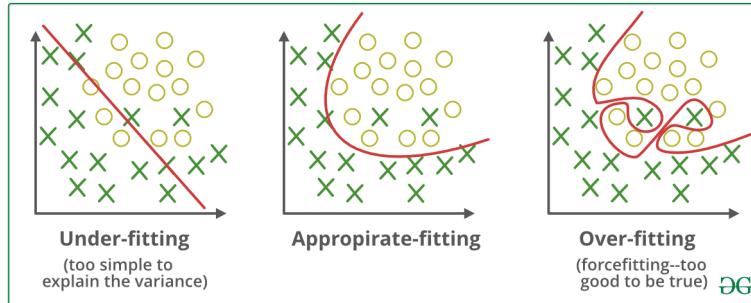


Figura 2.1: Rappresentazione di overfitting e underfitting

Gli **outliers**, ovvero i valori anomali, sono punti di dati che si discostano significativamente dagli altri punti del dataset. Gli outliers possono essere dovuti a vari motivi, come errori di misurazione, variazioni naturali o fenomeni imprevisti. Nel machine learning, i valori anomali possono influenzare negativamente l’addestramento dei modelli, distorcendo le stime e portando a prestazioni inferiori.

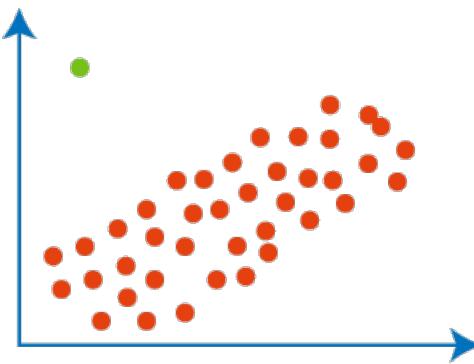


Figura 2.2: Esempio di valore anomalo

Gli **stalli in minimi locali** si verificano quando un algoritmo di ottimizzazione, come la discesa del gradiente, rimane bloccato in un minimo locale, ossia un punto in cui la funzione obiettivo ha un valore inferiore rispetto ai punti circostanti, ma non il valore minimo globale. Questo può impedire al modello di raggiungere le migliori prestazioni possibili, poiché l’algoritmo non riesce a superare questi minimi locali per trovare la soluzione ottimale globale.



Figura 2.3: Esempio di stallo in minimo locale

2.3 Ambiente

L’ambiente è stato creato con l’obiettivo di fornire un’area di simulazione per l’addestramento dell’agente. Lo spazio è considerato continuo e, pur essendo implementato in 3D, a livello modellistico è in sole due dimensioni, in quanto l’altezza non è mai rilevante; per questo viene visualizzato tramite una camera zenitale.

La Figura 2.4 mostra un esempio di uno degli ambienti presenti nel set di addestramento del modello base, contenente tutti gli elementi utilizzati per costruire tali ambienti. Di seguito, viene fornita una descrizione degli elementi presenti nell’immagine:

1. **Pavimento:** si trova sotto l’intero ambiente e funge da base per il posizionamento degli altri elementi.
2. **Muro:** è invalicabile e l’agente non può vedere attraverso di esso. La forma e le dimensioni possono variare.
3. **Target finale:** rappresenta l’obiettivo ultimo dell’agente. Quando l’agente raggiunge il target finale, viene assegnato un grande reward. Anche la forma e le dimensioni del target finale possono variare.
4. **Agente:** si muove all’interno dell’ambiente delimitato dai muri, cercando di raggiungere il target finale. Altri eventuali agenti bloccano la visuale e sono invalicabili.
5. **Target intermedio:** è un obiettivo secondario dell’agente, spesso utilizzato per guidarne il percorso attraverso l’ambiente. La collisione con un target intermedio assegna un reward all’agente. Anche in questo caso, forma e dimensioni possono variare.

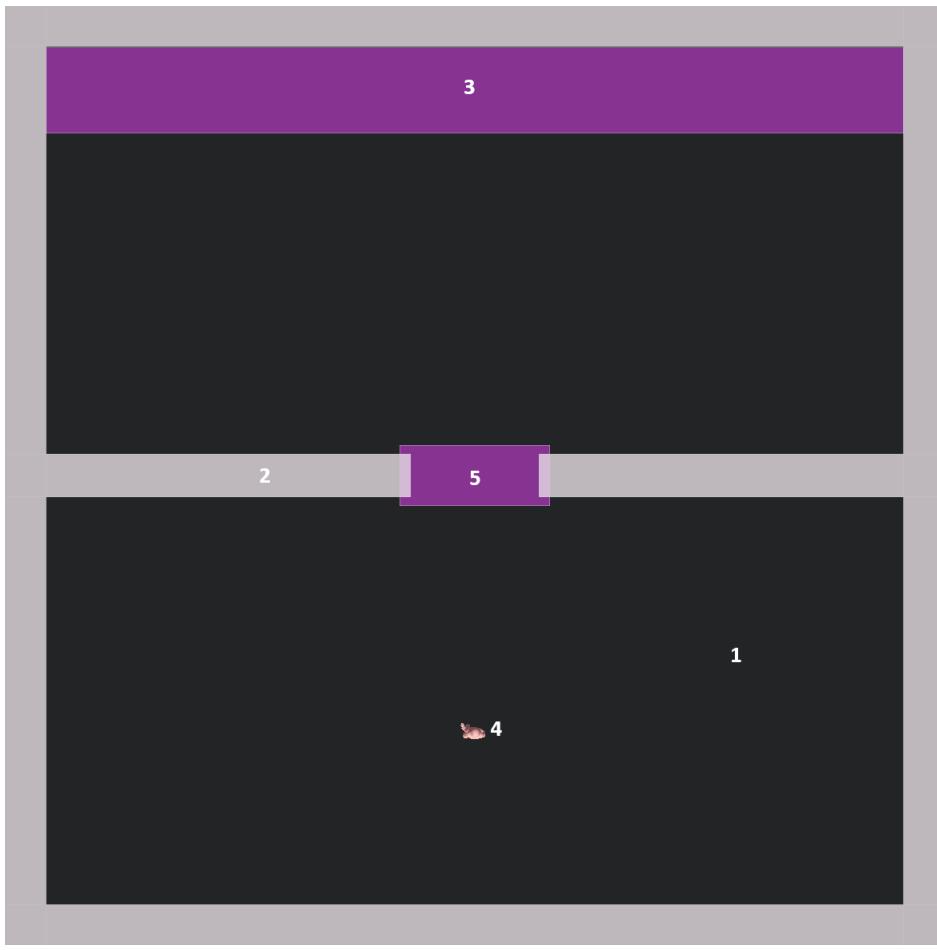


Figura 2.4: Esempio di un ambiente completo di dimensioni 20x20

2.4 Agente

Gli agenti utilizzati sono un'estensione degli agenti definiti nel framework Godot-RL-Agents, progettati specificatamente per apprendere attraverso la tecnica del reinforcement learning. Ogni agente, pur avendo dimensioni e capacità motorie identiche, è stato ulteriormente sviluppato per adattarsi meglio alle esigenze specifiche del nostro ambiente di simulazione. Le caratteristiche di base degli agenti sono le seguenti:

- Dimensioni: la forma del collisore del modello dell'agente è ottenuta tramite un collisore a capsula, con un'altezza di 1.8 metri e un raggio di 0.25 metri.
- Velocità e accelerazione massima: entrambe le caratteristiche verranno trattate in dettaglio nella Sezione 2.4.3.

2.4.1 Raggi

I sensori sono essenziali per un agente, in quanto permettono di raccogliere informazioni sull’ambiente circostante. Gli agenti implementati utilizzano due tipi di sensori: raggi, per la percezione della distanza, e archi, che aiutano a percepire la distanza sociale e personale con altre entità. Questi sensori simulano la capacità percettiva dell’agente e trasformano le informazioni raccolte in osservazioni utilizzabili dalla rete neurale per il processo di reinforcement learning. Questo permette all’agente di adattarsi e reagire in modo ottimale alle dinamiche dell’ambiente.

I raggi utilizzati dagli agenti sono creati mediante *raycast*. Questi si interrompono al contatto con un oggetto nell’ambiente, rilevando informazioni su distanza e tipo dell’oggetto impattato. Benché tutti i raggi forniscano queste informazioni di base, ne esistono vari tipi specifici per osservare differenti categorie di oggetti, come approfondiremo più avanti.

Per emulare la visione umana, i raggi non sono distribuiti uniformemente; sono infatti più densi al centro e si diradano progressivamente verso i lati. Questa particolare disposizione è regolata dall’Equazione 2.1, implementata per ottenere tale effetto di distribuzione.

$$\alpha_i = \min(\alpha_{i-1} + \delta \cdot i, \text{max_vision}) \quad (2.1)$$

Il parametro δ è fissato a 1.5, mentre il valore massimo di visione è impostato a 90 gradi e α_0 è inizializzato a 0. Di conseguenza, i raggi sono proiettati secondo le seguenti gradazioni: 0, 1.5, 4.5, 9, 15, 22.5, 31.5, 42, 54, 67.5, 82.5 e 90 gradi. Queste gradazioni sono distribuite sia in direzione positiva che negativa, garantendo agli agenti una visione complessiva di 180 gradi. Il numero totale di raggi restituiti dalla distribuzione è pari a 23, poiché il doppio raggio a 0 gradi viene eliminato in quanto superfluo.

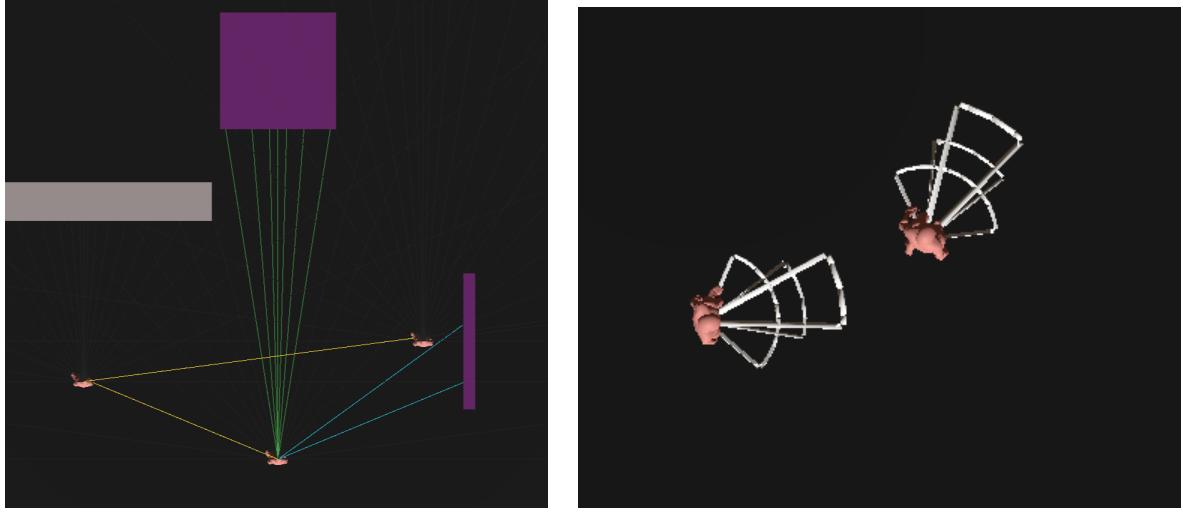
Tuttavia, è necessario disporre di diversi tipi di raggi in grado di fornire informazioni specifiche in base al tipo di oggetto osservato. Sono stati sviluppati due tipi principali di raggi:

- Raggi Muri+Target: Questi raggi sono progettati per rilevare collisioni con muri e target (sia i target intermedi che quelli finali).
- Raggi Muri+Agenti: Questi raggi sono configurati per rilevare collisioni con muri e altri agenti.

Di conseguenza, il numero totale di raggi per ciascun agente è fissato a 46. Una rappresentazione dei raggi è mostrato in Figura 2.5(a)

2.4.2 Archi

Al fine di fornire all’agente osservazioni riguardanti la prossemica, sono stati sviluppati degli archi con differenti raggio e ampiezza, progettati per rilevare la presenza di entità al loro interno. Questi archi possono rilevare sia muri che agenti. Gli archi sono posizionati a distanze di 0.6m, 1m e 1.4m. L’ampiezza di ciascun arco è determinata dal numero di raggi che contiene: l’arco meno ampio contiene 9 raggi, il secondo 11 raggi e l’arco più ampio ne contiene 17.



(a) I raggi di visione dell’agente

(b) Gli archi per la prossemica dell’agente

Figura 2.5: I raggi di visione e gli archi per la prossemica dell’agente

2.4.3 Spazio delle azioni

La rete neurale assegnata all’agente elabora le osservazioni raccolte dai sensori e le traduce in una serie di azioni di output. Queste azioni modificano la direzione e la velocità dell’agente secondo diverse regole. Ogni agente possiede una velocità desiderata media di 1.5 m/s con una deviazione standard di 0.2 m/s per simulare il passo di una persona. In questo modello, ogni agente può prendere tre decisioni al secondo, ognuna delle quali potrebbe cambiare la sua velocità e il suo orientamento.

Le azioni sono rappresentate da due valori continui nello spazio [-1,1] che vengono inseriti nelle equazioni utilizzate per modificare velocità e orientamento. Il primo valore, a_0 , viene utilizzato nell’Equazione 2.4 per modificare la velocità.

$$\text{speed}_t = \max \left(\text{speed}_{\min}, \min \left(\text{speed}_{t-1} + \frac{\text{speed}_{\max} \times a_0}{2}, \text{speed}_{\max} \right) \right) \quad (2.2)$$

Dove speed_{\min} è 0 e speed_{\max} è il valore assegnato all'agente secondo quanto detto in precedenza. L'equazione è modellata in modo che l'agente possa passare dalla velocità massima a fermarsi completamente in due azioni (circa 0.66 secondi).

Il secondo valore, a_1 , viene invece utilizzato nell'Equazione 2.3 per modificare la direzione dell'agente.

$$\alpha = \alpha + a_1 \times 25 \quad (2.3)$$

Modificando l'angolo in questo modo, l'agente può girarsi di un massimo di 25 gradi ad ogni azione (circa 0.33 secondi). L'obiettivo di questo modello è simulare la normale camminata di una persona in condizioni ordinarie.

2.4.4 Osservazioni

Le osservazioni sono normalizzate nell'intervallo [0,1] per ottimizzare le prestazioni della rete neurale. L'agente raccoglie informazioni sia attraverso i raggi che in modo intrinseco. Nella Tabella 2.1 è riportato l'elenco delle diverse osservazioni.

Tipo	Osservazione	Valore
Intrinseco	Velocità	Numero
Raggio Muri + Target	Distanza	Numero
	Tipo	One Hot Encoding
Raggio Muri + Agenti	Distanza	Numero
	Tipo	Booleano
	Direzione	Numero
	Velocità	Numero

Tabella 2.1: Osservazioni

La prima osservazione rappresenta la velocità attuale dell'agente, normalizzata secondo la formula 2.4:

$$\text{Velocità} = \frac{\text{speed} - \text{speed}_{\min}}{\text{speed}_{\max} - \text{speed}_{\min}} \quad (2.4)$$

dove speed è la velocità attuale dell'agente, mentre speed_{\min} e speed_{\max} sono rispettivamente la velocità minima e la velocità massima. Questa osservazione permette all'agente di capire a che velocità si sta muovendo.

Muri e Target

I muri e i target sono osservati dall'agente tramite i raggi. I raggi catturano informazioni riguardo al tipo di entità osservata (muro o target) e alla loro distanza. Per normalizzare la distanza, il valore osservato viene diviso per 10 (lunghezza massima di un lato dell'ambiente), e tutti i valori maggiori vengono riportati come 1 (Equazione 2.5).

$$\text{Distanza} = \begin{cases} \frac{\text{dist}}{10} & \text{se } \text{dist} < 10 \\ 1 & \text{altrimenti} \end{cases} \quad (2.5)$$

Per riconoscere l'entità osservata è stato utilizzato un one hot encoding (Equazione 2.6).

$$\text{Tipo} = \begin{cases} [1, 0, 0] & \text{se entità = muro} \\ [0, 1, 0] & \text{se entità = mid target non visitato} \\ [0, 0, 1] & \text{se entità = mid target già visitato} \end{cases} \quad (2.6)$$

L'osservazione dei mid target è utile all'agente per capire se sta percorrendo la strada giusta o se sta tornando indietro, allontanandosi dal target finale. Poiché i raggi emessi sono 23 e ogni raggio, in grado di percepire muri e target, ritorna 4 osservazioni, in totale abbiamo 92 osservazioni.

Muri e Agenti

I muri e gli agenti presenti nell'ambiente sono percepiti dall'agente sempre tramite l'ausilio di raggi. La distanza è uguale ai raggi del tipo precedente (Equazione 2.7).

$$\text{Distanza} = \begin{cases} \frac{\text{dist}}{10} & \text{se } \text{dist} < 10 \\ 1 & \text{altrimenti} \end{cases} \quad (2.7)$$

Per quanto riguarda la distinzione dell'entità che si sta osservando, in questo caso basta utilizzare un booleano in quanto si deve distinguere tra due sole classi (Equazione 2.8).

$$\text{Tipo} = \begin{cases} 1 & \text{se entità = agente} \\ 0 & \text{se entità = muro} \end{cases} \quad (2.8)$$

La terza osservazione riguarda la direzione relativa all'agente osservato. Essa viene calcolata come il prodotto scalare tra la propria direzione e la direzione dell'agente osservato, convertendola poi in un range [0,1] (Equazione 2.9).

$$\text{Direzione} = \begin{cases} \frac{(\text{dir}_{\text{agente}} \cdot \text{dir}_{\text{agente_oss}}) + 1}{2} & \text{se entità} = \text{agente} \\ 0 & \text{se entità} = \text{muro} \end{cases} \quad (2.9)$$

L'ultima osservazione è atta a misurare la velocità dell'agente che si sta osservando, normalizzata con la formula di normalizzazione della velocità vista precedentemente (Equazione 2.10).

$$\text{Velocità} = \begin{cases} \frac{\text{speed} - \text{speed}_{\min}}{\text{speed}_{\max} - \text{speed}_{\min}} & \text{se entità} = \text{agente} \\ 0 & \text{se entità} = \text{muro} \end{cases} \quad (2.10)$$

Nel caso in cui si stia osservando un muro, la direzione e la velocità saranno pari a 0. Anche in questo caso i raggi emessi sono 23 e da ognuno di essi vengono estratte 4 osservazioni, portando il numero totale di osservazioni, per questo tipo di raggi, a 92.

Osservazioni Totali

In totale, abbiamo una misura intrinseca, 92 osservazioni provenienti dal primo tipo di raggi e 92 osservazioni dal secondo tipo di raggi, portando il numero totale di osservazioni a 185.

$$n_{\text{oss}} = 185 = 1 + 23 \cdot 4 + 23 \cdot 4 \quad (2.11)$$

2.5 Rewards

I rewards sono il meccanismo alla base del reinforcement learning in quanto vanno a premiare o punire le azioni e interazioni dell'agente. A seconda di come vengono definiti i rewards l'agente svilupperà vari comportamenti diversi e avrà preferenze e bias verso alcuni pattern di movimento; uno studio sperimentale approfondito sui valori di reward è quindi fondamentale per la definizione di un agente efficace. Di seguito vengono elencati i rewards utilizzati per il modello base.

$$\text{Reward} = \begin{cases} +6 & \text{Raggiunto target finale} \\ +0.5 & \text{Raggiunto target intermedio non ancora visitato} \\ -1 & \text{Raggiunto target intermedio già visitato} \\ -0.5 & \text{Nessun target in vista} \\ -0.5 & \text{Collisione con un muro} < 0.6\text{m} \\ -0.5 & \text{Collisione con un agente} < 0.6\text{m} \\ -0.005 & \text{Vicinanza con un agente} < 1.0\text{m} \\ -0.001 & \text{Vicinanza con un agente} < 1.4\text{m} \\ -0.0001 & \text{Trascorso un timestep} \\ -6 & \text{Scaduti timesteps massimi} \end{cases}$$

Dettagli rewards

In questa sezione vengono spiegate nel dettaglio i vari rewards, analizzando il motivo della scelta dei valori e della struttura per ottenere il comportamento desiderato.

Raggiunto target finale (+6) Quando l’agente raggiunge il target finale, viene assegnata la ricompensa più alta. Questa ricompensa è fondamentale poiché insegna all’agente che, nonostante possa incontrare ricompense negative durante il suo percorso, il raggiungimento dell’obiettivo finale gli garantirà una grande ricompensa, compensando ampiamente le penalità precedenti.

Raggiunto target intermedio non ancora visitato (+0.5) Quando l’agente raggiunge un target intermedio per la prima volta, viene incoraggiato questo comportamento poiché i target intermedi sono posizionati per guidare l’agente nei percorsi ottimali per risolvere l’ambiente.

Raggiunto target intermedio già visitato (-1) Se l’agente raggiunge un target intermedio già visitato, riceve una penalità in quanto non ha seguito il percorso corretto per raggiungere il target finale.

Nessuna osservazione di target (-0.5) Se i raggi dell’agente non rilevano nessun target intermedio, viene assegnata una penalità, poiché questo comportamento indica che l’agente non sa dove si trovi il suo obiettivo (intermedio o finale) e avrà difficoltà a completare l’ambiente.

Collisione con un muro (entro 0.6m) (-0.5) Se l’agente rileva una collisione con un muro, viene assegnata una penalità. Questo comportamento è scoraggiato poiché non è realistico e può causare problemi agli agenti, che potrebbero rimanere bloccati vicino ai muri e avere difficoltà a riprendere il percorso corretto.

Collisione con un altro agente (entro 0.6m) (-0.5) Se l’agente rileva una collisione con un altro agente, viene assegnata una penalità. Questo comportamento è altamente scoraggiato poiché può creare situazioni di stallo con continue collisioni.

Vicinanza con un altro agente (entro 1m) (-0.005) Se l’agente rileva la presenza di un altro agente a una distanza inferiore a 1 metro, viene assegnata una piccola penalità. Questo comportamento è moderatamente scoraggiato poiché potrebbe portare l’agente a scontrarsi con un altro agente se non corregge la propria traiettoria. Questa penalità funge da avvertimento per l’agente a cambiare direzione.

Vicinanza con un altro agente (entro 1.4m) (-0.001) Se l’agente rileva la presenza di un altro agente a una distanza inferiore a 1.4 metri, viene assegnata una penalità lieve. Questo comportamento serve rilevare la presenza di un altro agente nelle vicinanze, dandogli tempo e spazio per cambiare direzione ed evitare una collisione.

Trascorso un timestep (-0.0001) Ogni timestep trascorso comporta una piccola penalità. Questa penalità è importante per insegnare all’agente a preferire una soluzione veloce rispetto a una lenta. La regolazione di questo parametro può influenzare drasticamente il comportamento dell’agente, imponendo maggiore o minore urgenza e potenzialmente portando a comportamenti indesiderati se l’urgenza è eccessiva.

Scaduti i timesteps massimi (-6) Se l’agente non riesce a completare l’ambiente entro il numero massimo di timesteps previsti, l’episodio di training fallisce, e viene assegnata la massima penalità negativa (pari in valore assoluto alla ricompensa per il successo). Questa penalità, insieme a quella per ogni timestep trascorso, insegna all’agente a risolvere gli ambienti entro il tempo previsto.

2.6 Curriculum

Il curriculum del modello base è composto da 11 ambienti, mostrati nella Tabella 2.2 e spiegati brevemente per caratteristiche e scopo.

Comportamento	Ambiente	Retraining
Camminare verso il target	StartEz	No
	Start	Sì
Guardare il target	Osserva	Sì
Navigare in corridoi stretti	StartCorridoio	No
Curvare e evitare prossemica muri	Curve	No
	CurveOstacoli	Sì
Evitare agenti nella stessa direzione	PortaStretta	Sì
Evitare agenti in direzioni diverse	Corridoio3vs3	Sì
	Incroci4	Sì
	T	Sì
Combinazione di tutti i comportamenti	DoppioPortaStretta	Sì

Tabella 2.2: Ambienti d’addestramento

StartEz

Ambiente iniziale, l’agente è posizionato vicino al target finale, come mostrato in Fig 2.6(a). Il fine primario è quello di introdurre l’agente alle interazioni dirette con i target finali, evidenziando come queste conducano a ricompense significative. La posizione dell’agente e del target finale sono fissati.

Start

In questo secondo ambiente, l’agente e l’obiettivo finale vengono posizionati casualmente all’inizio di ogni episodio. L’obiettivo è far sì che l’agente sviluppi la capacità di cercare l’obiettivo finale anche quando non è direttamente visibile, potenziando così la sua capacità di navigazione e di ricerca.

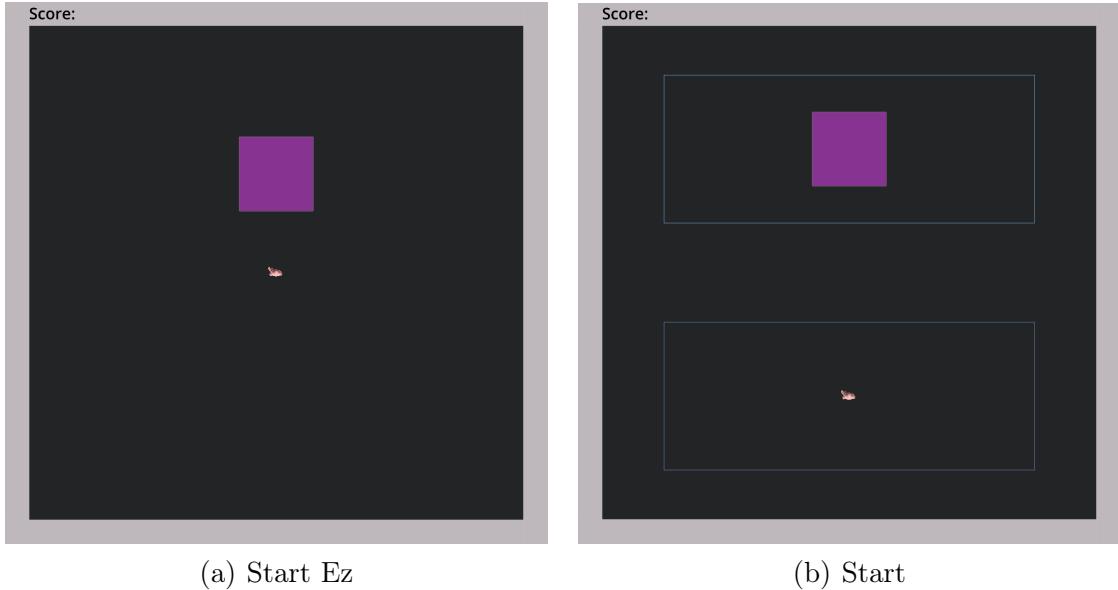


Figura 2.6: Ambienti “StartEz” e “Start” del curriculum.

Osserva

In questo ambiente, l’agente è limitato nella mobilità, potendo solo ruotare su se stesso per esaminare gli elementi circostanti. L’ambiente è caratterizzato da ostacoli e da un obiettivo finale posizionato casualmente all’inizio di ogni episodio. Senza la capacità di spostarsi fisicamente, per l’agente diventa impossibile raggiungere l’obiettivo finale e conseguire un punteggio positivo. Tuttavia, l’agente ha la capacità di mantenere l’obiettivo finale costantemente in vista, cercando così di minimizzare le penalità. L’obiettivo principale di questo ambiente è quindi sviluppare nell’agente abilità di osservazione avanzate.

Start Corridoio

In questo ambiente, l’agente è situato al centro di un corridoio, con l’obiettivo finale che appare casualmente a una delle due estremità. La finalità principale di questo scenario è di abituare l’agente alle interazioni con i muri, dato che all’avvio dell’episodio si trova in prossimità immediata di uno di essi, migliorare le sue strategie di localizzazione degli obiettivi, che non sono visibili all’inizio, e insegnargli a muoversi in linea retta.

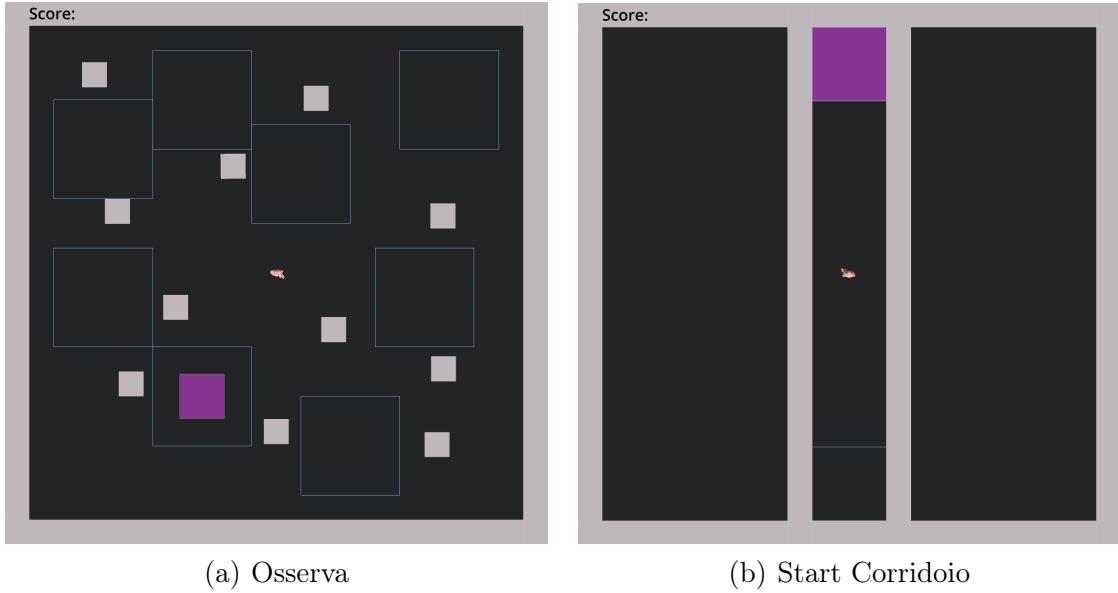


Figura 2.7: Ambienti “Osserva” e “Start Corridoio” del curriculum.

Curve

L’ambiente Curve consiste in un corridoio estremamente lungo, caratterizzato da cinque curve strette e angolate che l’agente deve affrontare prima di raggiungere l’obiettivo finale. L’obiettivo principale di questo ambiente è far acquisire all’agente familiarità con le curve in corridoi molto stretti. Per evitare il fenomeno dell’adattamento eccessivo e migliorare le capacità di generalizzazione dell’agente, una volta completato un episodio, l’ambiente ha una probabilità del 50% di essere ruotato di 90 gradi.

Curve Ostacoli

Questo ambiente presenta una progressione rispetto al precedente mantenendo la struttura di un unico corridoio lungo. Per aumentare la complessità, nei tratti rettilinei del corridoio, le pareti presentano sporgenze che costituiscono ostacoli aggiuntivi. Così come l’ambiente Curve, anche Curve Ostacoli vien ruotato per favorire la generalizzazione ed evitare adattamento eccessivo.

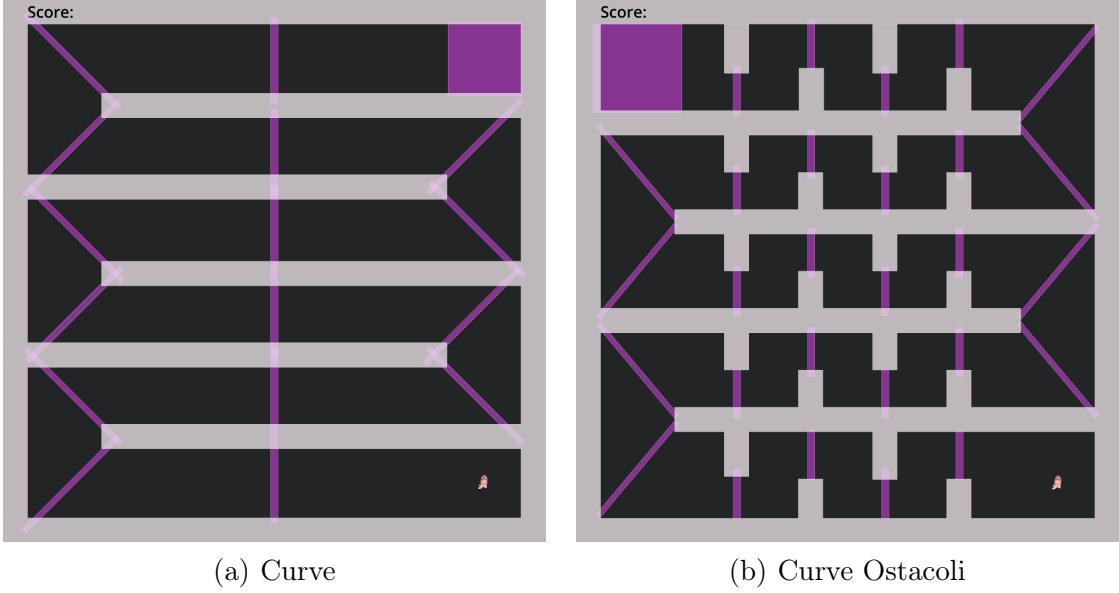


Figura 2.8: Ambienti “Curve” e “Curve Ostacoli” del curriculum.

Porta Stretta

In questo ambiente, si introduce per la prima volta la presenza di più agenti. Gli agenti vengono posizionati all’inizio di un ampio spazio prima di una piccola entrata che contiene un target intermedio. Per prevenire congestioni alla porta o subire penalità rilevanti, è necessario che gli agenti si coordinino e collaborino efficacemente per superare con successo le sfide poste dall’ambiente. Il passaggio viene posizionato casualmente all’interno di un’area specifica intorno al centro dell’ambiente all’inizio di ogni episodio.

Corridoio 3vs3

Questo ambiente rappresenta un’evoluzione rispetto all’ambiente Start Corridoio. In questo caso, sono presenti due gruppi opposti di agenti, ognuno con il compito di percorrere lo stesso corridoio. L’obiettivo cruciale qui è evitare collisioni e ingorghi, permettendo agli agenti di superare l’ambiente in modo efficiente.

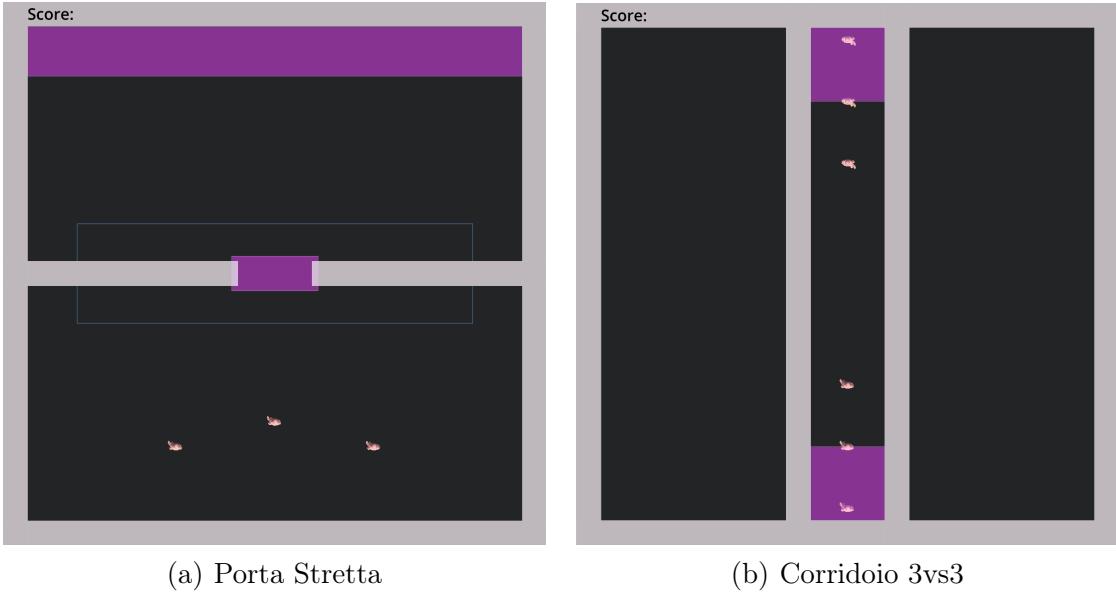


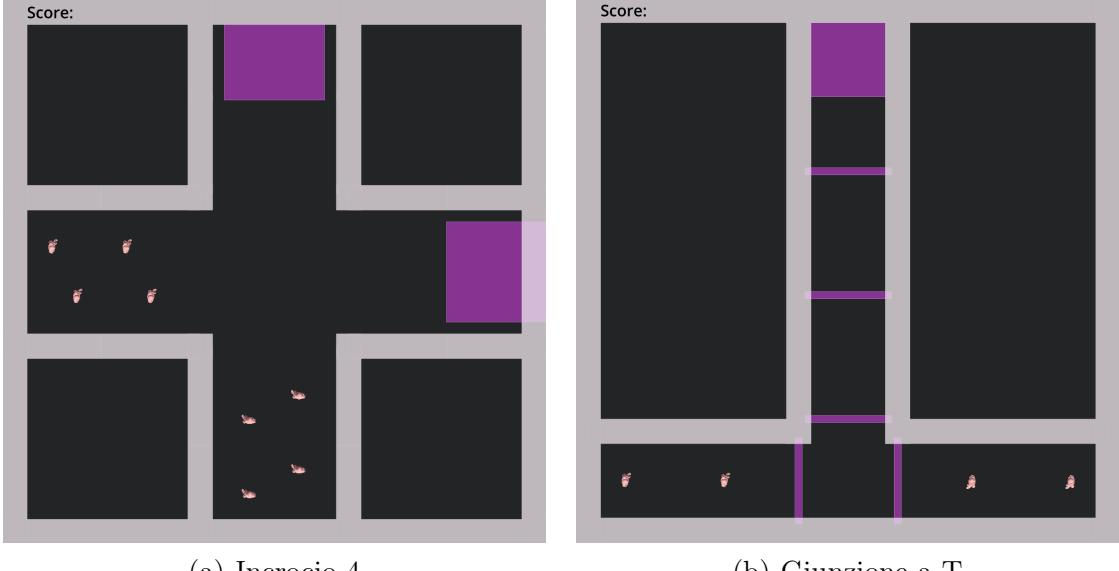
Figura 2.9: Ambienti “Porta Stretta” e “Corridoio 3vs3” del curriculum.

Incrocio4

Questo ambiente presenta un incrocio, con un gruppo di quattro agenti posizionato nella parte sinistra e un altro gruppo di quattro agenti posizionato nella parte inferiore. L’obiettivo di ciascun gruppo è raggiungere il lato opposto dell’ambiente. Tuttavia, per farlo, gli agenti devono affrontare un incrocio complesso formato da otto agenti in totale. Questa configurazione richiede un’interazione e una coordinazione avanzate tra i gruppi di agenti per raggiungere i rispettivi obiettivi senza collisioni o intoppi.

Giunzione a T

In questo ambiente, è presente un incrocio in cui si congiungono due flussi provenienti da due direzioni opposte. Al centro dell’incrocio, i due flussi si uniscono per raggiungere il target finale posizionato in alto. Target intermedi assistono nell’orientamento e nella percorrenza del corridoio. L’obiettivo principale di questo ambiente è insegnare agli agenti a mantenere una traiettoria minimale nelle curve, evitando di intralciare agenti appartenenti ad altri flussi.



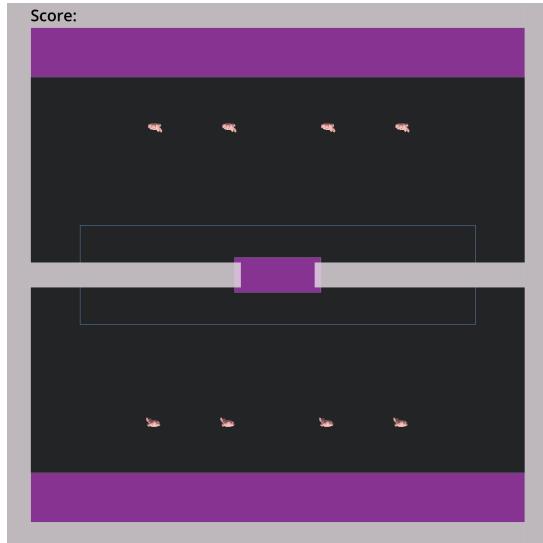
(a) Incrocio 4

(b) Giunzione a T

Figura 2.10: Ambienti “Incrocio 4” e “Giunzione a T” del curriculum.

Doppio Porta Stretta

L’ultimo ambiente rappresenta un’evoluzione di Porta Stretta e introduce un possibile ingorgo che divide due flussi, ognuno composto da quattro agenti, diretti verso i rispettivi target finali. L’obiettivo principale di questo ambiente è potenziare la capacità di coordinazione tra agenti indipendenti nell’attraversamento di spazi comuni con membri del proprio flusso o con flussi antagonisti.



(a) Doppio Porta Stretta

Figura 2.11: Ambiente “Doppio Porta Stretta” del curriculum.

2.7 Addestramento

L’addestramento dell’agente avviene attraverso la somministrazione graduale di ambienti sempre più difficili, come descritto nel dettaglio nella sezione precedente. Il passaggio da un ambiente all’altro è regolato da un punteggio medio minimo necessario: se questo valore viene superato, il training sull’ambiente corrente termina e si attiva l’ambiente successivo.

Il punteggio minimo viene calcolato ogni volta che vengono completati 36 episodi. Al termine di ciascun episodio (quando vengono raggiunti i time-steps massimi o tutti gli agenti hanno raggiunto il target finale), la media del punteggio ottenuto dagli agenti in quell’episodio viene salvata in una struttura dati. Una volta completati i 36 episodi, viene calcolata la media sfrondata, escludendo il 10% dei valori più alti e più bassi per evitare che valori anomali positivi o negativi influenzino il punteggio medio. Il valore ottenuto viene quindi confrontato con lo score medio minimo richiesto: se è superiore, l’agente passa al prossimo ambiente; altrimenti, inizia un altro ciclo di 36 episodi.

2.7.1 Early Fail

L’addestramento tramite reinforcement learning del caso di studio corrente non garantisce la convergenza teorica, considerata la complessità del task e la possibilità di incorrere in esecuzioni particolarmente ostiche durante l’addestramento. Per affrontare questo problema, è stato implementato un sistema di *early fail*.

Il funzionamento del sistema di *early fail* prevede un numero massimo di episodi associato a ciascun ambiente. Una volta superato questo limite, l’addestramento viene interrotto completamente e la run di addestramento viene considerata un fallimento. Prese in considerazione la media e la varianza degli ambienti, sono state create 3 classi di ambienti per determinare il numero massimo di cicli. Considerando che si utilizzano 36 episodi per calcolare la media e verificare se questa supera la soglia minima, abbiamo definito un numero massimo di cicli per poter eseguire una nuova run, in caso non si superi tale soglia:

- Negli ambienti più semplici, sono stati concessi 5 cicli, per un totale di 180 episodi.
- Negli ambienti di media difficoltà, sono stati forniti 10 cicli, per un totale di 360 episodi.

- Negli ambienti più complessi, sono stati concessi 15 cicli, per un totale di 540 episodi.

2.7.2 Fasi di Addestramento

L’addestramento dell’agente è composto da due fasi: training e retraining.

Training

Il training è caratterizzato dalla somministrazione di ambienti di difficoltà sempre maggiore, in accordo con i principi del *curriculum learning*. Per la fase di training si utilizzano 10 copie dello stesso ambiente in parallelo per velocizzare il processo. Una volta che lo score minimo dell’ambiente è raggiunto, viene somministrato l’ambiente successivo del curriculum. Alla fine dell’ultimo ambiente di training inizia la fase di retraining.

Retraining

Il retraining consiste nella ripetizione degli ambienti del curriculum di training con un numero di episodi massimi limitato. Vengono utilizzate 3 istanze per ogni ambiente, e l’addestramento termina dopo un numero di timesteps massimi prestabilito. Questa fase serve per riprendere alcuni degli insegnamenti che potrebbero perdersi durante il processo di training, specialmente se si protrae molto nel tempo.

2.7.3 Risultati addestramento

Risorse hardware utilizzate

Il modello base è stato addestrato più volte su macchine diverse per verificarne la consistenza e stimarne la variabilità. Le due macchine utilizzate supportano Windows 10 e Windows 11 come sistema operativo e hanno la seguente configurazione hardware:

Macchina	CPU	GPU	RAM
Macchina 1	Intel i7-10700F 2.90 GHz	NVIDIA GeForce RTX 3060 12GB	16 GB 3600 MHz
Macchina 2	AMD Ryzen 7 3700X 3.59 GHz	NVIDIA GeForce RTX 3070Ti 8GB	16 GB 3600 MHz

Tabella 2.3: Specifiche delle macchine utilizzate

L’addestramento è stato eseguito molteplici volte per testare la consistenza dell’utilizzo di risorse. Per quanto riguarda memoria e CPU utilizzata, i

valori risultano costanti per tutte le esecuzioni e sono riportati in Tabella 2.4:

Macchina	CPU Utilizzata		RAM Utilizzata	
	Python	Godot	Python	Godot
Macchina 1				
Singolo Agente	22%	17%	243MB	235MB
Multi Agente	55%	36%	243MB	241MB
Retraining	68%	44%	243MB	255MB
Macchina 2				
Singolo Agente	21%	13%	255MB	238MB
Multi Agente	58%	33%	255MB	239MB
Retraining	60%	36%	255MB	246MB

Tabella 2.4: Risorse utilizzate in fase di training

Andamento reward

L’analisi del grafico della reward è fondamentale per valutare l’efficacia del curriculum learning. Questo grafico mostra l’andamento della reward media durante l’addestramento, con il cambio degli ambienti nel curriculum. La parte più interessante è quella relativa agli ambienti di training, mentre la parte dedicata al retraining rappresenta una media di tutti gli ambienti somministrati in parallelo.

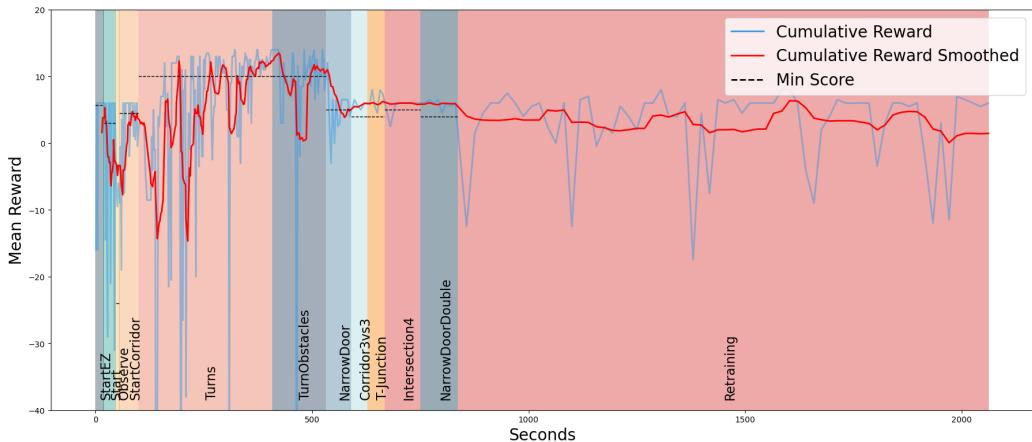


Figura 2.12: Reward Cumulativo

In generale, si osserva un aumento della reward nel tempo, con alcune cadute coincidenti con la fine dell’addestramento degli ambienti che raggiungono il punteggio minimo richiesto. Questo è dovuto al fatto che l’agente si concentra sugli ambienti che ancora necessitano di miglioramento.

Alcuni ambienti, come “Start Corridoio” e “Curve Ostacoli”, mostrano un andamento della reward in costante aumento. Altri, invece, come “Curve”,

presentano fasi di discesa alternate a fasi di salita. Questo comportamento è dovuto alla difficoltà di questi ambienti e alla necessità per l'agente di imparare nuovi comportamenti rispetto a quelli necessari per gli ambienti precedenti.

Per visualizzare invece la durata dei singoli episodi e il tempo totale dell'esecuzione possiamo utilizzare la Figura 2.13

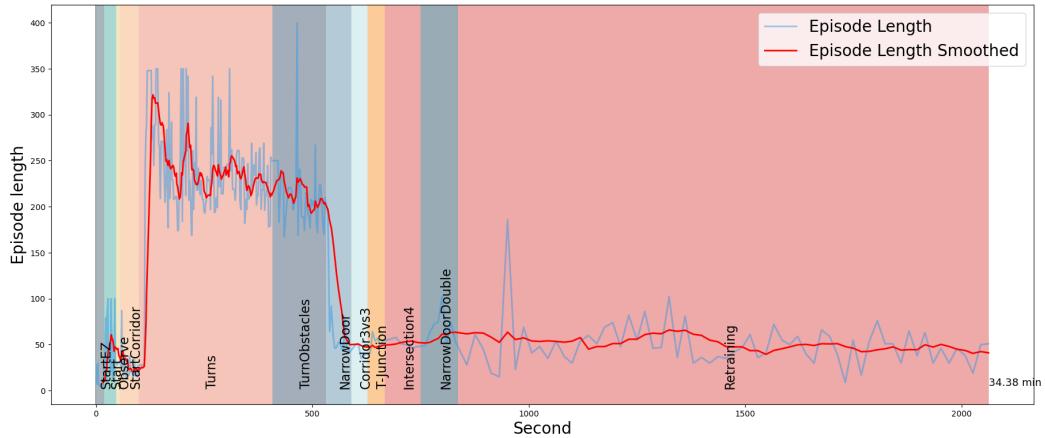


Figura 2.13: Lunghezza degli episodi

2.8 Test

La fase di test serve a misurare quanto efficacemente gli agenti, addestrati secondo il metodo presentato precedentemente, possono applicare le competenze acquisite in contesti nuovi e non previsti.

2.8.1 Ambienti di Test

Di seguito sono riportati gli ambienti di test. È importante notare che questi ambienti non sono noti agli agenti, quindi possono essere considerati una valutazione autentica della loro capacità di generalizzazione.

Curva C

L'ambiente Curva C, illustrato nella Figura 2.14(a), è estremamente basilare, caratterizzato dalla presenza di un unico agente che deve eseguire due curve semplici. Nonostante la sua semplicità, questo ambiente può rivelarsi utile per identificare comportamenti anomali.

Omega

L’ambiente Omega, mostrato nella Figura 2.14(b), è un ambiente di test che coinvolge un singolo agente incaricato di navigare attraverso diverse curve. Questo scenario è cruciale per valutare l’efficacia del modello nella navigazione.

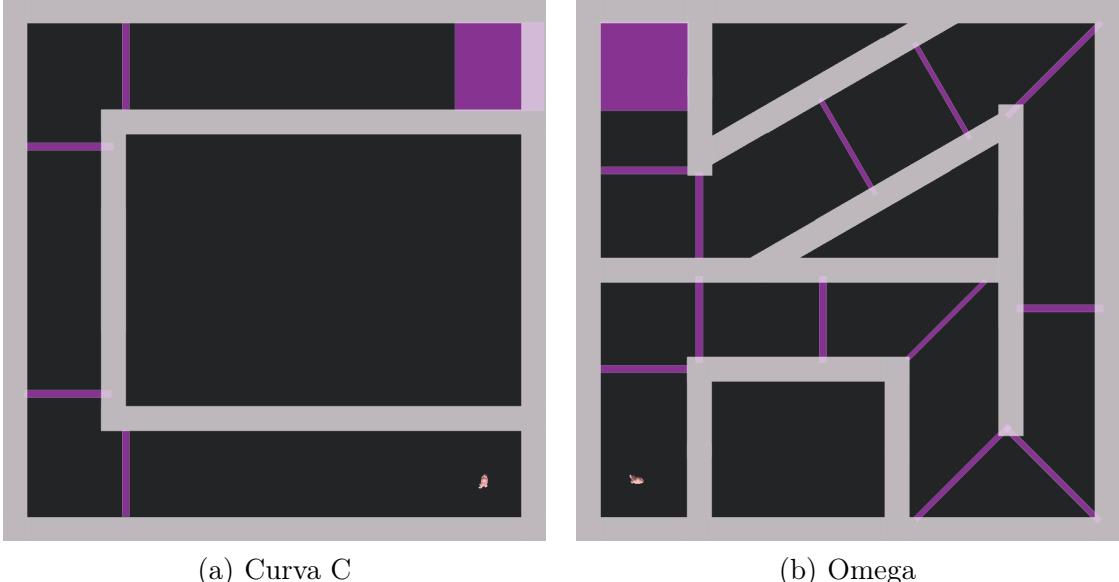


Figura 2.14: Ambienti “Curva C” e “Omega” della fase di test

Ancora

L’ambiente di test Ancora, illustrato nella figura 2.15(a) coinvolge quattro agenti divisi in due gruppi, che partono da corridoi opposti. Dopo un percorso rettilineo, si incontrano dietro una curva con un angolo cieco e devono coordinarsi per attraversare l’ultimo corridoio verso l’obiettivo finale, testando così la loro capacità di evitare congestioni.

Scelta Porta

L’ambiente Scelta Porta, illustrato nella figura 2.15(b), presenta un agente in una posizione casuale nella parte inferiore e l’obiettivo nella parte superiore. Tra loro ci sono due muri: il primo con due porte, il secondo con una sola porta. Questo ambiente testa la capacità decisionale degli agenti quando devono scegliere tra due alternative, elemento non presente negli ambienti di addestramento.

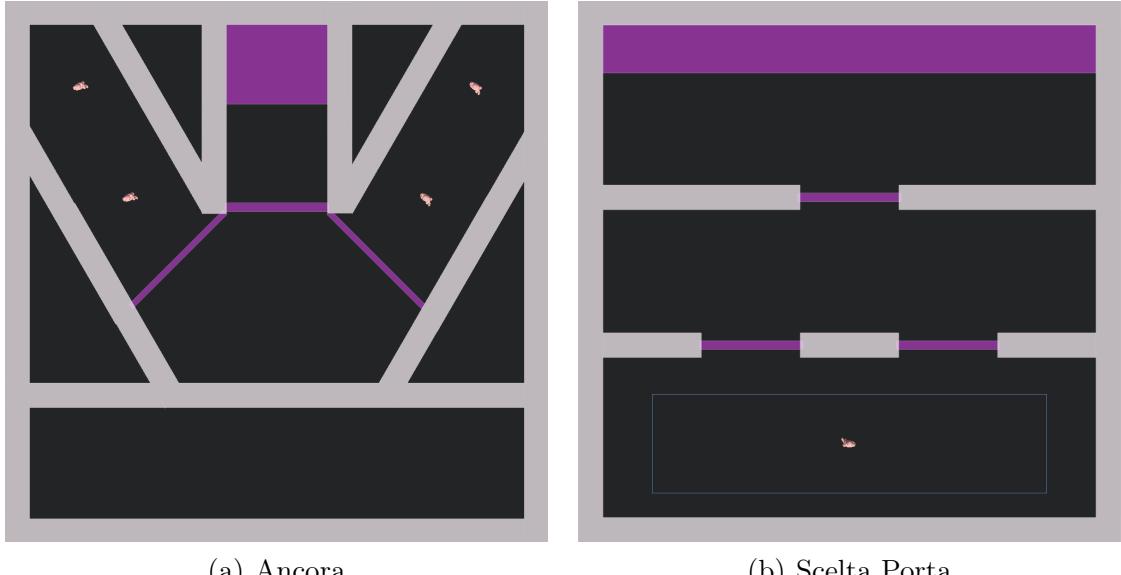


Figura 2.15: Ambienti “Ancora” e ”Scelta Porta” della fase di test

Doppia Porta Opposta

L’ambiente di test Doppia Porta Opposta, presentato nella Figura 2.16(a), include sei agenti divisi in due gruppi di tre. Un corridoio centrale, accessibile attraverso due porte, separa i due gruppi. Gli agenti sono chiamati a coordinarsi con precisione per attraversare simultaneamente le porte e il corridoio centrale, evitando così eventuali congestioni. La complessità di questo ambiente è accentuata sia dalla presenza di numerosi agenti sia dalla necessità di un’attenta coordinazione a causa delle doppie porte.

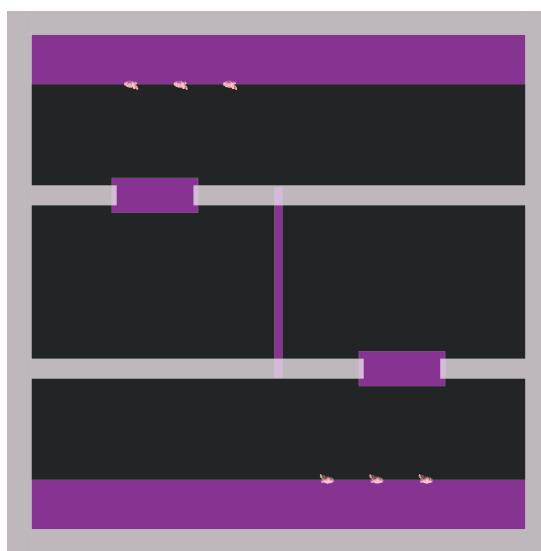


Figura 2.16: Ambiente “Doppia Porta Opposta” della fase di test

2.8.2 Analisi dei risultati

In questa sezione vengono esaminati i risultati ottenuti dai dati raccolti sugli agenti all'interno dell'ambiente di simulazione. Per questa analisi vengono eseguite 10 ripetizioni per ciascun ambiente. Ogni secondo vengono registrati 30 campionamenti riguardanti gli agenti. Quando gli agenti raggiungono l'obiettivo finale un numero prestabilito di volte, i dati vengono salvati su file per effettuare l'analisi.

I dati raccolti vengono successivamente elaborati in un notebook Python. Utilizzando PedPy, esaminiamo il comportamento degli agenti negli ambienti di prova, consentendo una comprensione dettagliata e approfondita del loro comportamento nelle diverse situazioni simulate.

Per il seguente modello, la visualizzazione più utile al nostro scopo, è quella delle traiettorie, in quanto ci permette di visualizzare come gli agenti affrontano i nuovi livelli e siano capaci di portare a termine l'ambiente, raggiungendo l'uscita delle stanze.

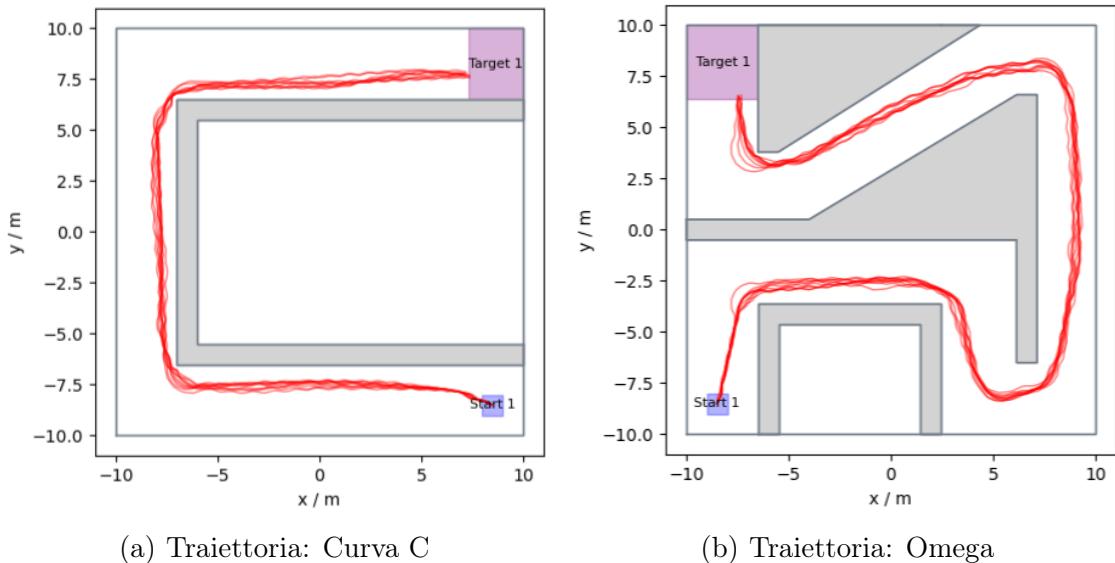
Parallelamente a questa relazione, è stata condotta un'analisi per studiare il comportamento di pedoni in ambienti ad elevata densità. A tale scopo sono stati introdotti nuovi tipi di grafici per visualizzare densità, velocità e tempi, che verranno riportati anche in questo studio solo verso la parte finale.

Curva C

Nell'ambiente “Curva C”, le traiettorie degli agenti seguono con precisione le due curve previste, dimostrando un comportamento coerente e prevedibile. Gli agenti completano il percorso senza deviazioni significative, mostrando la loro capacità di mantenere la traiettoria anche in presenza di curve.

Omega

Nell'ambiente “Omega”, le traiettorie degli agenti mostrano una navigazione fluida e senza interruzioni attraverso le curve. Gli agenti gestiscono con successo la varietà di curve presenti nel percorso, dimostrando un'efficace capacità di navigazione.

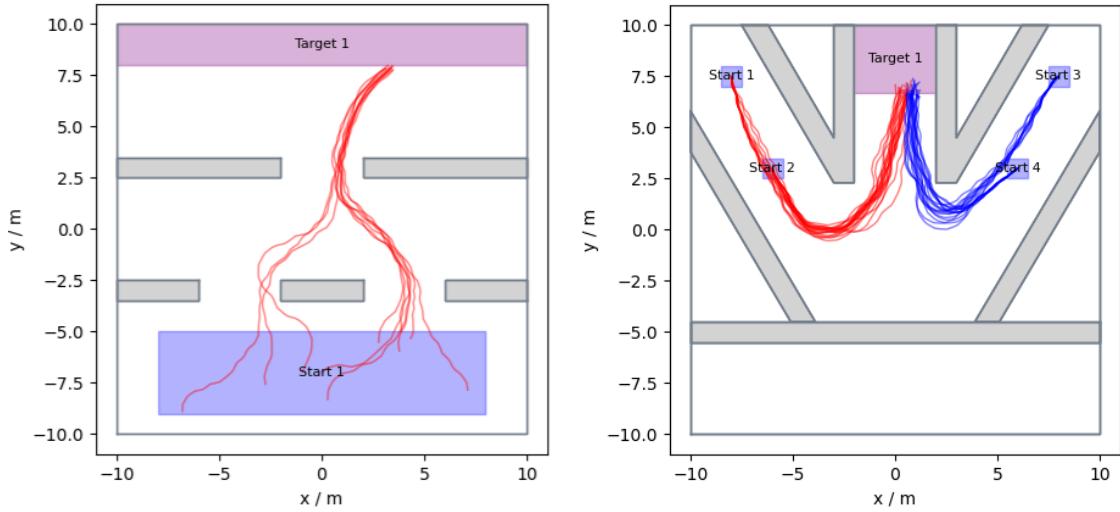


Scelta Porta

Le traiettorie nell’ambiente “Scelta Porta” dimostrano la capacità decisionale degli agenti. Gli agenti scelgono la porta più vicina al punto di partenza e proseguono attraverso la porta del secondo muro senza problemi, evidenziando una navigazione efficace e priva di difficoltà.

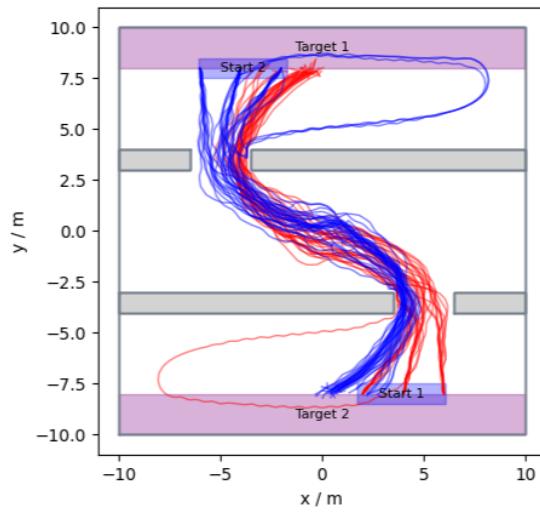
Ancora

Nell’ambiente “Ancora”, le traiettorie degli agenti mostrano un elevato livello di coordinazione. I due gruppi di agenti navigano attraverso i corridoi e si incontrano all’angolo cieco senza causare congestioni.



Doppia Porta Opposta

Nell'ambiente “Doppia Porta Opposta”, la maggior parte dei pedoni riesce a mantenere una buona traiettoria durante l'esecuzione, mentre una piccola parte, a causa di un'elevata densità ai passaggi, crea una diramazione e si allontana, mostrando un comportamento non ottimale. Inoltre, al centro dell'incrocio, non vi è un buon coordinamento tra agenti, in quanto i due flussi non sono ben delineati.



(e) Traiettoria: Doppia Porta Opposta

Capitolo 3

Descrizione del progetto software

In questo capitolo verranno analizzati la struttura e il funzionamento del progetto, al fine di fornire una comprensione più dettagliata della sua organizzazione e operatività. Per una comprensione completa, è possibile consultare il repository del progetto open source realizzato e rilasciato sotto licenza MIT¹. Il codice sorgente, la documentazione dettagliata e gli esempi possono essere visualizzati e scaricati dal repository, consentendo così una valutazione approfondita e la possibilità di contribuire allo sviluppo continuo del modello.

3.1 Componenti Godot

3.1.1 Training Scene

La **Training Scene** è la scena fondamentale che gestisce l’addestramento del modello. Il suo compito principale è eseguire il cambio di scena tra un ambiente di training e il successivo, mentre le condizioni per la transizione sono gestite direttamente lato **Python**. Al suo interno è memorizzato un array contenente gli ambienti di training, specificati nell’ordine in cui devono essere eseguiti. Per eseguire la fase di retraining, la **Retraining Scene** deve essere inserita come ultimo elemento in questo array.

3.1.2 Level Batch

Il **Level Batch** è la scena che istanzia in parallelo i livelli su cui addestrare il modello. Contiene il livello da istanziare e il numero di istanze da eseguire in parallelo. Al termine dell’esecuzione del livello corrente, il **Level Batch** notifica la **Training Scene** per effettuare il cambio di scenario.

¹<https://github.com/Ruben-2828/RL-Godot-Pedestrian-Simulation.git>

3.1.3 Sync

Il nodo **Sync**, aggiunto dalla libreria **godot-rl-agents**, gestisce la comunicazione con **Python**. Le sue funzioni principali sono:

1. Sincronizzarsi con il lato **Python**.
2. Inviare le informazioni relative all’ambiente corrente.
3. Ricevere le azioni scelte dal modello e impostarle ai pedoni corrispondenti.
4. Inviare i reward ottenuti in seguito all’esecuzione delle azioni ricevute.
5. Segnalare la fine del livello corrente quando riceve il messaggio di terminazione da **Python**.

La comunicazione tra **Python** e **Godot** avviene tramite lo scambio di messaggi in formato **JSON** su uno stream **TCP**.

3.1.4 Level Manager

Il **Level Manager** è il nodo che gestisce i singoli livelli. Contiene il livello di cui si occupa e le sue funzioni principali sono:

1. Inizializzare il livello e tutti i suoi elementi (pedoni, target, randomizzazione, ecc.).
2. Notificare la fine di un episodio di training.

3.1.5 Pedestrian

Il **Pedestrian** è l’elemento centrale di questa architettura, incaricato di imparare a orientarsi e spostarsi nei vari ambienti. Si occupa di eseguire le azioni ricevute da **Python** e di calcolare i reward relativi a quelle azioni. I suoi componenti principali sono:

1. Il pedone stesso, che esegue le azioni ricevute (muoversi in avanti e ruotare).
2. L’**AIController**, che riceve le azioni e trasmette i reward e le osservazioni.
3. I sensori, che rilevano ostacoli, target e altri pedoni, computando i reward e le osservazioni.

AIController

L’**AIController** è il nodo che colleziona le osservazioni del pedone e imposta le azioni che deve eseguire. Durante la fase di collezione delle osservazioni, l’**AIController** contatta i sensori del pedone per ottenerle e le invia al nodo **Sync**. Quando deve impostare le azioni del pedone, l’**AIController** riceve le azioni dal nodo **Sync** e le trasmette al pedone.

3.2 Componenti Python

3.2.1 Runner

Il **Runner** è il componente principale per l’addestramento del modello. Gestisce tutte le operazioni necessarie per completare il training. Le sue funzioni principali sono:

1. Caricare i livelli di training tramite il **ConfigParser**.
2. Eseguire ogni livello specificato nelle configurazioni.
3. Verificare se i criteri di terminazione di un livello sono stati raggiunti e notificare **Godot** in caso affermativo.
4. Mantenere un log delle transizioni degli ambienti.
5. Esportare il modello **ONNX** al termine dell’addestramento.

3.2.2 Config

Un elemento fondamentale del progetto è la configurazione lato **Python**. Questa si divide in tre file principali:

1. **Config File**: necessario per specificare gli iperparametri del modello.
2. **Curriculum File**: necessario per specificare gli ambienti del curriculum e i relativi valori associati.
3. **Config Parser**: necessario per validare i dati inseriti.

Curriculum File

Il **Curriculum File** è un file **YAML** che contiene tutte le configurazioni degli ambienti necessarie per l’esecuzione del training. Ogni livello del curriculum è definito con 3 parametri specifici:

1. **mean_reward**: soglia da raggiungere per considerare un livello superato
2. **episode_mean**: numero di episodi sulla quale eseguire la media
3. **cycles**: numero di cicli per la quale è possibile continuare l'esecuzione

Di seguito è riportato il file del curriculum utilizzato durante il modello base del progetto.

Codice 3.1: Curriculum File

```

1 Curriculum:
2   StartEZ:
3     mean_reward: 5.7
4     episode_mean: 36
5     cycles: 10
6   Start:
7     mean_reward: 3.0
8     episode_mean: 36
9     cycles: 5
10  Observe:
11    mean_reward: -24.0
12    episode_mean: 36
13    cycles: 5
14  StartCorridor:
15    mean_reward: 4.5
16    episode_mean: 36
17    cycles: 15
18  Turns:
19    mean_reward: 10.0
20    episode_mean: 36
21    cycles: 10
22  TurnObstacles:
23    mean_reward: 10.0
24    episode_mean: 36
25    cycles: 10
26  NarrowDoor:
27    mean_reward: 5.0
28    episode_mean: 36
29    cycles: 15
30  Corridor3vs3:
31    mean_reward: 4.0
32    episode_mean: 36
33    cycles: 15
34  T-Junction:
35    mean_reward: 4.0
36    episode_mean: 36
37    cycles: 5
38  Intersection4:
39    mean_reward: 5.0
40    episode_mean: 36
41    cycles: 15

```

```
42  NarrowDoorDouble:  
43      mean_reward: 4.0  
44      episode_mean: 36  
45      cycles: 15
```

Config File

Il **Config File** è un file YAML che contiene tutte le configurazioni del modello necessarie per l'esecuzione del training. Di seguito è riportato il file della configurazione del modello base del progetto.

Codice 3.2: Config File

```
1 hyperparameters:  
2     learning_rate: 0.0003  
3     n_steps: 32  
4     batch_size: 320  
5     n_epochs: 3  
6     gamma: 0.99  
7     gae_lambda: 0.95  
8     clip_range: 0.2  
9     clip_range_vf: null  
10    normalize_advantage: True  
11    ent_coef: 0.0001  
12    vf_coef: 0.5  
13    max_grad_norm: 0.5  
14    use_sde: False  
15    sde_sample_freq: -1  
16    target_kl: null  
17    stats_window_size: 1  
18    device: "auto"  
19    verbose: 0  
20    policy_kwargs:  
21        net_arch:  
22            - 256  
23            - 256  
24 max_steps: 1000000000000000000  
25 retraining_steps: 200_000
```

Config Parser

Il **Config Parser** è una classe Python utilizzata per caricare e validare i dati inseriti all'interno del **Config File** e del **Curriculum File**.

3.2.3 Callbacks

Le callbacks sono il principale metodo utilizzato in questa architettura per interrompere l'addestramento del modello. Ci sono due tipi di callback:

1. `EndTrainingOnMeanRewardReachedCallback` per fermare l'addestramento nel momento in cui le condizioni di terminazione sono state raggiunte
2. `EndTrainingOnEarlyFailCallback` per fermare interamente la fase di training quando le prestazioni del modello sono troppo scarse.

Per effettuare una valutazione della prima callback viene eseguita una media sfondata dei reward ogni 36 episodi. Se questa soddisfa i requisiti specificati nel file delle configurazioni del curriculum, il `Runner` interrompe l'ambiente corrente e istanzia il livello successivo sincronizzandosi con il nuovo ambiente lato `Godot`. Se dopo un certo numero di cicli prestabiliti la media non è ancora stata raggiunta, subentra la seconda callback che ferma interamente il training terminando l'intera esecuzione dello script `Python`.

3.3 Flusso di esecuzione

In questa sezione viene spiegato a grandi linee il normale flusso di esecuzione del programma. Per facilitarne la comprensione, il flusso è presentato in diverse sottosezioni, ciascuna delle quali descrive una parte specifica del processo.

3.3.1 Training Scene

La prima sottosezione riguarda la `Training Scene` con il `Level Batch`. All'avvio dell'esecuzione del progetto su `Godot`, lo script della `Training Scene` entra in un loop in cui istanzia il `Level Batch` del primo livello presente nell'array dei livelli.

Una volta che il primo livello terminerà, verrà istanziato il successivo e così via, fino all'ultimo livello: la scena di `Retraining`. Quando anche questa sarà terminata, verrà creato, lato `Python`, un modello in formato `ONNX` assegnabile al nodo `Sync` della `Testing Scene`.

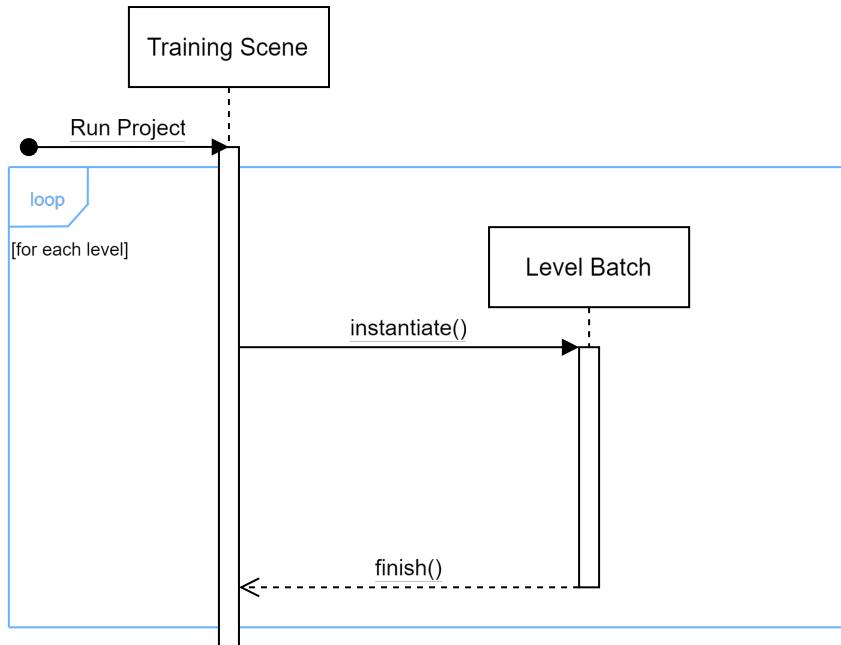


Figura 3.1: Diagramma di Sequenza: Training Scene

3.3.2 Level Batch

In questa fase, il **Level Batch** entra a sua volta in un loop in cui istanzia tanti **Level Manager** quanti specificati nella costante **batch_size**. Questo approccio consente l’addestramento parallelo, accelerando il processo di apprendimento.

Dopo aver istanziato i **Level Manager**, il **Level Batch** svolge un altro compito fondamentale: l’abilitazione del Reinforcement Learning. A tale scopo, il **Level Batch** imposta il nodo **Sync** a **ready**. Il nodo **Sync** si mette in ascolto con **Python** e, una volta stabilita la comunicazione, avviene lo scambio di messaggi. Fino a quando **Python** non invia un messaggio di tipo **close**, il nodo **Sync** continuerà a ricevere l’azione da intraprendere e a restituire il risultato conseguente.

Quando il messaggio di chiusura arriva al nodo **Sync**, questo termina la comunicazione con il **Level Batch**, che a sua volta interrompe la propria esecuzione, permettendo alla **Training Scene** di istanziare il livello successivo.

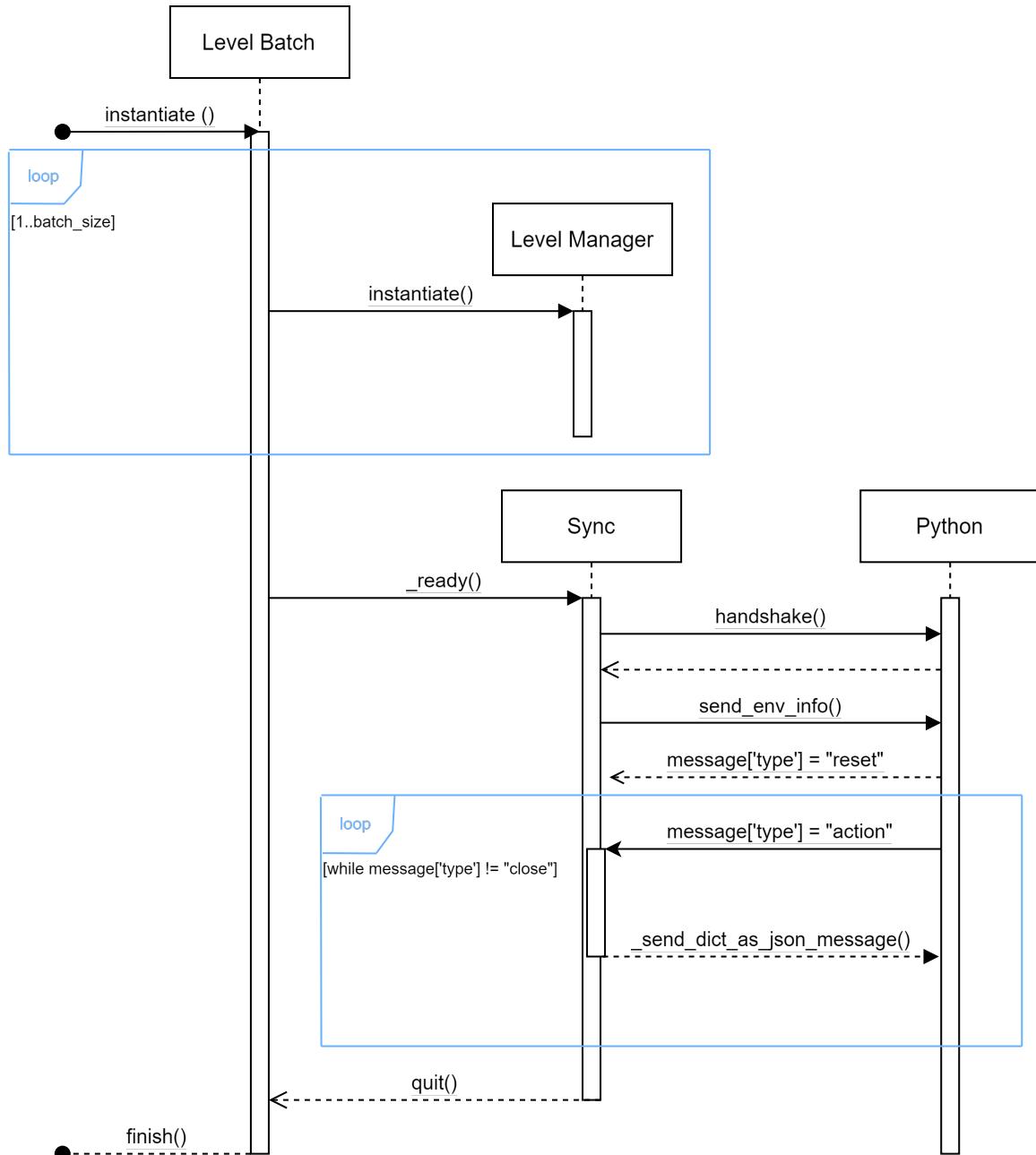


Figura 3.2: Diagramma di Sequenza: Level Batch

3.3.3 Level Manager

Il **Level Manager** è il nodo responsabile della gestione di tutti gli elementi all'interno di un singolo livello del curriculum. La prima operazione che esegue è l'istanziazione dell'effettivo livello, creando l'**environment** su cui eseguire il training. Una volta istanziato il livello, il **Level Manager** imposta a **ready** il nodo **PedestrianController**, che controlla tutti i pedoni presenti nella scena. I pedoni vengono istanziati e iniziano ad eseguire le proprie azioni.

Quando i pedoni terminano la loro esecuzione, ovvero quando tutti raggiungono il target finale o scade il numero massimo di timesteps predisposti per l'**environment**, vengono resettati dal **Pedestrian Controller**. Quest'ultimo notifica inoltre se un episodio è terminato. La terminazione dell'intero livello è gestita da **Python**, che verifica se il **Mean Reward**, definito nel file di configurazione del curriculum, è stato raggiunto, oppure se il training non è stato sufficientemente efficace, attivando la **Callback Early Fail**.

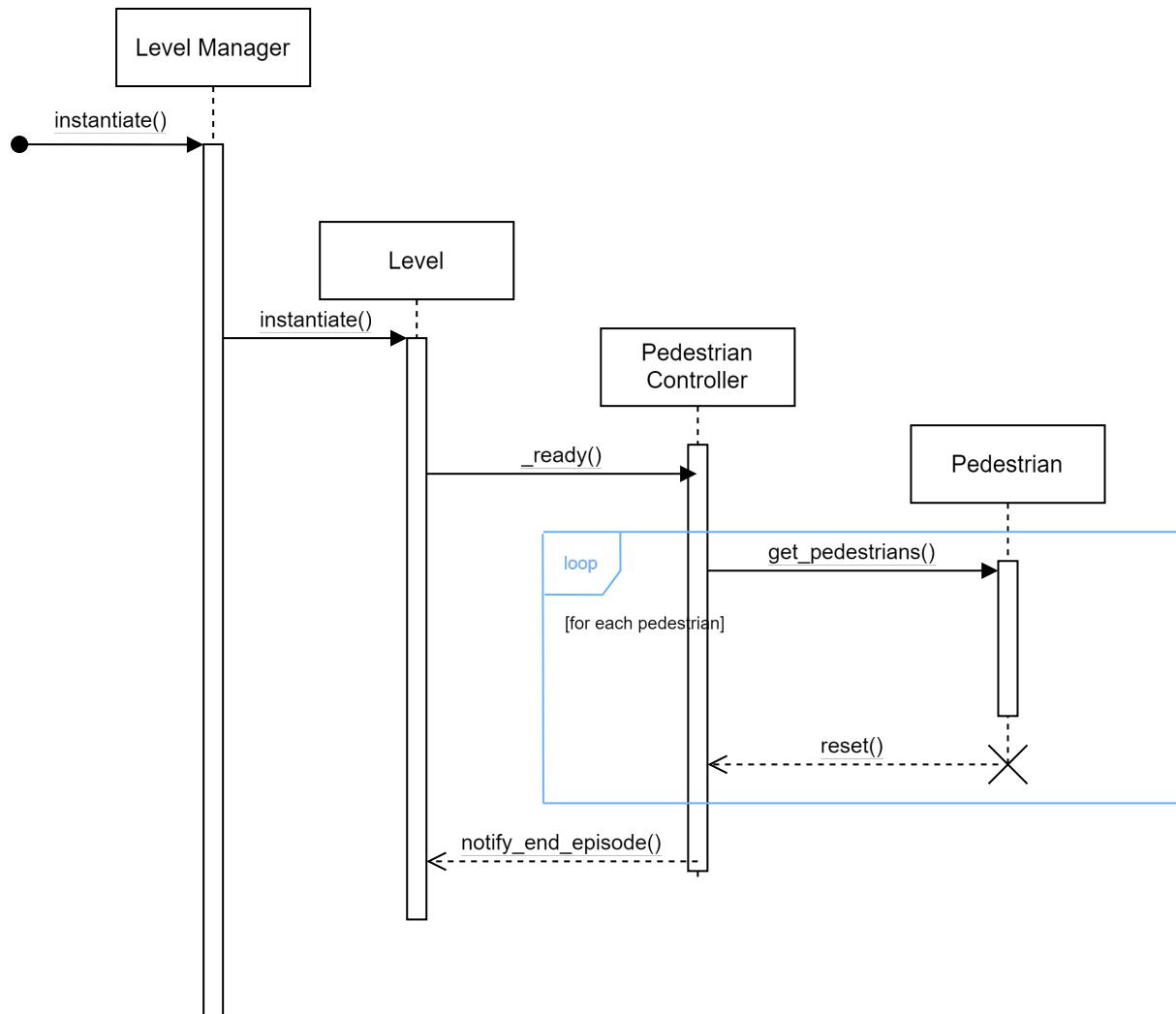


Figura 3.3: Diagramma di Sequenza: Level Manager

Capitolo 4

Analisi di Sensitività e di Iperparametri

4.1 Introduzione

In questo capitolo viene presentata un'analisi di sensitività e degli iperparametri del modello illustrato nei capitoli precedenti. L'obbiettivo principale sarà quello di ottimizzare i tempi di addestramento, mantenendo (o migliorando) le prestazioni ottenute dal modello di base. Successivamente, è stato eseguito una analisi ablativa degli ambienti del curriculum base. Per concludere, sono stati svolti dei test su ambienti ad elevata densità realizzati in uno studio svolto in parallelo a questo.

L'analisi di sensitività è uno studio volto a visualizzare come il cambiamento di qualche variabile (iperparametro) in input al modello, influenzino il comportamento finale dello stesso. In poche parole, si va ad osservare le differenze nei risultati finali ottenute applicando variazioni asincrone di singole variabili. Lo studio degli iperparametri è strettamente correlato all'analisi di sensitività, ma il suo scopo principale è quello di ottimizzare gli iperparametri del modello, piuttosto che visualizzare come il modello stesso cambi al variare di questi ultimi.

L'analisi ablativa (o studio ablativo) è un processo che consiste nel rimuovere selettivamente i componenti di un modello di machine learning, per identificare il loro ruolo e la loro importanza nel determinare le prestazioni del modello. Inoltre ci permette anche di ottenere una migliore comprensione generale sul comportamento del modello stesso.

I modelli addestrati sono stati testati su gli tutti ambienti di test base e sui nuovi ambienti di test (spiegati nella Sezione 4.1.1). Per comodità sono stati riportati solo i grafici degli esperimenti più rilevanti.

4.1.1 Modifiche alla fase di Test

Alla fase di test sono stati aggiunti due nuovi ambienti per visualizzare in modo migliore i cambiamenti e le prestazioni dei modelli addestrati. Gli ambienti di base utilizzati per testare il modello originale sono stati mantenuti.

Nuovi ambienti

Il primo ambiente creato è stato denominato “Curva Cieca”. Come si può notare dalla Figura 4.1(a), due pedoni partono in basso a sinistra e devono raggiungere il target situato nella parte destra. Durante il loro percorso, incontreranno altri quattro pedoni che camminano dall’alto verso il basso. Questo ambiente è stato ideato per verificare come dei gruppi di pedoni possano interagire tra loro, avvertendo però la presenza l’uno dell’altro solo quando sono già molto vicini tra loro.

Il secondo ambiente creato è stato denominato “Incrocio V”. Facendo riferimento alla Figura 4.1(b), ognuno dei due gruppi di pedoni deve muoversi verso l’angolo opposto rispetto a quello di partenza, andando ad incrociarsi durante il loro cammino. Questo ambiente sarà necessario soprattutto nella fase di analisi ablativa del modello, come aiuto per stabilire quali ambienti sono necessari nella fase di addestramento.

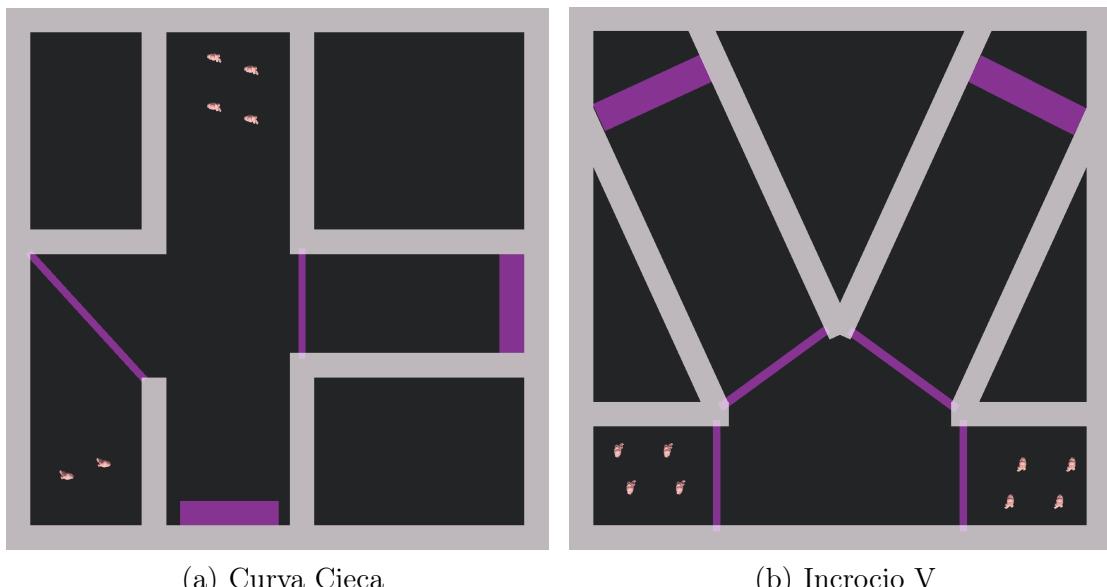


Figura 4.1: I nuovi ambienti “Curva Cieca” e “Incrocio V”

4.2 Analisi di Sensitività e degli Iperparametri

4.2.1 Studio sui principali iperparametri

Gli iperparametri studiati sono Learning Rate (LR) e Batch Size, il quale va di pari passo con n_steps. In Stable Baselines 3, il parametro n_steps indica per quanti step raccogliere dati prima di aggiornare il modello. Per un

addestramento ottimale, Batch Size deve essere uguale a n_steps * numero di ambienti in parallelo.

Learning Rate

Come primo esperimento, si è provato a incrementare notevolmente il Learning Rate da 0.0003 a 0.003, cosicché il modello possa apprendere maggiormente e più rapidamente dall'addestramento.

Come è possibile notare dalla Figura 4.2, l'addestramento ha impiegato un tempo notevolmente maggiore rispetto a quello del modello di base. Questo potrebbe essere attributo all'elevato Learning Rate, che porta il modello ad andare in overfit su alcuni ambienti. Nel momento in cui si passa da un ambiente al successivo, il modello si ostina quindi a provare ad eseguire le azioni imparate nell'ambiente appena passato, rallentando di molto il processo.

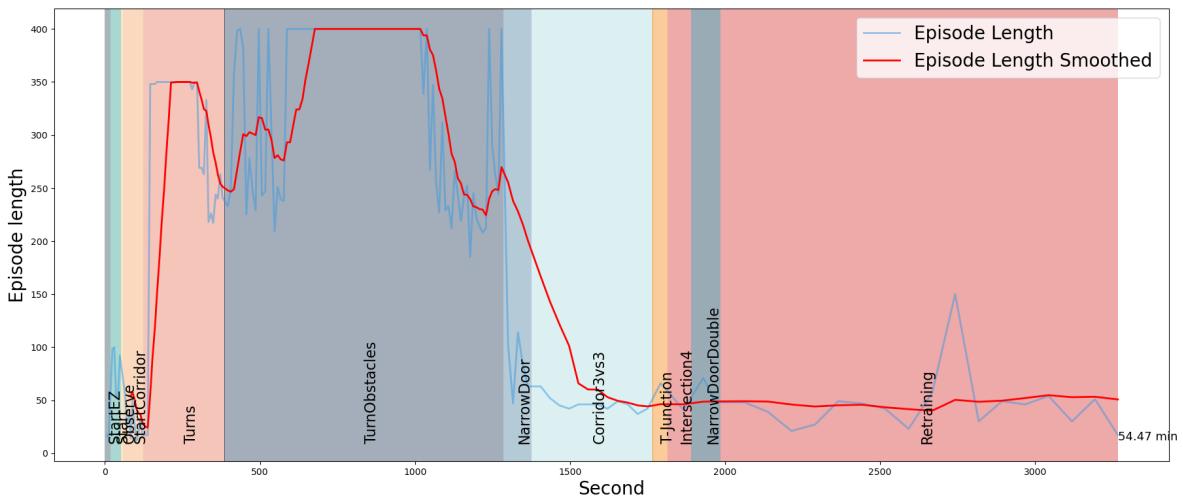


Figura 4.2: Lunghezza Episodi

Nell'ambiente “Doppia Porta Opposta” (Figura 4.3(a)) si può notare come il modello mantenga molto distanti i due flussi di pedoni, ma, allo stesso tempo, ci sono troppi pedoni con un comportamento anomalo. I pedoni più distanti dalla prima porta, nel momento in cui si trovano il primo sbocco già occupato da altri pedoni, preferiscono dirigersi lungo tutta la parete della zona iniziale per poi tornare alla prima porta e proseguire il percorso normalmente.

Nell'ambiente “Ancora” (Figura 4.3(b)) si può notare come il modello mantenga quasi perfettamente ad ogni esecuzione le stesse traiettorie per ogni pedone. Il gruppo di pedoni rosso esegue una curva stretta per arrivare più in fretta all'obiettivo, mentre il gruppo blu esegue sempre una curva più ampia per non incrociarsi con l'altro flusso. Questo grafico evidenzia

chiaramente come il modello abbia imparato a mantenere il lato sinistro del pedone il più vicino possibile al muro, dimostrando una tendenza costante a rimanere accanto al muro sul lato sinistro.

Nell'ambiente “Curva Cieca” (Figura 4.3(c)) si può notare come il modello nuovamente tende a mantenere la sinistra, ma senza un'unica traiettoria come nell'ambiente precedente. In prossimità dell'incrocio i pedoni si distanziano tra di loro e variano di poco la traiettoria per potersi evitare e proseguire verso il loro obiettivo. Alcuni pedoni del gruppo rosso vengono però distratti dai pedoni del gruppo blu e tendono a finire verso l'obiettivo finale errato.

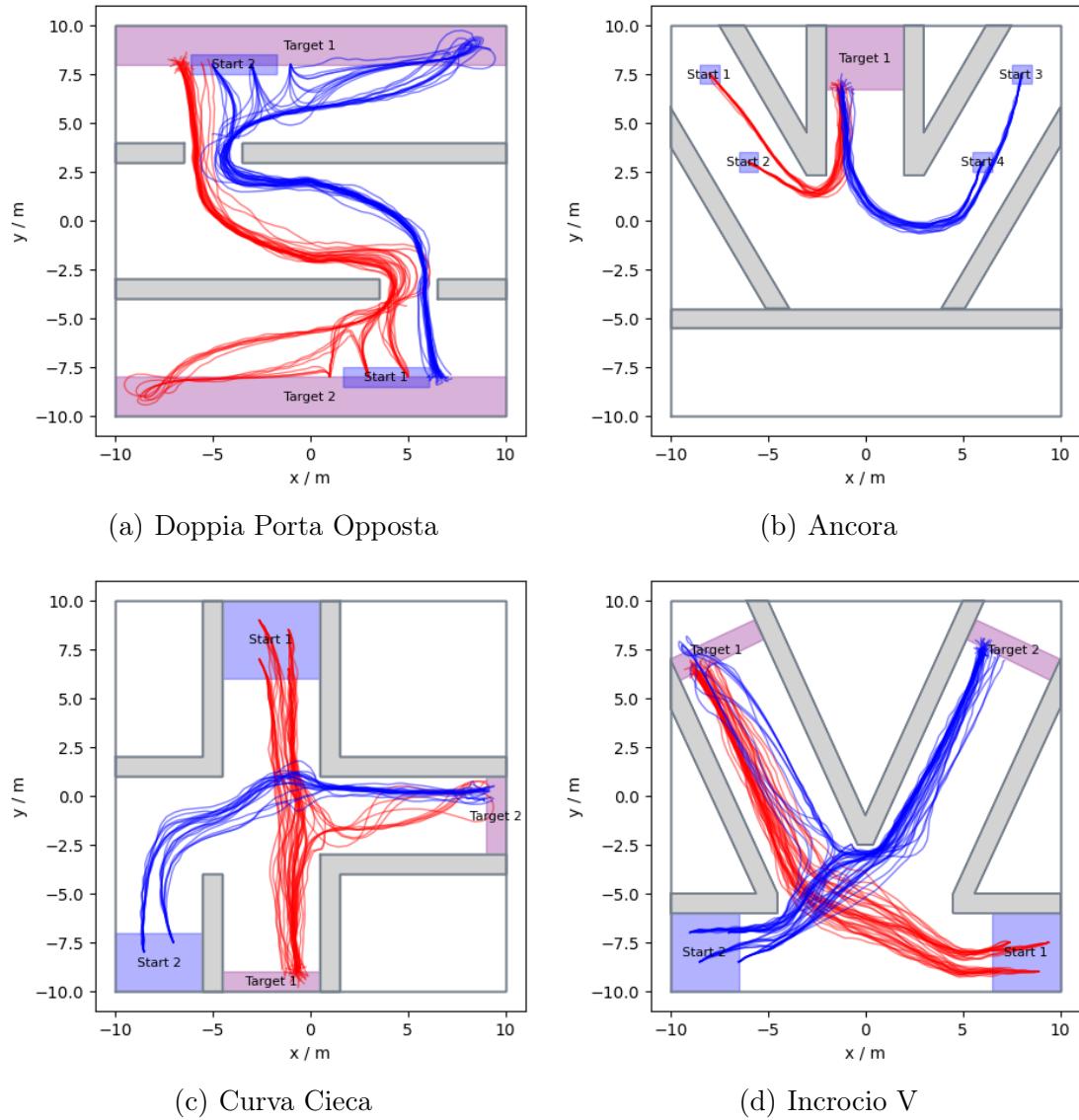


Figura 4.3: Risultati del modello con $LR = 0.003$

Nell'ambiente “Incrocio V” (Figura 4.3(d)) si può notare come le traiettorie del gruppo blu siano più vicine e simili tra loro, in quanto questo gruppo

di pedoni tende ad attraversare l’incrocio per primo. Ciò nonostante, quando alcuni pedoni del gruppo rosso nascono con una velocità abbastanza elevata, alcuni pedoni del gruppo blu vengono spinti verso sinistra e, di conseguenza, camminano verso l’obbiettivo finale errato. I pedoni del gruppo rosso cercano di mantenere traiettorie più ampie per evitare di scontrarsi con l’altro gruppo.

Negli altri ambienti, qui non riportati, il modello è riuscito ad eseguire il suo compito egregiamente senza la comparsa di casi anomali. In conclusione, l’innalzamento del Learning Rate da 0.0003 a 0.003 ha apportato un peggioramento nelle prestazioni del modello, causando un significativo overfitting e di conseguenza la comparsa di molti comportamenti anomali (outliers).

Batch Size

Come secondo esperimento, sono stati modificati il Batch Size e il parametro “n_steps”, mantenendo il Learning Rate al suo valore iniziale (0.0003). Inizialmente sono stati impostati rispettivamente a 1280 e 128, per poi passare a 640 e 64. In entrambi i casi l’addestramento è sempre finito in Early Fail, in alcune run non ha nemmeno superato l’ambiente Start. Questo potrebbe essere dovuto alla combinazione tra Learning Rate troppo basso e “n_steps” troppo elevato, che insieme portano il modello ad aggiornarsi solo dopo troppi timestep e ad apprendere una quantità di informazioni insufficienti.

Combinazione Learning Rate e Batch Size

Viste le considerazioni fatte nell’esperimento precedente, si è provato a modificare simultaneamente sia il Learning Rate che il Batch Size (e “n_steps”), per dare la possibilità al modello di aggiornarsi in modo meno frequente (riducendo il carico sull’hardware), ma allo stesso tempo di apprendere più informazioni ad ogni aggiornamento. I nuovi valori utilizzati sono un Learning Rate di 0.003, un Batch Size di 640 e un “n_steps” di 64.

Come è possibile notare dalla Figura 4.4, l’addestramento ha impiegato ancora un tempo di poco superiore a quello del modello di base, nonostante l’ottimizzazione dal lato Hardware

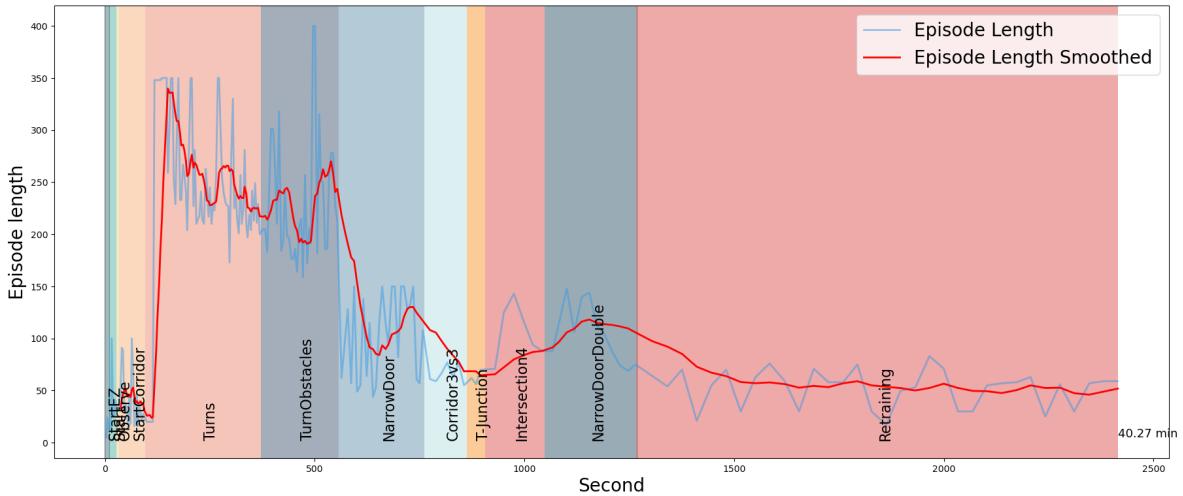


Figura 4.4: Lunghezza Episodi

Nell’ambiente “Doppia Porta Opposta” (Figura 4.5(a)) si può notare come il modello tenga abbastanza separati i due flussi di pedoni, con pochi casi in cui vanno a mischiarsi fra di loro. E’ però evidente che gli outliers sono parecchi, soprattutto se si prende in considerazione il percorso compiuto dal pedone più vicino il centro di ogni gruppo.

Nell’ambiente “Curva Cieca” (Figura 4.5(c)) si può notare come il comportamento dei pedoni di mantenere la destra metta in difficoltà il gruppo di pedoni rossi. Probabilmente a causa di un leggero overfitting in addestramento, i pedoni tendono a tenere un muro alla loro destra costantemente e ciò porta il gruppo rosso a spostarsi troppo verso la loro destra, finendo quasi nella via accanto a quella con l’obiettivo.

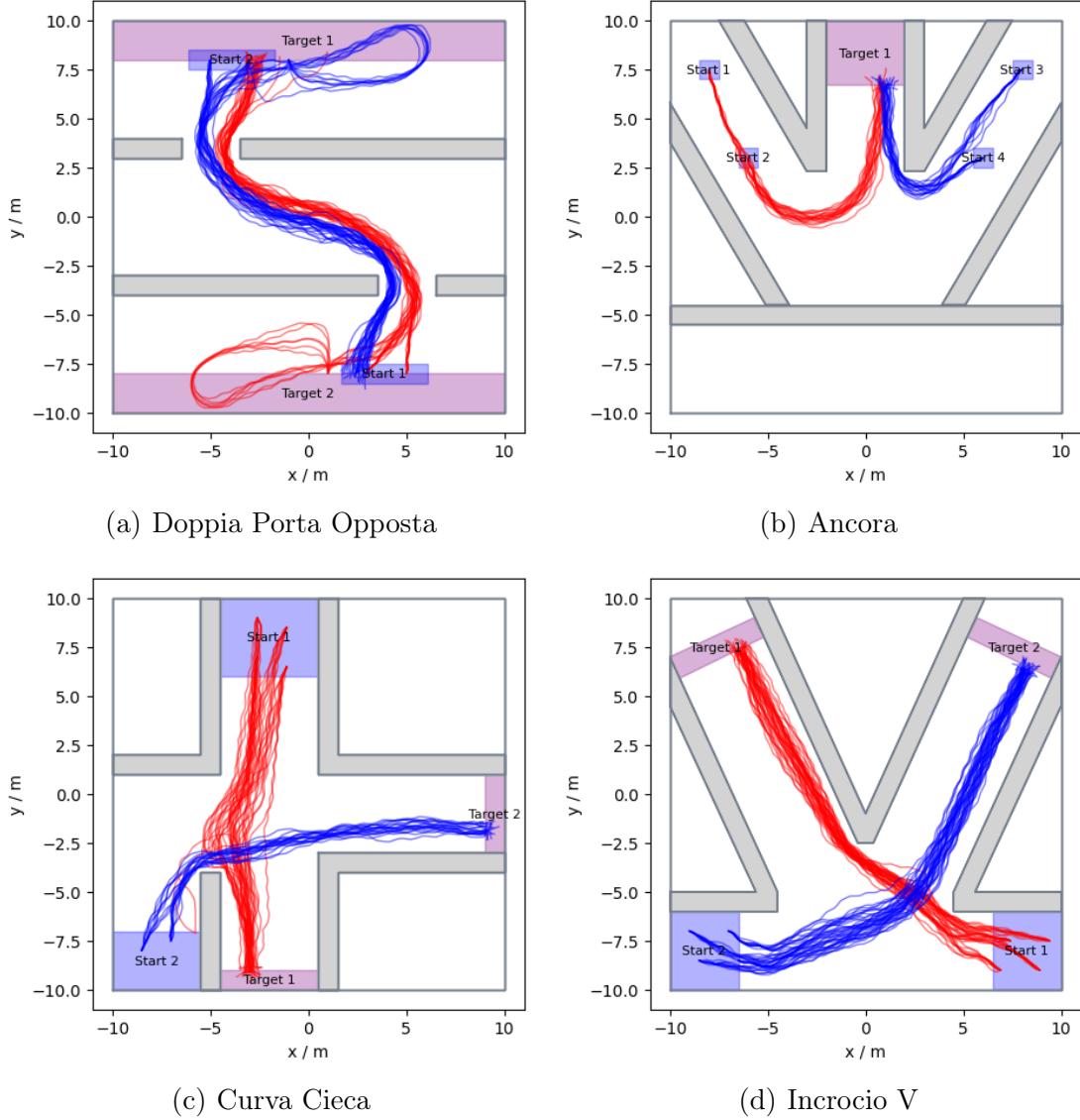


Figura 4.5: Risultati del modello con $LR = 0.003$, Batch Size = 640, n_steps = 64

4.2.2 Studio sull'architettura della rete neurale con strati nascosti di egual dimensione

Nella seconda parte dello studio di sensibilità e degli iperparametri, è stata presa in considerazione l'architettura della rete neurale utilizzata dall'algoritmo PPO. In particolare, sono stati modificati solo il numero di strati e la loro dimensione, mentre gli altri elementi non sono stati presi in considerazione. In Stable Baselines 3, le dimensioni dell'input layer e dell'output layer non sono modificabili, poiché il primo è definito dal numero di osservazioni che l'agente può eseguire, mentre il secondo è determinato dal numero di azioni che l'agente può compiere. Gli unici strati che possono essere modificati

sono quelli nascosti. Di conseguenza, nei titoli delle sezioni successive, sono riportati solo gli strati nascosti (ad esempio, con “Rete 128-128” si intende una rete neurale con due strati nascosti di ampiezza 128).

Rete 128-128

Il primo test eseguito sulla rete neurale è stato di ridurre l’ampiezza degli strati per diminuire il tempo di aggiornamento dei pesi della rete e tentare di velocizzare l’addestramento del modello, a discapito della capacità di apprendere della rete. Ogni strato nascosto è stato portato da 256 a 128.

Come è possibile notare dalla Figura 4.6, l’addestramento ha impiegato ancora un tempo di poco superiore a quello del modello di base. In particolare, il modello ha impiegato un tempo molto elevato nell’ambiente “Curve Ostacoli” (TurnsObstacles).

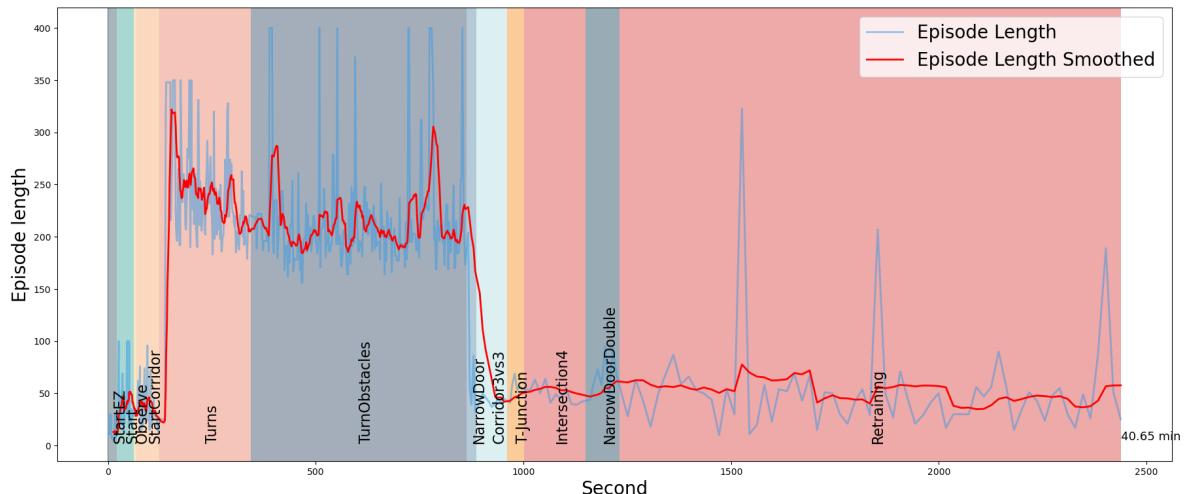


Figura 4.6: Lunghezza Episodi

Andando a vedere il grafico dell’andamento dei reward (Figura 4.7), è evidente che durante l’addestramento ci siano stati parecchi picchi negativi soprattutto nell’ambiente “Curve Ostacoli”.

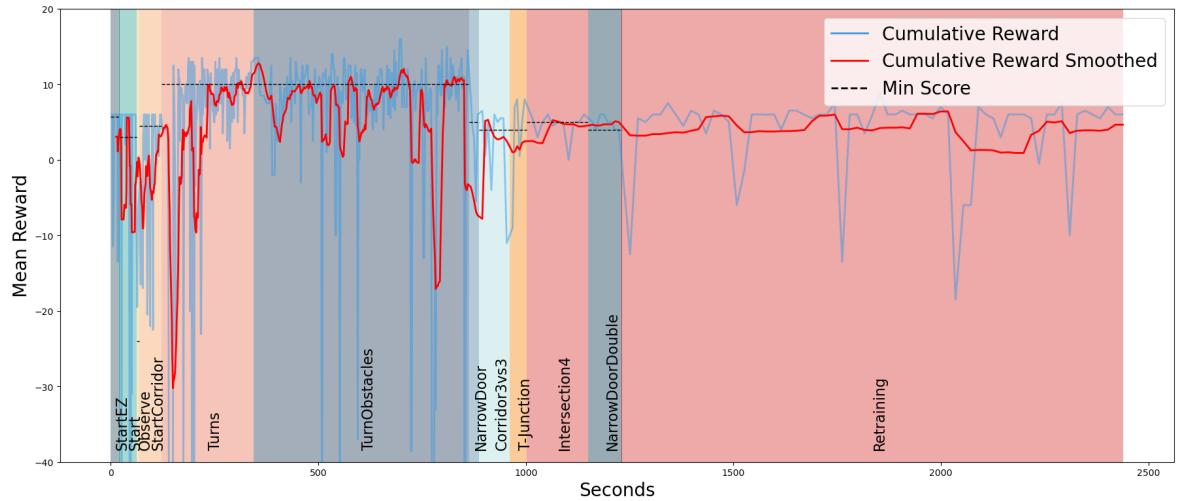
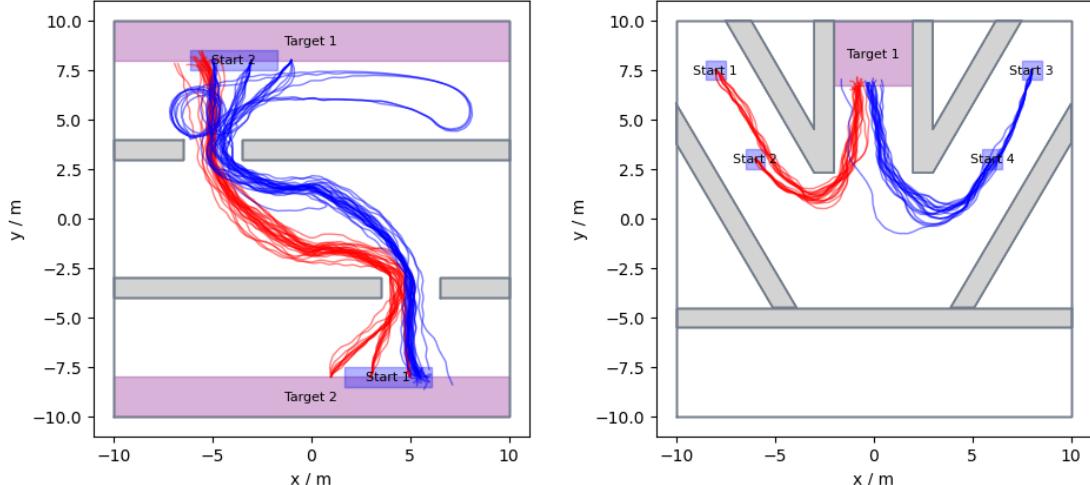


Figura 4.7: Reward Episodi

Nonostante questo piccolo rallentamento avvenuto in “Curve Ostacoli”, il modello ha portato a termine l’addestramento e durante la fase di test ha superato in modo eccellente tutti gli ambienti.

Le uniche anomalie comparse sono in “Doppia Porta Opposta”, dove, come mostra la Figura 4.8(a), qualche pedone ha preferito allontanarsi dalla prima porta, esattamente come avveniva negli esperimenti precedenti.



(a) Doppia Porta Opposta

(b) Ancora

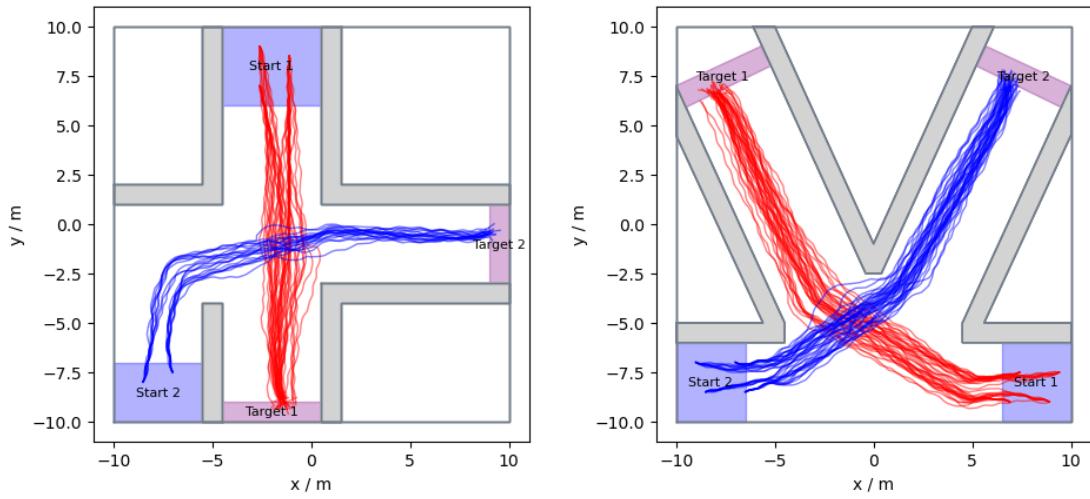


Figura 4.8: Risultati del modello con rete neurale da 128-128

In conclusione, con questa rete neurale il modello riesce ad apprendere i comportamenti necessari a completare la fase di test. La riduzione dell'ampiezza degli starti della rete neurale non è però servita a diminuire i tempi di addestramento, in quanto, a causa del ridotto numero di nodi per layer, il modello fatica a convergere in alcuni ambienti.

Rete 512-512

Il secondo test eseguito sulla rete neurale, è stato di aumentare l'ampiezza di ogni strato, in modo da ridurre la velocità di aggiornamento del modello, permettendo però una maggiore capacità di inferenza. Ogni strato nascosto è stato portato da 256 a 512.

Osservando il grafico tempi (Figura 4.9), l’addestramento ha impiegato un tempo ancora troppo elevato rispetto a quello del modello di base. Come nell’esperimento precedente, l’ambiente “Curve Ostacoli” (TurnsObstacles) ha creato ancora qualche problema nell’apprendimento e presenta molti picchi negativi (Figura 4.10).

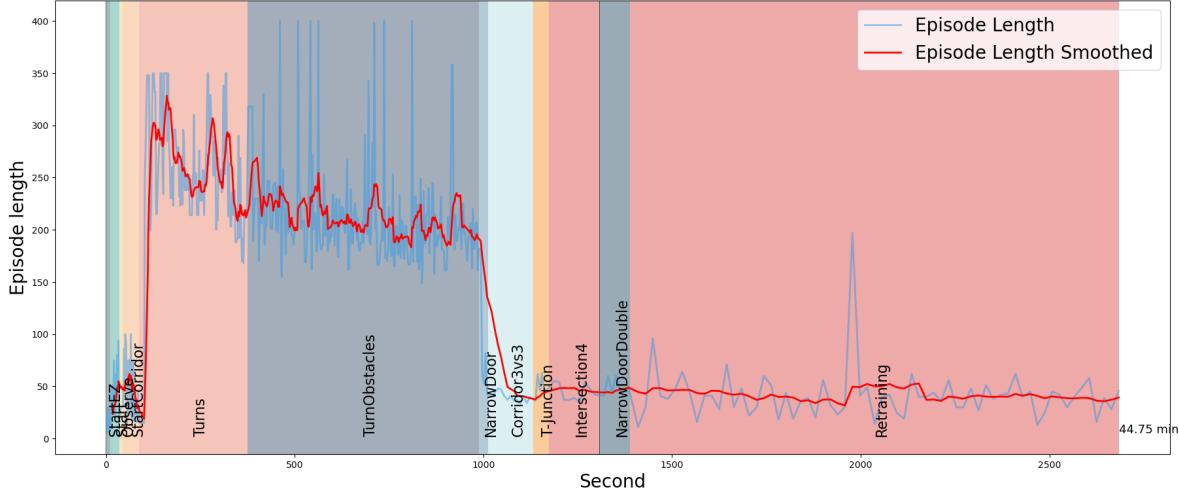


Figura 4.9: Lunghezza Episodi

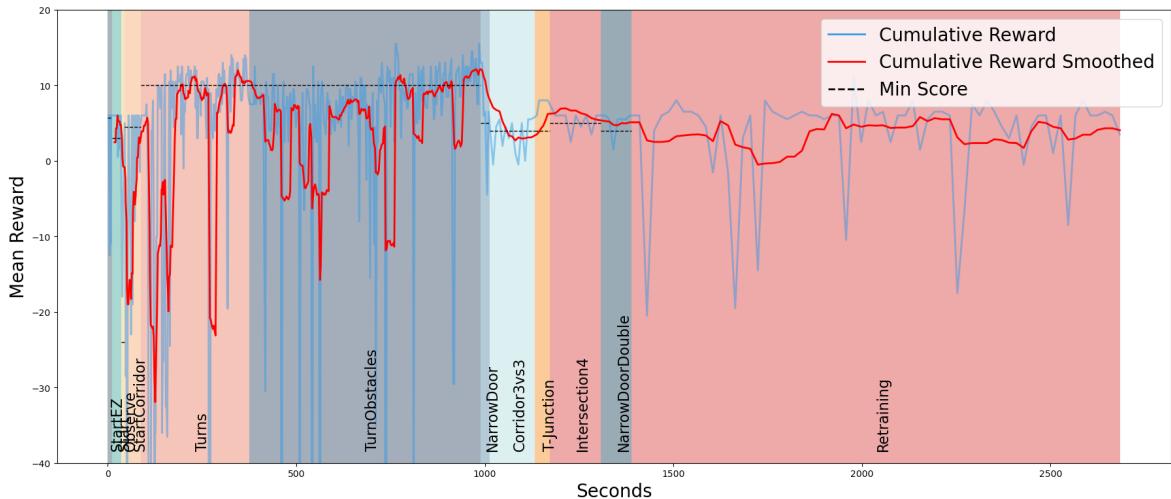


Figura 4.10: Reward Episodi

Nell’ambiente “Doppia Porta Opposta” (Figura 4.11(a)) si può notare come come alcuni pedoni si allontanino dalla prima porta. In questo test si verifica un comportamento anomalo mai verificatosi prima: alcuni pedoni, a metà percorso, cambiano la loro traiettoria e preferiscono tornare indietro. Nonostante queste leggere anomalie, il modello porta a termine questo ambiente in maniera ottimale.

L’ambiente “Curva Cieca” (Figura 4.11(c)) viene eseguito correttamente. Le uniche piccole anomalie si possono notare al centro dell’incrocio, dove

alcuni pedoni del gruppo rosso, per non urtare quelli del gruppo blu, eseguono una deviazione circolare prima di proseguire verso il loro obiettivo.

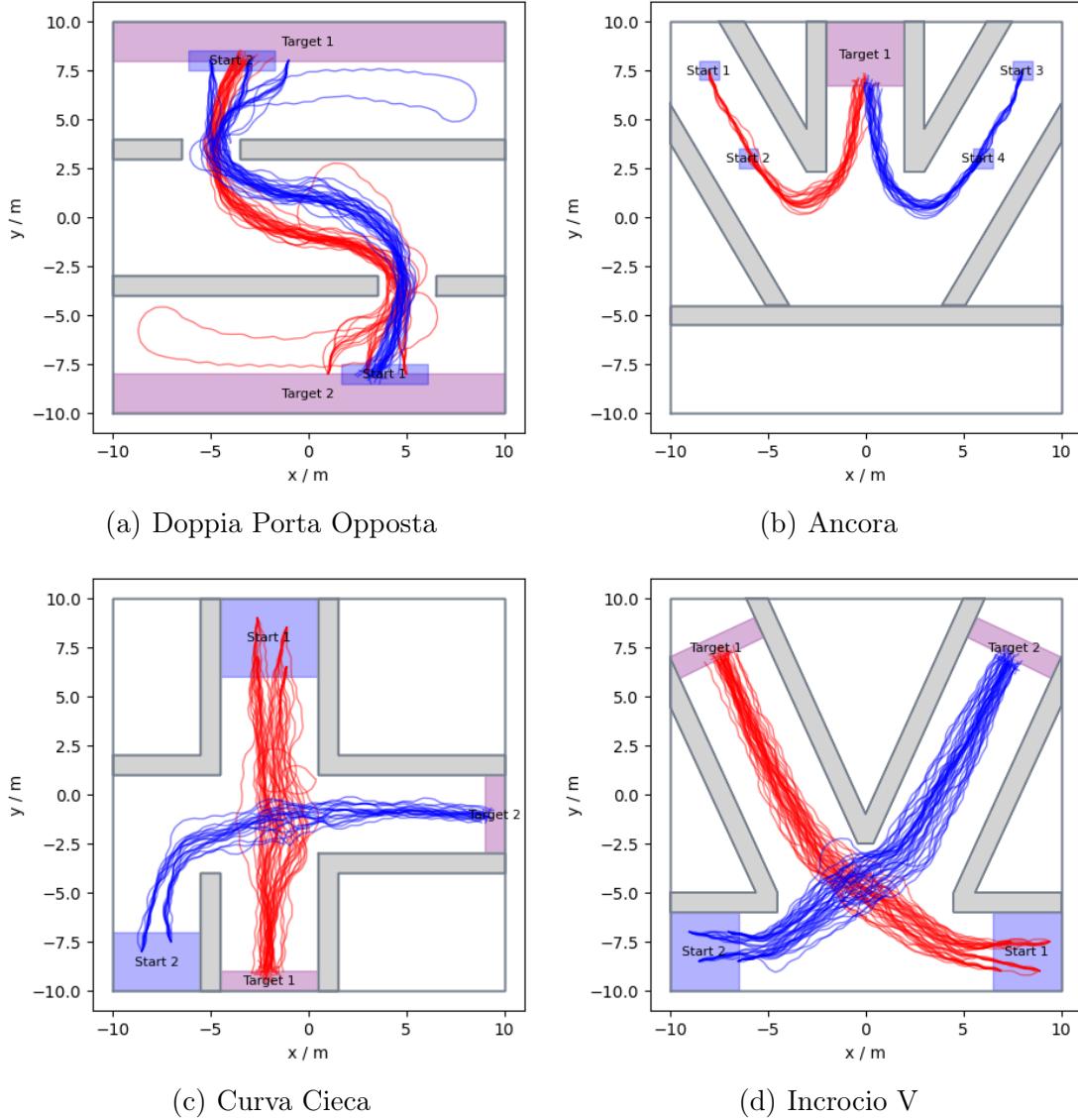


Figura 4.11: Risultati del modello con rete neurale da 512-512

Tutti gli altri ambienti, non riportati, sono stati eseguiti alla perfezione, senza alcuna anomalia. In conclusione il modello ottenuto esegue in modo eccellente tutti gli ambienti di test, tuttavia il tempo di addestramento è ancora troppo elevato.

Rete 128-128-128

In questo test si è provato a diminuire la dimensione degli strati e, contemporaneamente, ad aggiungerne uno in più. La rete risultante è quindi formata da tre strati da 128 nodi ciascuno. In questo modo si ottiene un compro-

messo tra capacità di inferenza della rete e tempo utilizzato per aggiornare i suoi pesi.

Osservando il grafico dei tempi (Figura 4.12), l’addestramento ha impiegato un tempo pressoché identico a quello del modello di base. A differenza degli esperimenti precedenti, l’ambiente “Curve Ostacoli” (TurnsObstacles) non ha dato particolari problemi nell’apprendimento e la sua esecuzione è stata molto più rapida (Figura 4.13).

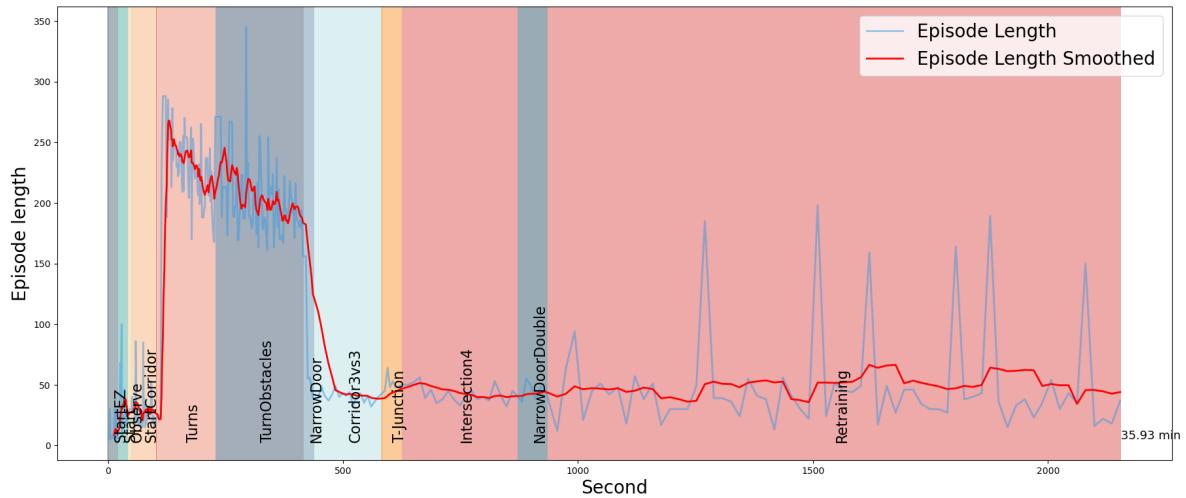


Figura 4.12: Lunghezza Episodi

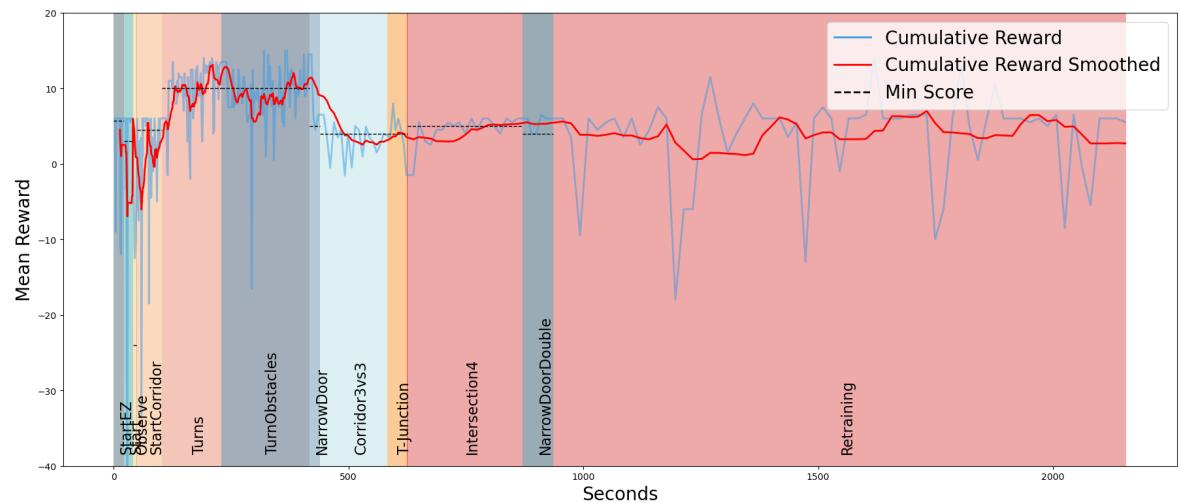


Figura 4.13: Reward Episodi

Nonostante i miglioramenti nella fase di addestramento, durante il testing compaiono ancora abbastanza casi anomali, in particolare negli ambienti “Doppia Porta Opposta” e “Incrocio V”. Questo potrebbe essere dovuto a un leggero overfitting causato dalla presenza di uno strato aggiuntivo nella rete.

Nell’ambiente “Doppia Porta Opposta” (Figura 4.14(a)) si può notare come qualche pedone si allontani dalla prima porta. Nonostante queste

piccole anomalie, il modello porta a termine questo ambiente in maniera soddisfacente, mantenendo una netta separazione tra i due flussi dei pedoni.

Nell'ambiente "Incrocio V" (Figura 4.14(d)) è comparso un nuovo tipo di anomalia. Qualche pedone del gruppo blu ha preferito tornare indietro, piuttosto che proseguire verso il suo obiettivo, per poi riprendere successivamente la strada corretta.

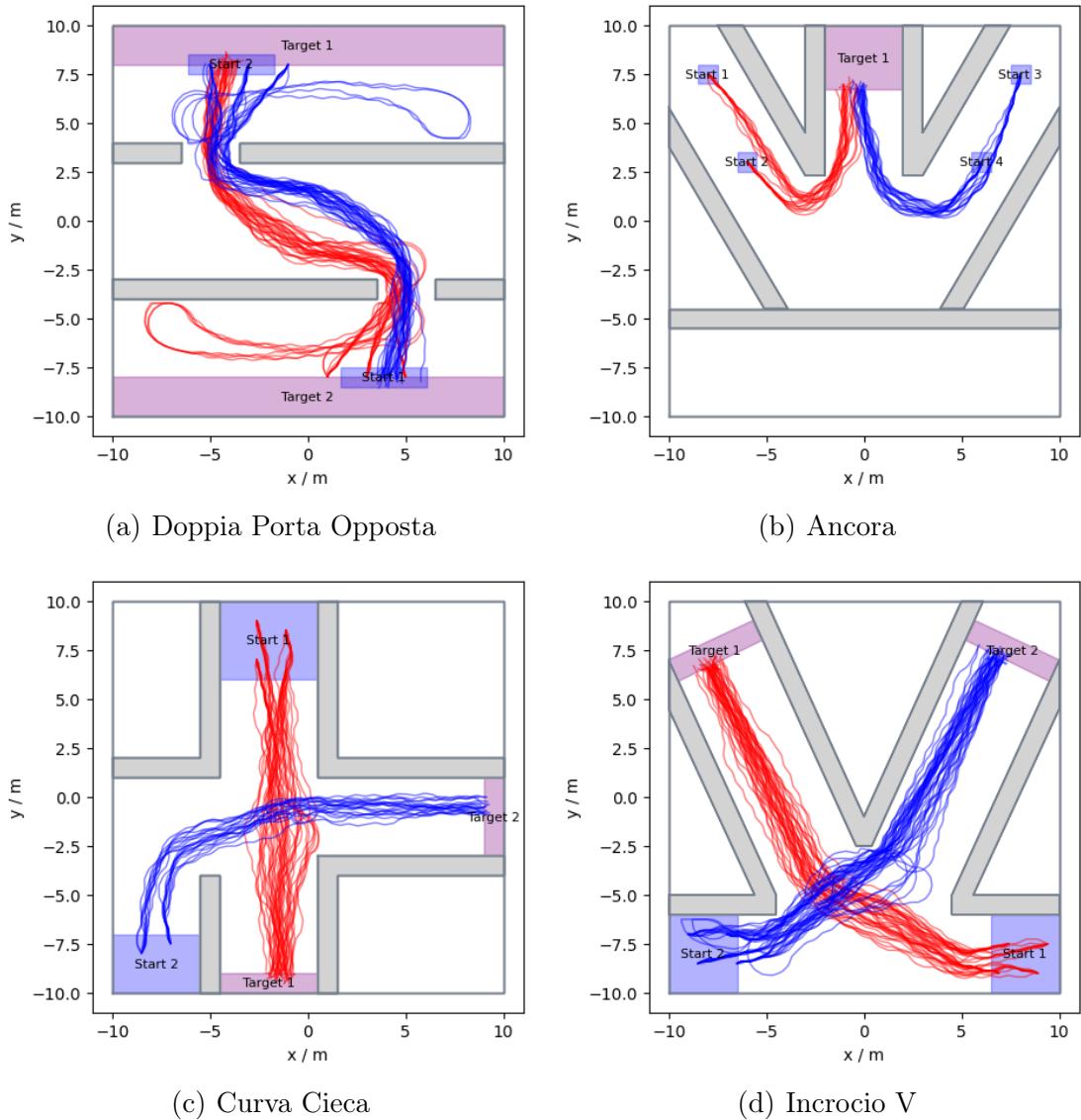


Figura 4.14: Risultati del modello con rete neurale da 128-128-128

Nonostante le anomalie appena mostrate, il modello si comporta egregiamente nel resto degli ambienti di test. In conclusione, questo modello è quello che più si avvicina a quello originale considerando i tempi impiegati in fase di addestramento.

4.2.3 Studio sul tempo di re-training

In questo studio il modello è stato addestrato utilizzando differenti periodi di retraining. Per quanto riguarda gli iperparametri, sono stati mantenuti quelli originali, a parte l'architettura della rete neurale. E' stata utilizzata quella con tre strati intermedi da 128 nodi ciascuno. Questa decisione è stata presa poiché questa architettura dovrebbe offrire una capacità di apprendere maggiore rispetto a quella originale. Inoltre, visti i tempi di retraining molto ridotti, potrebbe aiutare il modello a mantenere più informazioni sugli ambienti vecchi, che altrimenti andrebbero perse con la rete originale.

Retraining da 150.000 timestep

Il tempo di retraining è stato ridotto di 50.000 timestep, portandolo a 150.000 timestep totali. Come mostrato dalla Figura 4.15, il tempo di addestramento totale è diminuito di circa 7 minuti, ma il tempo dei singoli ambienti è rimasto pressoché lo stesso dell'esperimento precedente.

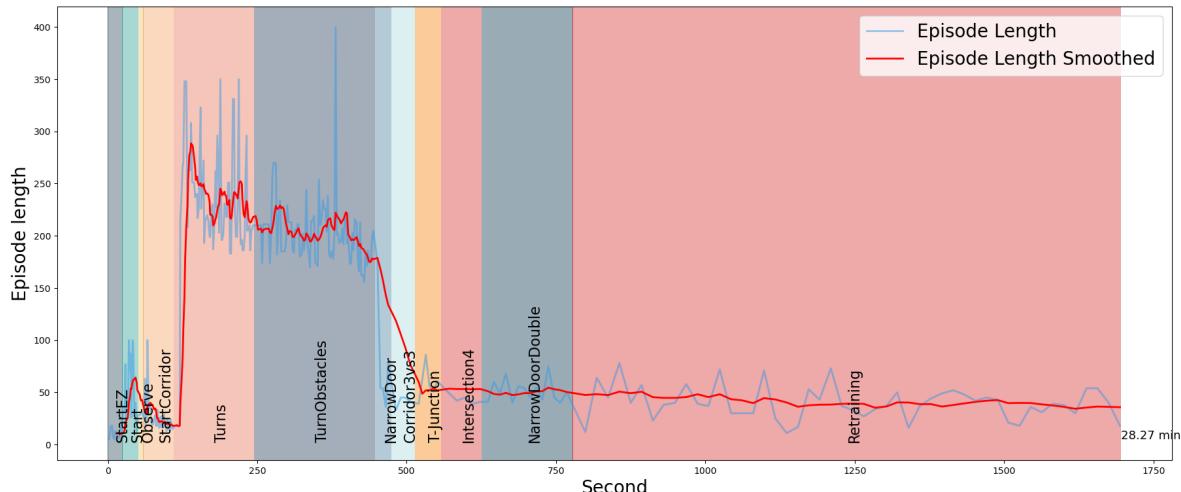


Figura 4.15: Lunghezza Episodi

Nell'ambiente “Doppia Porta Opposta” (Figura 4.16(a)) i casi anomali alle estremità dell'ambiente sono diminuiti, a discapito della comparsa di qualche caso al centro dell'ambiente.

Nell'ambiente “Incrocio V” (Figura 4.16(d)), non sono più presenti anomalie. In un caso, un pedone del gruppo rosso, spinto da quelli del gruppo blu, stava per imboccare il corridoio errato, ma è riuscito immediatamente a ritornare sulla via corretta.

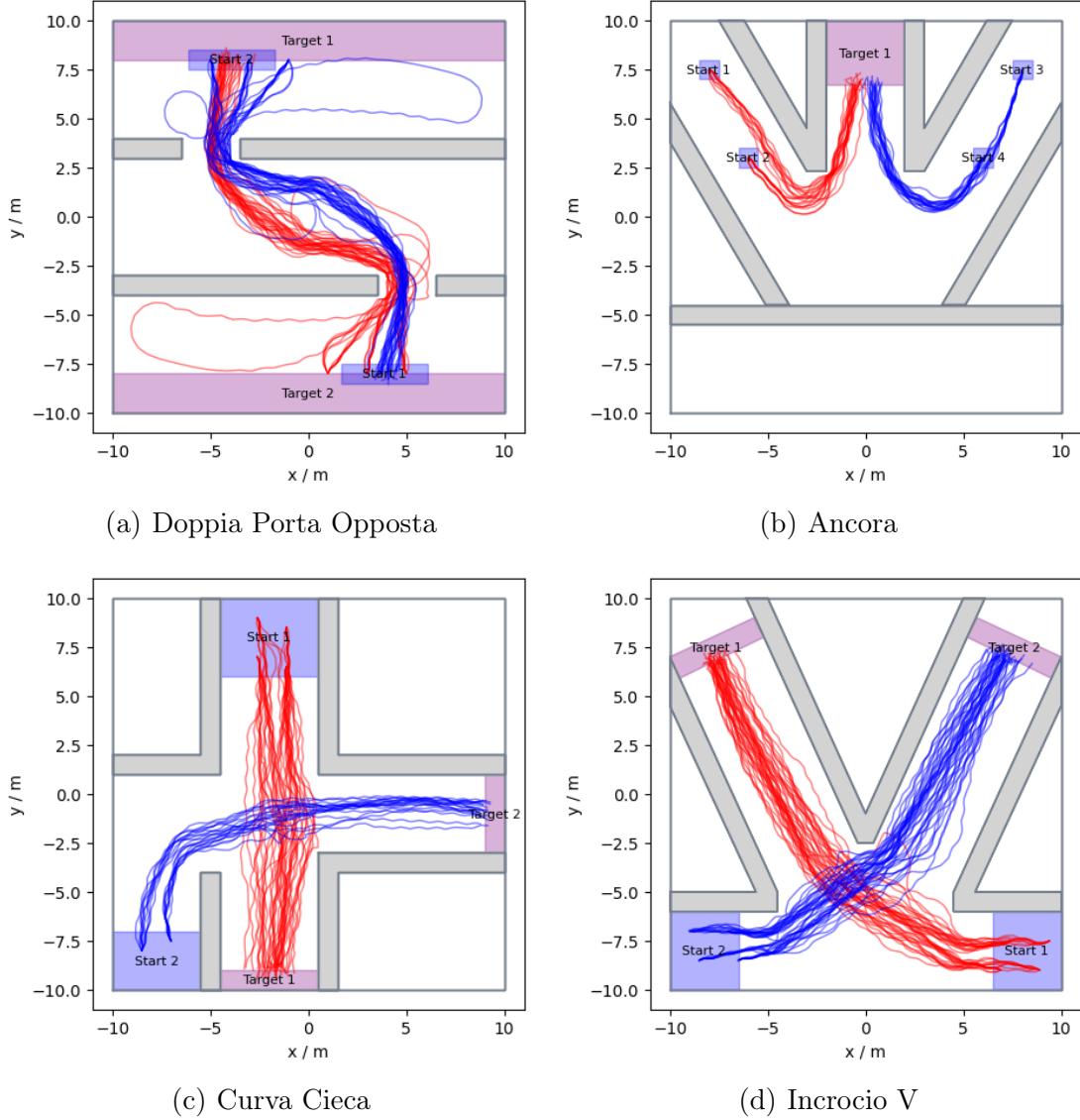


Figura 4.16: Risultati del modello con retraining da 150.000 steps

In conclusione, riducendo i tempi di retraining, il modello ottenuto ha mostrato dei comportamenti leggermente migliori rispetto a quello addestrato con un retraining di 200.000 timesteps. Questo ci porta a pensare che il processo di retraining, se svolto per troppo tempo, può confondere il modello piuttosto che aiutarlo a consolidare i suoi comportamenti.

Retraining da 100.000 timestep

Visti i risultati dell'esperimento precedente, si è deciso di ridurre di ulteriori 50.000 timestep il tempo di retraining, portandolo a 100.000 timestep totali. Come mostrato dalla Figura 4.17, il tempo di addestramento totale è diminuito solamente di circa 3 minuti, a causa dell'ambiente “Curve Ostacoli”.

coli". Quest'ultimo, probabilmente a causa di parecchie run sfortunate, ha impiegato un tempo eccessivamente elevato nella fase di addestramento.

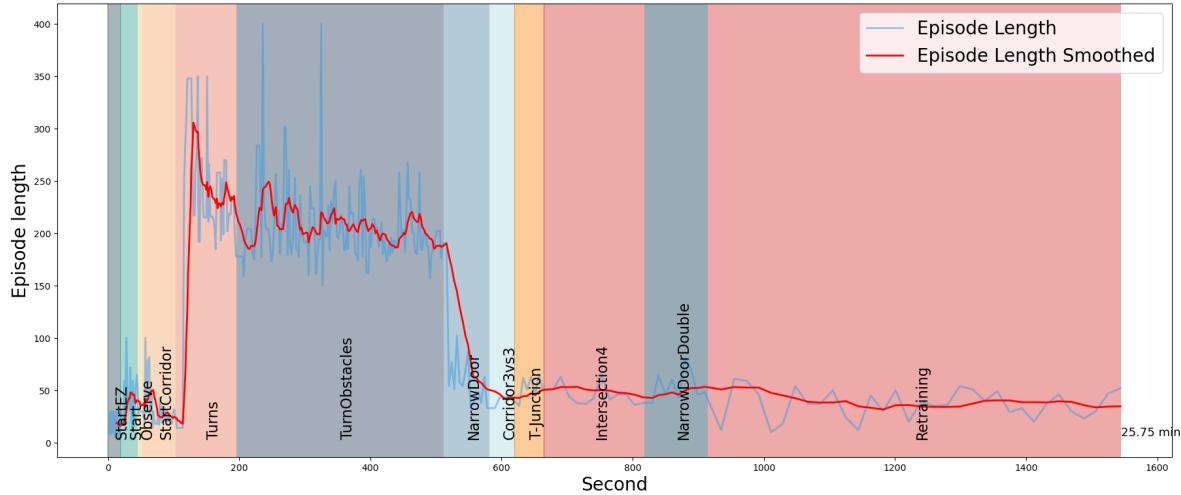
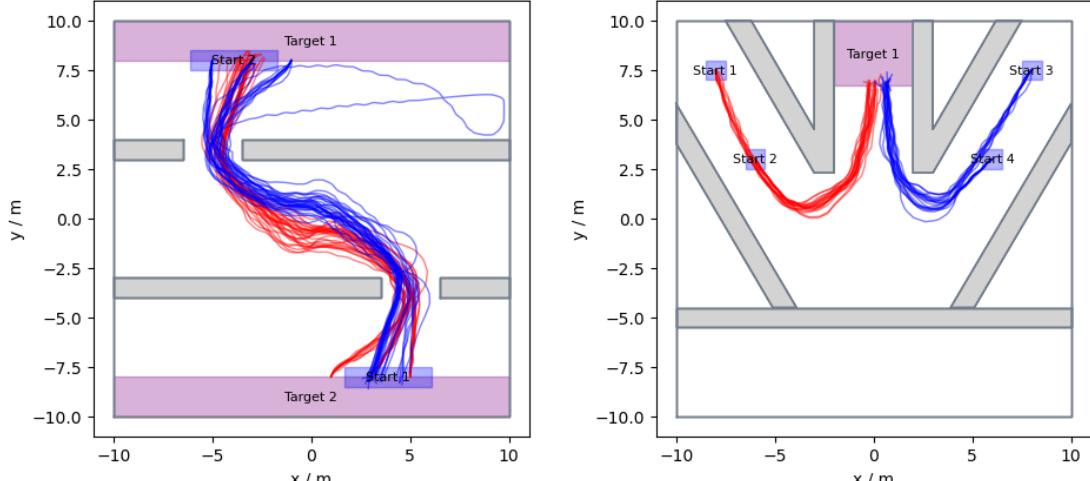


Figura 4.17: Lunghezza Episodi

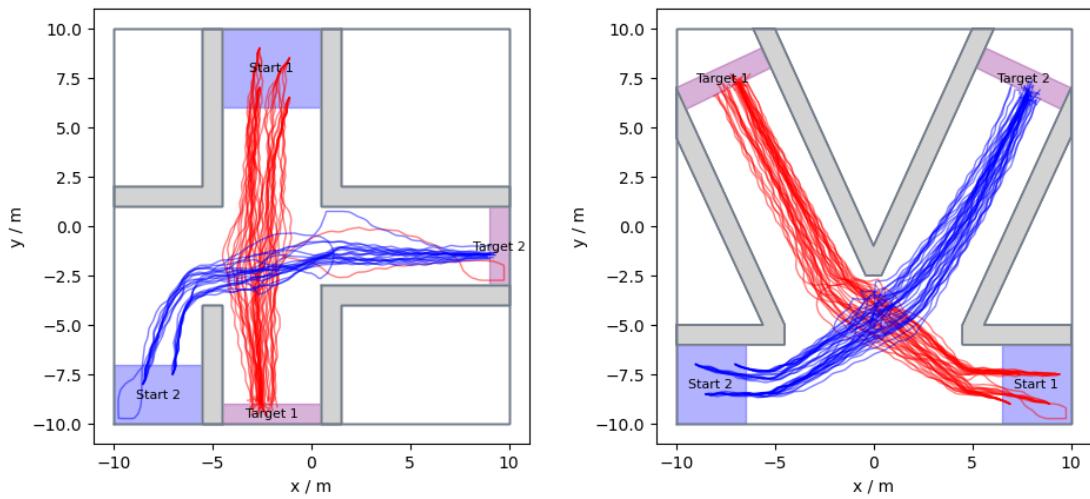
Nell'ambiente “Doppia Porta Opposta” (Figura 4.18(a)), in cui i casi anomali sono quasi assenti, i due flussi di pedoni non sono sufficientemente separati tra loro.

Nell'ambiente “Curva Cieca” (Figura 4.18(c)), si nota come in pochi casi i pedoni facciano confusione in concomitanza dell'incrocio.



(a) Doppia Porta Opposta

(b) Ancora



(c) Curva Cieca

(d) Incrocio V

Figura 4.18: Risultati del modello con retraining da 100.000 steps

Questi risultati mostrano che un tempo ancora più basso di retraining sembrerebbe aver migliorato nuovamente le prestazioni. A questo punto ci si pone una domanda: il retraining è un processo fondamentale o potrebbe essere eliminato? Questo ci porta all'esperimento successivo.

No retraining

Visti gli ottimi risultati dell'esperimento precedente, si è deciso di rimuovere completamente la fase di retraining. Come mostrato dalla Figura 4.19, il tempo di addestramento totale è di circa 14 minuti. Tutti gli ambienti sono stati eseguiti, a grande linea, nello stesso tempo dell'esperimento precedente.

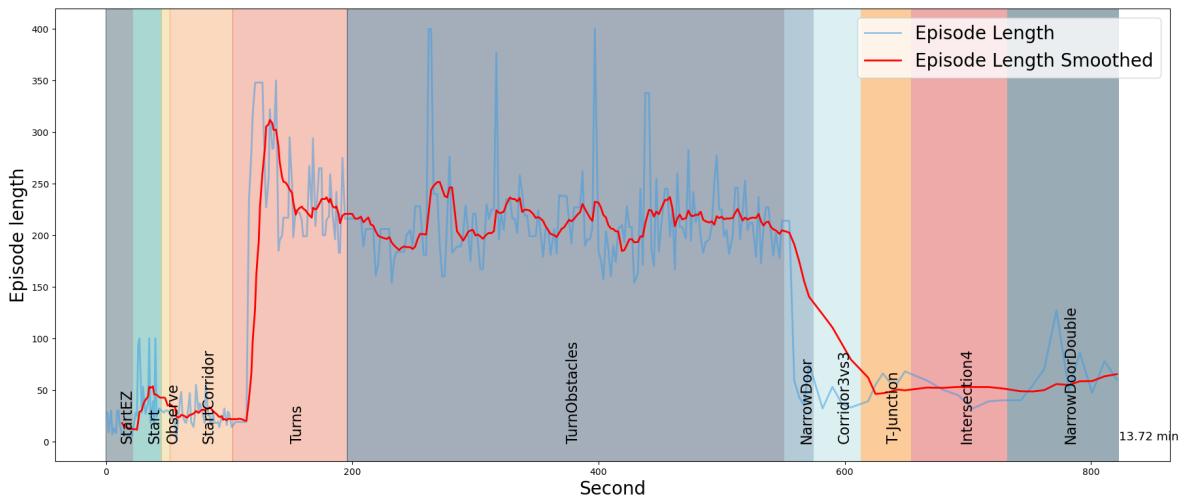


Figura 4.19: Lunghezza Episodi

La fase di test è stata svolta correttamente ed è comparsa una sola anomalia nell’ambiente “Doppia Porta Opposta” (Figura 4.20(a))

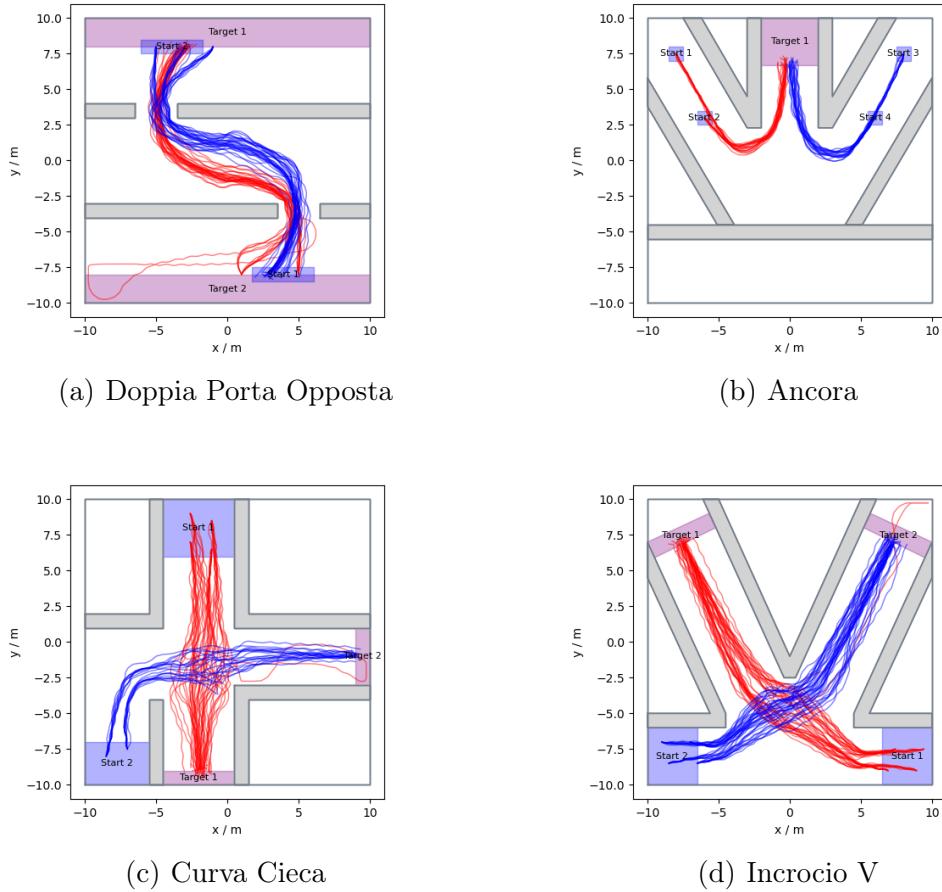


Figura 4.20: Risultati del modello senza retraining

Conclusioni

Come dimostrato in questi esperimenti, il retraining rappresenta un intralcio nella fase di addestramento. Non solo impiega molto tempo per essere eseguito, ma al suo termine il modello addestrato mostra dei chiari segnali di overfitting. In conclusione i modelli creati senza l'utilizzo del retraining sono più prestanti e possono essere addestrati molto più rapidamente.

4.2.4 Studio sull'architettura della rete neurale con strati nascosti di diversa dimensione

In questa fase dello studio è stata testata una rete con strati nascosti di diverse dimensioni. Sono stati utilizzati tre strati, il primo da 256, il secondo da 128, il terzo da 64. Per questo esperimento si è deciso di non utilizzare il retraining visti i risultati ottenuti in precedenza.

Come mostrato dalla Figura 4.21, l'addestramento ha impiegato un tempo molto elevato, considerando l'assenza del retraining. Gli ambienti "Intersezione 4" e "Doppio Porta Stretta" sono stati quelli in cui il modello ha avuto più difficoltà, nonostante l'assenza di grandi picchi negativi di reward (Figura 4.22).

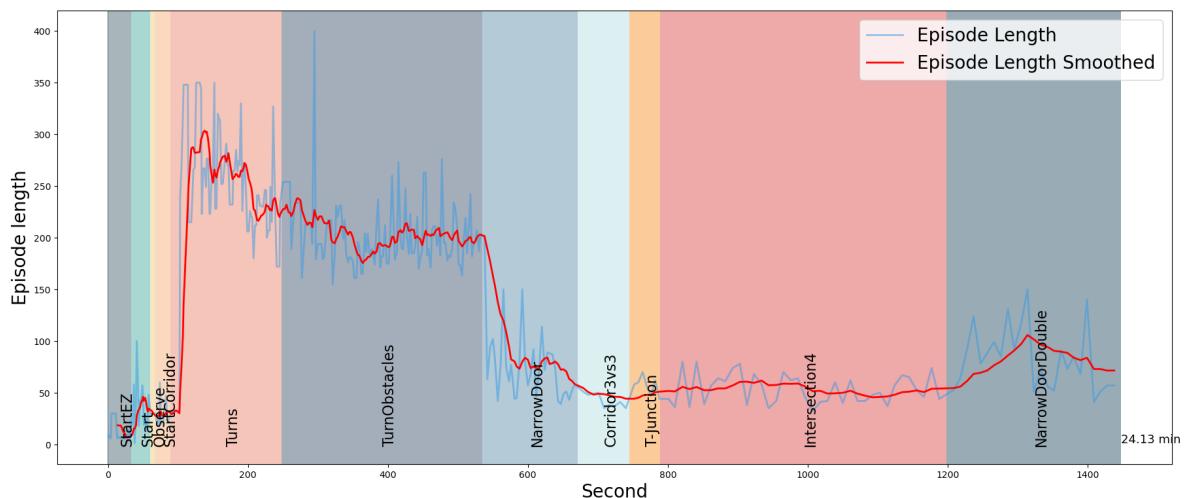


Figura 4.21: Lunghezza Episodi

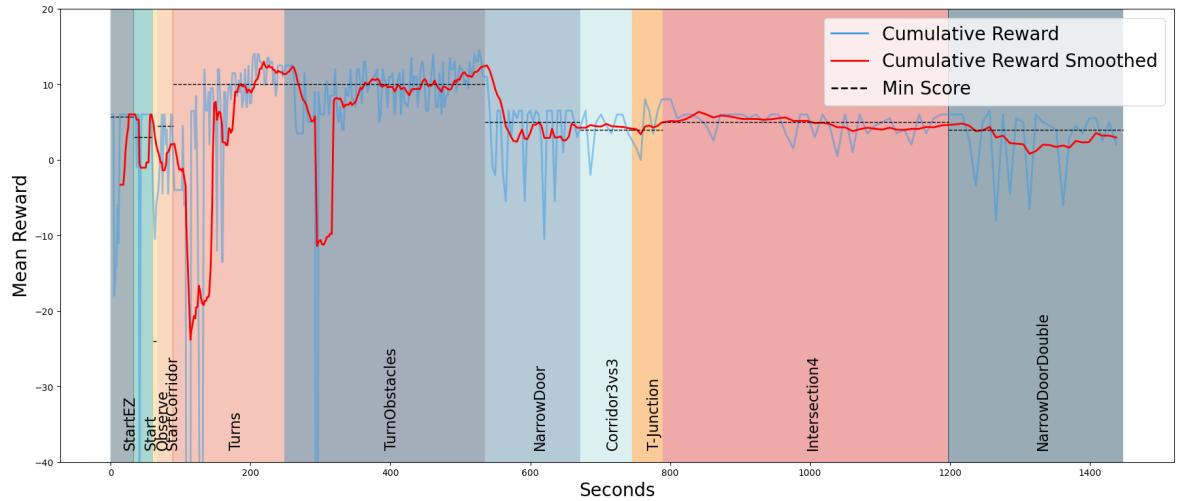


Figura 4.22: Reward Episodi

Nell’ambiente “Doppia Porta Opposta” (Figura 4.23(a)) sono presenti alcuni casi anomali. E’ importante notare il comportamento dei pedoni quando i due flussi si incontrano: il modello preferisce mantenere la stessa traiettoria per poi spostarsi solamente all’ultimo momento, piuttosto che mantenere due flussi separati fin da subito.

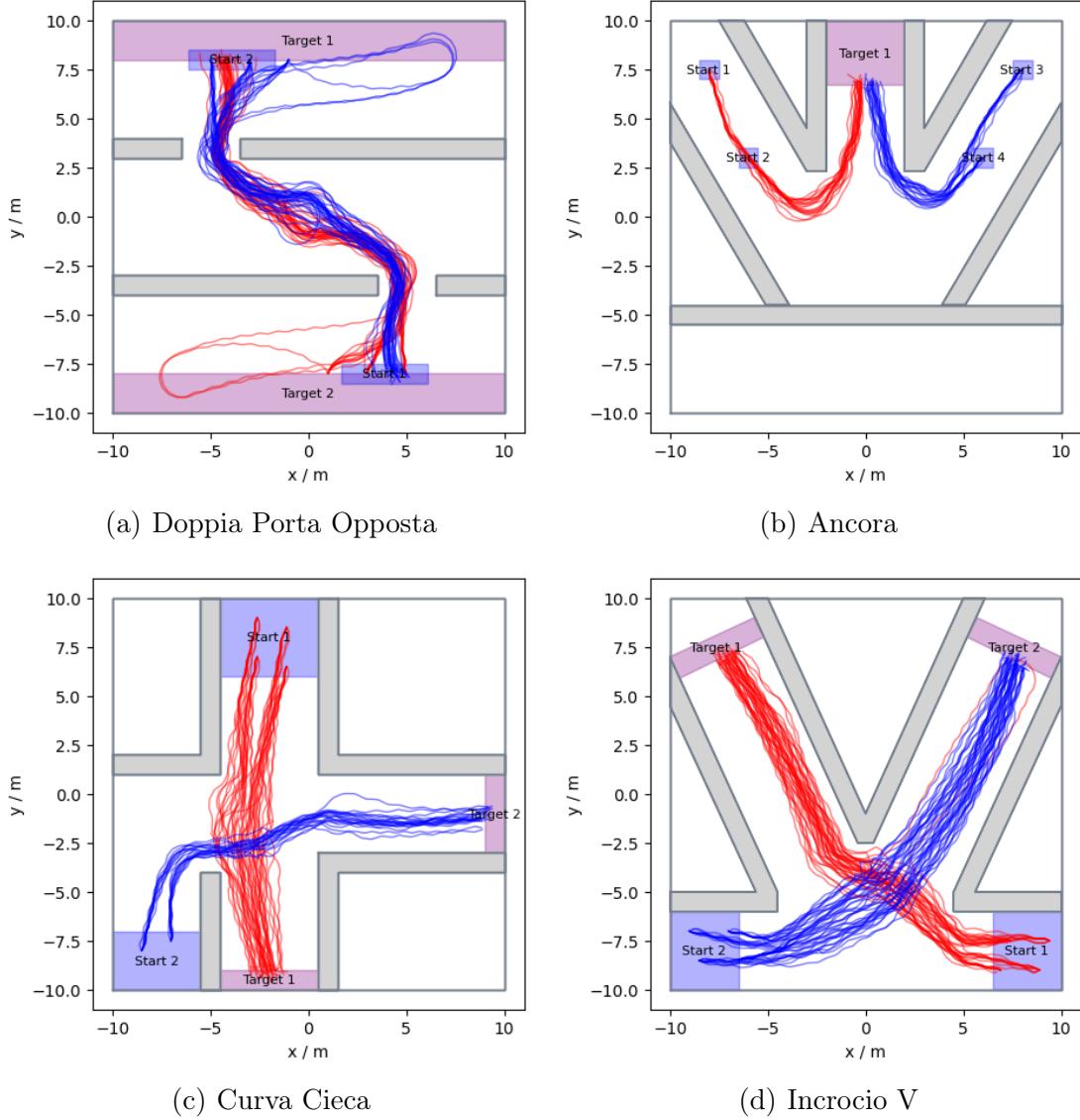


Figura 4.23: Risultati del modello con rete neurale da 256-128-64

In conclusione, i risultati visti nell’ambiente “Doppia Porta Opposta” fanno pensare che il modello abbia imparato correttamente ad evitare che i pedoni collidano fra di loro, ma, allo stesso tempo, sembra non sia stata sviluppata una forte preferenza su quale lato della strada mantenere.

4.3 Analisi Ablativa

Successivamente, è stata eseguita una analisi ablativa per visualizzare i componenti del curriculum di addestramento più importanti e quelli eliminabili. Gli ambienti presi in considerazione sono stati quelli a partire da “Porta Stretta”, in quanto quelli precedenti sono fondamentali per far apprendere

al pedone i comportamenti più importanti. Il modello utilizzato è quello descritto nella Sezione 4.2.3.

4.3.1 Curriculum senza Porta Stretta

L’addestramento senza “Porta Stretta” è stato portato a termine in poco più di 12 minuti (Figura 4.24) senza alcuna difficoltà. Si può, però, notare come il tempo nell’ambiente “Doppio Porta Stretta” sia aumentato considerevolmente.

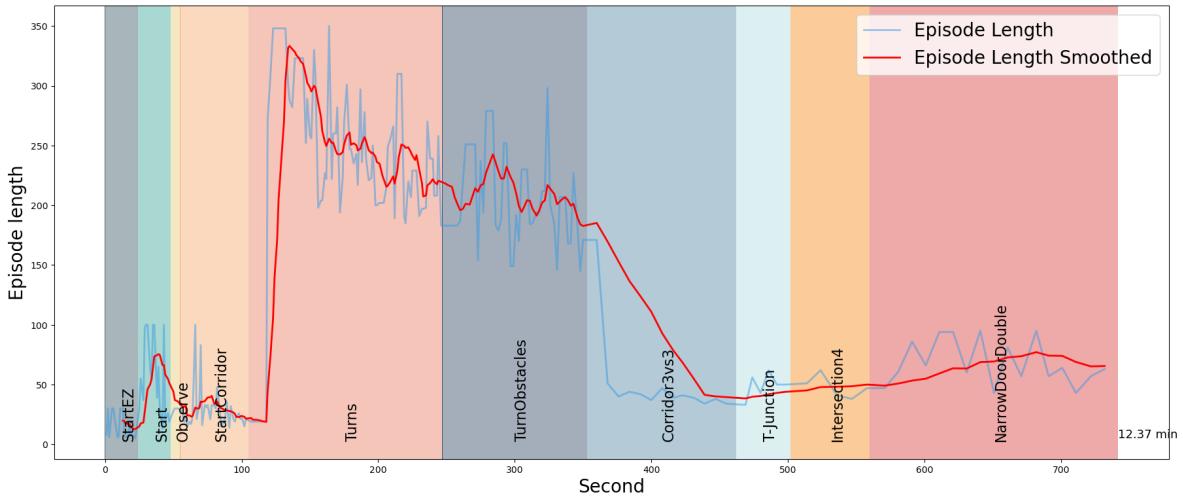


Figura 4.24: Lunghezza Episodi

Il modello risultante ha portato a termine la fase di test senza particolari problemi. Nell’ambiente “Doppia Porta Opposta” (Figura 4.25(a)) i due flussi di pedoni non sono particolarmente distinti. L’unico caso anomalo si può notare nell’ambiente “Curva Cieca” (Figura 4.25(c)) dove un pedone abbandona le solite traiettorie per percorrere un movimento circolare e poi riprendere il suo normale percorso.

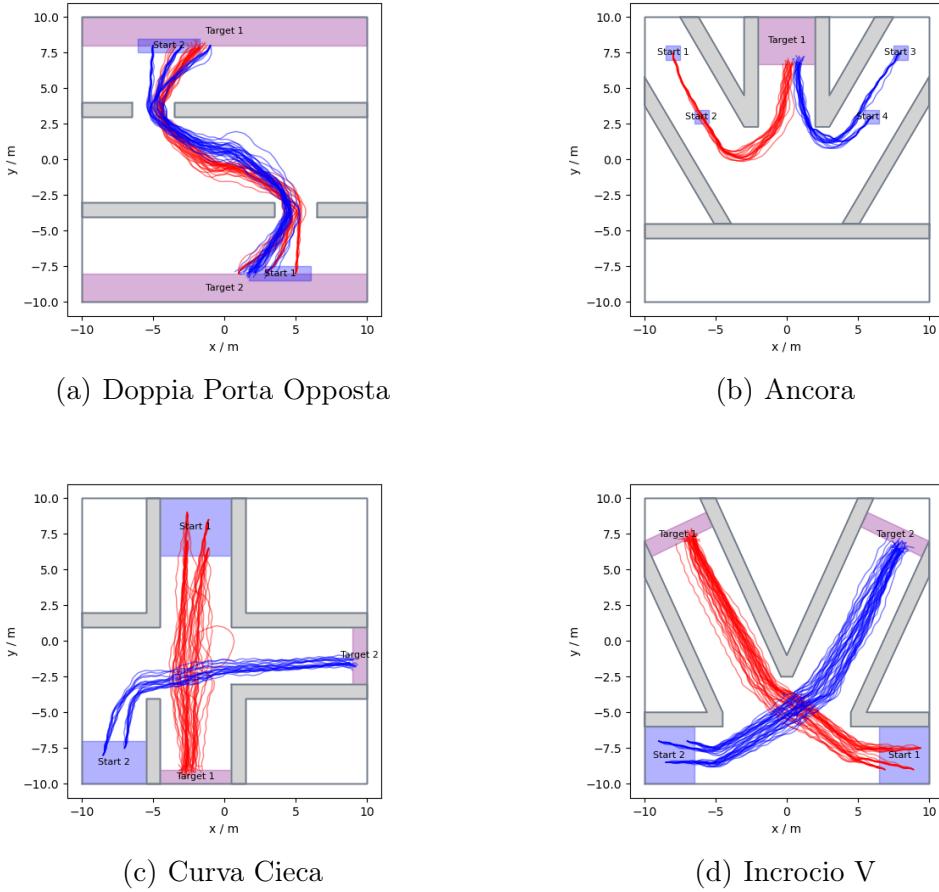


Figura 4.25: Risultati del modello senza “Porta Stretta”

In conclusione, l’ambiente “Porta Stretta” non è molto significativo nell’addestramento del modello. In sua assenza sono stati riportati dei lievi peggioramenti nel comportamento dei pedoni, che non compromettono la fase di test.

4.3.2 Curriculum senza Corridoio 3vs3

L’addestramento senza “Corridoio 3vs3” ha impiegato un tempo particolarmente elevato (Figura 4.26) a causa di molte run sfortunate negli ambienti “Curve” e “Curve Ostacoli”. Trascurando questo avvenimento, si può notare come il tempo di addestramento di “Intersezione 4” sia aumentato di molto: nell’addestramento normale è meno di 100 step, mentre in questo è di più di 200.

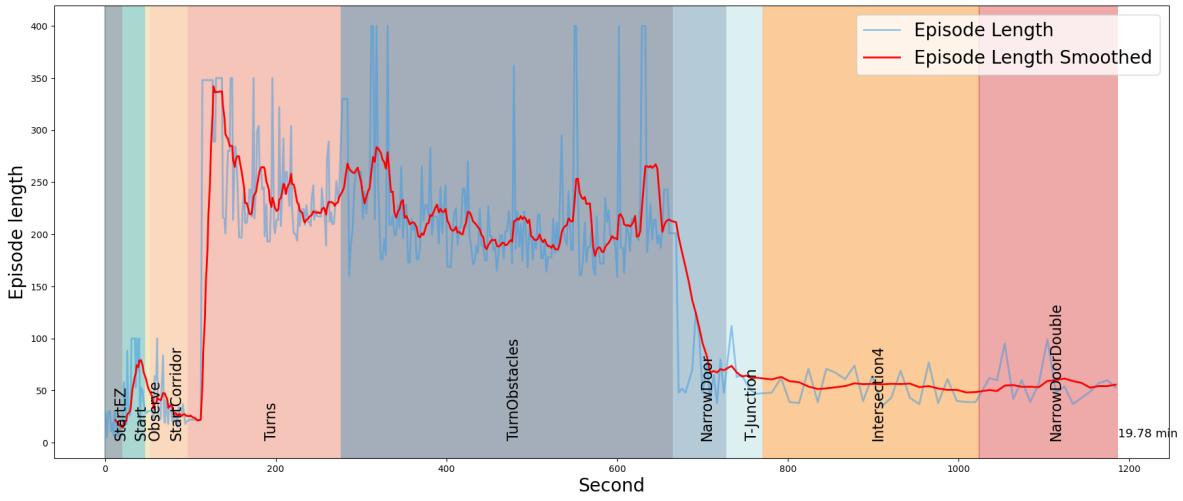
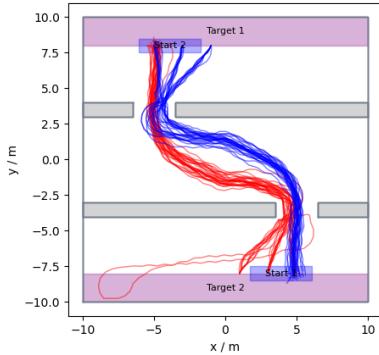
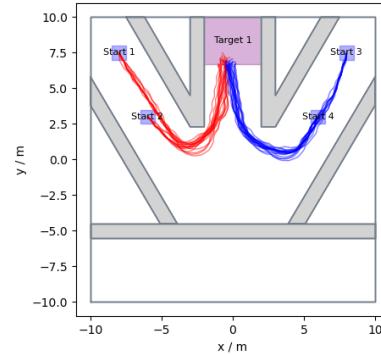


Figura 4.26: Lunghezza Episodi

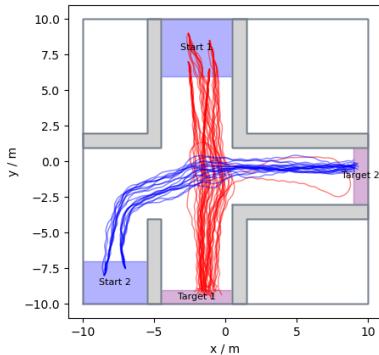
Nonostante il rallentamento dell’addestramento, il modello si comporta in modo eccellente in tutti gli ambienti di test. Sono presenti solo due outliers negli ambienti “Doppia Porta Opposta” (Figura 4.27(a)) e “Curva Cieca” (Figura 4.27(c)).



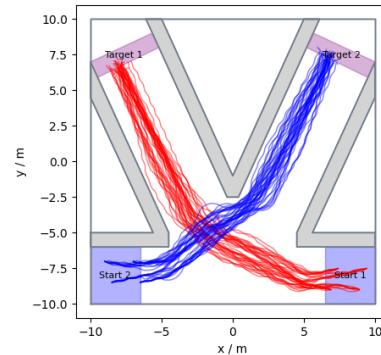
(a) Doppia Porta Opposta



(b) Ancora



(c) Curva Cieca



(d) Incrocio V

Figura 4.27: Risultati del modello senza “Corridoio 3vs3”

In conclusione, nonostante l'addestramento più lento di "Intersezione 4", il modello mostra un comportamento ottimale. E' evidente come l'assenza di "Corridoio 3vs3" non apporti cambiamenti fondamentali al modello.

4.3.3 Curriculum senza Giunzione a T

L'addestramento senza "Giunzione a T" ha impiegato un tempo particolarmente elevato (Figura 4.28), in parte, di nuovo a causa degli ambienti "Curve" e "Curve Ostacoli". Nonostante questi due intoppi, si può notare come i tempi di "Intersezione 4" e di "Doppio Porta Stretta" siano aumentati considerevolmente.

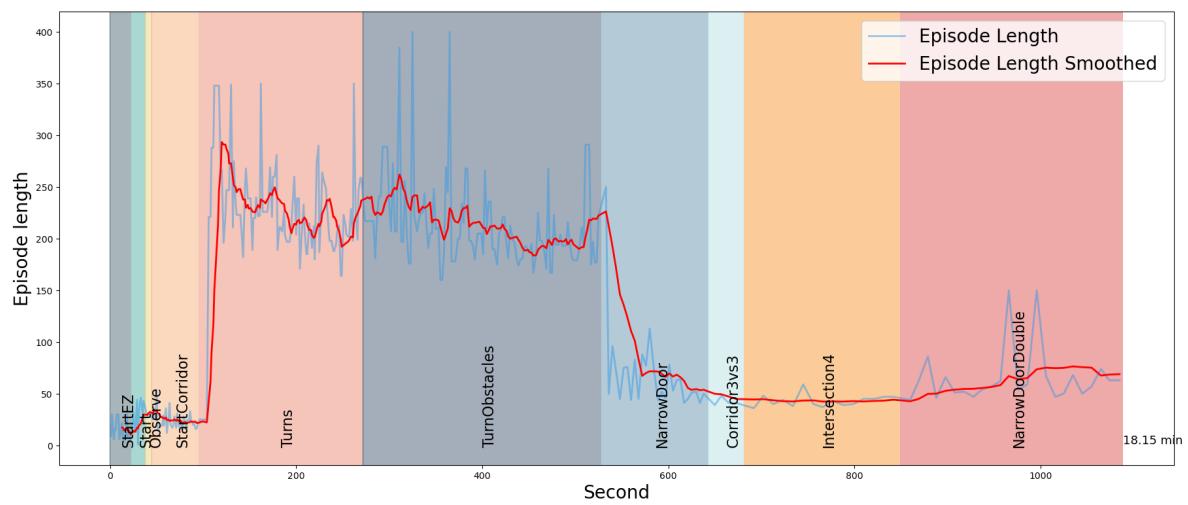


Figura 4.28: Lunghezza Episodi

Il modello ha portato a termine la fase di test in modo perfetto, senza alcuna anomalia.

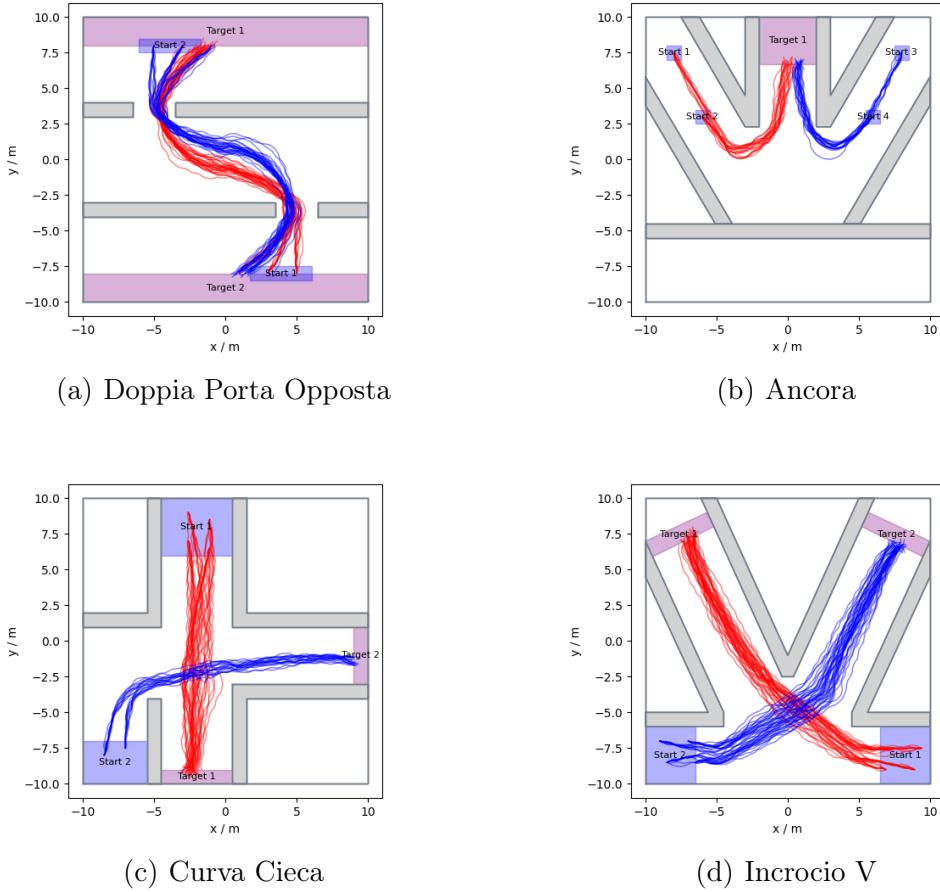


Figura 4.29: Risultati del modello senza “Giunzione a T”

In conclusione, il modello ottenuto senza “Giunzione a T” si comporta comunque come dovrebbe. Senza questo ambiente, però, i tempi di addestramento degli ultimi due ambienti del curriculum si alzano lievemente.

4.3.4 Curriculum senza Intersezione 4

L’addestramento senza “Intersezione 4” ha impiegato un tempo relativamente basso. Sorprendentemente, è possibile vedere come l’ambiente “Doppio Porta Stretta” impieghi un periodo di tempo molto basso (Figura 4.30) rispetto a tutti i test precedenti.

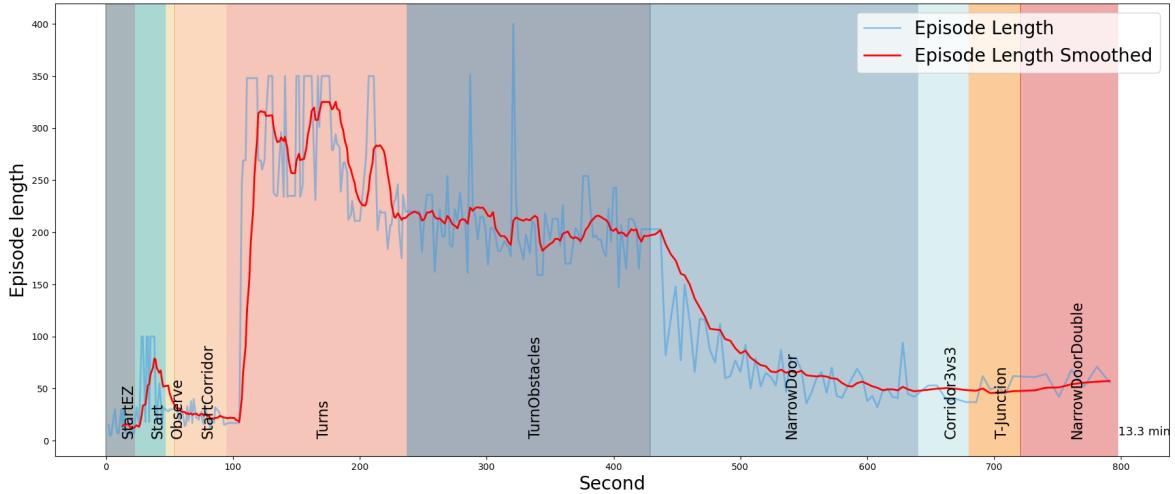


Figura 4.30: Lunghezza Episodi

Come è possibile notare dalla Figura 4.31(a), il modello addestrato presenta molte difficoltà nell’ambiente “Doppia Porta Opposta”, dove molti pedoni tendono ad allontanarsi dalla prima porta del loro percorso e altri tendono ad aggregarsi al gruppo sbagliato al centro dell’ambiente, tornando al loro punto di partenza. Questo potrebbe essere un segnale di underfitting: i tempi di addestramento molto ridotti di “Doppio Porta Stretta” e l’assenza di “Intersezione 4” possono aver causato una mancanza di informazioni fondamentali per l’esecuzione di ambienti complessi.

Nell’ambiente “Ancora” è evidente come alcuni pedoni del gruppo di sinistra vadano a sbattere inspiegabilmente contro il muro subito prima del corridoio centrale.

Nell’ambiente “Incrocio V” (Figura 4.31(d)) sono presenti alcuni casi anomali in concomitanza del corridoio per raggiungere l’obiettivo del gruppo blu.

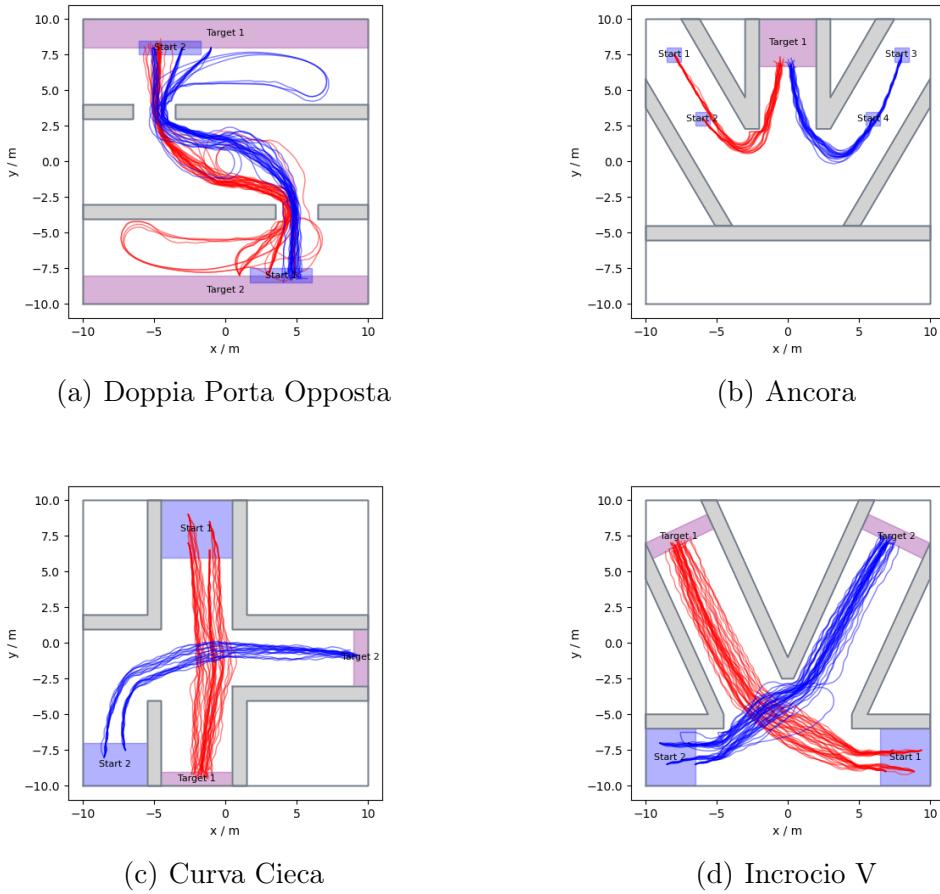


Figura 4.31: Risultati del modello senza “Intersezione 4”

In conclusione, è evidente come la mancanza di un ambiente complesso come “Intersezione 4” possa ridurre le prestazioni del modello addestrato, rendendolo incapace di attraversare ambienti complessi (“Doppia Porta Opposta”) e peggiorando le prestazioni anche in ambienti più semplici (“Ancora”).

4.3.5 Curriculum senza Doppio Porta Stretta

Essendo “Doppio Porta Stretta” l’ultimo elemento del curriculum, gli altri ambienti non hanno subito un rallentamento nei tempi di esecuzione (Figura 4.32).

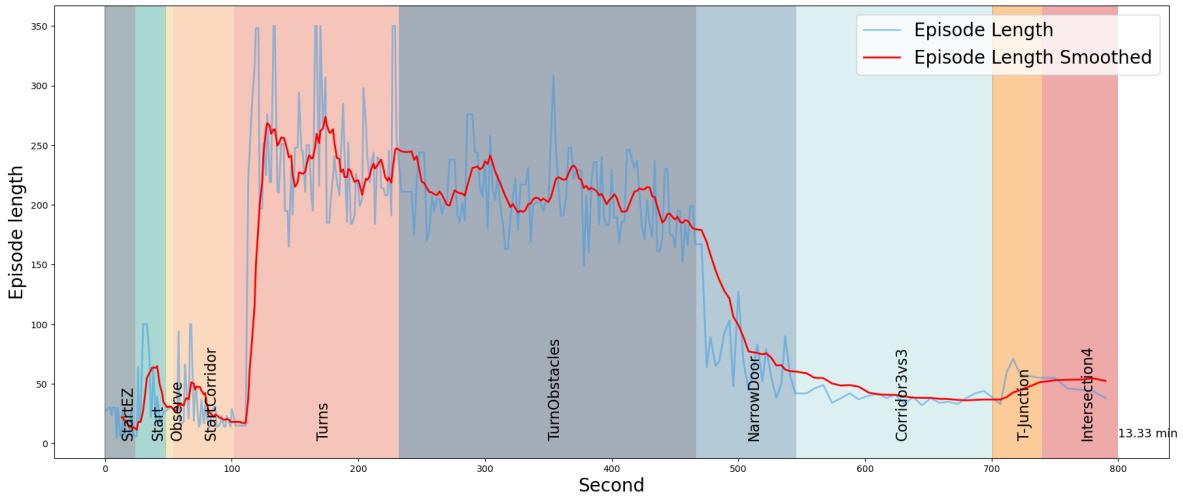
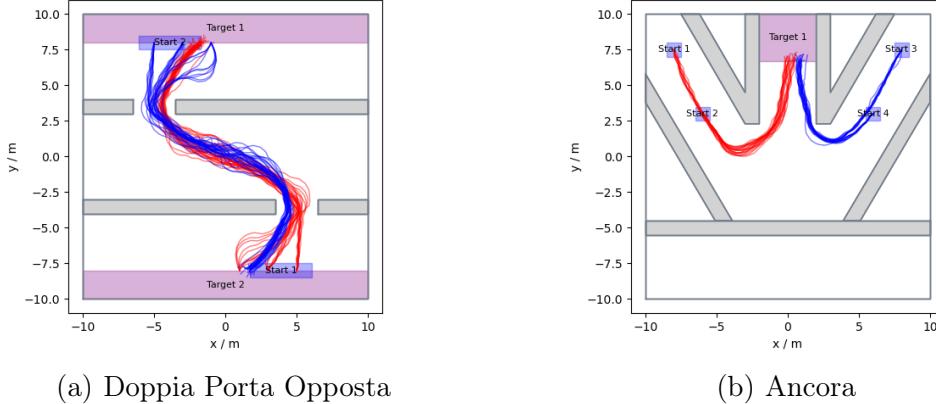


Figura 4.32: Lunghezza Episodi

Nell’ambiente “Doppia Porta Opposta” (Figura 4.33(a)) è evidente come non ci sia alcuna distinzione tra i due flussi di pedoni. Tutti i pedoni percorrono l’ambiente senza prestare nessuna attenzione a tutti gli altri.



(a) Doppia Porta Opposta

(b) Ancora

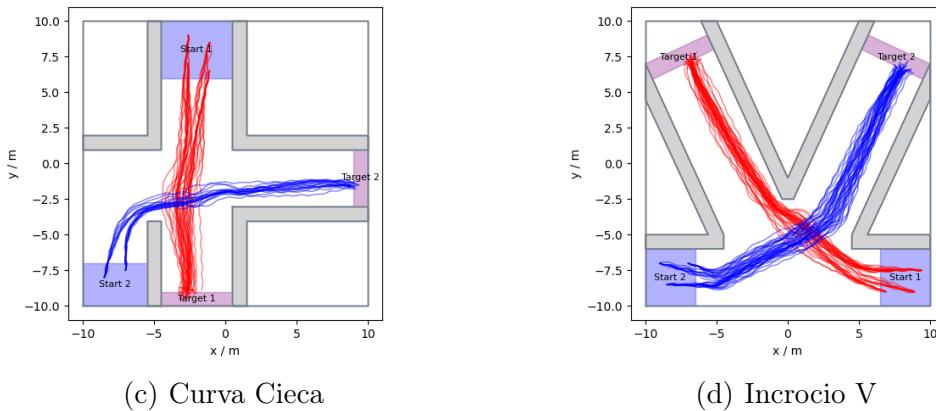


Figura 4.33: Risultati del modello senza “Doppio Porta Stretta”

In conclusione, “Doppio Porta Stretta” riveste un ruolo cruciale nel determinare il comportamento dei pedoni in situazioni in cui due gruppi si incontrano frontalmente. Il modello addestrato non mostra alcuna capacità di gestire correttamente questo tipo di situazione.

4.3.6 Conclusioni

Tramite questa analisi ablativa è stato possibile visualizzare i ruoli e l’importanza dei vari ambienti del curriculum. Nessun ambiente in particolare è fondamentale per l’addestramento del modello, ma alcuni si sono dimostrati cruciali per aumentare le prestazioni.

I primi tre ambienti analizzati (“Porta Stretta”, “Corridoio 3vs3” e “Giunzione a T”) non sono fondamentali per l’addestramento del modello, bensì si sono dimostrati utili per velocizzare le fasi successive del curriculum. L’ambiente “Intersezione 4” si è dimostrato fondamentale per affinare le prestazioni in ambienti complessi. Infine, l’ambiente “Doppio Porta Stretta” è molto utile per apprendere dei comportamenti specifici.

E’ infine opportuno specificare che i test sono stati eseguiti una volta ciascuno. Per una maggiore precisione nell’analisi, sarebbe interessante eseguire gli stessi test più volte per eliminare (o quantomeno ridurre) la stocasticità degli esperimenti.

4.4 Analisi ad Alta Densità

In parallelo a questo studio, è stata svolta un’analisi su ambienti ad elevata densità con l’obiettivo di ottimizzare i tempi di evacuazione di vari ambienti. In questa analisi è stato modificato il curriculum base per abituare il modello fin dall’inizio ad ambienti ad elevata densità. Nella fase di test è stato poi verificato che il modello agisca correttamente in tre tipi di ambienti:

1. **Modello base:** Normali ambienti di test, visti anche in precedenza.
2. **Modello aumentato:** Normali ambienti di test, ma con un numero di pedoni molto più elevato.
3. **Modello esteso:** Nuovi ambienti di test, pensati appositamente per analizzare i comportamenti del modello in scenari specifici.

In particolare, in questa analisi, verranno considerati solo i nuovi ambienti del modello esteso.

L'obiettivo sarà quello di verificare se il modello ottenuto nella sezione 4.2.3 sia in grado di portare a termine correttamente la fase di test in ambienti ad elevata densità.

Nella fase di test verranno utilizzati i classici grafici delle traiettorie e altri grafici relativi alla densità, sviluppati nello studio parallelo. In particolare saranno presenti i grafici nT per evidenziare il flusso di pedoni che passa attraverso i colli di bottiglia e dei diagrammi a clessidra per visualizzare i valori della densità all'interno dei colli di bottiglia.

4.4.1 Ambienti del Modello Esteso

Il primo ambiente introdotto è necessario per l'analisi della folla di fronte a un collo di bottiglia. Sono state create tre versioni dello stesso ambiente, ognuna con 60 pedoni. La differenza tra queste versioni consiste nella larghezza iniziale del collo di bottiglia: nella prima misura 5.6 metri, nella seconda 3.5 metri e nella terza 2.0 metri.

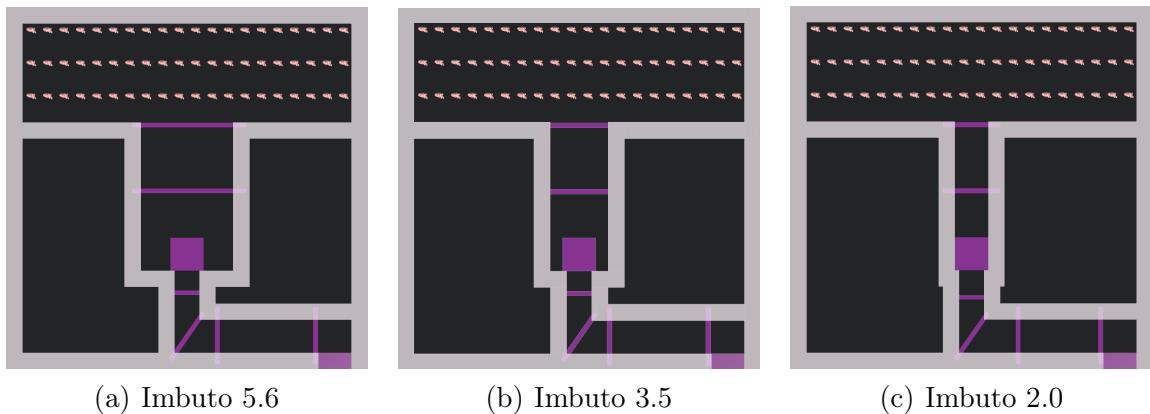


Figura 4.34: Ambienti “Imbuto 5.6”, “Imbuto 3.5” e “Imbuto 2.0” dello studio ad alta densità.

Successivamente sono stati creati altri due ambienti per l'analisi dei tempi di evacuazione di ambienti con porte situate al centro oppure ai lati. Entrambi gli ambienti hanno due stanze ed in ogni stanza sono presenti 20 pedoni, per un totale di 40. Come possibile vedere dalle Figure 4.35(a) e 4.35(b), la sostanziale differenza sta nella posizione del collegamento tra le stanze laterali e quella centrale: in “Collegamento ad Angolo” le stanze comunicano attraverso gli angoli, mentre in “Collegamento Centrale” le stanze sono collegate al centro delle pareti laterali.

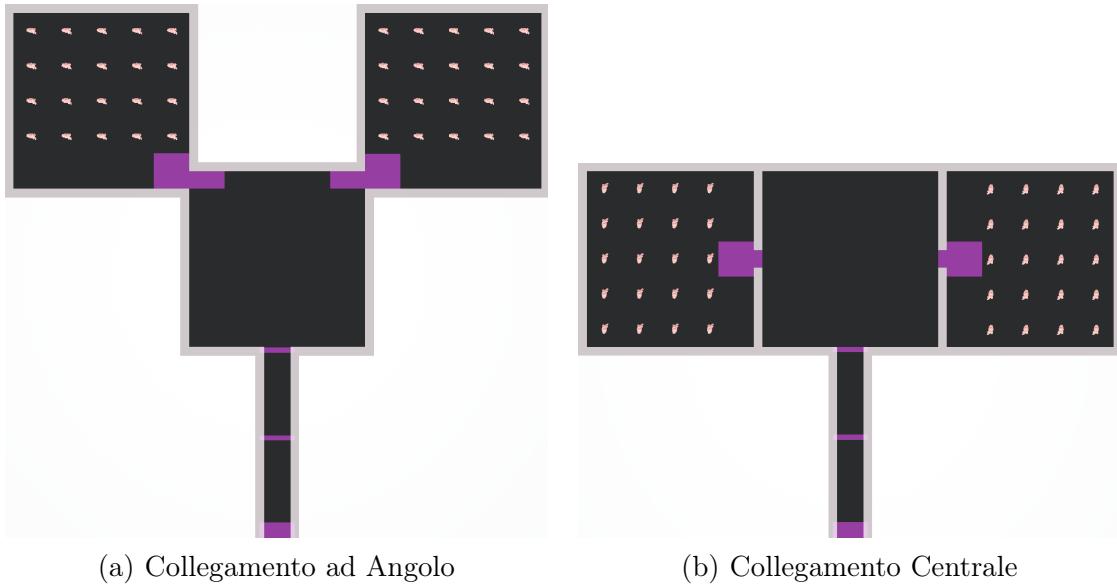


Figura 4.35: Ambienti “Collegamento ad Angolo” e “Collegamento Centrale” dello studio ad alta densità.

4.4.2 Risultati ottenuti

Di seguito verranno riportati i grafici dei risultati ottenuti in questo studio e nello studio in parallelo, per avere un chiaro confronto tra il modello addestrato con curriculum base e quello addestrato con curriculum ad alta densità.

Ambienti Imbuti

Come possibile notare dalla Figura 4.36, il modello addestrato con il curriculum base è chiaramente peggiore per ambienti ad elevata densità. I pedoni, invece che aspettare sul posto, preferiscono continuare a camminare in tondo finché non è disponibile un percorso non troppo affollato verso l’obiettivo finale. In particolare è possibile notare come i pedoni tendano a spingersi tutti contemporaneamente attraverso il collo di bottiglia, senza dare la precedenza o aspettare che gli altri pedoni fluiscano in modo omogeneo. Come nei test eseguiti nello studio in parallelo, man mano che si stringe il collo di bottiglia, si crea maggiore confusione nella zona iniziale.

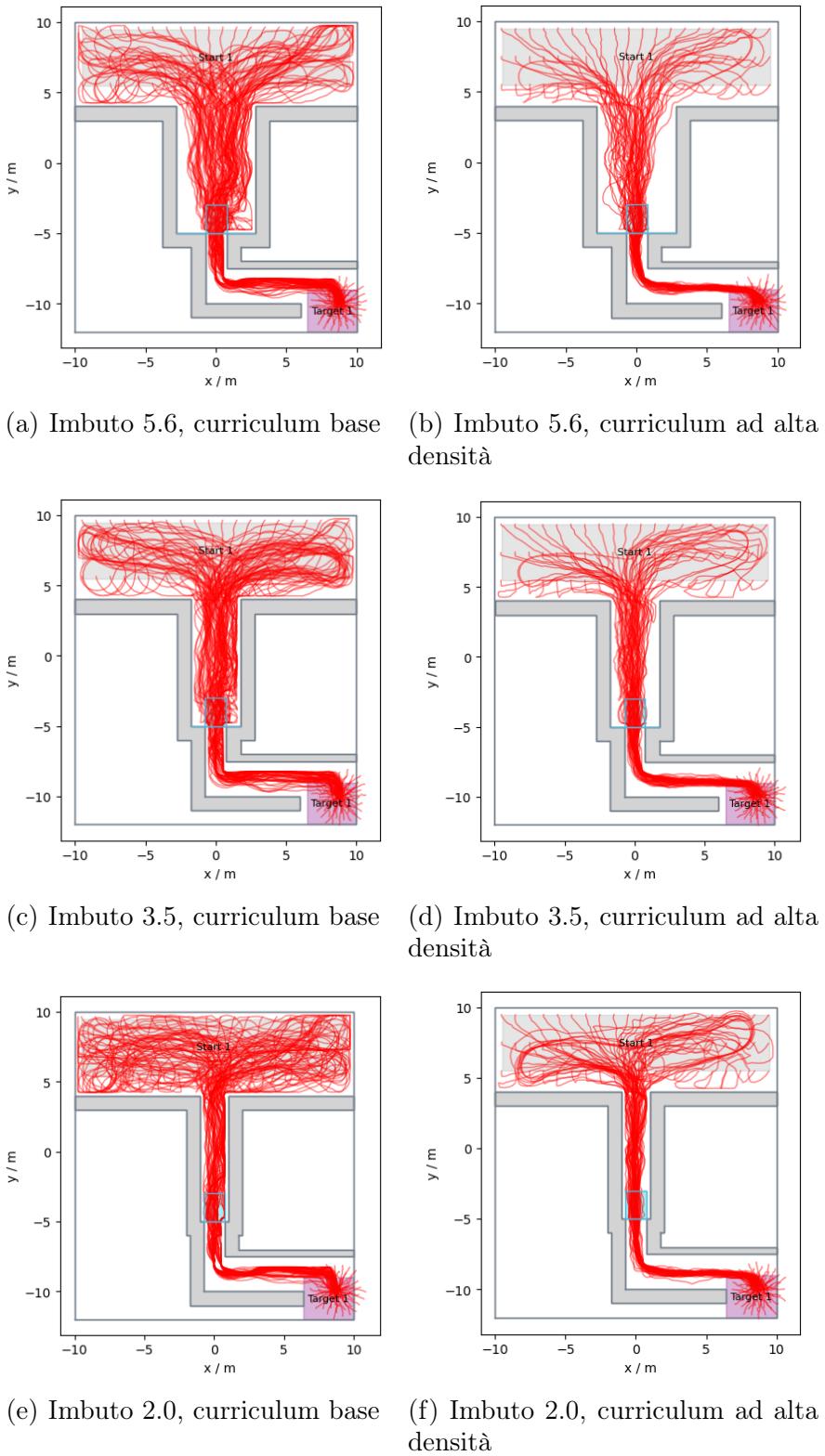


Figura 4.36: Confronto delle traiettorie degli ambienti Imbuto

Successivamente sono stati utilizzati dei grafici nT. Questi servono per visualizzare la quantità di pedoni che sono passati attraverso il bottleneck in un determinato istante di tempo. Sono particolarmente utili per visualizzare

in quanto tempo il flusso di pedoni passa attraverso il collo di bottiglia.

Come possibile vedere dalla figura 4.37, in tutti gli ambienti, il modello con curriculum base tende a smuovere i pedoni più velocemente, ma non con un andamento lineare, come invece avviene con il modello con il curriculum specializzato. Negli ambienti “Imbuto 3.5” e “Imbuto 2.0”, si nota come la grande maggioranza dei pedoni passino dal bottleneck in un tempo molto ristretto, per poi impiegare moltissimo tempo per far passare gli ultimi pedoni rimanenti. In “Imbuto 3.5” (Figura 4.37(c)) 59 pedoni impiegano circa 15 secondi per attraversare il bottleneck, mentre l’ultimo pedone impiega da solo 10 secondi in più degli altri. In “Imbuto 2.0” (Figura 4.37(e)) più di 50 pedoni impiegano circa 10 secondi per attraversare il bottleneck, mentre gli ultimi pedoni rimanenti impiegano più del doppio del tempo per completare il loro tragitto. L’ambiente “Imbuto 5.6” (Figura 4.37(a)) non presenta in modo molto accentuato questo comportamento, ma è comunque possibile notare come i pedoni completino l’ambiente in modo più rapido col modello con cv base. Questi grafici mostrano chiaramente l’incapacità del modello addestrato con curriculum base nell’eseguire ambienti ad elevata densità, alternando momenti in cui i pedoni fluiscano alla massima velocità e momenti in cui i pedoni non riescono a percorrere l’ambiente correttamente, facendo pensare che si creino molti intoppi che bloccano totalmente (o quasi) il passaggio attraverso il collo di bottiglia.

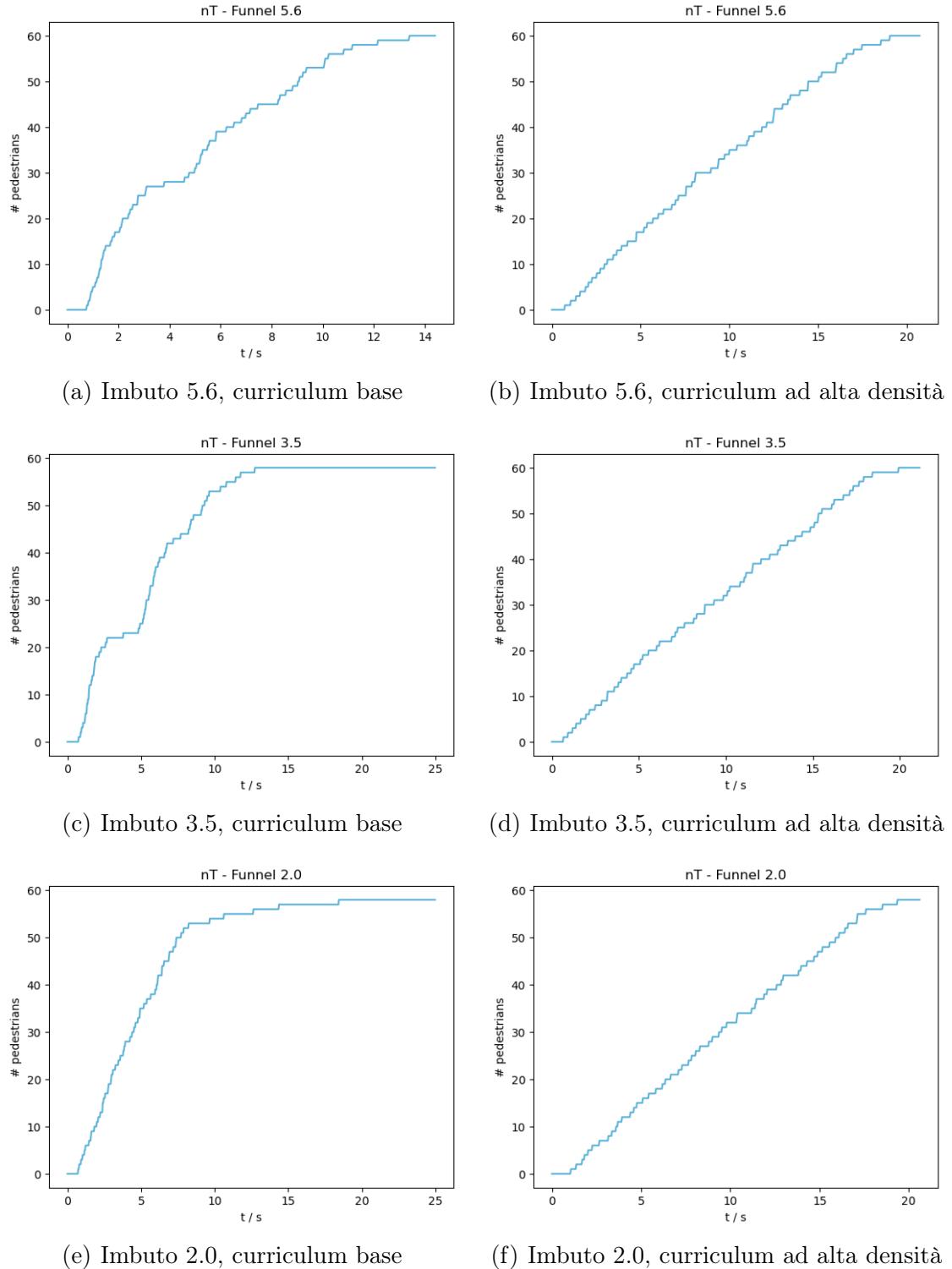


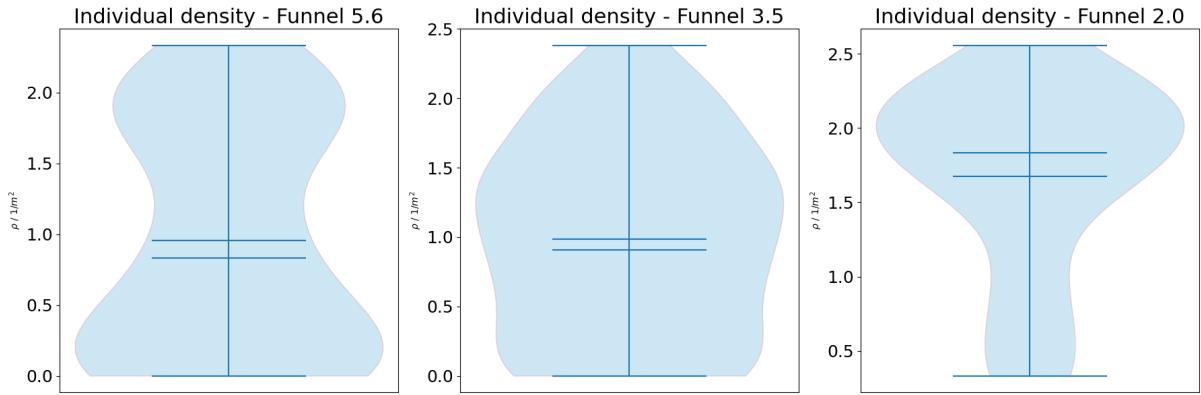
Figura 4.37: Confronto del flusso pedonale degli ambienti Imbuto

Per concludere sono stati utilizzati dei diagrammi a clessidra. Questi ci forniscono due tipi di informazioni: la quantità di densità presente nel collo di bottiglia, (in numero di persone per metro quadro) e indicazioni sul comportamento dei pedoni durante il passaggio. Se il grafico risulta simmetrico,

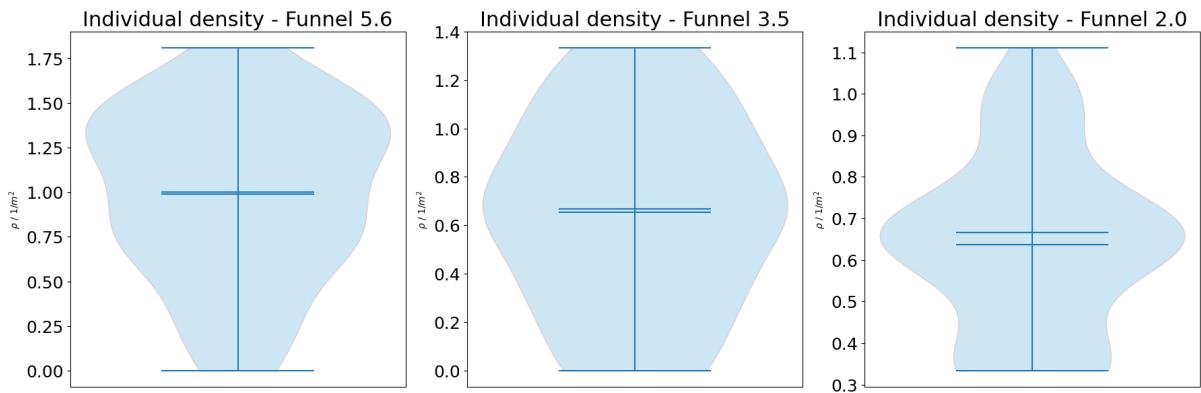
significa che i pedoni hanno mantenuto le norme comportamentali corrette e che il flusso di pedoni è rimasto costante durante l'esecuzione.

Come è possibile notare dalla figura 4.38, in tutti gli ambienti sono state registrate delle densità notevolmente più elevate nel modello con curriculum base. Questo ci fa pensare che i pedoni del modello con curriculum base non rispettino alcuna norma e che non abbiano appreso dei comportamenti necessari per navigare in modo efficacie ambienti ad alta densità. Analizzando brevemente la figura 4.38(a), si può notare che:

- Nell'ambiente “Imbuto 5.6” sono state registrate due densità prevalenti, una intorno a 2 e l'altra intorno a 0.2 pedoni al metro quadro. Questo ci fa pensare che una parte dei pedoni si siano lanciati all'interno del collo di bottiglia per raggiungere l'obiettivo finale più rapidamente possibile e l'altra parte ha, invece, aspettato il passaggio degli altri pedoni prima di proseguire.
- Il grafico dell'ambiente “Imbuto 3.5” è quello risultato più simmetrico, indicando che il flusso di pedoni sia passato attraverso il collo di bottiglia in maniera abbastanza regolare, ma comunque con troppa fretta e accalcandosi troppo, vedendo le elevate densità registrate.
- Nell'ambiente “Imbuto 2.0” si può notare come spicca una densità di 2 pedoni al metro quadro. Questo indica come, per tutta la durata del test, ci sia stato un flusso costante di pedoni che tentava di accalcarsi all'interno del bottleneck, senza alcun riguardo per la distanza tra un pedone e l'altro.



(a) Densità individuali degli ambienti Imbuti del curriculum base

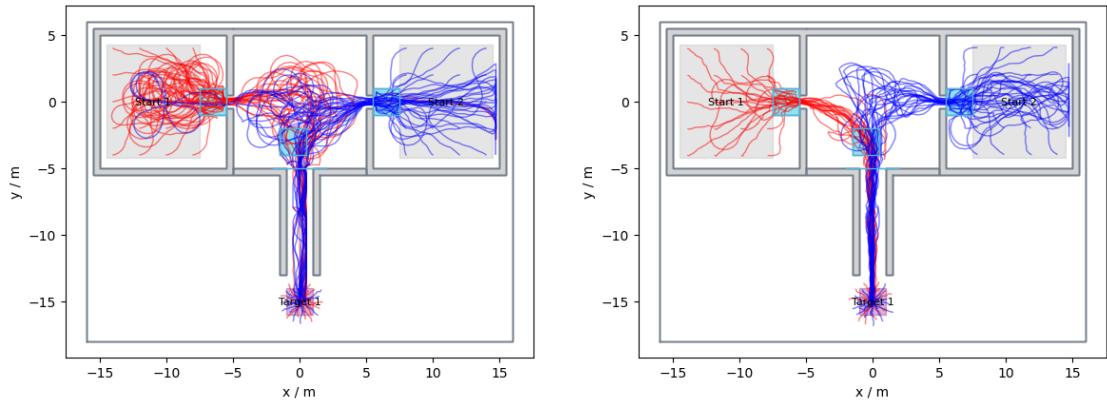


(b) Densità individuali degli ambienti Imbuti del curriculum ad alta densità

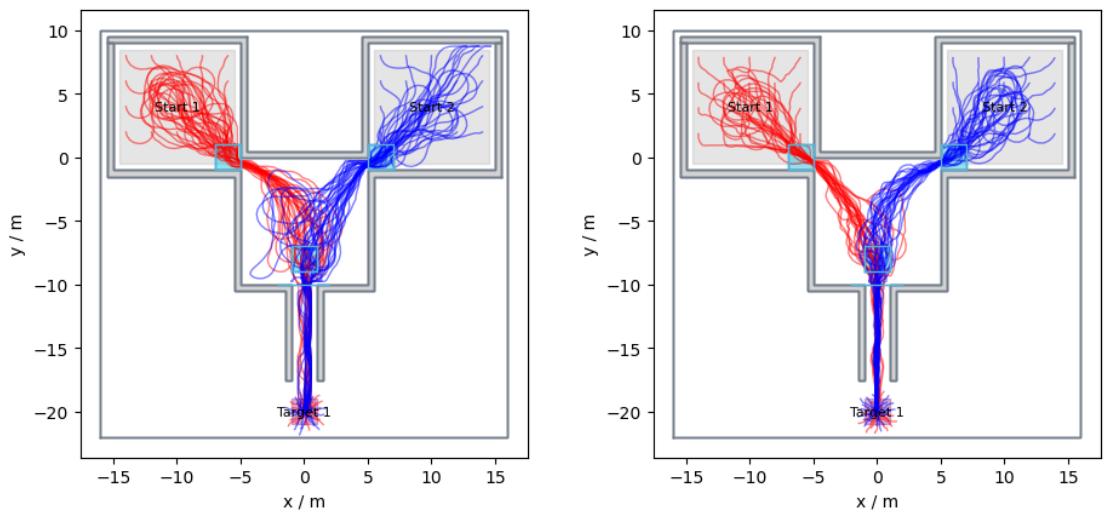
Figura 4.38: Confronto delle densità all'interno del bottleneck degli ambienti Imbuti

Ambienti Collegamenti

Come è possibile notare dalla figura 4.39, il comportamento dei pedoni del modello con cv base è peggiore e molto più confusionale rispetto a quello dell'altro modello. Nella stanza in comune è possibile notare una particolare confusione nel flusso dei pedoni. Molti pedoni, nel momento in cui trovano le uscite occupate, preferiscono continuare a vagare in attesa che si liberi l'uscita, piuttosto che stare fermi. Nella figura 4.39(a) è possibile notare, adirittura, come qualche pedone vada a finire nella stanza opposta, piuttosto che rimanere in attesa in quella centrale.



(a) Collegamento Centrale, curriculum base (b) Collegamento Centrale, curriculum ad alta densità



(c) Collegamento ad Angolo, curriculum base (d) Collegamento ad Angolo, curriculum ad alta densità

Figura 4.39: Confronto delle traiettorie degli ambienti Collegamento

Nei grafici nT (figura 4.40) è presente lo stesso comportamento visto anche negli ambienti Imbuto. In “Collegamento Centrale” (figura 4.40(a)), i primi 38 pedoni completano il loro percorso in circa 10 secondi, mentre i due restanti impiegano circa il doppio per attraversare il collo di bottiglia. Nell’ambiente “Collegamento ad Angolo” (figura 4.40(c)) questo comportamento è meno accentuato, ma il passaggio dei pedoni attraverso il bottleneck non è comunque lineare.

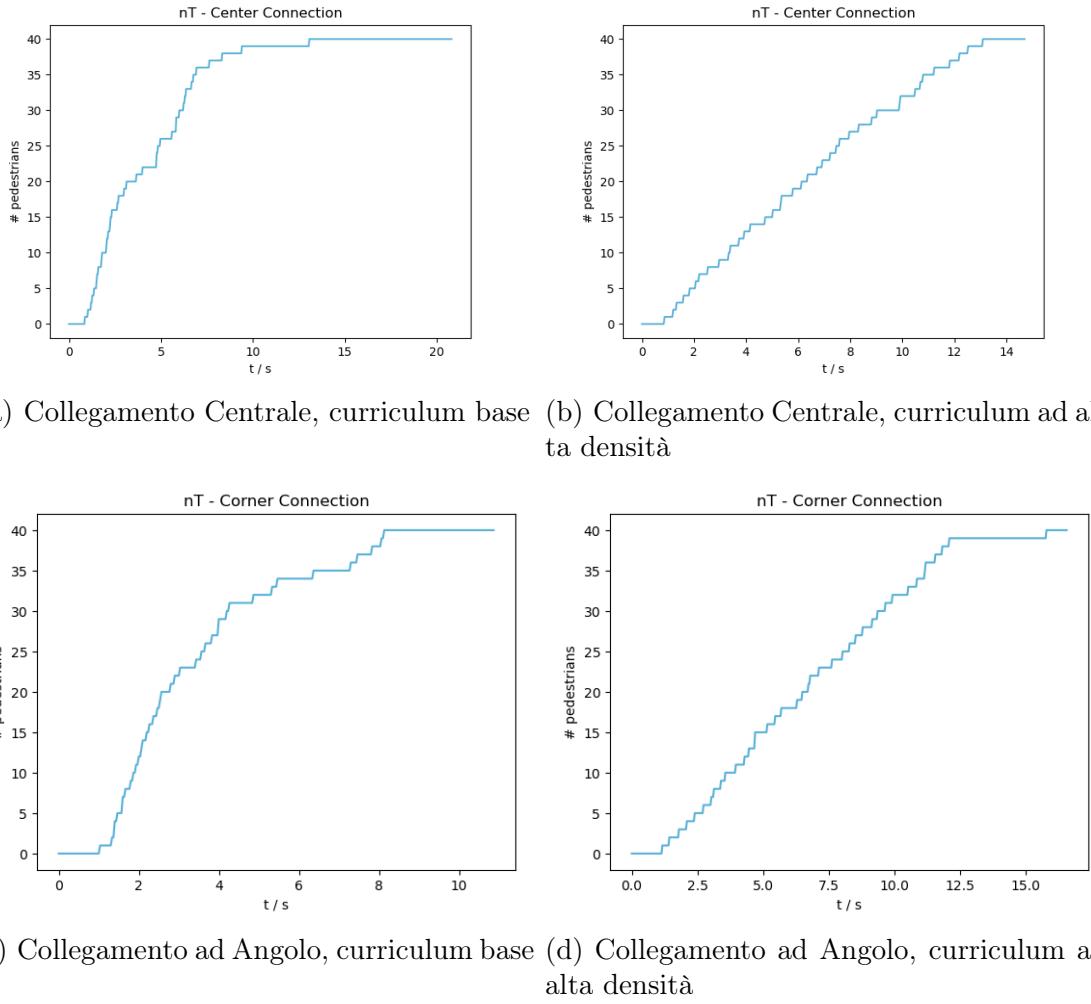


Figura 4.40: Confronto del flusso di pedoni degli ambienti Collegamento

4.4.3 Conclusioni

Come visto in precedenza, il modello addestrato con il curriculum base mostra delle scarse capacità di navigazione di ambienti ad alta densità. I pedoni risultano incapaci di attendere sul posto che si liberino i colli di bottiglia e preferiscono, invece, vagare liberamente all'interno dello spazio disponibile. Il modello sembra non badare ai reward negativi ricevuti dalla prossemica tra i pedoni, andando a causare in molti casi dei grossi ingorghi.

Ciò nonostante, è bene notare che, con questo modello, i tempi di evacuazione dei vari ambienti sono risultati inferiori a quelli del modello addestrato con cv ad alta densità. Questo è dovuto al fatto che i pedoni non si curano della vicinanza con gli altri pedoni e tendono a completare i vari ambienti il prima possibile.

In conclusione, questo modello non si è dimostrato capace di riprodurre dei comportamenti reali che assumono le folle in situazioni di alta densità. Un curriculum specializzato è necessario per riprodurre questi tipi di comportamenti.

Capitolo 5

Conclusione e sviluppi futuri

Questa tesi si è incentrata sulla realizzazione di agenti intelligenti utilizzando il Reinforcement Learning e sulla creazione di un sistema per la simulazione pedonale in cui allenarli e testarli.

Durante la prima parte del progetto, è stato sviluppato un sistema per la gestione completa delle simulazioni pedonali, offrendo la possibilità di creare nuovi ambienti e agenti. Il sistema è completamente e facilmente personalizzabile, permettendo di essere utilizzato sia per l'addestramento degli agenti sia per testare differenti modelli di simulazione pedonale.

Durante la seconda parte del progetto, sono stati condotti due studi principali che hanno fornito risultati significativi per l'ottimizzazione e la comprensione del modello di addestramento.

Il primo studio si è concentrato su un'analisi di sensibilità e degli iperparametri, con l'obiettivo di ottimizzare i tempi di addestramento e migliorare le prestazioni in fase di test. I principali iperparametri analizzati sono stati Learning Rate, Batch Size e l'architettura della rete neurale utilizzata. È stato condotto un ulteriore studio sulla fase di retraining, il quale, contrariamente alle aspettative, peggiorava le prestazioni del modello e di conseguenza aumentava i tempi di addestramento inutilmente.

Il secondo studio ha esaminato l'importanza del curriculum attraverso un'analisi ablativa, rimuovendo singoli ambienti uno alla volta per valutare il loro impatto sulle prestazioni del modello. È emerso che l'eliminazione di un singolo ambiente non comprometteva drasticamente le prestazioni, ma aumentava leggermente i tempi di addestramento. Questo suggerisce che ulteriori indagini potrebbero includere la rimozione simultanea di più ambienti o l'inversione della loro posizione nel curriculum. Tuttavia, i risultati di questi test si sono rivelati altamente stocastici, evidenziando la necessità di eseguire molteplici run per ottenere una media affidabile delle prestazioni dei modelli.

Successivamente, è stata effettuata un'analisi in ambienti ad elevata densità. Uno studio parallelo ha sviluppato un nuovo curriculum con una particolare enfasi su questi scenari, includendo quindi nuovi ambienti di test

specificamente progettati. I risultati hanno dimostrato che il modello addestrato con il curriculum base era inefficace nella gestione della navigazione dei pedoni in ambienti ad elevata densità. Questo sottolinea l'importanza di un curriculum specializzato per questo tipo di ambienti.

In conclusione, i risultati ottenuti evidenziano l'importanza e la complessità di una progettazione accurata del curriculum e della scelta degli iperparametri per l'addestramento dei modelli.

Il progetto descritto in questa relazione si distingue per la sua elevata personalizzabilità e facilità di modifica, aprendo la strada a numerosi possibili sviluppi futuri.

Un primo sviluppo possibile è l'introduzione della completa eterogeneità dei pedoni. In questo scenario, ogni pedone potrebbe essere dotato di un comportamento peculiare che non riguarda solo la velocità desiderata, ma anche la distanza che desidera mantenere dagli altri pedoni, la tendenza a stare vicino ai muri, e così via. La varietà di comportamenti individuali modellabili è estremamente ampia, il che potrebbe portare a simulazioni più imprevedibili e dinamiche. Questi miglioramenti non solo renderebbero le simulazioni più realistiche, ma permetterebbero anche di studiare una gamma più vasta di scenari, migliorando la nostra comprensione dei comportamenti pedonali in situazioni diverse.

Inoltre, un ulteriore sviluppo futuro potrebbe essere la creazione di un sistema multi-agente. Attualmente, il modello utilizzato è basato su un singolo agente, il che implica che le osservazioni e i reward sono centrati su un pedone considerato individualmente e non sul comportamento del gruppo. Questo può costituire un problema, soprattutto in situazioni con molti pedoni, poiché all'aumentare del numero di pedoni aumenta anche la stocasticità dell'ambiente. Non è sempre possibile determinare se un comportamento specifico è il risultato dell'azione di un singolo pedone o dell'interazione con altri pedoni. Implementare un sistema multi-agente permetterebbe di modellare meglio queste interazioni, offrendo una rappresentazione più accurata e dinamica dei comportamenti collettivi all'interno delle simulazioni pedonali.

Uno sviluppo futuro del progetto può riguardare l'automazione dei vari processi. Al momento, vi sono diverse configurazioni e file necessari per eseguire il progetto, il che porta al beneficio di un'alta personalizzazione delle fasi di training e testing, ma al tempo stesso rischia di creare discrepanze

all'interno dell'applicazione. Si potrebbe pensare di unificare alcuni di questi processi e implementare parser più efficienti per garantire un coordinamento ottimale tra i file. Questo migliorerebbe la coerenza del sistema, ridurrebbe il rischio di errori dovuti a configurazioni incoerenti e renderebbe il processo complessivo più snello e intuitivo, facilitando l'uso e la manutenzione del sistema.

Un ulteriore sviluppo del progetto può consistere nell'aggiunta di nuove zone o oggetti nell'ambiente simulato. Ad esempio, si potrebbero includere zone di attesa, zone in cui fare la fila, aree di maggiore interesse, oppure elementi come tornelli, macchinette per i biglietti, sedie, scale mobili, ascensori, ecc. Le possibilità di modifica sono infinite e possono permettere di modellare scenari più realistici, aumentando così la complessità e la validità delle simulazioni.

In conclusione, questi sviluppi futuri potrebbero significativamente migliorare la qualità e la versatilità del simulatore di pedoni e folle, rendendolo uno strumento ancora più potente per lo studio dei comportamenti pedonali e la progettazione di spazi pubblici efficienti.

Bibliografia

- [1] Armin Seyfried Antoine Tordeux Mohcine Chraibi e Andreas Schadschneider. «Prediction of pedestrian dynamics in complex architectures with artificial neural networks». In: *Journal of Intelligent Transportation Systems* 24.6 (2020), pp. 556–568. DOI: 10.1080/15472450.2019.1621756. eprint: <https://doi.org/10.1080/15472450.2019.1621756>. URL: <https://doi.org/10.1080/15472450.2019.1621756> (cit. a p. 3).
- [2] Stefania Bandini, Sara Manzoni e Giuseppe Vizzari. «Agent Based Modeling and Simulation: An Informatics Perspective». In: *Journal of Artificial Societies and Social Simulation* 12.4 (2009), p. 4 (cit. a p. 3).
- [3] Stefania Bandini et al. «Distance-based affective states in cellular automata pedestrian simulation». In: *Nat. Comput.* 23.1 (2024), pp. 71–83. DOI: 10.1007/S11047-023-09957-Y. URL: <https://doi.org/10.1007/s11047-023-09957-y> (cit. a p. 3).
- [4] Yoshua Bengio et al. «Curriculum Learning». In: 2016, pp. 41–48. URL: <https://dl.acm.org/doi/10.1145/1553374.1553380> (cit. a p. 12).
- [5] C Burstedde et al. «Simulation of pedestrian dynamics using a two-dimensional cellular automaton». In: *Physica A: Statistical Mechanics and its Applications* 295.3 (giu. 2001). ISSN: 03784371. DOI: 10.1016/S0378-4371(01)00141-8. URL: <https://www.sciencedirect.com/science/article/pii/S0378437101001418?via%3Dihub> (cit. a p. 3).
- [6] Felipe Leno Da Silva e Anna Helena Reali Costa. «A survey on transfer learning for multiagent reinforcement learning systems». In: *Journal of Artificial Intelligence Research* 64 (2019), pp. 645–703. ISSN: 10769757. DOI: 10.1613/jair.1.11396 (cit. a p. 4).
- [7] Stephen J. Guy, Ming C. Lin e Dinesh Manocha. «Modeling collision avoidance behavior for virtual humans». In: *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3*. A cura di Wiebe van der Hoek et al. IFAAMAS, 2010, pp. 575–582. URL: <https://dl.acm.org/citation.cfm?id=1838182> (cit. a p. 3).
- [8] Robert Junges e Franziska Klügl. «Programming Agent Behavior by Learning in Simulation Models». In: *Applied Artificial Intelligence* 26.4 (2012), pp. 349–375 (cit. a p. 3).
- [9] Raphael Korbacher, Huu-Tu Dang e Antoine Tordeux. «Predicting pedestrian trajectories at different densities: A multi-criteria empirical analysis». In: *Physica A: Statistical Mechanics and its Applications* 634 (2024), p. 129440. ISSN: 0378-4371. DOI: <https://doi.org/10.1016/j.physa.2023.129440>. URL: <https://www.sciencedirect.com/science/article/pii/S0378437123009950> (cit. a p. 3).

- [10] Francisco Martinez-Gil, Miguel Lozano e Fernando Fernández. «Emergent behaviors and scalability for multi-agent reinforcement learning-based pedestrian models». In: *Simulation Modelling Practice and Theory* 74 (mag. 2017). Publisher: Elsevier B.V., pp. 117–133. ISSN: 1569190X. DOI: 10.1016/j.smpat.2017.03.003. URL: <https://linkinghub.elsevier.com/retrieve/pii/S1569190X17300503> (cit. a p. 3).
- [11] Miguel Morales. *Grokking Deep Reinforcement Learning*. ISSN: 1098-6596 _eprint: arXiv:1011.1669v3. Manning Publications, 2020. ISBN: 978-85-7811-079-6 (cit. alle pp. 10, 11).
- [12] Sébastien Paris e Stéphane Donikian. «Activity-Driven Populace: A Cognitive Approach to Crowd Simulation». In: *IEEE Computer Graphics and Applications* 29.4 (2009), pp. 34–43 (cit. a p. 8).
- [13] Enrico Ronchi et al. *The Process of Verification and Validation of Building Fire Evacuation Models*. Gaithersburg, MD: National Institute of Standards e Technology, nov. 2013, pp. 1–85. DOI: 10.6028/NIST.TN.1822. URL: <https://nvlpubs.nist.gov/nistpubs/TechnicalNotes/NIST.TN.1822.pdf> (cit. a p. 14).
- [14] Stuart Russell e Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3^a ed. Prentice Hall, 2010 (cit. alle pp. 4–7).
- [15] Andreas Schadschneider et al. «Evacuation Dynamics: Empirical Results, Modeling and Applications». In: *Encyclopedia of Complexity and Systems Science*. A cura di Robert A. Meyers. New York, NY: Springer, 2009, pp. 3142–3176. ISBN: 978-0-387-30440-3. DOI: 10.1007/978-0-387-30440-3_187. URL: https://doi.org/10.1007/978-0-387-30440-3_187 (visitato il 16/04/2024) (cit. a p. 3).
- [16] Tobias Schrödter e The PedPy Development Team. *PedPy - Pedestrian Trajectory Analyzer*. Ver. 1.0.2. URL: <https://github.com/PedestrianDynamics/pedpy> (cit. a p. 15).
- [17] John Schulman et al. *Proximal policy optimization algorithms*. ISSN: 23318422 Publication Title: arXiv _eprint: 1707.06347. Lug. 2017. URL: <https://arxiv.org/abs/1707.06347v2> (cit. a p. 13).
- [18] Felipe Leno Da Silva e Anna Helena Reali Costa. «A survey on transfer learning for multiagent reinforcement learning systems». In: *Journal of Artificial Intelligence Research* 64 (2019), pp. 645–703. ISSN: 10769757. DOI: 10.1613/jair.1.11396 (cit. a p. 12).
- [19] Matthew E. Taylor e Peter Stone. «Transfer Learning for Reinforcement Learning Domains: A Survey». In: *Journal of Machine Learning Research* 10 (2009), pp. 1633–1685. ISSN: 18674542. DOI: 10.1007/978-3-642-27645-3_2 (cit. a p. 12).
- [20] Lisa Torrey e Jude Shavlik. «Transfer Learning». In: *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global, 2010, pp. 242–264. ISBN: 978-1-60566-766-9. DOI: 10.4018/978-1-60566-766-9.ch011. URL: [https://www.igi-global.com/chapter/transfer-learning/36988](https://www.igi-global.com/chapter/transfer-learning/www.igi-global.com/chapter/transfer-learning/36988) (visitato il 26/06/2024) (cit. a p. 12).

- [21] Giuseppe Vizzari, Daniela Briola e Federico Pisapia. «Curriculum-Based RL for Pedestrian Simulation: Sensitivity Analysis and Hyperparameter Exploration». In: *Prestigious Applications of Intelligent Systems (PAIS-2024)*. 2024 (submitted) (cit. a p. 16).

Ringraziamenti

In primo luogo, un sentito ringraziamento va al mio relatore, il Professor Giuseppe Vizzari. La sua guida esperta, la sua disponibilità e i suoi preziosi consigli sono stati fondamentali per la stesura di questo lavoro. Grazie per la pazienza e per avermi incoraggiato a dare sempre il massimo.

Un ringraziamento speciale va ad Andrea, il mio compagno di corso. Insieme abbiamo condiviso gli ultimi sei mesi e, nonostante alcune divergenze, insieme abbiamo raggiunto questo importante traguardo. Grazie per avermi supportato e sopportato incondizionatamente, senza di te questo percorso sarebbe stato molto più arduo e monotono.

Desidero esprimere un ringraziamento speciale alla mia ragazza, Chiara. Il tuo supporto incondizionato, la tua pazienza e la tua costante presenza sono stati fondamentali per me. Grazie per avermi incoraggiato e sostenuto con affetto durante questo percorso.

Non posso non ringraziare la mia famiglia, che mi ha sempre supportato in ogni fase del mio percorso accademico. La vostra fiducia, il vostro amore e il vostro incoraggiamento sono stati per me una fonte inesauribile di forza e motivazione. Senza di voi, nulla di questo sarebbe stato possibile.

Infine, un grazie di cuore va a tutti i miei cari amici. La vostra vicinanza, i momenti di svago e il vostro supporto mi hanno aiutato a superare le difficoltà e a mantenere un equilibrio tra studio e vita personale.

A tutti voi, va la mia più sincera gratitudine.