Università degli Studi di Napoli

Federico II

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

Scuola Politecnica e delle Scienze di Base

Corso di Laurea Triennale in Ingegneria Informatica

Approccio con agente DDPG per convoglio di treni ad alta velocità

Relatore:

Prof.ssa Santini Stefania

Correlatore:

Ing. Basile Giacomo

Candidato:

Fazzari Daniele

Matr. N46004470

Anno Accademico

2020/2021

Indice

Introduzione

Il settore dell'automazione sta riscuotendo un interesse sempre maggiore da parte della ricerca con conseguente incremento dell'utilizzo di sistemi di controllo, i quali consentono di ridurre la necessità dell'intervento umano nella gestione di macchine e processi, portando vantaggi in termini di affidabilità, sicurezza ed efficienza (oltre che comodità). In particolare un notevole incremento si sta riscontrando per quanto riguarda gli investimenti nel campo della guida autonoma: difatti nel settore automotive il tema della safety è particolarmente delicato a causa della frequenza degli incidenti stradali, alcuni dei quali portano anche alla perdita di vite umane, ed è dunque di notevole interesse l'introduzione di veicoli automatici in grado di soddisfare le principali capacità di trasporto dei veicoli tradizionali. Chiaramente la convergenza di investimenti non è giustificata unicamente dalla volontà di ridurre il numero di incidenti, bensì anche da ulteriori vantaggi come riduzione della congestione stradale, superamento parziale del problema dei parcheggi (il passeggero potrebbe essere portato nel luogo richiesto e successivamente il veicolo si occuperebbe di trovare posteggio autonomamente) e riduzione dei tempi di viaggio.

Nel caso dei trasporti ferroviari la ricerca in tale argomento è stata spinta dalla necessità pratica di aumento della capacità delle linee e di interoperabilità tra i sistemi di circolazione e sicurezza delle varie ferrovie europee. infatti svariate motivazioni, come ad esempio le preoccupazioni ambientali (in particolare il riscaldamento globale), stanno spingendo sempre più i governi a ridurre le emissioni

di gas serra e il consumo di combustibili fossili nel settore dei trasporti, ed una soluzione attualmente discussa è lo spostamento del volume dei trasporti dalla strada alla ferrovia, in quanto i treni sono considerati i mezzi di trasporto più efficienti dal punto di vista energetico ed ecologico. Dunque, considerando anche l'aumento della popolazione nelle città e quindi della domanda di trasporto, si ha la necessità di aumentare la capacità dei trasporti ferroviari che operano, ad oggi, in condizioni di quasi saturazione. Tale problematica può essere affrontata grazie ad esempio alla possibilità di accoppiare virtualmente i treni al fine di realizzare convogli virtuali per aumentare l'efficienza del trasporto. Difatti Shift2Rail, l'iniziativa istituita nel 2014 per gestire e coordinare tutte le attività di ricerca e innovazione incentrate sulle ferrovie, finanziate dall'iniziativa europea Horizon 2020, ha fissato tra gli obiettivi quello di migliorare il comfort dei passeggeri riducendo i tempi di attesa (con benefici quindi anche in termini economici) grazie ad una gestione del traffico che ottimizzi lo sfruttamento dell'infrastruttura.

Il Virtual Coupling (VC) rappresenta dunque una grande innovazione che tenta di riportare il concetto di "platoon", nato in campo automotive, anche in ambito ferroviario. Ad oggi praticamente tutte le auto di nuova generazione dispongono infatti di sistemi ADAS (Advanced Driver Assistance System), tra cui il più diffuso è l'ACC (Adaptive Cruise Control), ovvero il controllo di crociera adattativo che consente di adattare la velocità di un veicolo sulla base della distanza con il veicolo che lo precede. Dunque, grazie al progresso tecnologico e al crescente interesse sia da parte degli investitori che dei produttori, tale settore è in forte espansione, ma l'adozione e la successiva diffusione dei veicoli autonomi pone inevitabilmente delle sfide da affrontare. Ad esempio il VC in ambito ferroviario richiede non solo la comunicazione tra i singoli treni del covoglio bensì anche quella tra il treno e l'infrastruttura fisica, e tale scambio di informazioni avviene tipicamente tramite comunicazione radio GSM-R: è una tecnlogia che tuttavia potrebbe risultare pericolosa in termini di copertura del segnale e latenza di trasmissione, per tale motivo

si prevede la migrazione a nuove tecnologie di comunicazione come LTE o 5G. In aggiunta l'adozione del VC nel mondo reale deve assicurare la compatibilità con l'ERTMS e inevitabilmente sviluppi tecnologici avanzati per la valutazione delle prestazioni, l'analisi dei pericoli e della sicurezza, non semplici da realizzare. Non solo: un ulteriore passo in avanti è dotare il sistema di controllo di una capacità di apprendimento automatico, utilizzando tecniche di intelligenza artificiale, in modo tale che sia in grado di prendere decisioni valide in autonomia.

Il seguente lavoro di tesi ha come scopo l'analisi di tre treni in configurazione leader-follower che realizzano un convoglio tramite virtual coupling sfruttando un controllo mediante agente di reinforcement learning appositamente addestrato, il quale offre tra i suoi principali vantaggi quello di essere in grado di adattarsi a diversi contesti applicativi senza dover essere modificato. Il progetto si fonda su un controllo ACC implementato da MathWorks, adattato poi al contesto in esame. Durante la trattazione verranno innanzitutto analizzati gli algoritmi di apprendimento più popolari fino a quello impiegato per l'agente utilizzato (ovvero il DDPG), successivamente verrà analizzato l'intero scenario realizzato in ambiente Matlab/Simulink mostrando il processo di creazione dell'ambiente e dell'agente con conseguente validazione simulativa di quest'ultimo in diversi scenari. Dai risultati ottenuti si vuole confermare l'efficacia del sistema di controllo che deve dunque consentire di realizzare un convoglio di treni ad alta velocità garantendo i vincoli imposti dal problema (come il mantenimento di una opportuna distanza tra i treni del convoglio). E' tuttavia da specificare che l'intero lavoro è stato pensato per un contesto semplificato che nella pratica richiederebbe ulteriori analisi e approfondimenti.

Capitolo 1

Virtual Coupling

1.1 Virtual Coupling: l'idea

L'industria ferroviaria è nella direzione di adottare sistemi di signalling avanzati che riducano la distanza tra i treni incrementando affidabilità e sicurezza. Oggi esiste un processo manuale per i treni che consente di accoppiarli meccanicamente tramite un "coupler". Questo processo aiuta a mantenere distanze minime tra i treni stessi, ma per coloro che non possono essere accoppiati con coupler c'è bisogno di mantenere distanze maggiori. Il Virtual Coupling (VC) è una strategia di controllo di formazione che si propone di ridurre la distanza tra treni adiacenti che non possono essere accoppiati meccanicamente [11].

I set di treni accoppiati (*Virtually–Coupled Train Sets, VCTS*) sono un approccio che potrebbe aumentare la capacità della rete ferroviaria senza ingenti investimenti nell'infrastruttura di rete e senza abbassare le norme di sicurezza dell'industria ferroviaria [17]. Nei VCTS i treni non sono collegati fisicamente, bensì usano le informazioni sullo stato dinamico (posizione, velocità e accelerazione) di altri treni nel set per controllare la distanza tra loro, in modo simile al *plotoon* dei veicoli stradali. Con queste caratteristiche il flusso del treno può essere ottimizzato per migliorare la capacità della rete, la puntualità del treno, la flessibilità opera-

tiva e potenzialmente ridurre i costi di manutenzione. Dunque Il virtual coupling aggiunge ai sistemi di controllo automatico dei veicoli ferroviari (*Automatic Train Control*) la funzionalità di poter connettere virtualmente due o più treni tra loro.

La guida cooperativa è un componente essenziale dei sistemi di trasporto intelligenti del futuro, essa garantisce infatti numerosi vantaggi come:

- incremento della safety (per ad esempio riduzione degli incidenti dovuti a distrazione del conducente).
- aumento dell'utilizzazione delle infrastrutture (quindi riduzione della congestione del traffico).

Il VC è strettamente correlato alla guida cooperativa in quanto mira a trasferire al settore ferroviario alcune conquiste e risultati della ricerca automobilistica nel platooning dei veicoli sulle autostrade intelligenti. La maggior parte dei sistemi ATC sono driverless, ma alcuni dipendono ancora dal conducente, il quale riceve ed imposta informazioni come lo speed-profile da seguire, mentre la safety è garantita dal sistema. L'obiettivo del VC è realizzare un tracking di riferimento per far si che il treno follower sia in grado di seguire il treno antecedente garantendo il soddisfacimento delle specifiche di controllo e sicurezza richieste dalle norme Europee [4].

1.2 ERTMS/ETCS

ERTMS/ETCS (European Railway Traffic Management System / European Train Control System) è un set di standard per la gestione e l'interoperabilità delle ferrovie moderne. Esso nasce dall'esigenza di interoperabilità tra i diversi paesi Europei (sia in termini di sistemi di segnalamento che di regolamento, fino alla modalità di marcia stessa dei treni). La sua introduzione porta a notevoli vantaggi

in termini di risparmio sui costi di manutenzione, sicurezza, affidabilità, puntualità e capacità di traffico [18].

1.2.1 Struttura ERTMS

L'ERTMS è costituito da due sistemi :

- ETCS ("European Train Control System"). Si tratta di uno standard Europeo di controllo operativo della marcia dei treni basato su una serie di apparecchiature di bordo atte alla supervisione dei movimenti del treno, in grado di arrestarlo sulla base della velocità consentita in ciascuna sezione della linea, insime al calcolo e alla supervisione della velocità massima del treno.

- GSM-R ("Global System for Mobile Communications – Railways"), standard europeo di comunicazione radio per le operazioni ferroviarie. Esso è basato sulla tecnologia radio GSM utilizzando però bande di frequenza esclusive per permettere la comunicazione fra il treno e l'infrastruttura.

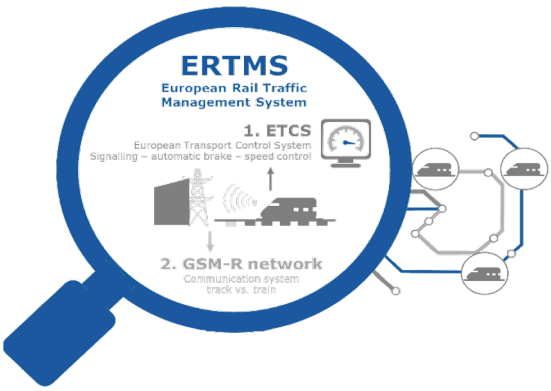


Figura 1.1: ERTMS

ERTMS/ETCS è suddiviso in tre livelli sulla base delle attrezzature impiegate e delle informazioni scambiate:

- **Livello 1**: prevede la supervisione continua del movimento del treno ed una comunicazione non continua tra il treno ed il binario (normalmente mediante le euro-balise). Sono necessari segnali lungo la linea e la rilevazione del treno è eseguita dalle apparecchiature di terra fuori dal campo di applicazione dell'ERTMS. Dunque l'apparecchiatura di posizione invia la posizione del treno al centro di controllo. Il centro di controllo, che riceve la posizione di tutti i treni che circolano sulla linea, determina la nuova autorità di movimento e la invia alla balise. Il treno passa sopra la balise e riceve la nuova autorità di movimento e traccia i dati. Il computer di bordo determina quindi il profilo di velocità dall'autorità di movimento e dal successivo punto di frenata.

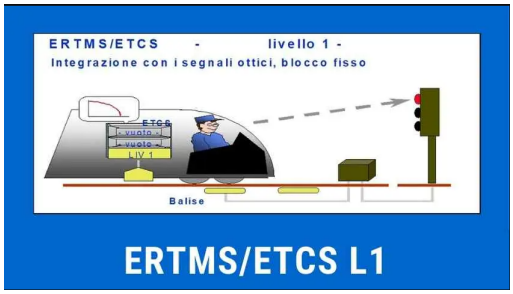


Figura 1.2: ERTMS Lv1

- **Livello 2** : prevede la supervisione del movimento del treno con una comunicazione continua, fornita da GSM-R, tra il treno e il binario, in particolare il computer di bordo consente la comunicazione con il Radio Block Center (RBC). I segnali lungo la linea sono facoltativi in questo caso e il rilevamento del treno viene eseguito dall'apparecchiatura di terra. Per cui l'apparecchiatura di rilevamento invia la posizione del treno al centro di controllo, il quale riceve la posizione da tutti i treni che circolano sulla linea, determina la nuova autorità di movimento e la trasmette continuamente via GSM-R al treno. Il computer di bordo calcola quindi il suo profilo di velocità (informazioni visualizzate dal conducente).

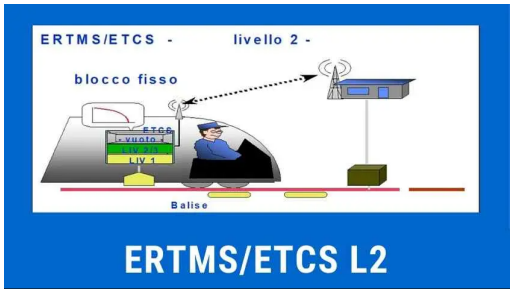


Figura 1.3: ERTMS Lv2

- **Livello 3**: Fornisce una supervisione continua del treno con comunicazione tra il treno e il binario. La differenza principale con il livello 2 è che la posizione e l'integrità del treno sono gestite nell'ambito del sistema ERTMS, vale a dire che non sono necessari segnali a bordo linea o sistemi di rilevamento del treno a bordo pista diversi dalle euro-balise. Ancora una volta è necessario un sistema di bordo che consenta la comunicazione con il centro di controllo. Dunque il treno che passa sopra la balise riceve una nuova indicazione di posizione. Il computer di bordo determina la posizione del treno e verifica se la velocità attuale è corretta per la distanza percorsa. Il treno invia la sua posizione all'RBC, che riceve la posizione di tutti i treni in circolazione, determina la nuova autorità di movimento e la trasmette al treno. Il computer di bordo calcola quindi il profilo di velocità e il successivo punto di frenata. Il percorso non viene quindi ripartito in sezioni di binari fissi (successivamente sarà approfondito tale concetto).

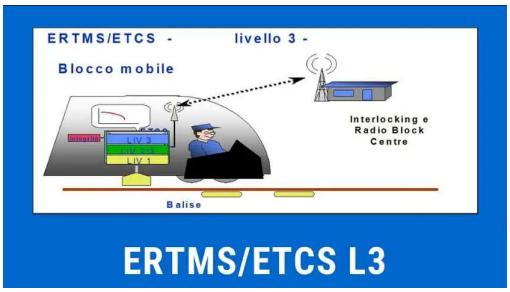


Figura 1.4: ERTMS Lv3

- **Livello 0**: pensato per i treni dotati di ETCS che circolano lungo linee non attrezzate.

Attualmente sulle linee ferroviarie Europee sono implementati ERTMS/ETCS di Livello 1 o Livello 2, mentre la linea di base di Livello 3 è ancora oggetto di studi sperimentali [18].

1.3 Modalità di funzionamento e tecnologie utilizzate

Dal punto di vista delle manovre, il VC può essere suddiviso in tre processi: avvicinamento, guida praticamente accoppiata, e separazione [17]. Per mancanza di connessione fisica, un treno in VC può accoppiarsi o disaccoppiarsi facilmente dai set, anche durante il viaggio. Con queste caratteristiche il flusso del treno può essere ottimizzato per migliorare la capacità della rete, la puntualità del treno e la flessibilità operativa [20].

Il meccanismo di accoppiamento virtuale si basa su due livelli funzionali. In primo luogo, è richiesto un affidabile sistema di comunicazione wireless per comandare e monitorare tutte le parti virtualmente accoppiate dalla cabina principale, oltre a quello per lo scambio di informazioni tra i membri del platoon. In secondo luogo, l'accoppiamento meccanico è sostituito da un meccanismo di sicurezza per garantire una certa distanza tra i treni del convoglio.

Distinzione fondamentale è tra le due diverse tecniche: "Fixed Block" e "Moving Block" (figura 1.5). La prima concepisce la linea come suddivisa in sezioni di blocco fisse, ognuna di una certa lunghezza, e ciascuna delle quali può essere occupata, in un certo momento, da un singolo treno in transito. La seconda, impiegata a partire dal livello 3, invece, fa riferimento al concetto di distanza di sicurezza tra

i treni sulla base della frenata assoluta (statico) o tale da consentire una distanza tra i treni che garantisca di fermarsi in sicurezza (dinamico) [4].

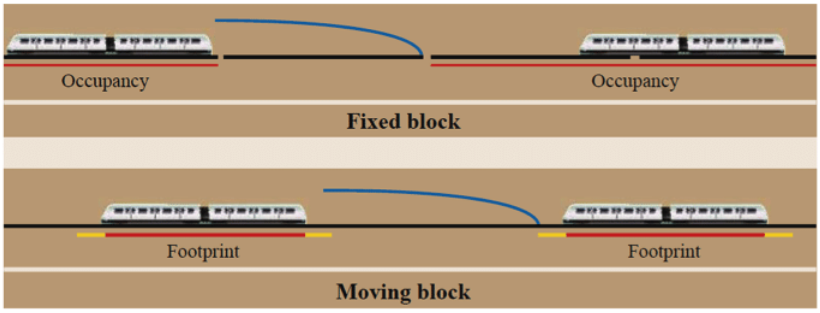


Figura 1.5

Dal punto di vista del controllo, il VC potrebbe teoricamente funzionare senza alcuna comunicazione tra treni, in modo simile al sistema ACC sviluppato per le automobili. In questo caso, i sensori vengono utilizzati per determinare lo stato dinamico relativo (posizione relativa o distanza, velocità relativa e accelerazione relativa) in relazione al veicolo davanti, e controllare la distanza tra i treni regolando la propria accelerazione e velocità. La gamma di sensori commerciali relativi è normalmente alla distanza di 10–250 m, e può essere influenzata da diversi fattori come il tempo, la luce e gli ostacoli sul lato della pista che bloccano la linea di vista (LoS) nelle curve. Tuttavia per velocità elevate non esiste una distanza possibile che consenta ai treni di essere abbastanza vicini da consentire ai sensori relativi di rilevarsi a vicenda e allo stesso tempo abbastanza lontano da reagire nel tempo. In alternativa all'utilizzo dei soli sensori relativi, lo stato dinamico assoluto di ciascun treno (posizione assoluta, velocità e accelerazione) potrebbe essere misurato utilizzando sensori di bordo (o assoluti): i dati misurati vengono scambiati tra i treni tramite un adeguato collegamento di comunicazione. La comunicazione Train to Infrastructure (T2I) ha un raggio illimitato ma dipende dalla disponibilità del segnale, che limita VCTS a posizioni con una buona copertura. Anche la comunicazione Train to Train (T2T) presenta una gamma limitata, sebbene possa essere significativamente più ampia della gamma tipica di sensori

relativi. Inoltre, la comunicazione è generalmente più tollerante alle condizioni meteorologiche e alla mancanza di LoS, rispetto ai sensori relativi. Tuttavia, VC-TS che utilizzano solo sensori assoluti hanno i propri limiti: è difficile misurare con precisione la posizione e la velocità assolute del treno utilizzando contachilometri e tachimetri (ad esempio a causa dello slittamento delle ruote) e misurare la posizione integrando il segnale di velocità o integrando due volte l'accelerazione aggiunge errori. I sistemi di posizionamento basati sul segnale dipendono dalla disponibilità del segnale stesso e potrebbero non essere disponibili in tunnel o aree remote. L'unione di diversi sensori mediante algoritmi di fusione dei sensori può migliorare le misurazioni e l'uso di più sensori fornisce ridondanza, migliorando l'affidabilità.

Dunque Il VC richiede il Livello 3 di ERTMS, e la differenza con il livello 2 consiste nel modo in cui si calcolano le *Movement Authorities* (Autorità di movimento). In particolare il componente responsabile per la separazione tra i treni è un controllore denominato RBC (*Radio Block Cente*r): esso trasmette con continuità a ciascun treno, via radio GSM-R, la velocità e la distanza da rispettare in funzione della posizione dei treni presenti sulla linea e dei vincoli imposti dal percorso. Nello stesso tempo i treni inviano, sempre via radio, la loro posizione al blocco centrale grazie alla comunicazione treno-infrastruttura (T2I). Le moderne aree di ricerca in futuro prevedono sicuramente la migrazione dal GSM-R a nuove tecnologie di comunicazione come LTE o 5G. Tipicamente si utilizzano balise fisse per la localizzazione dei vari treni anche se ad oggi sono in atto sforzi per l'utilizzo in congiunzione con sistemi satellitari. ERTMS Lv3 non è ancora sufficiente per garantire un tracking completo, nel caso di perdita di informazioni infatti potrebbe lavorare in condizioni non sicure, dunque è possibile immaginare un'implementazione del VC come un sistema che offra le stesse funzionalità del Lv3 di ETCS con un livello aggiuntivo di comunicazione treno-treno (T2T) per lo scambio di opportune informazioni. Nel VC la comunicazione T2T efficiente diventa essenziale per

massimizzare la probabilità di consegna dei messaggi, infatti i VCTS richiedono tempi di reazione piccoli e quindi una latenza ridotta per la sincronizzazione multi-treno. Attraverso la combinazione dei dati trasmessi dall'RBC e dal livello T2T, ciascun treno aggiorna la propria autorità di movimento contentente la distanza di sicurezza, la velocità, l'accelerazione e ulteriori informazioni di interesse.

1.4 Topologia della rete di comunicazione

E' possibile immaginare diverse topologie:

- **Fully connected**: Il RBC comunica con l'intero convoglio di treni ed ogni treno (a turno) inoltra alcune informazioni ai suoi vicini.
- **Chain like**: Il RBC comunica col primo treno del convoglio, il quale inoltra il messaggio al treno seguente come in un routing multi-hop.

Si sottolinea che Il leader del convoglio può essere sia fisico, cioè un vero e proprio treno situato in testa al plotone (come per le automobili) che virtuale, cioè simulato dal RBC che invia i dati relativi allo stato in cui dovrebbe essere un treno con comportamento ideale.

1.5 Full Supervision plus Virtual Coupling

La modalità di Supervisione Completa (*Full supervision*, FS) è possibile qualora tutti i treni e dati di tracciamento necessari al supervisionamento sono disponibili a bordo. L'implementazione del VC deve assicurare la compatibilità con l'ERTMS aggiungendo una nuova modalità operativa, al di sopra della Full Supervision, denominata *Full Supervision plus Virtual Coupling* (FSVC). Lo switch da FS al FSVC è pilotato da messaggi ERTMS inviati dall'RBC che orchestra il coupling del platoon mentre i treni sono ancora in corsa (da notare che si considera uno scenario quasi-stazionario in contesti non critici al fine di valutare l'accoppiamento) [4].

Capitolo 2

Approccio con agente DDPG

2.1 Controllo con AI

L'utilizzo dell'intelligenza artificiale nella progettazione ed implementazione di sistemi di controllo è una pratica sempre più diffusa e ricercata dalle aziende del settore. La guida autonoma sta dunque trasformando il concetto di mobilità riducendo sempre più l'interazione con il conducente umano, fino ad arrivare anche ad eliminare completamente tale interazione, portando a notevoli vantaggi sotto svariati aspetti (sicurezza ed efficienza ne sono un esempio). Ciò è chiaramente possibile grazie all'utilizzo di una combinazione di sensori i quali sono atti al monitoraggio delle condizioni di guida: dotando inoltre il controllo di una capacità di apprendimento (grazie all'approccio con AI) esso è in grado di prendere decisioni autonome e valide sulla base delle informazioni ricevute dall'ambiente stesso e dunque di adattarsi a diversi contesti applicativi.

2.2 Reinforcement Learning

Il **reinforcement learning** (apprendimento per rinforzo) è una tipologia di machine learning (tecnica di apprendimento automatico) che si concentra su come

un agente potrebbe operare in un ambiente al fine di massimizzare una data metrica di ricompensa senza l'intervento di un operatore umano. Si tratta di una tecnica in cui un agente impara a svolgere un'attività tramite ripetute interazioni di tipo "trial-and-error" (eseguite per tentativi ed errori) con un ambiente dinamico, e le cui performance vengono valutate tramite una funzione di ricompensa. Questo approccio all'apprendimento consente all'agente di adottare una serie di decisioni in grado di massimizzare una metrica di ricompensa per l'attività, senza essere esplicitamente programmato per tale operazione e senza l'intervento dell'uomo. In particolare, a differenza del machine learning con e senza supervisione, il reinforcement learning non si affida a un set di dati statici, ma opera in un ambiente dinamico e apprende dalle esperienze raccolte durante l'addestramento. Non risulta dunque necessario raccogliere, pre-elaborare ed etichettare i dati prima dell'addestramento. In pratica un modello di reinforcement learning è in grado di iniziare ad apprendere un comportamento autonomamente, senza supervisione (dell'uomo). I problemi complessi di reinforcement learning spesso si basano su reti neurali profonde, dando luogo a ciò che viene chiamato deep reinforcement learning [13].

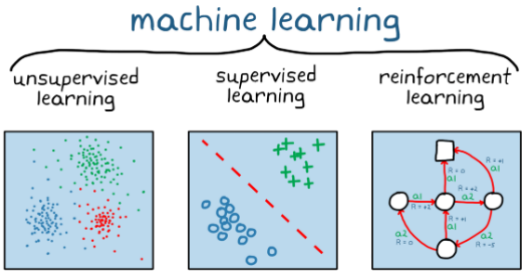


Figura 2.1: Tre categorie di machine learning

Le reti neurali profonde addestrate con il reinforcement learning sono in grado di codificare comportamenti complessi. Ciò consente di adottare un approccio alternativo per le applicazioni che, diversamente, sarebbero intrattabili o più difficili da affrontare con metodi più tradizionali. Ne sono un esempio i controlli avanzati:

il controllo dei sistemi non lineari è un problema abbastanza complesso che spesso viene trattato linearizzando il sistema in corrispondenza di diversi punti operativi, mentre il reinforcement learning può essere direttamente applicato al sistema non lineare.

Le entità da considerare sono diverse. L'agente (*agent*), il decisore che interagisce con l'ambiente in cui è collocato, con interazioni che si verificano in sequenza nel tempo. Ad ogni passo temporale l'agente ottiene una rappresentazione dello stato (*state*) in cui l'ambiente si trova, e a partire da tale rappresentazione poi intraprende una azione (*action*), la cui esecuzione fa transitare l'ambiente in un nuovo stato. L'agente riceve una ricompensa (*reward*) sulla base della azione che ha intrapreso nello stato precedente con l'obiettivo (*goal*) di massimizzare la quantità di ricompense (*rewards*) che riceve intraprendendo determinate azioni in determinati stati dell'ambiente, ovvero si preoccupa di massimizzare la ricompensa cumulativa (*cumulative reward*) piuttosto che quella immediata (ovvero la reward massima che può ottenere dal passaggio allo stato successivo) [10]. Dunque il *return* (ovvero il guadagno complessivo dell'agente) si può definire come la somma di tutte le ricompense che otterrà in futuro (con T tempo allo step finale), ovvero:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T$$

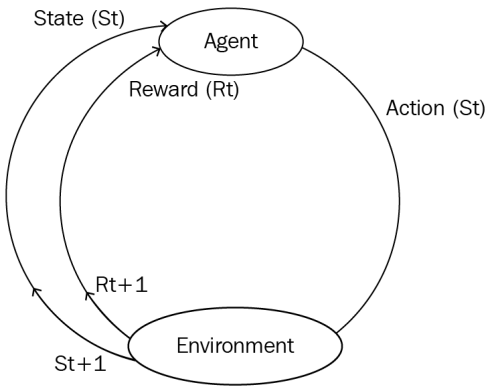


Figura 2.2: Interazione agente-ambiente (agent-environment)

L'obiettivo dell'agente è quello di massimizzare il ritorno atteso (*expected return*), è infatti ciò a guidarlo nella scelta delle azioni da intraprendere. Notiamo che è possibile definire un tempo T finale quando l'interazione tra l'agente e l'ambiente può essere suddivisa in sottosequenze chiamate episodi, ma qualora ciò non fosse possibile e l'interazione continui senza interruzione si tratterebbe di *continuos tasks* e risulta necessario ridefinire il guadagno G_t introducento il concetto di discount: L'obiettivo dell'agente diventa dunque massimizzare lo sconto atteso delle ricompense (*expected discount return*) [8].

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Questa uguaglianza mostra che l'agente dà maggior peso alla ricompensa più immediata rispetto a quelle future (le quali verranno maggiormente scontate, essendo il *fattore di sconto* γ un valore compreso tra 0 ed 1).

Il *reinforcement learning agent* (RL agent) si costituisce di due parti: *policy* e algoritmo di apprendimento. L'agente segue una policy, la quale definisce il modo di comportarsi dell'agente in un determinato momento, ovvero è una mappatura dagli stati percepiti dell'ambiente alle azioni da intraprendere in tali stati. All'istante temporale t, seguendo la policy π, la probabilità di selezionare l'azione "a" nello stato "s" è $\pi(a|s)$. Per la policy π la *State-value function* ci da il valore di uno stato, il quale è pari all'expected return partendo dallo stato "s" al tempo t, seguendo la politica π.

$$v_\pi(s) = E_\pi[G_t|S_t = s] = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s]$$

Associata alla policy invece definiamo l'*Action-value function* (q_π), la quale indica la bontà di un agente nel selezionare un'azione partendo da un dato stato "s" seguendo la policy π, e fornisce il valore dell'azione pari all'expected return

partendo dallo stato "s" al tempo t intraprendendo l'azione "a" (seguendo la policy π).

$$q_\pi(s,a) = E_\pi[G_t \mid S_t = s, A_t = a] \\ = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right].$$

Figura 2.3

Tale funzione (figura 2.3) è indicata come *Q-function* e l'output per ogni coppia stato-azione è detto *Q-value* [6]. L'algoritmo di apprendimento invece si occupa di aggiornare i parametri della policy sulla base delle azioni intraprese dall'agente, ricompense ed osservazioni. Esempi di algoritmi popolari sono le *deep Q-network* (DQN) e i *deep deterministic policy gradient* (DDPG). L'obiettivo di un algoritmo di RL è quello di trovare una policy che produce la maggior quantità di rewards, ovvero cercare tra tutte le policy quella che permette all'agente di ricevere il massimo return: questa viene definita una policy *ottima* a cui si associata una *action-value function* ottima (figura 2.4).

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right]$$

Figura 2.4

Una delle prioprietà della action-value function ottima è che soddisfa l'*equazione di Bellman*: per ogni coppia stato-azione (s,a) al tempo "t" l'expected return, partendo da uno stato "s", intraprendendo l'azione "a" e seguendo la policy ottima, sarà pari all'expected reward che otteniamo intraprendendo l'azione "a" nello stato "s", il quale è pari ad R_{t+1} sommato al massimo expected discounted return che può essere raggiunto da una qualsiasi coppia (s',a') successiva. Fintantochè

l'agente segue una policy ottima, lo stato succesivo "s'" è lo stato da cui la miglior azione "a'" prossima può essere intrapresa al tempo t+1 [9].

2.3 Q-learning

il Q-learning è tra i più utilizzati algoritmi di reinforcement learning. E' possibile dimostrare che converge al valore ottimo purchè i valori stato-azione abbiano una rappresentazione di tipo discreta. La regola che gli agenti che lo implementano seguono per aggiornare i propri Q-values è la seguente:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a)]$$

dove α è il *learning rate* (tasso di apprendimento) che può assumere valori compresi tra 0 ed 1. Tale algoritmo si occupa di aggiornare iterativamente il Q-value per ogni coppia (s,a) utilizzando l'equazione di Bellman finchè la Q-function non converge a quella ottima q^* (approccio noto come *value iteration*). Importante per la scelta delle prime azioni da intraprendere è la differenza tra *exploration* e *exploitation* (*agent dilemma*):

- l'exploration consiste nell'esplorare l'ambiente con il fine ultimo di ricavare informazioni su di esso.
- l'exploitation consiste nello sfruttare delle informazioni che già si conoscono sull'ambiente al fine di massimizzarne il guadagno.

E' dunque necessario un trade-off tra le due e ciò viene realizzato seguendo la strategia *Epsilon-Greedy*. Dunque si definisce un fattore di esplorazione ϵ il quale rappresenta la probabilità che l'agente esplorerà l'ambiente piuttosto che sfruttarlo. Man mano che l'agente esplora l'ambiente tale percentuale diminuirà e l'agente diventerà più *greedy* ("avido") nello sfruttare le informazioni che ha

ricavato. Per determinare se l'agente farà exploration o exploitation è generato un numero casuale [6].

Passi del Q-learning:

1. Inizializzazione di tutti i Q-values nella Q-table a 0 (In genere si utilizza una *Q-tabel* per memorizzare i Q-values per ogni coppia stato-azione).

2. In ogni time-step di ogni episodio: Si sceglie una azione (considerando il trade-off tra exploration ed exploitation), si osserva il reward e lo stato successivo e si aggiorna la *Q-value function*.

In casi semplici il Q-learning utilizza delle tabelle per memorizzare i Q-values (di dimensione pari al prodotto tra il numero di stati e il numero di azioni), tuttavia tale approccio perde di affidabilità al crescere del livello di complessità del sistema. Una possibile soluzione può essere l'uso di una rete neurale artificiale: La combinazione di Q-learning con una rete neurale prende il nome di *Deep-Q-Learning*, e una rete neurale(*deep neural network*) che approssima la Q-function su uno spazio di alta dimensione, è chiamata *Deep-Q-Network (DQN)* [1].

2.4 Reti neurali artificiali

Una rete neurale artificiale è un modello matematico composto da "*Neuroni*" artificiali. Esse cercano di simulare all'interno di un sistema informatico il funzionamento di sistemi nervosi biologici del cervello, il quale contiene circa 10 miliardi di neuroni , ciascuno dei quali è connesso ad altri attraverso le *sinapsi* (figura 2.5), andando a costituire un sistema parallelo di elaborazione di informazioni.

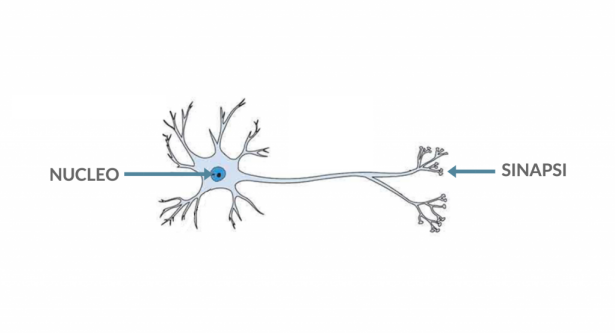


Figura 2.5: Neurone (biologico)

L'elemento di base di calcolo è chiamato nodo (o neurone). Esso riceve degli input da altre unità o dall'esterno. Ad ogni ingresso è associato un peso ω modificabile, che determina il tipo e l'ampiezza dell'effetto eccitatore o inibitore. Tale neurone calcola una funzione f della somma pesata dei suoi ingressi, $y_i = f(\sum_j(\omega_{ij}\, y_j) - \Theta_i)$, dove Θ_i è chiamata *soglia di eccitazione del neurone*, la quale rappresenta il valore che deve essere superato affinchè il neurone si attivi (nota che può essere eliminato aggiungendo un ingresso fittizio unitario chiamatao *bias*) [5].

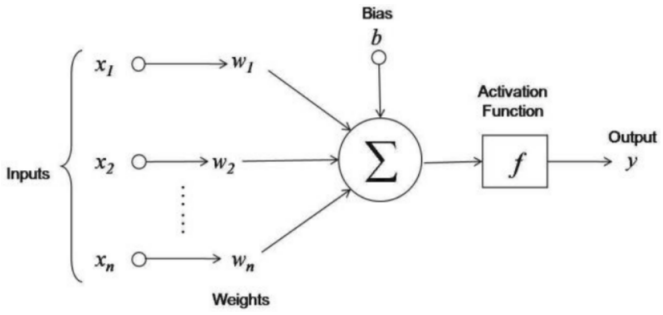


Figura 2.6: Neurone di una rete neurale

Una rete neurale si suddivide in *layers* (livelli), seguiti da una serie di funzioni di attivazione (lineari e non lineari). Gli strati che costituiscono una rete neurale, come mostrato in figura 2.7, sono almeno tre: Input layer, uno o più hidden layer (strato nascosto) e output layer.

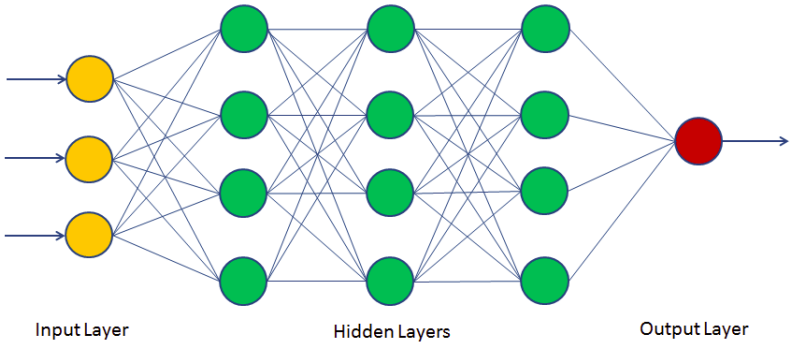


Figura 2.7: Livelli di una rete neurale

Una rete è definita "*deep*" (profonda) se contiene due o più strati nascosti. Lo scopo delle reti neurali è di essere delle *Universal Function Approximator*, ovvero essere in grado di approssimare qualsiasi funzione, e per fare questo è necessario introdurre un fattore di non linearità, da qui la funzione di attivazione. Inoltre le funzioni di attivazione non lineari sono anche tali da comprimere i valori di output in un range ben definito. Tra le più utilizzate ci sono la sigmoide, la tangente iperbolica e la *funzione ReLu*. La funzione ReLU (*Rectifier linear unit*, figura 2.8) è molto semplice da calcolare appiattendo a zero la risposta per tutti i valori negativi e lasciando tutto invariato per valori uguali o maggiori di zero: tale semplicità (unita al fatto di ridurre drasticamente il problema del *vanishing gradient*) la rende una funzione utile nei layer intermedi, dove la quantità di passaggi e di calcoli è particolarmente onerosa [19].

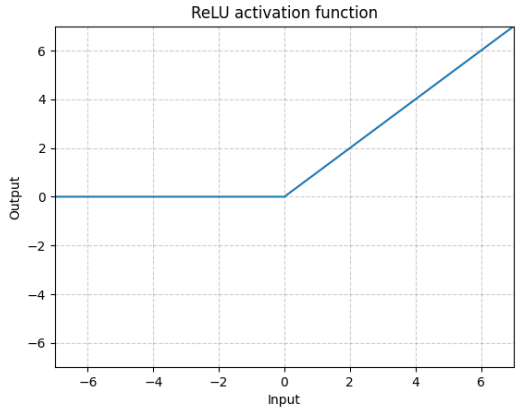


Figura 2.8: Funzione ReLu

2.4.1 Addestramento della rete neurale

L'addestramento della rete neurale si basa sull'utilizzo di un algoritmo noto come *Stocastic Gradient Descent (SGD) with Backpropagation*. Innanzitutto occorre definire la *Loss Function* (funzione di perdita):

$$J(\omega, b) = \frac{1}{2} \sum_i (t_i - y_i)^2$$

Questa da indicazioni sulla discrepanza tra il risultato ottenuto e quello atteso. Nei livelli della rete neurale ogni nodo riceve l'input dal livello precedente, il quale è una somma pesata di ciascuna connessione moltiplicata per l'output del livello precedente, questo valore è poi passato come argomento di una funzione di attivazione il cui risultato è l'output di un particolare nodo poi passato come parte di input al livello successivo. Questo processo prende il nome di *forward propagation*. Raggiunto il livello di uscita, otteniamo l'output del modello per il dato input fornito. Durante la fase di training SGD opera per minimizzare la funzione di perdita aggiornando i pesi, in direzione opposta a quella del gradiente della funzione stessa (rispetto ai pesi del modello). Difatti il gradiente indica la direzione di massimo incremento della funzione, e muovendoci in direzione opposta ci si dirige verso il minimo con l'obiettivo di ricercare il minimo globale [19].

$$\nabla J \equiv \left[\frac{\partial J}{\partial \omega 11}, \cdots, \frac{\partial J}{\partial \omega ij}\right]^T$$

Figura 2.9

Inizialmente tutti i pesi e i valori di bias vengono inizializzati casualmente, dunque nella prima iterazione la risposta della rete sarà casuale, e, con alta probabilità, completamente errata. Chiaramente deve essere possibile calcolare il

gradiente Δ della funzione di perdita J in relazione ad ogni peso ω_{ij} della rete. Esso indica come una piccola variazione influirà sul peso complessivo dell'errore J.

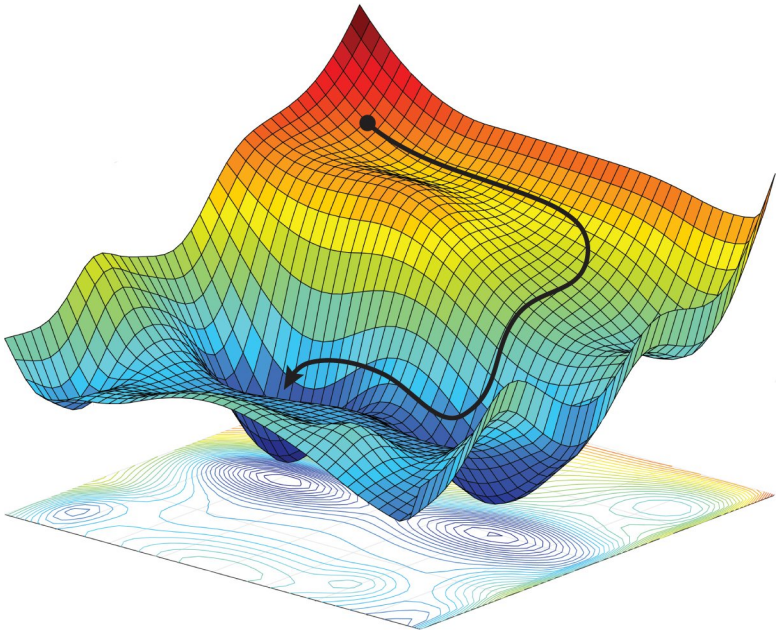


Figura 2.10: Gradient descent

Il Gradient Descent inizia osservando l'output attivato tra i nodi di uscita e comprende che il valore di uno di essi dovrebbe essere incrementato mentre quello di tutti gli altri dovrebbe diminuire. Un modo per farlo è aggiornare i pesi delle connessioni col nodo di output, oppure cambiare l'output di attivazione del livello precedente. Tuttavia non è possibile cambiare direttamente l'output di attivazione poichè è un calcolo che si basa sui pesi e gli output dei livelli precedenti. Il processo continua fino al raggiungimento del livello di input. In particolare l'aggiornamento dei pesi procede nel seguente modo:

$$\omega_{ij} \to \omega_{ij} - \eta \frac{\partial J}{\partial \omega_{ij}} \qquad b_i \to b_i - \eta \frac{\partial J}{\partial b_i}$$

Figura 2.11

dove η è il **tasso di apprendimento** che determina quanto cambiano i pesi ω ad ogni passo dell'algoritmo. Se η è troppo piccolo ne consegue che per convergere

l'algoritmo impiegherà molto tempo, viceversa si potrebbe verificare perdita di controllo sulla loss function (ovvero l'algoritmo diverge).

Apprendimento batch e on-line

Esistono due tipologie di apprendimento: batch ed online. Nella prima tipologia, prima di aggiornare i pesi, si accumulano i contributi di pendenza per tutti i punti dei dati nel training set. Nel secondo i pesi si aggiornano subito dopo aver preso in considerazione ogni punto del set di dati. E' da notare che per l'apprendimento online la tecnica di gradient descent è tra quelle migliori utilizzabili.

2.5 Deep Q-learning

Il Q-learning utilizza una tabella per la memorizzazione delle coppie stato-azione, la quale è utile per la stima della value-function, dato che ad ogni configurazione dell'ambiente viene associato l'expected return appreso durante le iterazioni della policy. Tuttavia l'utilizzo di una tabella risulta possibile ed applicabile solo ad ambienti con un numero ridotto di stati ed azioni. Il Deep Q-Learning (DQN) è un algoritmo di apprendimento appartenente alla famiglia del Deep Reinforcement Learning che consente di implementare reti neurali profonde per realizzare funzioni approssimatrici. Esso rappresenta quindi un'evoluzione del metodo base Q-Learning in quanto la tabella stato-azione è sostituita da una rete neurale, con lo scopo di approssimare la funzione valore ottima q* che soddisfa l'equazione di Bellman. In tal caso infatti l'apprendimento non consiste nell'aggiornamento della tabella, bensì nell'applicazione dell'algoritmo di SGD e Backpropagation alla rete stessa, con l'obiettivo di minimizzare ancora la loss function:

$$q_*(s,a) - q(s,a) = loss$$

$$E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right] - E\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right] = loss$$

Dunque il problema da risolvere è il medesimo ma cambia l'algoritmo impiegato: piuttosto che utilizzare un'iterazione dei vari valori viene impiegata una *deep neural network*. Inoltre il DQN utilizza funzioni approssimatrici in modo stabile e robusto grazie ad un addestramento off-policy con campioni provenienti da un replay buffer per limitare le correlazioni tra i campioni e Q-network target per fornire dei target consistenti. Con l'experience replay si immagazzina l'esperienza dell'agente (nei suoi istanti temporali) in un data-set chiamaato *replay memory*. L'esperienza dell'agente al tempo t è indicata con $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$. Tale data-set è quello dal quale si campiona randomicamente (*mini-batches*) per addestrare la rete, e tale processo prende il nome di experience replay. Prima del training il data-set di memoria D viene inizializzato in termini di dimensioni e con pesi randomici. Un campione di esperienza prelevato dalla replay memory viene preprocessato e passato come input alla rete. Ci si riferisce ad essa come la policy network dato che il suo obiettivo è l'approssimazione della Q-function ottima. Dunque lo stato di input si propaga attraverso la rete ed il modello fornisce in output un Q-value stimato per ogni possibile azione da un dato stato di input. Viene poi determinata la perdita comparando il Q-value ottenuto in uscita dalla rete con quello ottimo (che è il target). Per determinare il termine massimo presente nell'equazione della loss function passiamo "s'" alla policy network che fornirà in uscita il Q-value per ogni coppia (s,a) usando "s'" come stato e ogni possibile azione prossima "a'". Dopo aver calcolato la perdita il gradiente viene determinato per riaggiustare i pesi nella nostra rete per minimizzare la perdita stessa, il che vuol dire ottenere un Q-value in uscita dalla rete per ogni stato-azione che approssima il Q-value target fornito dall'equazione di Bellman.

Algorithm 3 Deep Q learning con Experience Replay

Inizializza Replay Memory D
Inizializza Q(s,a) con pesi random
repeat
  Osserva stato iniziale s_1
  for t=1 **to** T **do**
    Seleziona un azione a_t usando Q (es: ϵ-greedy)
    Esegui azione a_t
    Osserva il reward r_t e il nuovo stato s_{t+1}
    Salva la transizione (s_t, a_t, r_t, s_{t+1}) nel Replay Memory D
    Preleva un campione di transizioni (s_j, a_j, r_j, s_{j+1}) da D
    Calcola il target T per ogni transizione
    if s_{j+1} è Terminale **then**
      $T = r_j$
    else
      $T = r_j + \gamma \max_a Q(s_{j+1}, a_j)$
    end if
    Addestra la rete Q minimizzando $(T - Q(s_j, a_j))^2$
  end for
  until terminazione

Figura 2.12: DQN con Experience Replay

Piuttosto che passare una seconda volta attraverso la policy network per calcolare il Q-value target viene utilizzata una rete separata denominata target network, questa costituisce un clone delle policy network i cui pesi sono congelati a quelli della rete originale e aggiorniamo i vari pesi dopo un certo numero di step temporali [1].

2.6 Approccio con Actor-Critic e algoritmo DDPG

Il DQN non può essere applicato direttamente ad un dominio continuo poichè fa riferimento alla ricerca dell'azione che massimizza la action-value function, che nel caso continuo, richiede un processo di ottimizzazione iterativo ad ogni step. E' possibile però approcciare l'utilizzo del DQN con la discretizzazione: tuttavia ciò comporta una serie di limitazioni soprattutto circa la dimensionalità, poiché il numero di azioni cresce esponenzialmente col numero di gradi di libertà, dunque uno spazio di azioni del genere risulta complesso da esplorare efficientemente (ed inoltre ciò comporterebbe una perdita di azioni che potrebbero essere essenziali nella risoluzione del problema) [7]. Un approccio interessante come mezzo per

risolvere i problemi di controllo che ha guadagnato particolare interesse negli ultimi anni è stato l'uso di metodi *policy-gradient*: Una classe di algoritmi on-line che modifica i parametri di una politica stocastica nella direzione che ottimizza la ricompensa attesa ed in grado di aggiornare i pesi della policy utilizzando una stima del gradiente ad ogni iterazione; Inoltre sono algoritmi che possono funzionare con variabili continue, hanno garanzia di convergenza, sono approcci model free (senza modelli) ovvero possono apprendere un comportamento efficiente attraverso l'interazione diretta con l'ambiente, e reagiscono bene all'apprendimento sotto dinamiche rumorose. La policy in questo caso, essendo una rete neurale profonda una function approximator, in uno spazio di azione continuo può essere associata ad una distribuzione di probabilità. Se invece la policy è deterministica restituisce direttamente l'azione più probabile. Il DDPG (*Deep Deterministic Policy Gradient*) è un algoritmo che in maniera concorrente apprende la Q-function ed una politica. Esso utilizza dati off-policy e l'equazione di Bellman per determinare la Q-function e usa la Q-function per apprendere la politica stessa. Il DDPG lavora in un ambiente con spazio-azioni continue di alta dimenesione, e ciò è legato al modo in cui viene calcolato il $max_a Q^*(s, a)$. Infatti quando lo spazio delle azioni è continuo non possiamo valutare esaustivamente lo spazio e il problema di ottimizzazione non è banale, in taluni casi il calcolo dovrebbe essere svolto ogni volta che l'agente vuole compiere un'azione all'interno dell'ambiente. Dunque il DDPG può essere immaginato come una estensione del deep Q-learning per spazi di azione continui [16], difatti rappresenta una tecnica di reinforcement learning che combina Q-learning e Policy Gradient. In particolare l'utilizzo del DDPG con una tencica di tipo actor-critic (attore-critico) consiste di due modelli:

- Attore: è una policy network che a partire dallo stato fornito come input restituisce l'esatta azione, invece di una distribuzione di probabilità definita sulle azioni stesse.

- Critico: è una Q-value network che accetta stati e azioni e restituisce il Q-value.

Seguendo l'osservazione dello stato, la rete dell'attore seleziona un'azione mentre la rete del crtiico valuta("critica") l'azione scelta migliorando la rete dell'attore. Un buffer è utilizzato per memorizzare e campionare le varie esperienze di addestramento in modo da aggiornare i parametri della rete neurale. Essendo la policy deterministica, allo scopo di promuovere l'esplorazione tipicamente viene aggiunto del rumore Gaussiano all'azione scelta. Inoltre per stabilizzare l'apprendimento vengono di solito create delle reti target, sia per il critico che per l'attore, che fanno in modo che il valore atteso non sia dipendente dai parametri utilizzati nelle reti principali, evitando di calcolare tramite la stessa rete anche il valore target e dunque evitando la divergenza tra i due valori [12].

- Il target Actor ha la stessa struttura e parametrizzazione dell'Attore e produce in uscita l'azione a' rispetto allo stato successivo s'.

- Il target Critic ha la stessa struttura e parametrizzazione del Critico e produce in uscita il valore futuro atteso Q(s',a') che servirà a calcolare la value-function.

La loss function per il critico (Q) e l'attore (μ) è data da:

$$J_Q = \frac{1}{N}\sum_{i=1}^{N}(r_i + \gamma(1-d)Q_{targ}(s_i', \mu_{targ}(s_i')) - Q(s_i, \mu(s_i))^2$$

$$J_\mu = \frac{1}{N}\sum_{i=1}^{N} Q(s_i, \mu(s_i))$$

J_μ è la somma dei Q-values per i vari stati. Per calcolare i Q-values utilizziamo la rete del Critico passandogli l'azione selezionata dalla rete dell'Attore. L'obiettivo è massimizzare il risultato dato che lo scopo è ottenere il massimo return o Q-values. Nella J_Q viene utilizzata la rete target per computare il Q-value per il

prossimo stato. Occorre minimizzare questa perdita. Nota che, al fine di propagare correttamente l'errore, abbiamo bisogno che le funzioni siano differenziabili.

Per la perdita del critico la derivata del Q-value è diretta considerando μ come delle costanti, per l'attore invece la funzione μ è contenuta all'interno del Q-value, per cui per quest'ultima utilizzeremo la regola della catena (*chain rule*)

$$J_\mu = E\left[Q(s, \mu(s))\right]$$
$$\nabla_{\theta^\mu} J_\mu = E\left[\nabla_\mu Q(s, \mu(s)) \nabla_{\theta^\mu} \mu(s)\right]$$

Le reti target sono delle copie ritardate temporalmente delle reti originali di attore e critico. Secondo un approccio soft-update solo una frazione dei pesi vengono trasferiti (con τ parametro che viene scelto vicino ad 1) [2].

$$\theta^\mu{}_{targ} \leftarrow \tau\theta^\mu{}_{targ} + (1-\tau)\theta^\mu$$
$$\theta^Q{}_{targ} \leftarrow \tau\theta^Q{}_{targ} + (1-\tau)\theta^Q$$

Inoltre, quando l'apprendimento viene realizzato a partire da un vettore di osservazione a bassa dimensione i vari componenti potrebbero avere diverse unità fisiche (posizione, velocità) con i relativi ranges che variano nell'ambiente. Per risolvere questo problema si utilizza una tecnica tipica del Deep-Learning nota come batch normalization. Questa tecnica normalizza le varie dimensioni dei campioni nel mini-batch per avere media e varianza unitaria. Nelle reti profonde è usato per minimizzare la covarianza durante il training [3].

Algorithm 1 DDPG algorithm

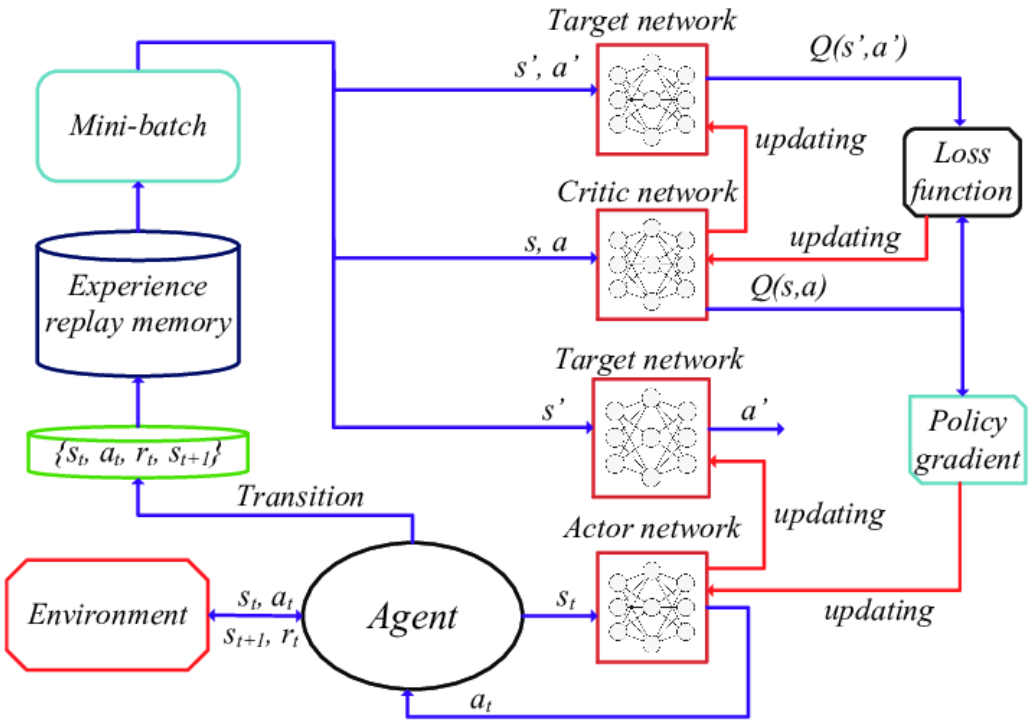
Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for



Capitolo 3

Modello ed implementazione

3.1 Specifico problema

Tramite l'utilizzo di un agente DDPG è stato implementato un controllo che permettesse di realizzare un convoglio di treni "on-the-fly", partendo da un controllore ACC realizzato mediante il medesimo approccio da Mathworks [15]. In particolare la configurazione considerata è quella di una tripla di veicoli ferroviari in configurazione leader-follower: Il treno che segue (follower train o ego train) utilizza le informazioni dinamiche del treno antecedente (leader train o lead train), quali posizione e velocità, per attuare il comportamento desiderato. Durante il controllo deve essere sempre rispettata la distanza di sicurezza rispetto al treno precedente: tale distanza è chiaramente funzione della velocità del treno stesso, in quanto una maggiore velocità implica un maggiore spazio di frenata necessario. Nel momento in cui il treno è sufficientemente vicino al treno antecedente (quindi è in quella che possiamo definire come distanza di accoppiamento nella quale riceve il comando dal RBC), l'ego train insegue la dinamica della velocità del lead train per mantenersi ad una distanza tale da rispettare la distanza di sicurezza. Possiamo dunque affermare che, in una situazione di regime in cui il primo ego train (ego_train_1) insegue la velocità del lead train ed il secondo ego train (ego_train_2) insegue

la velocità del primo ego train (che sarà leader per esso), allora i tre treni si trovano in una formazione a convoglio, il tutto senza la necessità di utilizzare dei collegamenti meccanici.

3.1.1 Semplificazioni del problema

Rispetto al concetto rigoroso di virtual coupling sono state fatte svariate semplificazioni, le quali tuttavia non rendono inesatto quanto detto durante la trattazione. Difatti il problema è stato analizzato ad un livello di astrazione superiore, evitando la modellazione di alcuni aspetti. Le semplificazioni considerate sono le seguenti:

- La comunicazione segue una topologia unidirezionale in cui un treno del convoglio riceve le informazioni dinamiche (in particolare posizione e velocità) unicamente sul treno antecedente.

- Non è stato modellato il Radio Block Center (RBC), ovvero il centro di controllo che gestisce il segnalamento. In particolare, all'interno del modello simulink, la ricezione delle informazioni dinamiche dall'ambiente sul treno antecedente è stata modellata con un collegamento diretto senza ritardo (che è evidentemente una semplificazione rispetto al caso reale di ritardo di propagazione del segnale e anche di disponibilità di esso) e senza esplicitare la particolare tecnologia utilizzata.

- Non si è esplicitata una eventuale comunicazione diretta tra i treni stessi che potrebbero eventualmente utilizzare sensori di vario genere per vigere al vincolo del rispetto della distanza di sicurezza. Le manovre di accoppiamento in cui la distanza di frenata (dinamica) è inferiore alla gamma del sensore relativo, potrebbero essere eseguite utilizzando solo sensori relativi, implementando difatti un ACC in cui la velocità di crociera è determinata dall'andamento treno antecedente. L'accoppiamento a velocità più elevate,

dove la distanza di frenata supera la portata del sensore relativo, richiede una comunicazione diretta o indiretta tra i treni ed i relativi sensori fungerebbero da sistema ridondante nei momenti finali dell'approccio.

- I treni utilizzati implementano innanzitutto un controllo ACC (Adaptive cruise control, utilizzato in contesto automotive) e dunque prima dell'accoppiamento (ad una oppurtuna distanza prima del raggiungimento della safe distance) inseguono il profilo di velocità di crociera v_set impostato. Tuttavia tale semplificazione potrebbe anche essere trascurata in quanto potremmo suppore che la v_set possa essere impostata dinamicamente sotto comando del RBC: prima dell'accoppiamento la v_set è una velocità tale da permettere al treno di raggiungere quello antecedente. Ciò produrrebbe il comportamento desiderato (ovvero la realizzazione di un convoglio) in quanto una volta accoppiato il treno inseguirà esclusivamente la velocità a cui sta andando il treno antecedente. Un possibile valore per la v_set è dunque la velocità massima del treno sulla tratta, ovviamente supposto che il treno che insegue abbia una velocità massima di percorrenza minore durante l'accoppiamento.

- La topologia presa in esame prevede la presenza di soli tre treni ma potrebbe essere estesa ad un numero N di treni.

3.2 Dataset di base

Di seguito invece si riporta il dataset di base realizzato in ambiente Matlab:

```
x0_lead_train = 470;    %posizione iniziale treno lead [m]
v0_lead_train = 75;     %velocità iniziale treno lead [m/s]
x0_ego_train_1 = 250;   %posizione iniziale treno intermedio [m]
v0_ego_train_1 = 60;    %velocità iniziale treno intermedio [m/s]
x0_ego_train_2 = 50;    %posizione iniziale treno in coda [m]
v0_ego_train_2 = 55 ;   %velocità iniziale treno in coda [m/s]
amin_ego = -4;          %decelerazione minima treno [m/s^2]
amax_ego = 1;           %accelerazione massima treno [m/s^2]
D_default = 10+90;      %distanza di default tra i treni [m]
t_gap = 1.4;            %time gap [s]
v_set = 80;             %velocità impostata [m/s]
coupling_distance = 40;
Ts = 0.1;               %sampling time [s]
Tf = 500;               %durata della simulazione [s]
```

Figura 3.1

- Innanzitutto vengono specificate le posizioni e le velocità iniziali per i treni costituenti il convoglio (da x0_lead_train a v0_ego_train).

- Le variabili amin_ego ed amax_ego definiscono l'intervallo di saturazione del segnale di accelerazione per gli ego train, ciò per considerare le limitazioni fisiche dettate dalla loro dinamica. Per il lead train non è stata considerata saturazione in quanto si presuppone che in ingresso riceva un segnale di accelerazione compreso nell'intervallo di saturazione.

- Le variabili $D_{default}$ (distanza di default) e t_{gap} (tempo di reazione alla frenata) sono invece utilizzate per determinare la distanza di sicurezza che i treni devono rispettare: difatti la safe distance è calcolata come la distanza di default sommata al prodotto tra la velocità attuale dell'ego train e il time gap ($safe\_distance = t_{gap}\ actual\_velocity + D_{default}$), ciò per modellare il fatto che al crescere della velocità lo spazio necessario per arrestare completamente il treno cresce anch'esso (e quindi deve aumentare la distanza di sicurezza). Inoltre la D_default è calcolata come somma di due contributi: il secondo rappresenta la lunghezza canonica dei treni (in tal caso 90 metri), in quanto la distanza di sicurezza la valutiamo rispetto alla testa del treno antecedente (per motivi di posizionamento assoluto dei treni nel convoglio)

- la variabile v_set rappresenta la velocità di crociera impostata per il treno. Inoltre essa può rappresentare, nel nostro particolare caso, anche la velocità massima che il treno può raggiungere lungo la sua tratta (per le motivazioni illustrate nel paragrafo precedente).

- Sono poi definiti il tempo di campionamento Ts e la durata della simulazione Tf (quindi il tempo assoluto di arresto della simulazione).

- coupling_distance è una variabile utile all'accoppiamento, determina l'istante in cui cambia il riferimento in velocità per un treno follower (di seguito sarà mostrato il suo utilizzo).

3.3 Modello simulink

Di seguito si mostra lo schema a blocchi complessivo (realizzato a partire dallo schema a blocchi fornito da Mathworks per il controllo ACC [15]):

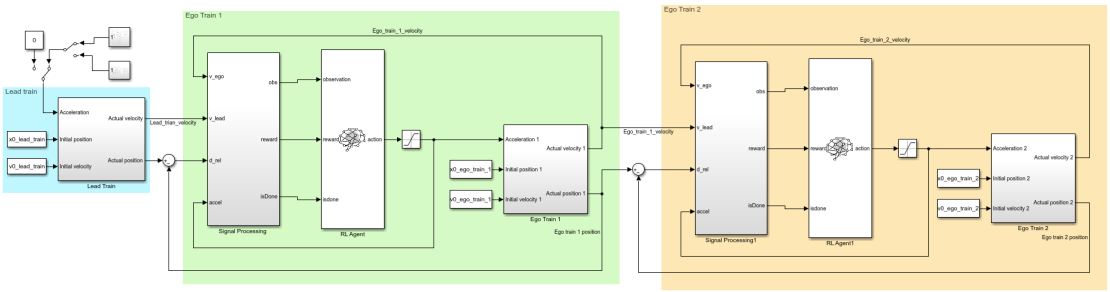


Figura 3.2: Modello simulink complessivo

3.3.1 Dinamica treno

La dinamica tra accelerazione e velocità dei treni è descritta da una funzione di trasferimento del secondo ordine (mostrata in figura 3.3).

$$G = \frac{1}{s\,(0.5s + 1)}$$

Figura 3.3: Dinamica treno

In figura 3.4 si mostra il modello simulink utilizzato per il calcolo della posizione attuale (actual position) e velocità attuale (actual velocity).

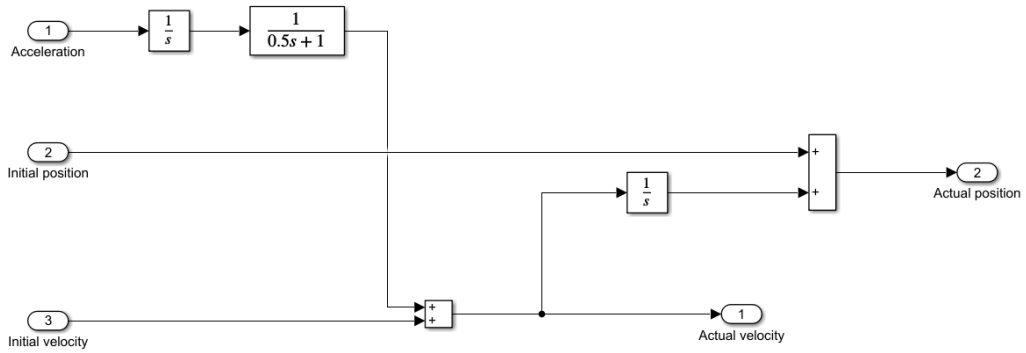


Figura 3.4: Modello Simulink treno

1. La posizione attuale del treno viene utilizzata per ricavare la distanza relativa con il treno che lo segue (quindi come differenza tra la posizione del treno leader e la posizione attuale del treno follower che viene ricavata in feedback).

2. La velocità attuale è passata direttamente al blocco di signal processing per l'agente RL (che controlla un treno follower) mostrato nella sezione successiva.

3.3.2 Signal processing

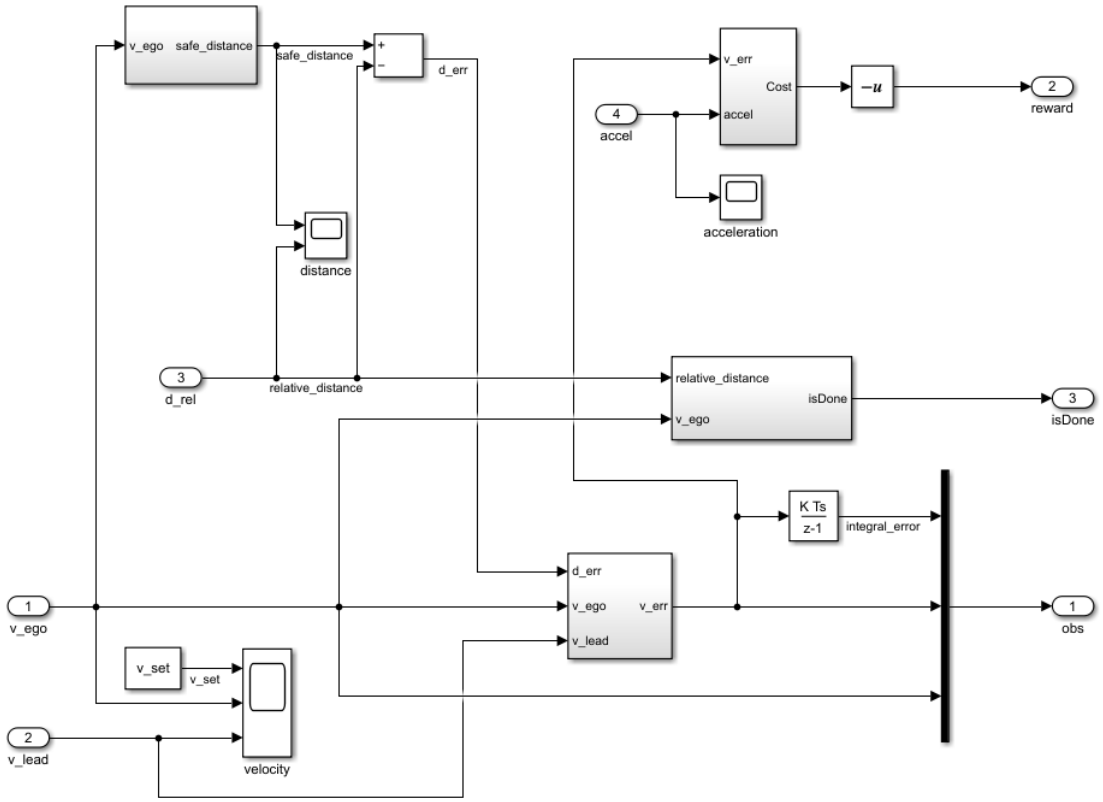


Figura 3.5: Blocco di signal processing

L'agente RL riceve segnali di osservazione provenienti dall'ambiente (**observation**), il segnale di **reward** calcolato in base ai dati osservati e utilizzato durante l'addestramento per massimizzare la ricompensa attesa, **isDone** per specificare condizioni di terminazione del singolo episodio di addestramento e **action** (elaborata dall'agente sulla base di osservazioni e ricompense).

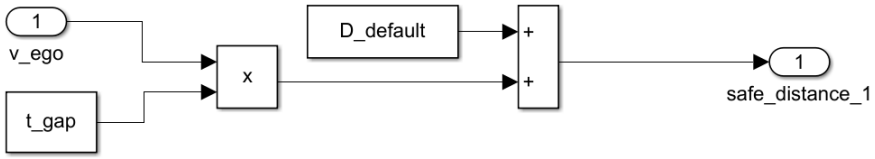


Figura 3.6: safe distance processing

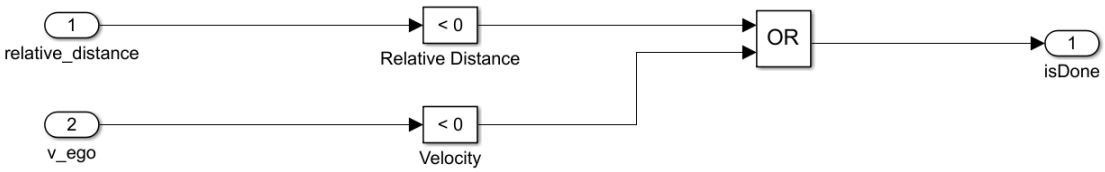


Figura 3.7: isDone processing

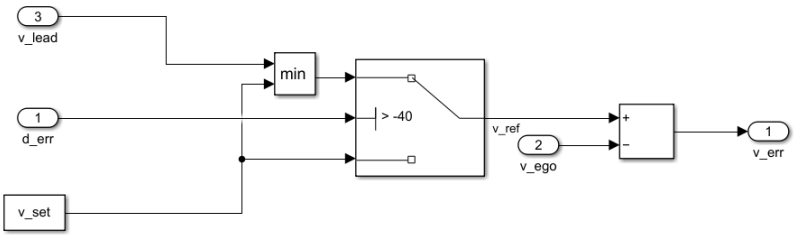


Figura 3.8: v_err processing

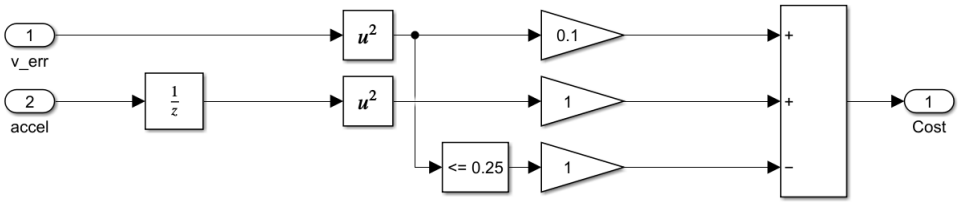


Figura 3.9: reward processing

Tutti i sottoblocchi mostrati contribuiscono alla costruzione dei tre segnali che poi entreranno nell'agente RL (blocco che rappresenta l'interfaccia tra agente e ambiente).

- Lo schema a blocchi in figura 3.6 calcola la distanza di sicurezza.
- Lo schema a blocchi in figura 3.7 dà informazioni sulle condizioni di terminazione dell'addestramento, il quale avviene quando la velocità longitudinale del treno scende al di sotto di zero, dunque il treno subisce un arresto, oppure la distanza relativa tra i due treni diventa minore di zero, che manifesta uno scontro.

- Lo schema a blocchi in figura 3.8 rappresenta invece la definizione dell'errore di velocità rispetto alla velocità di riferimento. In particolare la velocità da inseguire varia a seconda dell'errore di distanza (definito come la differenza tra la distanza relativa e la distanza di sicurezza). Grazie allo switch se $d\_err \leq -(coupling\_distance)$ allora il treno insegue la v_set, altrimenti intercetta il minimo tra la v_set e la velocità del treno antecedente (per rispettare la distanza di sicurezza ed aggiungersi al convoglio).

- Lo schema a blocchi in figura 3.9 calcola il segnale di ricompensa proveniente dall'ambiente per ogni azione impressa dall'agente su di esso: essa sarà massimizzata se l'agente si avvicinerà all'obiettivo prefissato dal controllo, minimizzata se viceversa. La reward function utilizzata è la seguente: $r_t = -(0.1\ v_{err}^2 + u_{t-1}^2) + M_t$, Dove u_{t-1} è l'input di controllo relativo al time step precedente. Il valore logico M_t è pari ad 1 se l'errore di velocità è $e_t^2 \leq 0.25$, altrimenti è nullo. Notiamo che l'uscita "Cost" di tale blocco sarà poi invertita di segno per realizzare il segnale di reward.

3.3.3 Agente RL

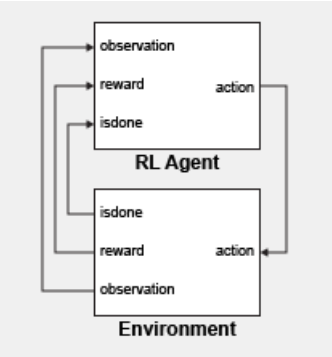


Figura 3.10: RL Agent

Il blocco di RL (disponibile nel *Reinforcement Learning Toolbox* di Matlab [14]) rappresenta l'interfaccia dell'agente con l'ambiente, dal quale riceve osservazioni

e su cui agisce a partire da osservazioni e ricompense. Per guidare il processo di apprendimento, il blocco RL utilizza un segnale scalare di ricompensa proveniente dall'ambiente che, data un'osservazione, misura la fattibilità di effettuare una data azione e la performance dell'agente nel rispetto dell'obiettivo fissato. Durante l'addestramento l'agente aggiorna la sua politica, in particolare lo fa attraverso la funzione train, basandosi sulle ricompense ricevute per diverse combinazioni di osservazione-azione. In generale viene fornita una ricompensa positiva per incoraggiare certe azioni compiute dall'agente, e una penalità per scoraggiarne altre, al fine di massimizzare la ricompensa attesa a lungo termine.

3.4 Matlab

3.4.1 Definizione interfaccia con l'ambiente

Per la creazione dell'interfaccia con l'ambiente bisogna innanzitutto definire le specifiche delle azioni e delle osservazioni, per poi creare l'*enviroment interface* tramite la funzione rlSimulinkEnv. Inoltre è definita una *reset function* che randomizza le posizioni iniziali del lead train.

```
observationInfo = rlNumericSpec([3 1],'LowerLimit',-inf*ones(3,1),'UpperLimit',inf*ones(3,1));
observationInfo.Name = 'observations';
observationInfo.Description = 'information on velocity error and ego velocity';
actionInfo = rlNumericSpec([1 1],'LowerLimit',-3,'UpperLimit',2);
actionInfo.Name = 'acceleration';
env = rlSimulinkEnv(mdl,agentblk,observationInfo,actionInfo);
env.ResetFcn = @(in)localResetFcn(in);
rng('default')
```

Figura 3.11: Interfaccia con l'ambiente

```
function in = localResetFcn(in)
% Reset the initial position of the lead car.
in = setVariable(in,'x0_lead',40+randi(60,1,1));
end
```

Figura 3.12: Reset function

3.4.2 Creazione agente DDPG

Per la generazione del critico e dell'attore, che daranno origine anche alle reti target rispettive, si utilizza una rete neurale con un numero di neuroni pari a 48. Per il Critico la rete neurale sarà composta da due input (l'osservazione e l'azione) e un output. Per l'attore la rete neurale sarà composta da un input (ovvero l'osservazione) e un output che sarà l'azione che dovrà eseguire l'agente. La programmazione delle reti neurali viene svolta in ambiente Matlab con le seguenti linee di codice:

```
L = 48; % number of neurons
statePath = [
    featureInputLayer(3,'Normalization','none','Name','observation')
    fullyConnectedLayer(L,'Name','fc1')
    reluLayer('Name','relu1')
    fullyConnectedLayer(L,'Name','fc2')
    additionLayer(2,'Name','add')
    reluLayer('Name','relu2')
    fullyConnectedLayer(L,'Name','fc3')
    reluLayer('Name','relu3')
    fullyConnectedLayer(1,'Name','fc4')];

actionPath = [
    featureInputLayer(1,'Normalization','none','Name','action')
    fullyConnectedLayer(L, 'Name', 'fc5')];

criticNetwork = layerGraph(statePath);
criticNetwork = addLayers(criticNetwork, actionPath);

criticNetwork = connectLayers(criticNetwork,'fc5','add/in2');
criticOptions = rlRepresentationOptions('LearnRate',1e-3,'GradientThreshold',1,'L2RegularizationFactor',1e-4);

critic = rlQValueRepresentation(criticNetwork,observationInfo,actionInfo,...
    'Observation',{'observation'},'Action',{'action'},criticOptions);
```

```
actorNetwork = [
    featureInputLayer(3,'Normalization','none','Name','observation')
    fullyConnectedLayer(L,'Name','fc1')
    reluLayer('Name','relu1')
    fullyConnectedLayer(L,'Name','fc2')
    reluLayer('Name','relu2')
    fullyConnectedLayer(L,'Name','fc3')
    reluLayer('Name','relu3')
    fullyConnectedLayer(1,'Name','fc4')
    tanhLayer('Name','tanh1')
    scalingLayer('Name','ActorScaling1','Scale',2.5,'Bias',-0.5)];

actorOptions = rlRepresentationOptions('LearnRate',1e-4,'GradientThreshold',1,'L2RegularizationFactor',1e-4);
actor = rlDeterministicActorRepresentation(actorNetwork,observationInfo,actionInfo,...
    'Observation',{'observation'},'Action',{'ActorScaling1'},actorOptions);
```

Figura 3.13

```
agentOptions = rlDDPGAgentOptions(...
    'SampleTime',Ts,...
    'TargetSmoothFactor',1e-3,...
    'ExperienceBufferLength',1e6,...
    'DiscountFactor',0.99,...
    'MiniBatchSize',64);
agentOptions.NoiseOptions.Variance = 0.6;
agentOptions.NoiseOptions.VarianceDecayRate = 1e-5;
agent = rlDDPGAgent(actor,critic,agentOptions);
```

Figura 3.14

3.4.3 Addestramento agente DDPG

Per l'addestramento dell'agente specifichiamo le opzioni con le istruzioni mostrate in figura 3.15: viene eseguito l'allenamento per massimo 5000 episodi, con ogni episodio che dura al massimo Tf/Ts timesteps. L'allenamento è interrotto quando l'agente ottiene un reward pari a 260 o superiore. E' inoltre impostato come vero il parametro "UseParallel" che permette di sfruttare un pool di esecuzione parallela (in tal caso sui core della CPU) per velocizzare il training. Settando la variabile doTraining a "false" è possible caricare un agente preaddestrato nel worskpace, settandola invece al valore "true" viene eseguita la funzione *train* per addestrare l'agente definito [15].

```
maxepisodes = 5000;
maxsteps = ceil(Tf/Ts);
trainingOpts = rlTrainingOptions(...
    'MaxEpisodes',maxepisodes,...
    'MaxStepsPerEpisode',maxsteps,...
    'Verbose',false,...
    'Plots','training-progress',...
    'StopTrainingCriteria','EpisodeReward',...
    'StopTrainingValue',260,"UseParallel", true);

doTraining = false;
if doTraining
    % Train the agent.
    trainingStats = train(agent,env,trainingOpts);
else
    % Load a pretrained agent for the example.
    load('TrainedAgent.mat','agent')
end
```

Figura 3.15: Training agente DDPG

L'addestramento è un processo ad alta intensità di calcolo che richiede dai minuti ai giorni per essere completato (a seconda di svariati fattori come complessità del problema o struttura dell'agente). Dopo ogni episodio la funzione train aggiorna i parametri dell'agente fino al raggiungimento del valore di reward prefissato. Nella fase di training, l'agente viene addestrato ad eseguire il controllo rispetto al treno che lo precede il quale è caratterizzato da un profilo di guida del conducente fortemente variabile. In particolare si è considerato in ingresso al sistema del lead train l'onda sinusoidale $u(t) = 0.6\ sin(0.2\ t)$ con sample-time impostato a 0.1s (mostrata in figura 3.16). Infine alla fase di training segue una fase di validazione dell'agente rispetto alle specifiche di controllo.

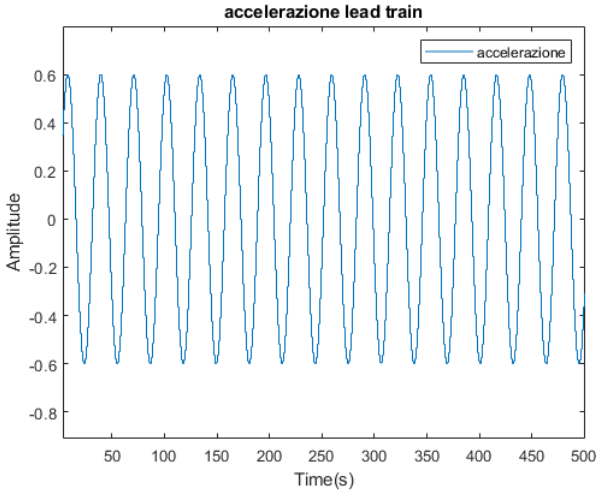


Figura 3.16: input di accelerazione per il lead train in fase di training

Capitolo 4

Risultati Sperimentali

4.1 Risultati sperimentali

Per la validazione sono stati considerati due diversi casi verosimili di accoppiamento: innanzitutto si valida l'agente per accelerazione nulla in ingresso al leader (dunque velocità costante), successivamente per una dinamica di accelerazione trapezoidale. In entrambi i casi si vuole mostrare come, a regime (o nella finestra di osservazione), i treni assumano una configurazione a convoglio in cui mantengono tra di essi una distanza costante durante la tratta che sia maggiore della distanza di sicurezza (garantendo dunque la safety). Infine si mostra un caso di disaccoppiamento tra i treni.

4.1.1 Profilo a velocità costante

Il treno leader del convoglio parte da una velocità iniziale di 75 m/s (circa 270 km/h) che sarà dunque mantenuta per l'intero intervallo di osservazione. Per la validazione dell'accoppiamento mostriamo che:

- la distanza relativa tra i treni del convoglio dopo l'accoppiamento resta costante, dunque la posizione, in funzione del tempo, ha lo stesso andamento per tutti i componenti (si mostra tale comportamento in figura 4.1).

- La velocità dopo l'accoppiamento è la stessa per tutti i componenti del convoglio (si mostra tale comportamento in figura 4.2).

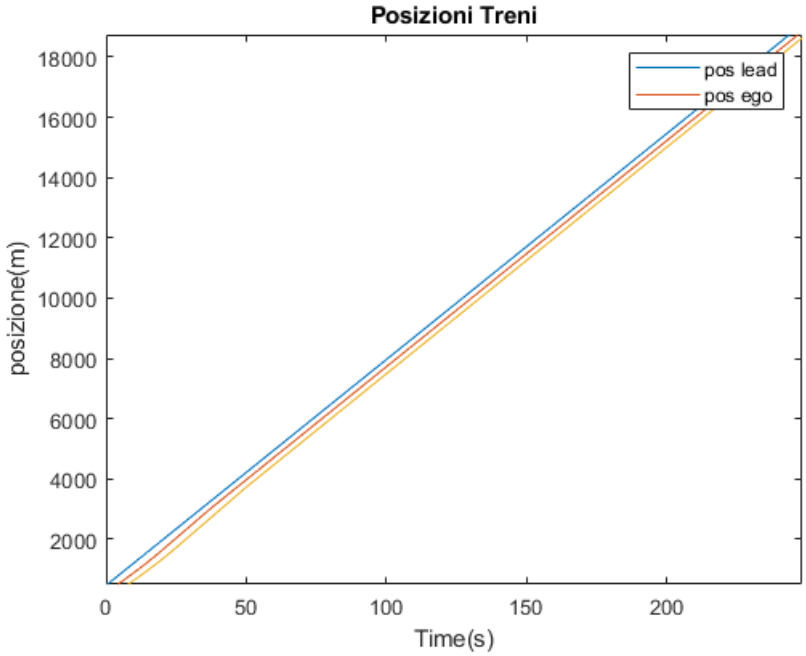


Figura 4.1: Confronto posizioni treni del convoglio

Si può notare (in figura 4.2) come nell'intervallo temporale tra 20 e 45 secondi circa l'ego_train_1 insegue la velocità v_set (80 m/s) in quanto la distanza relativa glielo permette, difatti si sta avvicinando al treno leader al fine di accoppiarsi. Da notare che la velocità v_set deve necessariamente essere più alta della velocità fissa alla quale sta viaggiando il treno leader, altrimenti si avrebbe un progressivo ed inevitabile allontanamento dei due. Notiamo che l'ego_train_2 non insegue la velocità del lead train, bensì dell'ego_train_1 (che per esso funge da leader). Infatti il terzo ed ultimo treno del convoglio si preoccupa di inseguire unicamente il treno antecedente mantenendo una opportuna distanza, ne consegue che l'ordine temporale degli accoppiamenti dipende dalle condizioni iniziali utilizzate dato che ogni componente del convoglio si accoppia indipendentemente.

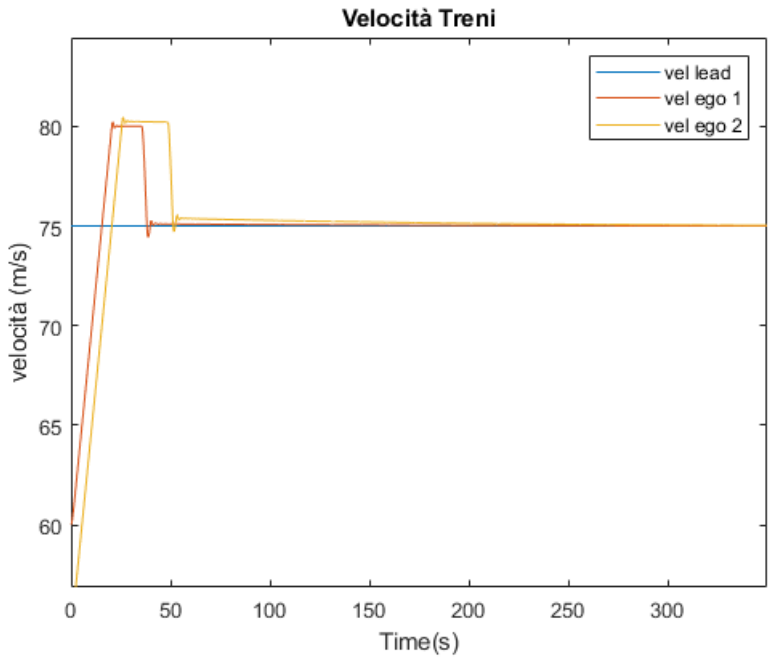


Figura 4.2: Confronto velocità dei treni del convoglio

Si mostra in figura 4.3 e 4.4 l'andamento della distanza relativa confrontata con la distanza di sicurezza. Secondo la specifica del controllo, la prima non deve essere mai minore della seconda (necessaria ad esempio in casi di emergenza per evitare collisione).

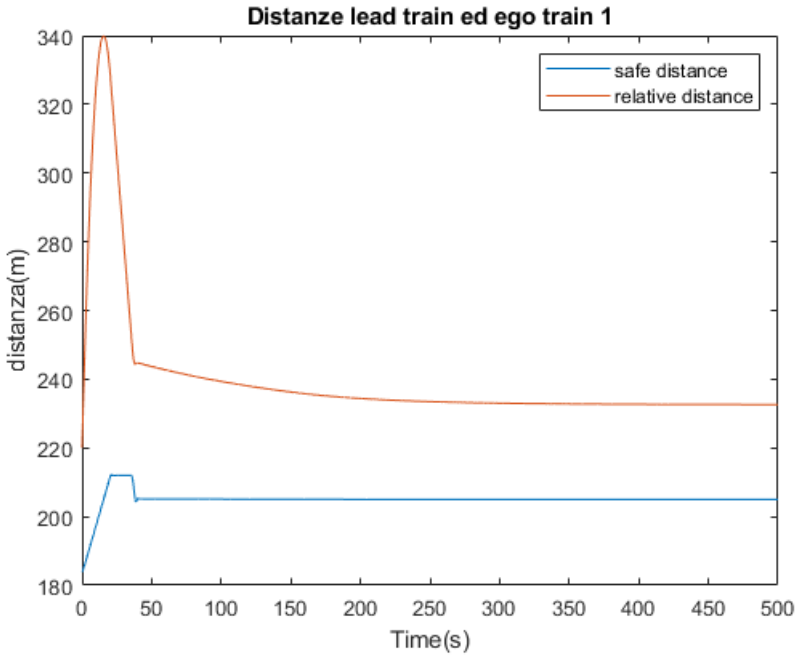


Figura 4.3

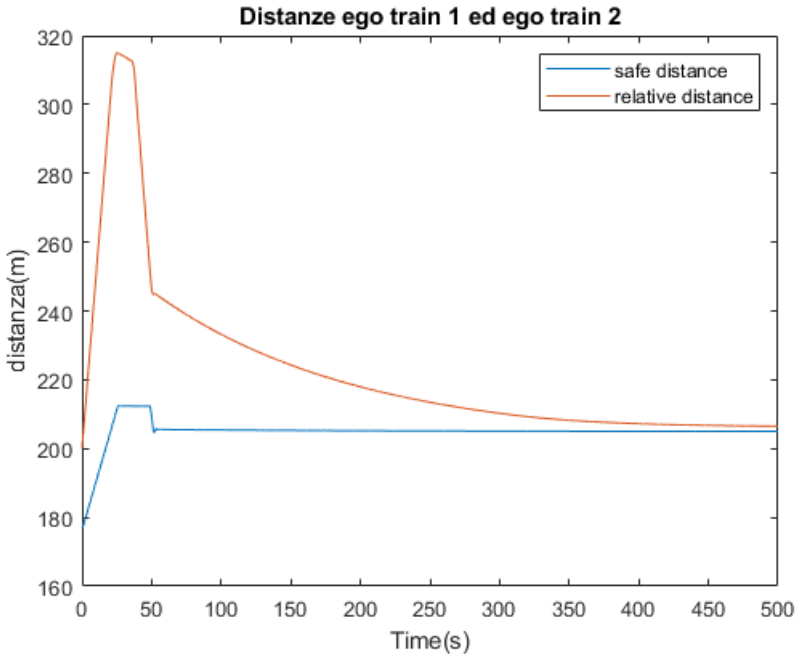


Figura 4.4

E' da precisare che non è nostro obiettivo quello di ottimizzare il controllo affinchè la distanza relativa arrivi a coincidere esattamente con la distanza di sicurezza.

4.1.2 Profilo ad accelerazione trapezoidale

Si riporta un caso di andamento reale del treno leader, il quale prima accelera e successivamente decelera con andamento trapezoidale (come mostrato in figura 4.5).

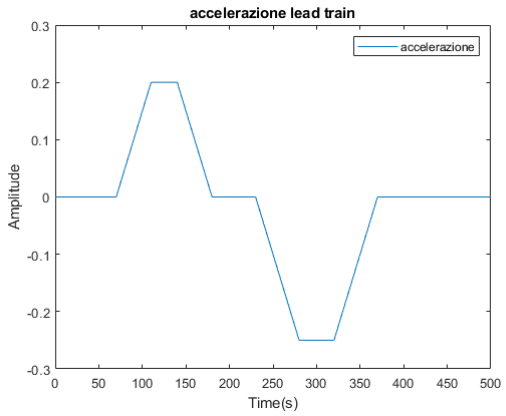


Figura 4.5: input di accelerazione per il lead train

Nell'intervallo temporale compreso tra 100 secondi e 130 secondi (circa) si può notare (in figura 4.7) come i tre treni si muovano a convoglio esibendo la medesima dinamica in velocità. I treni follower non eccedono mai la v_set , dunque il treno leader a circa 130 secondi inizia ad allontanarsi da entrambi gli ego train per poi frenare e raggiungere una velocità minore della v_set. Quando l'ego_train_1 entra nella distanza di accoppiamento allora inizia a decelerare per inseguire la velocità del lead train, mentre l'ego_train_2 segue il suo andamento. Definitivamente si può verificare che l'accoppiamento avviene correttamente e l'intero convoglio viaggia ad una velocità costante di circa 67 m/s.

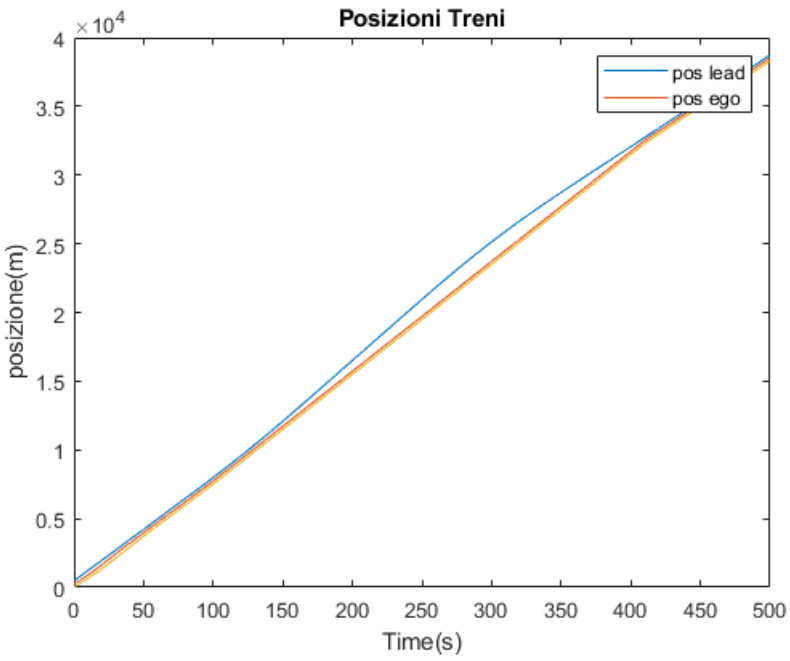


Figura 4.6: Confronto posizioni treni del convoglio

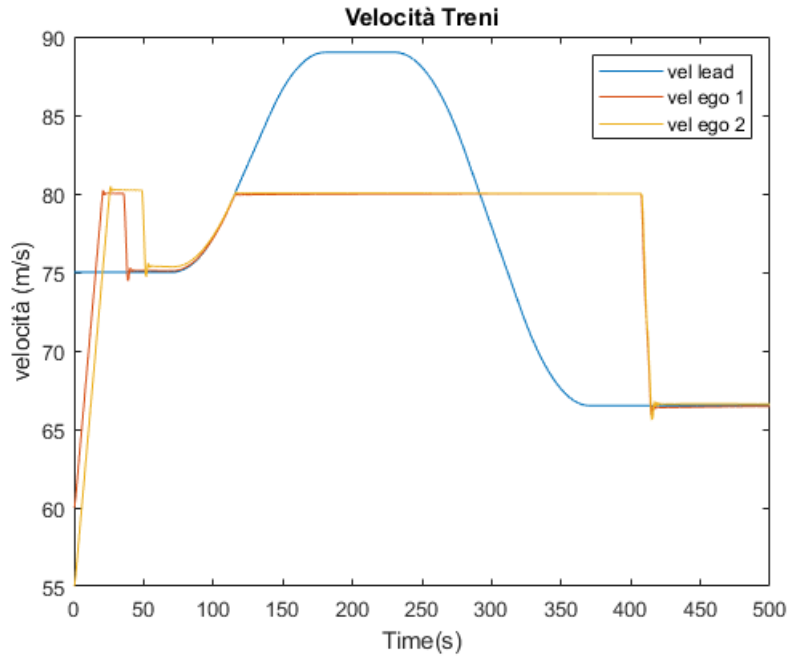


Figura 4.7: Confronto velocità dei treni del convoglio

Anche in questo caso è verificata la condizione di sicurezza (come mostrato in figura 4.8 e 4.9)

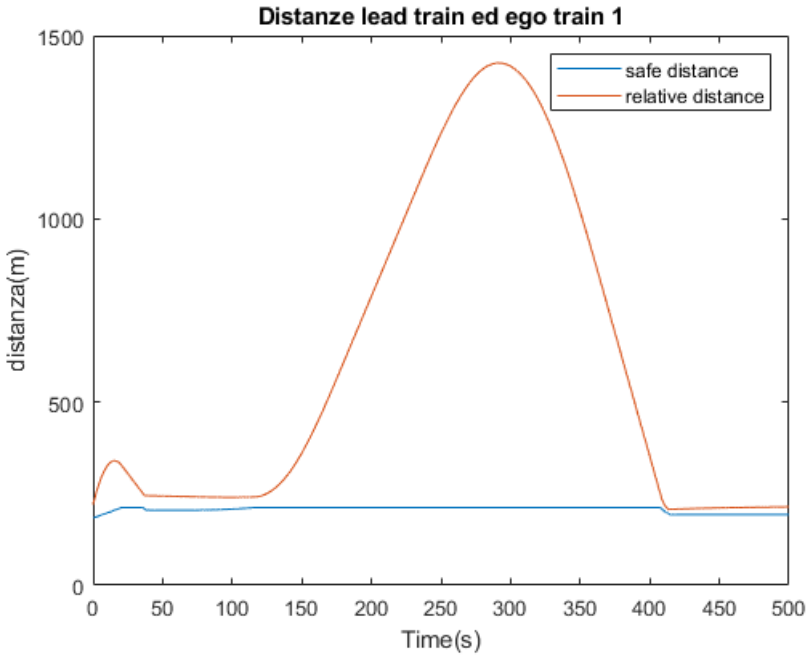


Figura 4.8

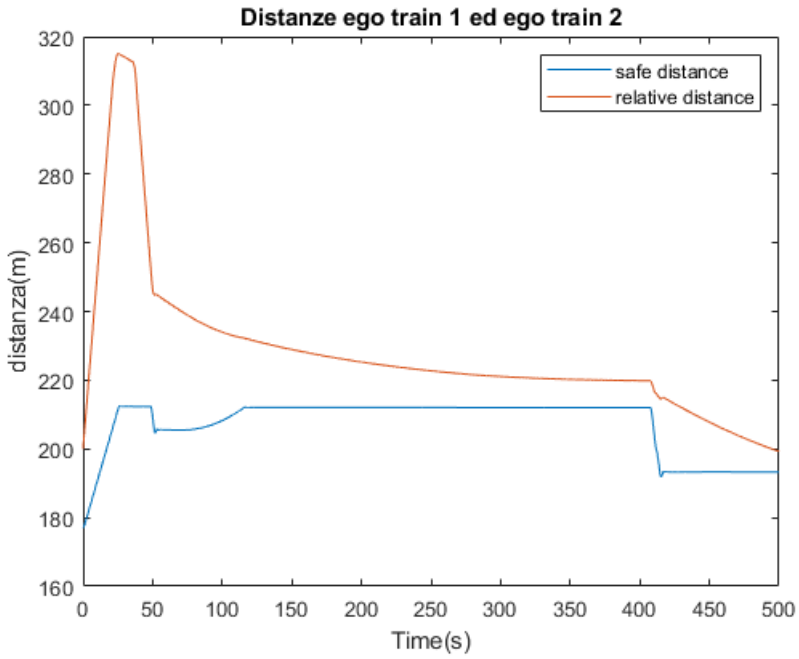


Figura 4.9

4.1.3 Disaccoppiamento

Infine si mostra un caso di disaccoppiamento e quindi di scioglimento del convoglio. Si è dato in ingresso al treno leader un profilo di accelerazione trapezoidale (figura 4.5) che lo porta ad una velocità maggiore della v_set dei treni ego e dunque ad allontanarsi dal convoglio.

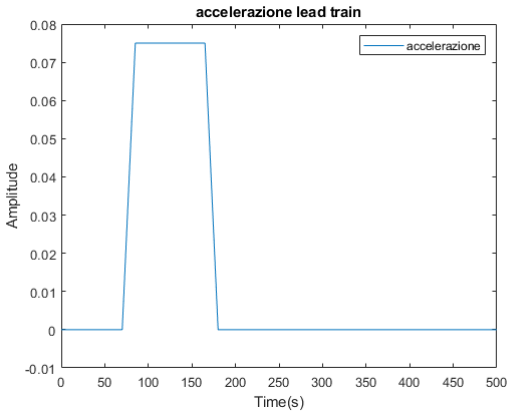


Figura 4.10: input di accelerazione per il lead train

Si può verificare come entrambi gli ego train, successivamente al disaccoppiamento, proseguano la loro tratta alla velocità v_set, così come da specifica per il controllo. E' da sottolineare che per motivi progettuali non è possibile imporre un andamento desiderato ai treni follower, quindi al solo fine di mostrare il corretto funzionamento del controllo in fase di disaccoppiamento è stata posta la velocità v_set dell'ego_train_2 ad un valore di 75 m/s (più basso degli 80 m/s della v_set originaria) in modo da poter apprezzare anche l'allontamemento tra esso e l'ego_train_1.

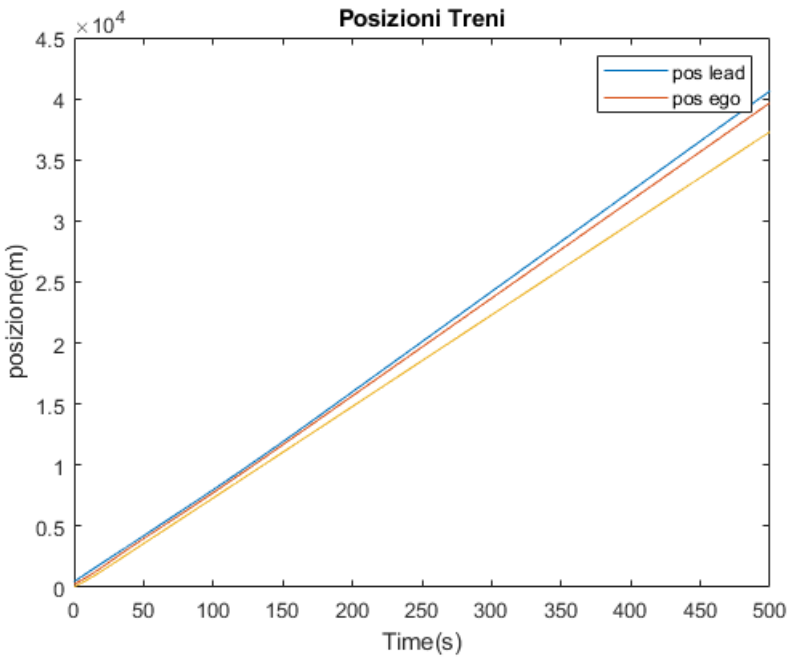


Figura 4.11: Confronto posizioni treni del convoglio

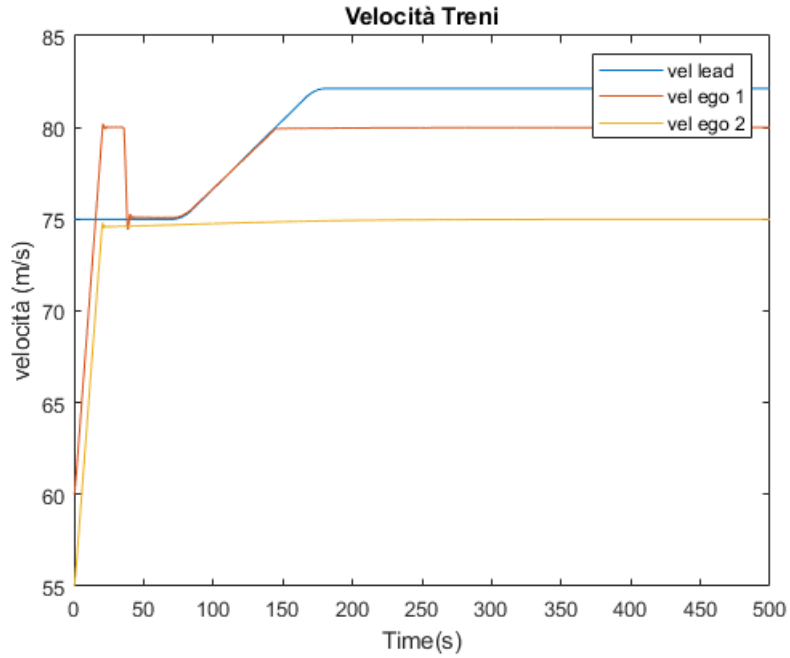


Figura 4.12: Confronto veloctià dei treni del convoglio

Delle figure 4.13 e 4.14 è possibile difatti notare come la distanza relativa (tra le coppie di treni) abbia andamento monotono crescente dopo il disaccoppiamento, infatti i treni definitivamente si assestano a velocità decrescenti.

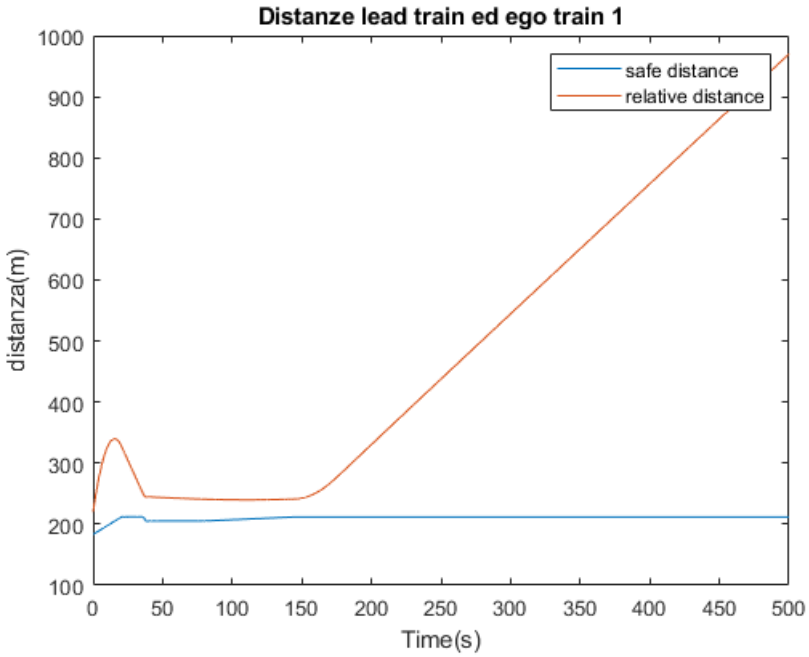


Figura 4.13

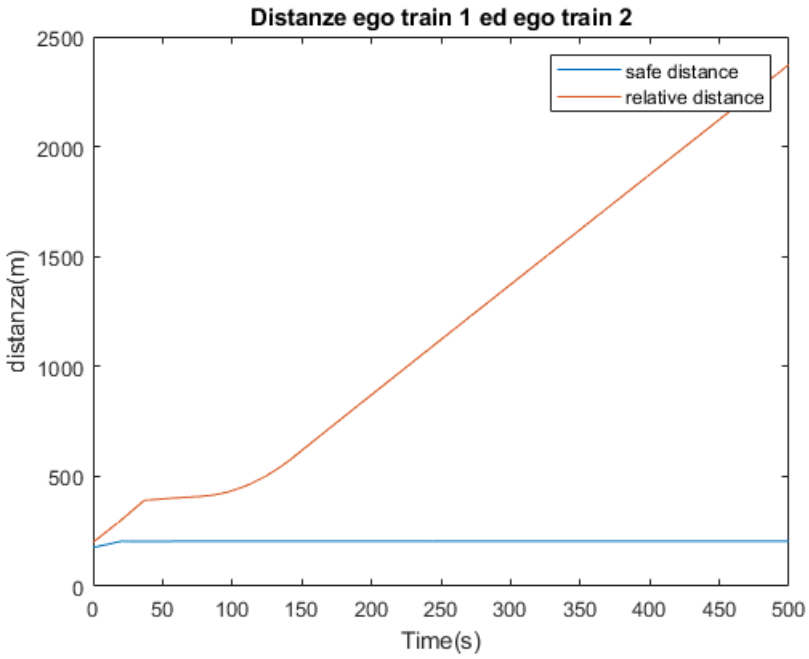


Figura 4.14

Capitolo 5

Conclusioni

Il trasporto ferroviario è una delle modalità di trasporto che sicuramente negli ultimi anni sta vivendo una fase di continua e rapida crescita, con ingenti investimenti atti al suo sviluppo. Ciò è dovuto a diversi fattori che lo rendono più competitivo rispetto alle altre modalità di trasporto: il treno è il mezzo di trasporto meccanico via terra più efficiente dal punto di vista energetico. Le rotaie, fornendo un supporto piano e molto solido alle ruote, ne permettono il rotolamento con un attrito più basso rispetto ai sistemi equivalenti su gomma e strada, dunque ne consegue che diminuendo le perdite energetiche per attrito è sufficiente una potenza minore per trainare quantità maggiori di persone e merci. Altri vantaggi sono: ridotta emissione di sostanze inquinanti per passeggero trasportato, riduzione del traffico sulle strade, uso più efficiente dello spazio (a parità di tempo, una linea ferroviaria a doppio binario porta un numero di passeggeri o di tonnellate maggiore di una strada a quattro corsie) e maggiore sicurezza e comfort dei passeggeri. Molte delle ferrovie ad alta velocità sono correntemente basate su ERTMS L2 che consente un buon sfruttamento della capacità della linea tuttavia ancora lontano dall'efficienza teorica massima: il Virtual Coupling appare quindi una soluzione molto promettente per incrementare la capacità delle reti ferroviarie, la loro interoperabilità e rendere più accattivante agli occhi dei clienti l'utilizzo del treno come mezzo di

trasporto, grazie al miglioramento del servizio in termini di costo, puntualità e flessibilità, oltre che di sicurezza.

Dalle simulazioni effettuate possiamo affermare che l'impiego di un agente intelligente addestrato per il controllo di velocità adattivo risulta essere efficace nelle situazioni ordinarie per realizzare accoppiamento virtuale. Difatti dai risultati ottenuti si riesce a confermare il corretto funzionamento, in svariati casi simulativi verosimili, del controllo tramite agente DDPG per l'accoppiamento e disaccoppiamento di un convoglio di treni. L'approccio impiegato presenta inevitabilmente dei vincoli, infatti, come nella maggior parte degli algoritmi Model-Free, anche per il DDPG il numero di episodi minimi al fine di addestrare l'agente in maniera opportuna è molto elevato rischiando una divergenza rispetto all'obiettivo di controllo. Inoltre l'utilizzo di approssimatori di funzioni non lineari annulla qualsiasi garanzia di convergenza.

Sebbene questa tesi fornisca un'analisi preliminare dell'accoppiamento virtuale ferroviario, le sfide poste dal coordinamento sicuro dei convogli di treni accoppiati aprono diverse direzioni future come presupposto per una reale attuazione. A tal fine, sono necessari ulteriori sforzi di ricerca per valutare le implicazioni dell'accoppiamento virtuale negli scenari operativi ERTMS.

Bibliografia

[1] Ankit Choudhary. Deep q-learning. `https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python`.

[2] Towards data science. Deep deterministic policy gradient: Ddpg explained. [https://towardsdatascience.com/deep-deterministic-policy-gradients-explained-2d94655a9b7b].

[3] Towards data science. Deep deterministic policy gradient (ddpg): Theory and implementation. [https://towardsdatascience.com/deep-deterministic-policy-gradient-ddpg-theory-and-implementation-747a3010e82f].

[4] Francesco Flammini, Stefano Marrone, Roberto Nardone, Alberto Petrillo, Stefania Santini, and Valeria Vittorini. Towards railway virtual coupling. In *2018 IEEE International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference (ESARS-ITEC)*, pages 1–6. IEEE, 2018.

[5] Crescenzio Gallo and Michelangelo De Bonis. Reti neurali artificiali tutorial (draft). //, ||.

[6] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–18, 2021.

[7] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[8] Deep Lizard. Expected return. https://deeplizard.com/learn/video/a-SnJtmBtyA.

[9] Deep Lizard. Optimal policies. https://deeplizard.com/learn/video/rP4oEpQbDm4.

[10] Deep Lizard. Structuring a reinforcement learning problem. https://deeplizard.com/learn/video/my207WNoeyA.

[11] Xiaolin Luo, Tao Tang, Hongjie Liu, Lei Zhang, and Kaicheng Li. An adaptive model predictive control system for virtual coupling in metros. In *Actuators*, volume 10, page 178. Multidisciplinary Digital Publishing Institute, 2021.

[12] Mathworks. Deep deterministic policy gradient agents. https://it.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html.

[13] Mathworks. Reinforcement learning. https://it.mathworks.com/discovery/reinforcement-learning.html.

[14] Mathworks. Reinforcement learning toolbox. https://it.mathworks.com/products/reinforcement-learning.html.

[15] Mathworks. Train ddpg agent for adaptive cruise control. https://it.mathworks.com/help/reinforcement-learning/ug/train-ddpg-agent-for-adaptive-cruise-control.html.

[16] OpenAI. Deep deterministic policy gradient. https://spinningup.openai.com/en/latest/algorithms/ddpg.html.

[17] Riccardo Parise, Holger Dittus, Joachim Winter, and Andreas Lehner. Reasoning functional requirements for virtually coupled train sets: Communication. *IEEE Communications Magazine*, 57(9):12–17, 2019.

[18] Comunità scientifica delle ferrovie. Ertms: "european rail traffic management system". https://www.ferrovie.academy/ferrovie-ertms-etcs-come-funziona.

[19] Spindox. Machine learning: reti neurali demistificate. https://www.spindox.it/it/blog/ml1-reti-neurali-demistificate/.

[20] DLR Transport. Innotrans2018: Dynamic virtual coupling (dvc). https://verkehrsforschung.dlr.de/en/innotrans-2018/dynamisches-fluegeln-1.