POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi Laurea Magistrale

Utilizzo di Machine Learning per la rilevazione di attacchi al sistema di controllo automatico della velocità

Relatori
prof. Fulvio Risso
dott. Marco Iorio

Candidato
Daniele Gatti

Tutor aziendale
Italdesign
dott. ing. Massimo Reineri

Anno accademico 2019-2020

Alla mia famiglia.

Ringraziamenti

Ringrazio il professor Fulvio Risso per l'opportunità e la fiducia concessami con questo lavoro di tesi, in particolare il dott. Marco Iorio che mi ha seguito, con la sua infinita disponibilità, in ogni step della realizzazione dell'elaborato.

I più grandi ringraziamenti vanno alla mia famiglia, senza i loro insegnamenti e senza il loro supporto, questo lavoro di tesi non esisterebbe nemmeno. Non finirò mai di ringraziarvi per avermi permesso di arrivare fin qui.

Riassunto

I sistemi di trasporto intelligenti cooperativi (C-ITS, Cooperative Intelligent Transport Systems) consentono agli attori dei sistemi di trasporto di interagire e collaborare scambiando informazioni tramite reti di comunicazione wireless. La comunicazione wireless tra i veicoli in questi sistemi viene spesso definita comunicazione da veicolo a veicolo (V2V), mentre la comunicazione tra veicoli e infrastrutture stradali è nota come comunicazione da veicolo a infrastruttura (V2I). Insieme, questi due tipi di comunicazione sono indicati V2X, che sta per veicolo-a-tutto. Oltre alla connettività, gli attori dei C-ITS utilizzano la comunicazione wireless e l'automazione dei veicoli per cooperare e interagire tra loro al fine di raggiungere gli obiettivi di miglioramento dei sistemi di trasporto mediante un uso più efficiente dello spazio stradale ed un migliore consumo di carburante. Ci sono diverse sfide da superare prima che possano essere implementate e dispiegate su strade pubbliche.

Tra le sfide più importanti vi sono i test e la valutazione per garantire la sicurezza delle applicazioni C-ITS. Questa tesi si concentra sul test e la valutazione delle applicazioni C-ITS per quanto riguarda la loro sicurezza mediante simulazioni. Un'applicazione principale dei C-ITS è il platooning, che è gestito da algoritmi di Cooperative Adaptive Cruise Control (CACC). Il framework di simulazione è stato convalidato dimostrando diversi casi d'uso nell'ambito del platooning. In particolare, viene utilizzato per dimostrare e analizzare la sicurezza del platooning.

La tecnica adottata per identificare gli attacchi in corso è in grado di valutare la coerenza delle informazioni ricevute da diverse fonti (es. veicoli e/o sensori locali), e si pone come obbiettivi principali: rilevare gli attacchi in tempi brevissimi ed evitare situazioni catastrofiche, quali incidenti. Principalmente verrà esplorata la possibilità di applicare diversi algoritmi di apprendimento automatico con il fine di valutare le loro prestazioni confrontando i risultati dei vari approcci. In particolare, si sono ottenuti degli ottimi risultati con algoritmi che facevano uso delle reti neurali ricorrenti, infatti, i valori considerati variano nel tempo seguendo la logica del platooning, rispettando le leggi del moto, quindi hanno una certa dipendenza nel tempo, caratteristica che si sposa perfettamente con modelli ricorrenti. Tale approccio è stato in grado di rilevare quasi la totalità degli attacchi (tra il 95% e il 100%) con tempi medi di detection tra 0.8 e 6 secondi e con un tasso di incedenti intorno allo 0.1%.

Indice

Capitolo 1

Introduzione

L'inquinamento atmosferico, le fonti limitate di combustibili fossili e il bisogno sempre crescente di trasporti rapidi e sicuri per la popolazione mondiale in crescita sono le maggiori preoccupazioni in tutto il mondo. L'aumento della domanda di trasporto ha comportato una maggiore intensità del traffico, tempi di viaggio più lunghi e un maggior rischio di incidenti. I sistemi avanzati di assistenza alla guida (Advanced Driver Assistance Systems - ADAS) possono aiutare ad alleviare e ridurre questi problemi. Gli ADAS utilizzano sensori a bordo veicolo a basso costo e migliorano l'interazione tra il conducente e l'ambiente che lo circonda. Questi sistemi possono ridurre gli errori di guida migliorando aspetti come la sicurezza, il comfort, il risparmio di carburante e il flusso del traffico riducendo al minimo le azioni necessarie da parte del conducente, soprattutto in situazioni di emergenza.

Tra i vari tipi di ADAS, quelle più promettenti sono l'Adaptive Cruise Control (ACC) e il Cooperative Adaptive Cruise Control (CACC). Nell'ACC, uno o più radar di bordo sono in grado di raccogliere molteplici informazioni tra qui la distanza e la velocità tra due veicoli in corsa. L'ACC utilizza questi dati per seguire una traiettoria di riferimento e per mantenere un gap inter-veicolare sicuro [1]. Il CACC estende l'attuale versione dell'ACC utilizzando le comunicazioni wireless Vehicle-to-Vehicle (V2V) e Vehicle-to-Infrastructure (V2I). V2V fornisce informazioni più utili sul veicolo precedente e anche su altri veicoli limitrofi, con ritardi inferiori rispetto ai radar. Utilizzando V2V e radar, il CACC può consentire ad un gruppo di veicoli di creare un plotone e di muoversi in modo sicuro, efficiente e cooperativo per brevi o lunghe distanze, migliorando il throughput del traffico e riducendo gli incidenti.

Le comunicazioni V2V vengono sviluppate all'interno della cosiddetta VANET (Vehicular Ad-hoc Network), questa è una rete dove i nodi corrispondono a veicoli che condividono la stessa area geografica, nei limiti infrastrutturali della tecnologia di comunicazione adottata, tipicamente tecnologie a corto raggio, come le wireless LAN (WLAN) [2], oppure le più recenti LTE e 5G.

1.1 Motivazioni

Sebbene l'introduzione della comunicazione V2X porta molti vantaggi, l'affidamento a dati generati da peer in remoto apre, allo stesso tempo, la possibilità di attacchi dannosi; infatti dati falsificati possono causare calcoli errati che, in casi estremi, potrebbero causare un arresto anomalo oppure nelle peggiori delle ipotesi delle collisioni. Negli ultimi anni, gli organismi di standardizzazione hanno compiuto notevoli sforzi per definire i livelli di rete necessari per abilitare la VANET, insieme alle proprietà di sicurezza associate che devono essere garantite nello scambio V2V, come la loro autenticità, integrità e riservatezza, se necessario. Tuttavia, queste proprietà si rivolgono esclusivamente alla protezione dai cosiddetti aggressori esterni, quelli non autorizzati a partecipare alla comunicazione.

In questa ricerca ci si vuole concentrare sugli attacchi condotti da avversari interni (ad esempio auto hackerate), quelli a cui attualmente viene concesso l'accesso alla rete come veicoli legittimi. In tale situazione, i metodi tradizionali basati sulla crittografia non sono adatti per rilevare comportamenti scorretti, poiché gli stessi aggressori possiedono anche le chiavi necessarie per stabilire la comunicazione. Quindi oltre a mantenere e migliorare gli attuali protocolli di sicurezza di autenticazione, si rende necessaria una logica in grado di verificare la plausibilità dei messaggi ricevuti.

In particolare, precedenti ricerche hanno dimostrato che gli attacchi più significativi che possono mettere a serio repentaglio la sicurezza di questi algoritmi sono principalmente due:

- gli attacchi di tipo *Jamming*, dove nel dettaglio si va a ritardare o impedire la ricezione dei pacchetti, effettuando ad esempio attacchi DoS;
- gli attacchi di tipo *Injection*, dove un'attaccante è in grado di modificare il contenuto dei pacchetti leciti mitigando la corretta interpretazione di questi;

In questo elaborato si andranno ad analizzare in profondità gli attacchi di iniezione, ritenuti più disastrosi soprattutto perché più difficili da rilevare rispetto ai primi, infatti questi potrebbero essere semplicemente rilevati dal protocollo di comunicazione utilizzato.

1.2 Obiettivi

Questo progetto di ricerca si propone di indagare la possibilità di estendere e generalizzare un primo approccio sviluppato in un precedente studio [3]; brevemente, la tecnica in questione mirava ad identificare gli attacchi in corso valutando la coerenza delle informazioni ricevute da diverse fonti (ad esempio diversi veicoli e/o sensori locali). Sebbene i risultati ottenuti fossero promettenti, questo approccio dipendeva da una serie di parametri sintonizzati manualmente, rendendo difficile l'adattamento a diversi scenari.

Pertanto l'obbiettivo principale di questa tesi è progettare un algoritmo per mettere a punto automaticamente i diversi parametri in base alla situazione specifica, col fine di migliorare le prestazioni e abbreviare il tempo di rilevamento per prevenire il verificarsi di incidenti catastrofici. In particolare, verrà esplorata la possibilità di applicare diversi algoritmi di apprendimento automatico alla circostanza specifica e valutare le loro prestazioni confrontando i risultati dei vari approcci.

1.3 Struttura dell'elaborato

Questa tesi è organizzata come segue:

- Nel Capitolo 2 verrà introdotto il background necessario per l'applicazione degli algoritmi di CACC, valutandone l'evoluzione dall'ACC. Verrà data una panoramica generale per quanto concerne gli aspetti di sicurezza in una rete VANET, alcuni cenni sul Kalman Filter.

- Nel Capitolo 3 un'introduzione sugli algoritmi principali di Machine Learning approfondendo le cosiddette RNN (*Recurrent Neural Network*), reti neurali ricorrenti.

- Nel Capitolo 4 sarà presente una panoramica degli attacchi pubblicati e una revisione dello stato dell'arte dei sistemi di rilevamento più recenti. Fornendo anche una panoramica di applicazioni delle RNN, e lo stato dell'arte sul rilevamento di anomalie.

- All'interno del Capitolo 5 verranno introdotte le varie soluzione elaborate in questa ricerca, descrivendo il modello di minaccia e definendo il modo in cui verranno valutate le prestazioni. Seguirà un'introduzione del rilevamento basato su algoritmi di Machine Learning; spiegando, come questi metodi vengono valutati e confrontati.

- Il Capitolo 6 presenterà la configurazione sperimentale e come sono stati preparati i dati. Seguirà una spiegazione di come sono stati addestrati i modelli e le motivazioni che hanno portato a questo tipo training. La sezione terminerà presentando i risultati del rilevamento, mostrando come si comportano i rilevatori nei vari scenari proposti.

- Infine, nel Capitolo 7 verranno presentate le conclusioni e verrà definito il percorso per un lavoro futuro.

Capitolo 2

Background

2.1 Introduzione

In questo capitolo si andranno ad analizzare nel dettaglio i requisiti richiesti per comprendere al meglio il lavoro svolto, esso sarà strutturato come segue: nella prima parte verrà data un'inquadratura generale su cosa sia il CACC e come mai c'è molta ricerca in quest'area, successivamente verranno dati alcuni cenni sul V2V e sugli aspetti di sicurezza legati alle VANET, dopodiché nel capitolo successivo si andranno ad analizzare i tipici algoritmi di Machine Learning fino alle reti neurali ricorrenti.

I sistemi avanzati di assistenza alla guida (ADAS) come l'Adaptive Cruise Control (ACC) e il Cooperative Adaptive Cruise Control (CACC) si sono dimostrati utili al fine di rendere il traffico il più efficiente possibile. Sebbene la caratteristica principale sia quella di ridurre il divario tra i veicoli, questi sistemi hanno diversi altri vantaggi derivati da quest'ultima caratteristica, come ad esempio la riduzione della resistenza aerodinamica e il calo delle emissioni [4]. Questo è uno dei motivi principali del crescente interesse della ricerca in questo settore. Inoltre, il tempo di risposta per il sistema ACC è dell'ordine di 0,1–0,2 sec., che è trascurabile se confrontato con il tempo di reazione umano di circa 1 s [5]. I plotoni di veicoli impiegano indistintamente l'ACC o il CACC o addirittura una combinazione di entrambi per il controllo longitudinale. Pertanto, in un plotone il veicolo anteriore acquisisce maggiore importanza rispetto ai veicoli che seguono; maggiori dettagli del perché questo accade saranno più chiari al termine di questo capitolo.

2.2 Panoramica del Vehicle Platooning

I veicoli autonomi sono in questo momento un tema molto gettonato. Il platooning dei veicoli può essere visto come la fase iniziale della guida autonoma, consentendo ai veicoli di seguire il veicolo principale senza alcun input da parte del conducente.

Il Vehicle Platooning può essere definito come il raggruppamento di più veicoli uno dopo l'altro in modo tale che tutti i veicoli seguano il veicolo di testa con una distanza definita tra di loro. Il veicolo anteriore in uno scenario realistico è guidato manualmente mentre tutti gli altri veicoli ricevono i vari input necessari per seguire il veicolo in testa.

Un plotone di veicoli ideale è quello in cui tutti i veicoli possono mantenere le distanze intra-veicolari desiderate in ogni momento. L'errore di questa implementazione è definito dalla deviazione della posizione di un veicolo dalla posizione desiderata; questo errore viene quindi utilizzato per spingere il veicolo nella posizione desiderata.

A seconda dei veicoli coinvolti nel plotone, il plotone può essere classificato in:

- plotone *omogeneo*: ovvero tutti i veicoli nel plotone sono uguali, e quindi anche i controller utilizzati per i veicoli nel plotone dovrebbero essere gli stessi.
- oppure, plotone *eterogeneo*: scenario più realistico dove i veicoli considerati sono differenti, ad esempio combinazioni di auto e autocarri, pertanto si rendono necessari dei controlli specifici in grado di gestire questo tipo di situazioni.

Una delle proprietà importanti da considerare nel platooning è il progresso (*headway*). L'headway può essere definito come una misura della distanza o del tempo tra due veicoli. Quindi, se un veicolo impiega 2 secondi per raggiungere la parte posteriore del veicolo che lo precede, il tempo di avanzamento è di 2 secondi. Quando il progresso è misurato in distanza, si chiama "distance headway"; se misurato nel tempo, si chiama "time headway". Quindi, la distanza che un veicolo deve mantenere con il veicolo precedente viene definita utilizzando il parametro di headway. Quando un veicolo viaggia alta velocità, è opportuno avere una distanza maggiore tra esso e il veicolo che precede per ragioni di sicurezza. Quindi, avere un progresso costante nel tempo (constant time headway) ha senso per questo tipo di applicazioni, riducendo le oscillazioni nel moto del veicolo.

2.3 Architettura di controllo per un sistema autostradale automatizzato

Una popolare architettura per un sistema autostradale automatizzato [6] è gerarchica ed è composta da 4 livelli, come mostrato in Figura 2.1. I 4 livelli sono:

1. Il livello di rete
2. Il livello di collegamento
3. Lo strato di coordinamento

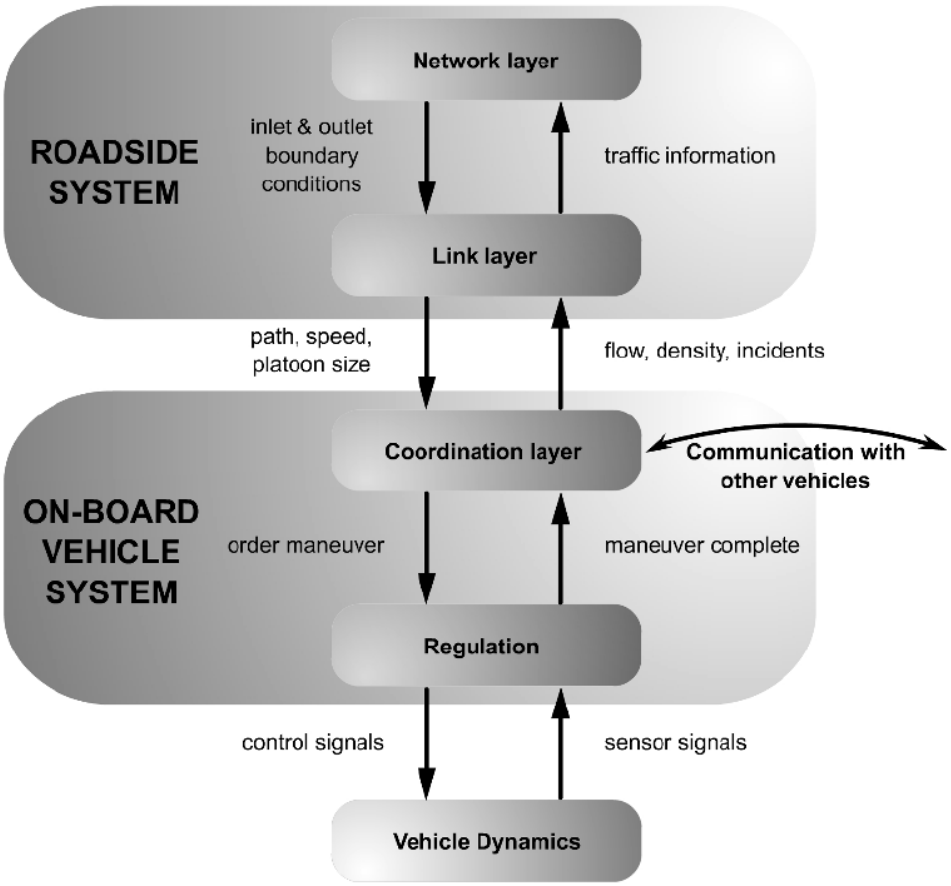

Figura 2.1: Architettura di controllo di alto livello di un sistema automatizzato.

4. Lo strato di regolazione

Il livello di rete controlla il traffico in entrata sull'intera rete e assegna un percorso a ciascun veicolo quando entra nel sistema.

Lo strato di collegamento trasmette i valori target della velocità e le dimensioni del plotone per quella sezione di strada. Questi valori target sono determinati in base allo stato del traffico (considerando densità, velocità, flusso), e stima la proporzione di veicoli destinati a varie uscite consigliando ai veicoli dove cambiare corsia per raggiungere le uscite. Riceve informazioni su incidenti o congestioni e riassegna di conseguenza i percorsi dei veicoli.

Lo strato di coordinamento risiede su ogni veicolo. Determina quale manovra avviare in qualsiasi momento per conformarsi al percorso assegnato; coordina inoltre tale manovra con i veicoli vicini in modo che la manovra possa essere eseguita in sicurezza.

Queste manovre eseguite dallo strato di regolazione, includono:

- mantenimento della corsia, controllo della velocità o l'inseguimento di un altro veicolo.

- cambio di corsia, uscita/ingresso autostradale, distaccamento da un plotone per unirsi ad un altro.

Lo strato di regolazione, acquisisce maggiore importanza proprio perché è responsabile di tutte le manovre possibili in uno scenario cooperativo, quindi volendo astrarre la sua struttura, essa può essere progettata ancora una volta in modo gerarchico, dove, però in questo caso i livelli di controllo sono soltanto due, come in Figura 2.2:

- Il livello superiore che determina l'accelerazione longitudinale desiderata per ogni veicolo.

- Mentre il livello inferiore determina i comandi dell'acceleratore e/o del freno necessari per seguire l'accelerazione desiderata.

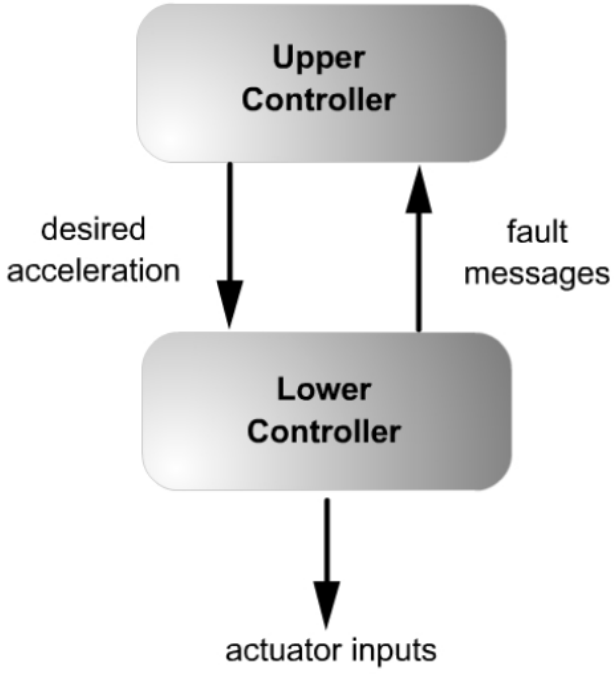

Figura 2.2: Struttura gerarchica del sistema di controllo.

2.4 Evoluzione del Cruise Control

I controllori longitudinali giocano un ruolo centrale nel controllo del veicolo, essendo responsabili del mantenimento della velocità, dell'accelerazione e della distanza tra le vetture consecutive sulla stessa corsia. Un'azione di controllo in questi scenari viene eseguita mediante l'azionamento di una valvola a farfalla o di un freno. In questa sezione verrà analizzata l'evoluzione che nel tempo si ha avuto dal semplice Cruise Control al più complesso e moderno CACC.

Il Cruise Control è uno dei controller longitudinali di base. Il suo obiettivo è mantenere la velocità preimpostata dal guidatore regolando la posizione dell'acceleratore. Il sistema non necessita di sensori di rilevamento, quindi non reagisce all'avvicinamento pericoloso al veicolo che precede. In una situazione così pericolosa, il conducente è obbligato a disinserire il sistema assumendo il controllo manuale del pedale del freno, per questo motivo il primo aggiornamento che ha ricevuto questo controllore è stato aggiungere un parametro adattivo, trasformandolo nell'Adaptive Cruise Control, abbreviato comunemente ACC.

Il sistema di Adaptive Cruise Control è quindi un'estensione del Cruise Control. Esso utilizza un radar o un altro sensore di rilevamento a bordo, in grado di misurare la distanza relativa e la velocità relativa del veicolo precedente (spesso chiamato anche veicolo host). L'estensione principale rispetto al CC è la capacità di regolare la velocità del veicolo in modo tale da mantenere una distanza desiderata tra i veicoli controllando l'azione dell'acceleratore e del freno. Quando l'auto davanti esce dalla corsia, un sistema ACC ripristina automaticamente la velocità di riferimento impostata manualmente dal pilota. L'Adaptive Cruise Control funziona in due modalità:

1. controllo della velocità
2. controllo della distanza, chiamata anche time gap control [7].

La transizione tra questi due stati avviene automaticamente, senza l'influenza del guidatore.

Quando viene implementato l'ACC, il ritardo di rilevamento e risposta è cumulativo dal leader ai veicoli più a valle (tranne nei rari casi in cui i radar possono rilevare i movimenti di un veicolo che precede il suo predecessore) dando luogo al cosiddetto effetto molla. Se il ritardo totale (del sensore, dell'elaborazione dei dati e del controllo) è di 1,5s (per un veicolo ad alte prestazioni), il quarto veicolo impiegherebbe 4,5 s per rilevare indirettamente cosa è successo al veicolo di testa. Questo ritardo potrebbe essere drasticamente ridotto se il veicolo leader potesse comunicare direttamente il suo stato a tutti i suoi inseguitori (ad esempio, ipotizzando che il ritardo di lettura del sensore sia equivalente al ritardo di trasmissione, il ritardo totale tra il primo e il quarto potrebbe scendere ad 1,5s).

Come miglioramento del sistema ACC, sono stati introdotti i sistemi di Cooperative Adaptive Cruise Control, che oltre ai dati ricevuti dai sensori di bordo utilizzano

quelli scambiati dai veicoli tramite la comunicazione wireless. Tale miglioramento consente di ridurre la distanza tra due veicoli consecutivi a pochi metri [8], il che consente di migliorare il rendimento del traffico della rete stradale; nella sezione successiva verranno analizzati nel dettaglio i vantaggi che questo tipo di sistema potrebbe portare se venisse applicato in larga scala.

2.5 Motivazioni legate allo sviluppo del CACC

Il primo aspetto che verrà analizzato è legato alla densità del traffico. La densità del traffico è in aumento a causa del numero crescente di veicoli. La Figura 2.3 mostra la stima del numero di auto presenti sulla strada nel periodo 1950-2030, questo denota come unendo auto e autocarri nel 2030 si stima ci siano quasi 2 miliardi di veicoli [9].

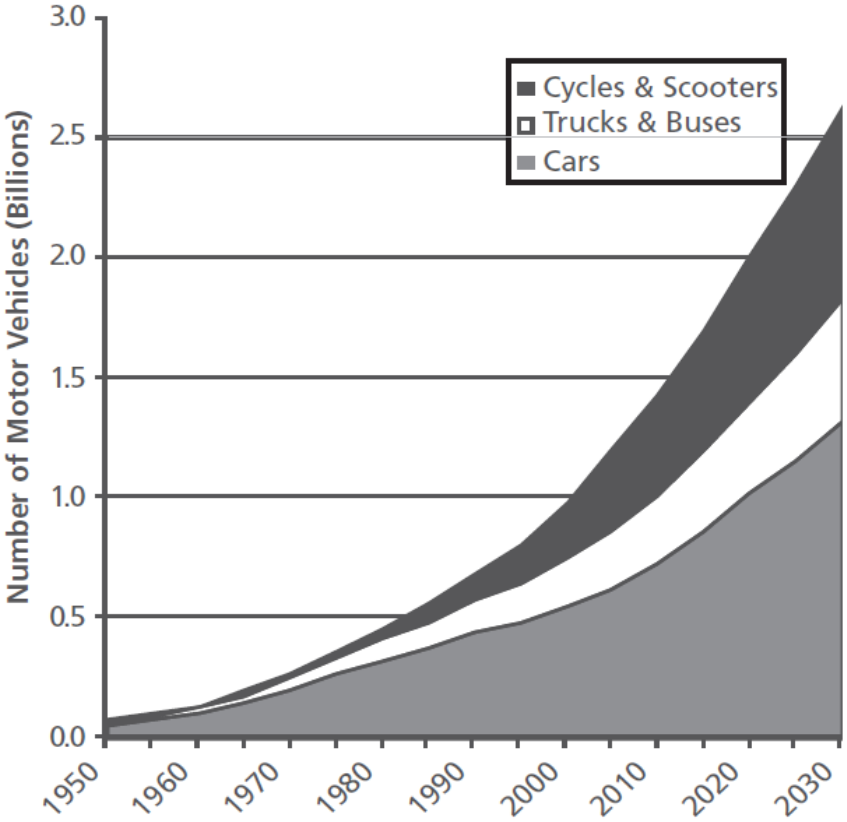

Figura 2.3: stima del numero di auto presenti sulla strada nel periodo 1950-2030

Poiché il numero di veicoli aumenta più rapidamente rispetto alla costruzione di strade pubbliche, è più facile che si verifichino delle congestioni con tempi di percorrenza più lunghi e più incidenti. La maggior parte di queste situazioni si

verifica a causa dell'intervento umano, con tempi di reazione lenti e/o decisioni imprudenti.

Il desiderio è quello di aumentare la capacità stradale rendendo i viaggi più sicuri e confortevoli. Il Cooperative Adaptive Cruise Control (CACC), che è parte del settore automotive, è una soluzione in grado di soddisfare questa richiesta.

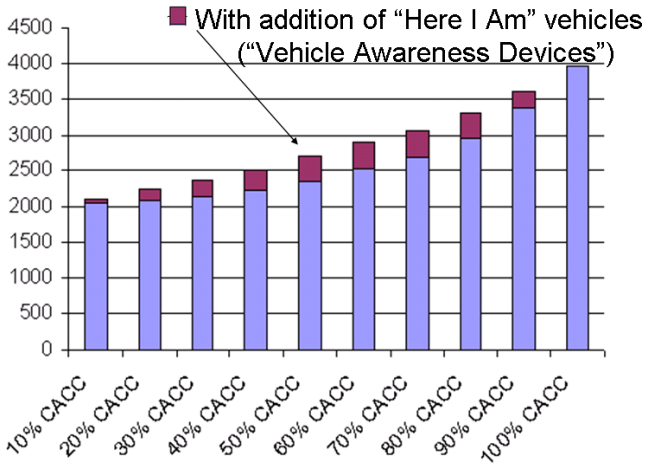

Figura 2.4: Aumento throughput di una corsia all'aumentare del numero di veicoli che implementano il CACC.

Nella Figura 2.4 è anche indicato il miglioramento della capacità di corsia mediante un sistema cosiddetto Here I Am (HIA). Il sistema HIA è una comunicazione radio a corto raggio utilizzata nelle auto a guida manuale [10] (in pratica rappresentano i veicoli leader). Essa scambia la sua velocità e posizione con veicoli con tecnologia CACC, che sono in grado di seguire il veicolo che lo implementa con un gap più breve.

La Figura 2.5 è un'altra rappresentazione di come il throughput stradale aumenta all'aumentare del numero di veicoli che implementano il CACC, confrontandolo anche con l'ACC, mostra gli effetti relativi sulla capacità di corsia dell'ACC a 1.4 s di time gap e al CACC a 0.5 s di intervallo (informazioni fornite da California PATH [11]).

Il restringimento della distanza tra i veicoli consente anche di diminuire il consumo di carburante [12], causato da una riduzione della resistenza aerodinamica, come mostrato in Figura 2.6, dove viene applicato lo stesso concetto utilizzato in F1, ovvero l'effetto scia.

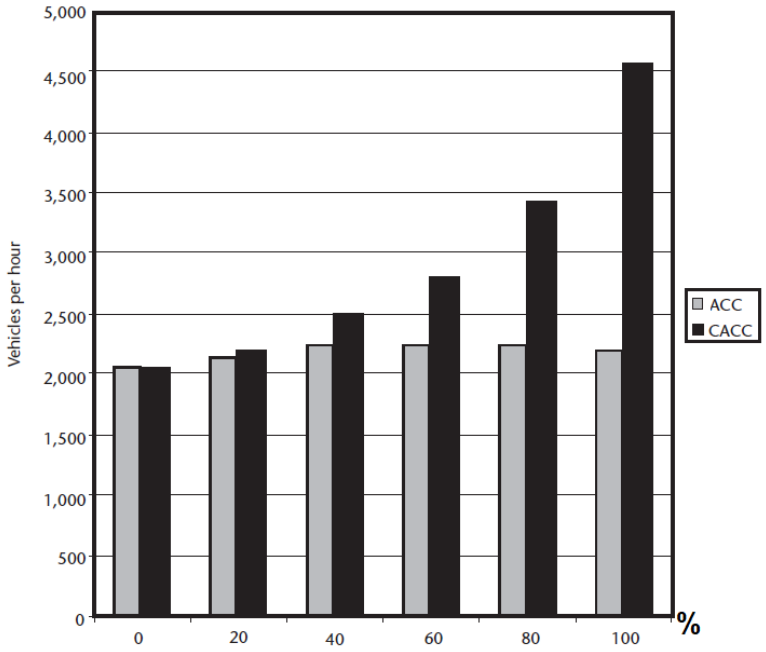

Figura 2.5: Confronto ACC con CACC in termini di throughput stadale.

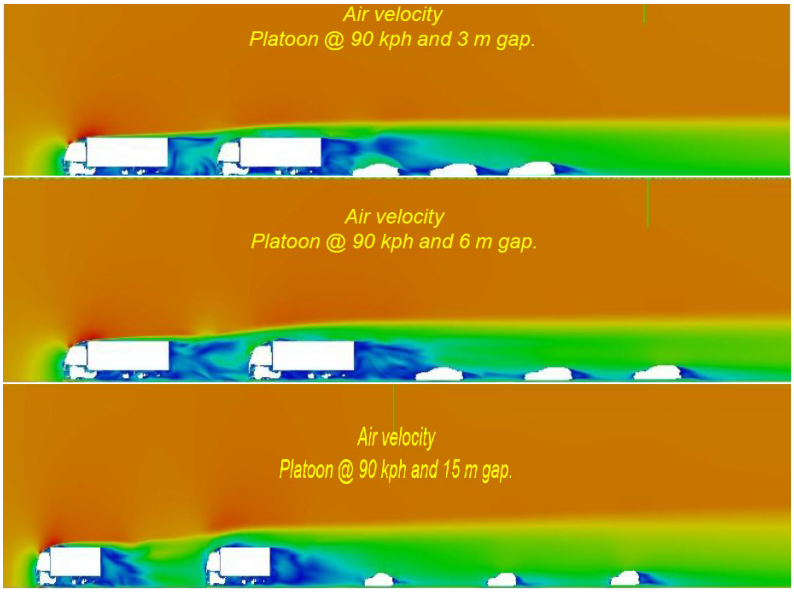

Figura 2.6: Effetto del flusso aerodinamico in base alla distanza [13]

2.6 Cooperative Adaptive Cruise Control

L'inseguimento automatico del veicolo basato sullo scambio di dati tramite comunicazione wireless, oltre ai dati ottenuti dal radar, è comunemente indicato come Cooperative Adaptive Cruise Control (CACC), illustrato in Figura 2.7.

L'idea principale è condividere informazioni come accelerazione, velocità, posizione, tramite comunicazione wireless, per migliorare la reattività del controllore longitudinale, riducendo il ritardo di risposta al comportamento del veicolo precedente.

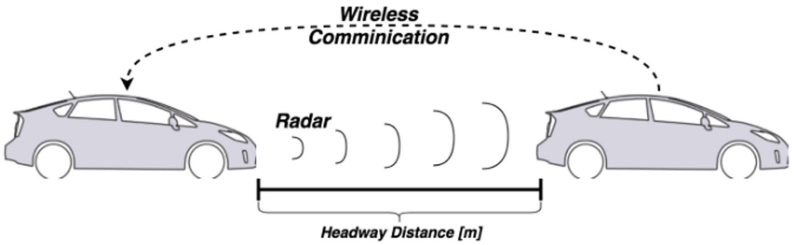

Figura 2.7: Esempio di comunicazione tra due veicoli [14]

Sebbene ACC e CACC siano esempi di guida autonoma di Livello 1 e 2 rispettivamente, come definito sia da SAE che da NHTSA, i vantaggi, soprattutto quelli del CACC del veicolo che segue possono essere ottenuti in ogni scenario e sono prestazioni attribuibili a livelli di automazione più elevati.

Nella Figura 2.7 viene utilizzata una topologia in cui la comunicazione avviene del veicolo leader verso i veicoli successi. Questa è l'unica topologia di comunicazione considerata in questo progetto, analizzata nel dettaglio nella sezione successiva. Altre possibili topologie di comunicazione sono la topologia bidirezionale completa (trasmissione delle informazioni tra tutti i veicoli in entrambe le direzioni) o la topologia bidirezionale con il leader (la trasmissione avviene in entrambe le direzioni ma solo con il leader, gli inseguitori non comunicano tra di loro).

2.7 V2V e VANET

Vehicle-to-Vehicle (V2V) è il concetto base generale su cui i veicoli si scambiano informazioni per prendere decisioni intelligenti. Una comunicazione V2V efficiente avviene quando esistono le cosiddette "pervasive networks" (letteralmente: reti pervasive) che consentono l'accesso onnipresente alle informazioni man mano che diventano disponibili. Le pervasive network consentono il collegamento in rete e il supporto per diverse applicazioni in ambienti diversi, in modo dinamico. Ciò consente l'elaborazione delle applicazioni e delle informazioni e la connettività di rete sia per il terminale fisso che per gli agenti mobili.

Per realizzare la comunicazione V2V, il Medium Access Control (MAC) fisico e i livelli di rete hanno dato luogo a nuove tecniche realizzate su misura per consentire uno scambio rapido e senza interruzioni di informazioni attraverso reti wireless, utilizzando ad esempio reti cellulari 3/4G, WIMAX IEEE 802.16 e Wi-Fi IEEE 802.11, con estensione alla 802.11p, oggetto delle simulazioni effettuate in questa tesi. Esempi applicativi: l'IEEE 802.11ah è un Wi-Fi a lungo raggio che può essere utilizzato soprattutto quando i veicoli sono dispersi per distanze superiori a 1 km. Mentre in ambienti in cui i veicoli circolano a una velocità superiore ai 160 km/h, l'uso di una WiMAX è molto consigliato.

In termini di diverse tecnologie di comunicazione V2V, IEEE 802.11p si basa sul meccanismo Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) ed è stato standardizzato dall'IEEE nel 2010 [15], ulteriori dettagli riguardante questa tecnologia saranno forniti nella sezione successiva.

Grazie a questa comunicazione V2V aggiuntiva, CACC è in grado di raggiungere la stabilità con intervalli di tempo significativamente inferiori ad 1 s. Inoltre, ogni veicolo può ottenere informazioni dai veicoli circostanti utilizzando la comunicazione wireless.

Uno dei componenti critici della comunicazione V2V è la "mobile ad hoc-network" (MANET) che è essenzialmente un sistema autonomo composto da diverse stazioni mobili. Queste reti ad hoc vengono formate in modo dinamico. I sensori (dei veicoli) nelle reti V2V non sono considerati come nodi ordinari perché questi hanno delle capacità particolari, come la raccolta di informazioni altamente dinamiche, l'uscita e l'ingresso continuo in diversi cluster di veicoli sotto reti diverse. In una rete V2V relativamente sviluppata, i nodi sono in grado di trasmettere informazioni a determinati intervalli, ad es. sullo stato della strada, il livello di congestione, segnalazioni di incidenti, ecc. La disponibilità tempestiva di tali informazioni può aiutare gli automobilisti a prendere decisioni intelligenti riducendo congestioni ed incidenti.

Esistono altre varianti della comunicazione V2V. Ad esempio:

- *Vehicle to Infrastructure* (V2I) consente ai veicoli di comunicare con diversi nodi e dispositivi come le unità a bordo strada (roadside-unit RSU).

- *Vehicle to Everything* (V2X) il veicolo è in grado di comunicare letteralmente con tutto ciò che lo circonda, come cartelli, semafori, persone, altri veicoli ecc.

VANET è una sottoclasse delle MANET, applicabile specificatamente per veicoli collegati tramite una rete wireless. Fino a poco tempo fa le VANET venivano utilizzate principalmente per fornire sicurezza e comfort ai conducenti in ambienti veicolari. Una delle caratteristiche distintive delle VANET è la distribuzione incentrata sul contenuto: il contenuto è importante, piuttosto che la fonte, in netto contrasto con Internet in cui un agente richiede informazioni da una fonte specifica.

I protocolli di comunicazione VANET sono simili a quelli utilizzati nelle reti cablate, in cui ogni host ha un indirizzo IP. Tuttavia, qui, assegnare tali indirizzi ai veicoli in movimento è tutt'altro che banale e spesso richiede un server DHCP (Dynamic Host Configuration Protocol), un'eresia per le reti ad hoc che operano senza alcuna infrastruttura, utilizzando protocolli di auto-configurazione.

La topologia della rete in VANET è determinata dalla velocità e dalla posizione del veicolo. La topologia più semplicistica è una peer to peer (P2P) generica, facile da progettare e distribuire in ambienti reali ed è meno costosa di architetture client-server; quindi, altamente adatte al contesto V2V. In una tipica rete V2V, il livello di comunicazione è realizzato mediante l'utilizzo dei tradizionali protocolli di acquisizione delle informazioni e di instradamento come i sistemi di posizionamento (GPS). La topologia della rete è principalmente influenzata da strutture dinamiche a causa dei cluster di veicoli altamente mobili. Le topologie utilizzano spesso algoritmi a grafo come l'Open Shortest Path First (OSPF).

2.7.1 IEEE 802.11p

L' IEEE 802.11p è un'estensione dell'IEEE 802.11, in cui sono stati apportati vari miglioramenti e modifiche per adattarlo alla comunicazione V2X, principalmente per le comunicazioni V2V e V2I. Alcune delle modifiche importanti sono state apportate al livello 2 dello stack OSI del protocollo di comunicazione 802.11 eliminando o accorciando alcune delle sue fasi. I veicoli che utilizzano l'IEEE 802.11p operano sulla banda DSRC (Dedicated Short-Range Communication) a 5,9 GHz negli Stati Uniti e nell'UE, ed utilizzano il CSMA/CA per coordinare l'accesso multiplo, mentre il giapponese l'ARIB STD-T109 utilizza lo stesso strato fisico, ma nella banda di frequenza a 700 MHz ed utilizza un livello MAC adattato che mescola CSMA/CA con TDMA [16].

Poiché la comunicazione V2V e V2I avviene in una banda di frequenza dedicata per la sicurezza, non è necessario eseguire la scansione del canale. Pertanto, IEEE 802.11p elimina il ritardo di trasferimento del livello 2 dovuto alle fasi di rilevamento, autenticazione, associazione e creazione delle chiavi. Tuttavia, esiste una disposizione per proteggere la comunicazione V2I al di fuori del livello MAC. Queste modifiche generali rendono lo standard IEEE 802.11p veloce e adatto alla comunicazione V2X.

In molti contesti, la comunicazione in VANET è ottenuta mediante Dedicated Short-Range Communication (DSRC), proposta come parte dello standard IEEE 802.11p. La tecnologia DSRC è pronta per essere implementata in ambienti reali ottenendo una grande fetta delle attuali comunicazioni. In questi ambienti dedicati i veicoli possono scambiare fino a 10 messaggi in un secondo. L'architettura di base di DSRC, concettualizzata sull'idea presentata in [17], è mostrata in Figura 2.8.

Come si può vedere nella Figura 2.8, il Robust Header Compression (ROHC) viene utilizzato nella compressione dei pacchetti in modo che siano conformi al throughput del canale di comunicazione. In particolare, ROHC è uno standard che

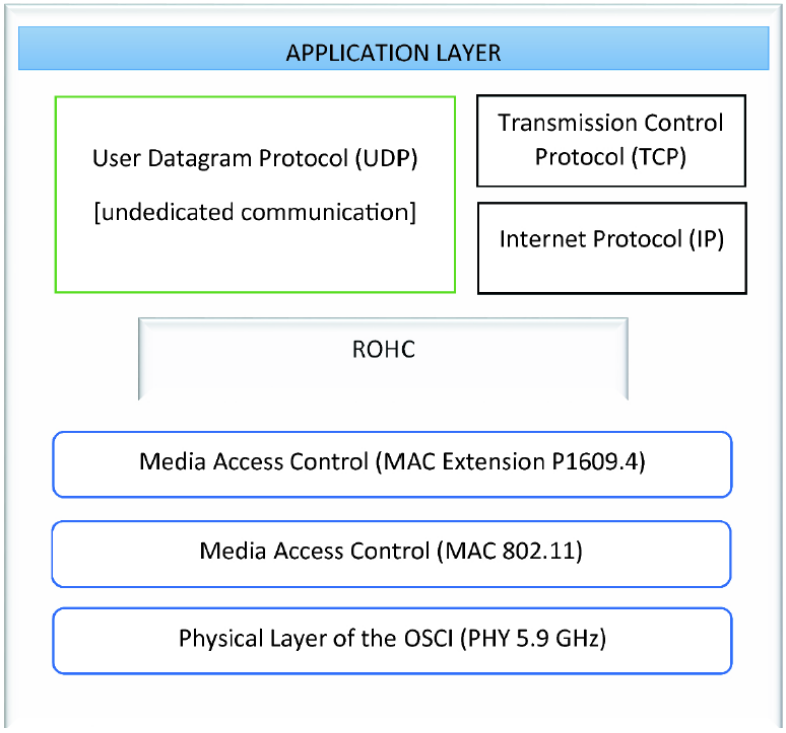

Figura 2.8: Architettura di base del Dedicated Short-Range Communication (DSRC).

comprime le intestazioni IP, UDP, UDP-Lite, RTP e TCP dei pacchetti Internet. Una volta che le intestazioni sono state compresse, i pacchetti sono in grado di risalire il canale di comunicazione attraverso i livelli MAC (Media Access Control) e fisico (PHY).

Sebbene l'IEEE 802.11p sia stato migliorato a livello PHY, MAC e di rete per incrementare le prestazioni e ridurre al minimo la latenza dei livelli 2 e 3, deve comunque affrontare diverse sfide, come:

- Bassa diffusione: a differenza dell'implementazione cellulare e Wi-Fi, l'implementazione basata su DSRC/IEEE 802.11p è diffuso solo negli Stati Uniti, in Europa e in Giappone è ancora in fase di implementazione.

- Elevati costi di implementazione: si stima che un'implementazione tale da fornire copertura completa su autostrade e aree urbane sia molto costosa.

- Velocità di trasmissione bassa: DSRC/IEEE 802.11p può fornire un massimo di 54 Mbps quando vengono combinati due canali adiacenti di 10 MHz. Oggigiorno la maggior parte delle applicazioni richiede un'elevata velocità di trasferimento dati e necessita di un elevata larghezza di banda; quindi, la velocità fornita può essere un collo di bottiglia per varie applicazioni V2X,

basti pensare all'invio dei dati forniti dai vari sensori, come ad esempio le telecamere in altissima risoluzione.

Nel 2018 è stato annunciato dall'IEEE la formazione di un nuovo gruppo di studio con l'obiettivo di migliorare la tecnologia 802.11 esistente per le comunicazioni V2X di prossima generazione, si chiamerà 802.11bd e sfrutterà l'infrastruttura 5G, migliorando le VANET per quanto riguarda la latenza dei dati, la copertura e la QoS [18].

2.7.2 Requisiti di sicurezza

Fornire sicurezza nelle VANET è un compito impegnativo a causa della loro elevata mobilità, dinamicità e le frequenti interruzioni dei collegamenti wireless non aiutano. Anche se esiste un modello di sicurezza di base comune per le VANET, applicazioni diverse possono richiedere misure di sicurezza aggiuntive. Ad esempio, un'applicazione di infotainment necessita di un approccio di sicurezza diverso rispetto a un'applicazione di sicurezza che diffonde un messaggio di allarme. Ciò aumenta il costo delle comunicazioni nelle VANET, e il fatto che sia una rete auto-organizzata senza infrastruttura la rende più vulnerabile agli attacchi.

Inoltre, l'utilizzo di tecnologie wireless rende queste reti vulnerabili a minacce e attacchi che possono sfruttare la tecnologia radio utilizzata. L'assenza di sicurezza nelle VANET le espone a diversi tipi di attacchi che possono avere gravi conseguenze per la sicurezza dei conducenti sulla strada, inclusa la morte. La privacy degli utenti deve essere protetta, poiché le informazioni scambiate tra i nodi del veicolo contengono informazioni private sul conducente e sul veicolo. Pertanto, una comunicazione sicura e attenta alla privacy deve essere una componente integrante delle VANET. Inoltre, se un messaggio di sicurezza legittimo inviato da una componente VANET viene modificato, ritardato o eliminato da un aggressore, può avere gravi conseguenze anche per le vite umane. Ciò significa che solo le informazioni legittime e veritiere dovrebbero essere comunicate tramite VANET [19].

I requisiti che una VANET deve avere per essere definita sicura sono:

- **Integrità**: un messaggio inviato da un mittente deve essere ricevuto senza modifiche, il contenuto del messaggio non deve essere modificato, cancellato o eliminato.

- **Autenticazione e identificazione**: tutti i veicoli (nodi) devono essere autenticati prima di essere autorizzati a unirsi e trasmettere qualsiasi dato. Questa autenticazione serve a garantire che a un veicolo dannoso venga negato l'accesso alla rete e che i messaggi inviati da tali nodi vengano bloccati prima di essere comunicati. L'autenticazione può essere ottenuta utilizzando un certificato da parte del mittente.

- **Disponibilità**: gli utenti (veicoli) dovrebbero essere in grado di inviare e ricevere messaggi in ogni momento anche quando una VANET è sottoposta a un attacco come un D/DoS o un attacco jamming o qualsiasi attività dannosa. L'importanza della disponibilità deriva dal fatto che alcuni messaggi sono sensibili al ritardo e devono essere trasmessi in tempo reale, altrimenti perderanno il loro valore.

- **Privacy**: le informazioni sul veicolo devono essere preservate durante la comunicazione come: l'effettiva identità del veicolo, del conducente e del luogo non deve essere divulgata a parti non autorizzate. È necessario mantenere un certo livello di anonimato, anche se alcune autorità dovrebbero essere in grado di ottenere l'identità del conducente in caso di problemi di responsabilità, ad esempio dopo un sinistro stradale.

- **Riservatezza**: i dati trasmessi tra due veicoli o tra un veicolo e una RSU devono essere protetti da perdite e intercettazioni. Questa si ottiene mediante l'attuazione di schemi di crittografia simmetriche o asimmetriche.

- **Non ripudio**: consente di identificare gli aggressori fornendo prove forensi sull'origine del messaggio, utile per monitorare lo scambio di comunicazioni in caso di incidente. Tutte le informazioni sul veicolo, come velocità e timestamp, sono memorizzate in un dispositivo a prova di manomissione e solo le autorità autorizzate possono recuperare tali informazioni.

- **Controllo degli accessi**: si tratta di implementare diritti e privilegi di accesso per gli utenti della rete. Ogni utente deve avere accesso solo ai servizi e alle applicazioni necessari per svolgere il proprio compito e non avere accesso ad altri servizi.

- **Tracciabilità e revoca**: questi sono necessari affinché un'entità autorizzata sia in grado di rintracciare un veicolo dannoso nella rete. L'autorità è responsabile della revoca del certificato del veicolo abusante nella rete.

- **Dati verificati**: solo i messaggi veri e legittimi dovrebbero essere consentiti sulla rete. Questo per escludere messaggi errati e ridurre il rischio di minacce e attacchi che tentano di iniettare false informazioni nella rete.

2.7.3 Possibili minacce

Questa sezione presenta una tassonomia dei diversi attacchi alle VANET rispetto ad alcuni dei requisiti di sicurezza discussi nella sezione precedente, quali: autenticità, disponibilità, integrità, affidabilità, riservatezza e non ripudio.

Attacchi comuni all'autenticità sono [20] [21]:

- Manomissione del messaggio: la comunicazione tra due entità deve essere protetta. In questo tipo di attacco, un messaggio inviato da un utente autenticato viene catturato da un nodo dannoso che manomette il messaggio lo inoltra alla destinazione. Questo attacco viola anche il principio di integrità. Un meccanismo di verifica dei dati può aiutare a mitigare un simile attacco controllando e verificando il contenuto del messaggio.

- Furto d'identità: un avversario utilizza l'identità di un utente affidabile per utilizzare risorse e servizi di rete o per ottenere l'accesso a informazioni sensibili e inviare messaggi fraudolenti sulla rete.

- Replay: in questo attacco, l'avversario replica l'identità di un vero nodo (cioè un veicolo) sulla rete.

- GPS Spoofing: il segnale GPS è vitale nei VANET poiché fornisce informazioni sulla posizione e l'ora di un veicolo. L'avversario cerca di modificare il segnale GPS fornendo un segnale più forte del segnale GPS originale. Ciò fa sì che i veicoli compromessi nella VANET si sincronizzino con il nuovo orario GPS non autorizzato.

- Sybil Attack: un attaccante inserisce false informazioni in una VANET facendo credere che lo stesso veicolo è in più posti contemporaneamente. Questo alla fine induce in errore altri veicoli a prendere decisioni sbagliate sulla base delle false informazioni ricevute. Inoltre, influisce sulla consistenza in tutto il VANET.

Attacchi alla disponibilità della VANET includono [20] [22] [23] [24]:

- Denial of Service (DoS) e Distributed DoS (DDoS): in questo caso, un nodo non è in grado di svolgere le sue normali attività a causa di uno o più utenti malintenzionati che sovraccaricano le risorse del nodo (ad esempio, processore, memoria) con più traffico da gestire. Questo attacco di solito colpisce molti nodi della rete ed è difficile ripristinarlo anche dopo il rilevamento. Può essere lanciato contro nodi veicolari o RSU. Il D/DoS può causare condizioni pericolose, ad esempio la soppressione della diffusione di un messaggio di allarme contenente informazioni per evitare situazioni pericolose sulla strada.

- Black hole: aumenta la latenza e impedisce ad alcuni nodi di ricevere un messaggio in modo tempestivo, influenzando il loro tempo di risposta.

- Spamming: in questo attacco vengono trasmessi messaggi di spam sulla rete occupando inutilmente la larghezza di banda ed aumentando la latenza dei messaggi leciti.

- Reordering: l'attaccante riceve i messaggi e li inoltra ai nodi con un ordine diverso, così che gli altri nodi avviino meccanismi di riordino aumentando ancora una volta la latenza.

- Jamming: lo scopo di questo attacco è interrompere la connessione sovrastando il segnale radio originale con uno più forte.

Tipici attacchi contro la Privacy comprendono [20] [22] [23] [24] [25]:

- Man-in-the-middle: l'avversario si posiziona tra il mittente e il destinatario (V2V o V2I), ciò gli consente di essere in grado di vedere cosa viene comunicato e ottenere l'accesso per modificare, eliminare o rilasciare uno o più messaggi.

- Intercettazione: l'attacco viene eseguito ottenendo la chiave della sessione e ascoltando la sessione di comunicazione.

- Analisi del traffico: lo scopo di questo attacco è raccogliere informazioni sulla vittima e analizzare e classificare i flussi di traffico per eseguire successivamente un attacco contro la vittima designata.

- Home attack: l'avversario dirotta il controllo del veicolo da un altro utente sulla rete. Ciò consente all'avversario di controllare tutti i sistemi del veicolo.

Attacchi contro l'integrità e l'affidabilità sono [20] [23] [26]:

- Replay attack: attacco che consiste nell'inviare in rete un vecchio messaggio presentando agli altri nodi una situazione diversa da quella che c'è attualmente.

- Masquerading (spoofing) attack: l'avversario falsifica l'identità di un altro veicolo nella rete e lo utilizza per iniettare informazioni pre-fabbricate nella rete. Anche questo è considerato un attacco contro l'autenticità. Ad esempio, un avversario può fingere di essere un veicolo di emergenza per liberarsi la strada da solo.

- Illusion attack: in questo attacco, un attaccante inserisce informazioni contraffatte nella rete, come la segnalazione di un incidente o di una congestione stradale. Questo inganna gli altri conducenti sulla strada e li porta a prendere la decisione sbagliata sulla base di informazioni false e fuorvianti.

- Manomissione del messaggio: qui lo scopo è inviare informazioni errate nella rete modificando il contenuto di un messaggio. Ad esempio, l'avversario potrebbe inviare un falso messaggio di congestione su un percorso per fare in modo che altri veicoli sulla strada cambino il loro percorso.

- Timing attack: in questo attacco l'avversario riceve i messaggi, attende un po', quindi li invia. Questa latenza introdotta fa sì che i messaggi arrivino più tardi del previsto, il che vanifica lo scopo di questi messaggi, soprattutto se sono di allarme.

La maggior parte delle minacce e degli attacchi presentati in precedenza possono essere mitigati implementando tecniche crittografiche, firme digitali, protocolli di routing sicuri, sistema di rilevamento delle intrusioni (Intrusion Detection System -IDS) e implementando le cosiddette Public Key Infrastructure (PKI).

Data la sua importanza, i requisiti di privacy e sicurezza sono stati standardizzati in IEEE 1609 e ETSI C-ITS con il fine di ottenere l'autenticazione dei messaggi utilizzando firme digitali e procedure di autenticazione con chiave asimmetrica [27].

2.8 Definizione di guida autonoma secondo la SAE

Una definizione precisa quale sia il compito della guida autonoma può essere derivata dalla SAE, la Society of Automotive Engineers ([28]). Questa associazione lo denomina "Dynamic Driving Task" (DDT), definendolo come "l'insieme delle funzioni operative e strategie elaborate in tempo reale necessarie per far funzionare un veicolo nel traffico stradale".

Secondo la SAE, queste funzioni possono essere:

- Controllo del movimento laterale tramite lo sterzo.
- Controllo del movimento longitudinale del veicolo tramite accelerazione e decelerazione.
- Monitoraggio dell'ambiente tramite rilevamento di oggetti ed eventi, riconoscimento e classificazione.
- Gestire le risposte ad oggetti ed eventi.
- Manovre locali.

Da questi sottoinsiemi è esclusa la schedulazione del viaggio attraverso la scelta della destinazione e il calcolo dei waypoint, come altra funzione strategica che si ritiene non strettamente correlata al Dynamic Driving Task.

SAE definisce anche il concetto di Automated Driving System (ADS) come "l'hardware e il software che sono collettivamente in grado di eseguire l'intera attività di guida autonoma". L'attività di guida autonoma generale può essere eseguita in tutto o in parte da un conducente umano o da un sistema di guida automatizzato. A seconda di quale attore sta eseguendo determinati compiti e quali sono le condizioni operative, il SAE distingue sei livelli esclusivi di automazione della guida:

- Livello 0 o "**No Driving Automation**". In questo livello, l'intera attività di guida viene eseguita dal conducente umano in qualsiasi momento. È possibile, comunque, che a bordo del veicolo siano presenti sistemi di supporto della sicurezza attiva, come l'Anti-blocking System (ABS) o il sistema Electronic Stability Program (ESP).

- Livello 1 o "**Driver Assistance**". A questo livello, un sistema di guida automatizzata può assumere il controllo laterale o longitudinale, ma solo sotto un dominio di progettazione operativo limitato. I restanti compiti sono sotto il controllo del conducente umano. Esempi di ADS corrispondenti a questo livello possono essere l'Adaptive Cruise Control (ACC) o il Lane-Keep Assist (LKA).

- Livello 2 o "**Partial Driving Automation**". In questo livello, un sistema di guida automatizzato può assumere il controllo sia laterale che longitudinale, ma anch'esso solo in un dominio operativo limitato. Il conducente ha la responsabilità di eseguire le altre attività, come il rilevamento e la risposta ad oggetti ed eventi e di supervisionare l'esecuzione delle attività automatizzate. Come nel livello 1 il conducente deve essere reattivo e assumere immediatamente il controllo dell'intero. Un esempio di ADS di livello 2 può essere l'uso cooperativo di un Adaptive Cruise Control e di un Lane Keeping Assist.

- Livello 3 o "**Conditional Driving Automation**". Da questo livello in poi, il sistema di guida automatizzata è in grado di svolgere gran parte delle attività in scenari comunque limitati, ma più ampi dei livelli precedenti. Tuttavia, il conducente umano è responsabile di prendere immediatamente il controllo del veicolo in caso di guasto o comportamento anomalo del sistema.

- Livello 4, o "**High Driving Automation**". A questo livello, il sistema di guida automatizzata assume gli stessi compiti del livello precedente ma è anche responsabile della procedura di fallback in caso di guasto. In questo livello, il conducente umano non è responsabile di alcun compito e non è previsto che sia reattivo a una procedura di fallback.

- Livello 5 o "**Full Driving Automation**". In questo livello, il sistema di guida automatizzata è responsabile di tutte le attività, senza restrizioni. Come nel livello 4, la procedura di fallback è sotto il controllo del sistema, mentre il conducente umano non ha alcuna responsabilità sul controllo del veicolo e sul suo comportamento.

Sebbene la guida autonoma sia un argomento in cui l'industria automobilistica sta investendo in modo importante, allo stato dell'arte della produzione di veicoli e del rilascio del prodotto, i sistemi di guida automatizzata attualmente venduti

appartengono tipicamente al livello 2 della classificazione SAE. Uno degli esempi più importanti di implementazione in questo livello è l'Autopilota Tesla.

In termini di prossimo futuro, molte aziende stanno attualmente testando e convalidando sistemi di guida automatizzata a livello 3 e 4 della classificazione SAE. Uno degli sviluppi più avanzati è portato avanti da Waymo, società controllata da Google che ha recentemente raggiunto i 16 milioni di chilometri percorsi, mentre altre società come Tesla, Uber e GM hanno rilasciato dichiarazioni ottimistiche affermando che i veicoli di livello 3 e 4 potrebbero essere pronti prima del 2025.

2.9 Filtro di Kalman

Il filtro di Kalman è un efficiente ed efficace filtro ricorsivo che a partire da una serie di misure soggette a rumore valuta lo stato di un sistema dinamico, discreto o continuo. Per le sue caratteristiche intrinseche è un filtro ottimo per rumori e disturbi agenti su sistemi a media nulla. Di seguito verrà trattata una breve introduzione sul suo funzionamento elencando le sue caratteristiche principali [29].

Il filtro di Kalman affronta il problema generale del tentativo di stimare lo stato $x \in \Re^n$ di un processo a tempo discreto che è governato dall'equazione alle differenze lineare:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$

con l'uscita $z \in \Re^m$

$$z_k = Hx_k + v_k$$

Dove, le variabili casuali w_k e v_k rappresentano il rumore di processo e il rumore di misurazione (rispettivamente). Si presume che siano indipendenti (l'uno dall'altro) con una distribuzione di probabilità normale:

$$p(w) \sim N(0, Q)$$
$$p(v) \sim N(0, R)$$

In pratica, le matrici di covarianza del rumore di processo Q e del rumore di misura R potrebbero cambiare a ogni passo temporale o misurazione, tuttavia per non complicarne la spiegazione saranno considerate costanti.

Definiamo $\hat{x}_k^- \in R^n$ come la stima a priori al passo k data la conoscenza del processo prima del passo k, e $\hat{x}_k \in \Re^n$ come la stima di stato a posteriori al passo k data la misura z_k. Possiamo quindi definire errori di stima a priori e a posteriori come:

$$e_k^- \equiv x_k - \hat{x}_k^-, \text{ and } e_k \equiv x_k - \hat{x}_k$$

Derivando le equazioni del filtro di Kalman si ottiene un'equazione che calcola una stima di stato a posteriori $\hat{x}_k$ come combinazione lineare di una stima a priori $\hat{x}_{k-}^-$ e una differenza ponderata tra una misurazione effettiva z_k e una previsione di misurazione $H\hat{x}_k^-$, come mostrato di seguito:

$$\hat{x}_k = \hat{x}_k^- + K\left(z_k - H\hat{x}_k^-\right)$$

La differenza $(z_k - H\hat{x}_k^-)$ è chiamata innovazione di misura o residuo. Il residuo riflette la discrepanza tra la misurazione prevista $H\hat{x}_k^-$ e la misurazione effettiva z_k. Un residuo pari a zero significa che sono concordi.

La matrice K $n \times m$ è scelta come guadagno o fattore di miscelazione che minimizza la covarianza dell'errore a posteriori $P_k = E\left[e_k e_k^T\right]$. Una delle possibili K che minimizza l'errore a posteriori è ottenuta da:

$$\begin{aligned} K_k &= P_k^- H^T \left(H P_k^- H^T + R\right)^{-1} \\ &= \frac{P_k^- H^T}{H P_k^- H^T + R} \end{aligned}$$

Alcune conclusioni che si possono sono: man mano che la covarianza dell'errore della stima P_k^- a priori si avvicina a zero, la misurazione effettiva z_k è sempre meno attendibile, mentre la misurazione prevista $H\hat{x}_k^-$ è sempre più attendibile.

Il filtro di Kalman stima un processo utilizzando una sorta di feedback. Pertanto, le equazioni per il filtro di Kalman si dividono in due gruppi: equazioni di aggiornamento temporale ed equazioni di aggiornamento della misurazione. Le equazioni di aggiornamento temporale sono responsabili della proiezione in avanti (nel tempo) dello stato corrente e delle stime di covarianza dell'errore per ottenere le stime a priori per la fase temporale successiva. Le equazioni di aggiornamento delle misurazioni sono responsabili del feedback, ovvero, per incorporare una nuova misurazione nella stima a priori per ottenere una stima a posteriori migliorata.

Le equazioni di aggiornamento temporale possono anche essere pensate come equazioni predittive, mentre le equazioni di aggiornamento delle misurazioni possono essere pensate come equazioni correttrici.

La covarianza del rumore di misura viene solitamente misurata prima della messa in funzione del filtro. La determinazione della covarianza dell'errore di misurazione è generalmente possibile, pertanto si deve essere in grado di effettuare alcune misurazioni del campione off-line. La determinazione della covarianza del rumore di processo è generalmente più difficile in quanto tipicamente non è possibile osservare direttamente il processo che si sta stimando. In entrambi i casi, indipendentemente dal fatto che si disponga o meno di una base razionale per la scelta dei parametri, spesso è possibile ottenere prestazioni del filtro superiori (statisticamente parlando) regolando i parametri del filtro: Q e R. La configurazione viene solitamente eseguita off-line, spesso con l'aiuto di un altro filtro Kalman.

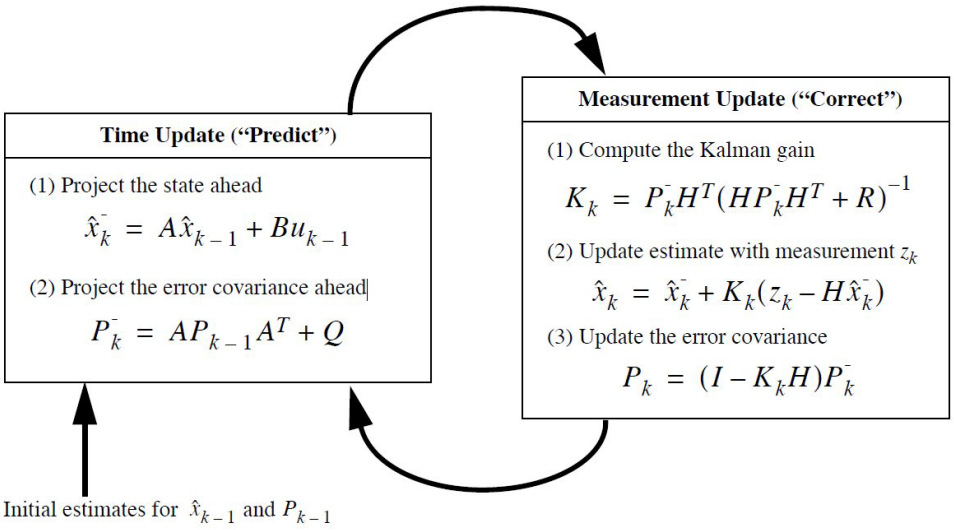

Figura 2.9: Fasi del filtro di Kalman.

Nella Figura 2.9 è possibile osservare tutte le operazioni effettuate dal filtro di Kalman e come vengono aggiornati sia lo stato che la misura.

Concludendo, è bene precisare che la versione del filtro presentata è la più semplice, infatti sono state sviluppate anche estensioni e generalizzazioni al metodo, come l'Extended Kalman Filter (EKF) e l'Unscented Kalman Filter (UKF) che funzionano su sistemi non lineari.

Capitolo 3

Machine Learning

3.1 Introduzione

Oggi si sente sempre più spesso parlare di Machine Learning, Big Data, Data Science e Deep Learning. In questo capitolo si spiegheranno questi concetti, indirizzando il discorso in quelle motivazioni che hanno portato la scelta di una determinata tecnica rispetto ad un'altra.

Il Machine Learning è una branca dell'Intelligenza Artificiale, disciplina che si occupa di elaborare algoritmi "intelligenti", cioè in grado di risolvere compiti tipici dell'intelligenza umana. Gli algoritmi di Machine Learning trasformano automaticamente i dati in informazioni ipotizzando delle regolarità (patterns) nei dati. A partire dalle ipotesi formulate è possibile poi, prendere delle decisioni (istintive) o applicare tecniche inferenziali (quando ci si basa solo su una parte delle informazioni) per ricavare conoscenza, cioè per assegnare un grado di bontà a quanto ipotizzato dall'algoritmo. Il Machine Learning usa tecniche induttive o bottom-up: l'elaboratore parte dai dati per formulare ipotesi che spiegano i dati. L'apprendimento statistico è invece deduttivo o top-down: consente ad un utente di applicare delle tecniche per assegnare un giudizio quantitativo sulla bontà delle proprie ipotesi [30].

Un desiderio comune è quello di far formulare direttamente alla macchina delle ipotesi sui dati non note in partenza all'utente di un database. La formulazione di ipotesi a partire dai dati può essere fatta applicando tecniche di Machine Learning, sviluppando una nuova disciplina chiamata Data Mining, che si occupa dell'uso del Machine Learning per la formulazione di ipotesi non note a priori a partire dai dati memorizzati in un database. Tali regolarità (pattern) nei dati sono usate come modello (ipotesi) per spiegare le informazioni analizzate.

La disciplina del Knowledge Discovery in Databases (nota con l'acronimo KDD) è nata con l'obiettivo di trasformare, quanto più possibile, le ipotesi prodotte durante le attività di Data Mining in conoscenza e quindi in decisioni consapevoli. I termini di Data Mining e di Knowledge Discovery in Databases sono spesso usati come

sinonimi. In realtà il KDD si riferisce all'intero processo di creazione di conoscenza mentre il Data Mining si riferisce ad una singola fase di questo processo: la fase di individuazione dei pattern e di formulazione delle ipotesi che spiegano i dati. Il processo di Knowledge Discovery in Databases prevede i seguenti passaggi:

1. identificazione degli obiettivi della ricerca;
2. formulazione di ipotesi di base;
3. raccolta dei dati e la loro memorizzazione;
4. eventuale ricodifica dei dati (spesso i dati raccolti devono essere ricodificati per poter applicare gli algoritmi di Machine Learning - è il tipico caso della normalizzazione di variabili numeriche per farle rientrare in un range compreso fra zero ed uno);
5. applicazione dell'algoritmo di Machine Learning (fase di Data Mining) per formulare delle ipotesi che spiegano i dati;
6. fase di interpretazione dei risultati e di generazione della conoscenza;
7. fase di decisione ed utilizzo dei dati;

La fase di interpretazione dei risultati si basa su:

- conoscenza pregressa del fenomeno in analisi;
- uso di misure che calcolano la bontà dell'apprendimento della macchina basandosi su dei parametri legati allo specifico algoritmo di Machine Learning applicato;
- verifica delle ipotesi ripetendo l'analisi con altri dati a disposizione;

L'enorme mole di dati che è possibile raccogliere e analizzare è in genere indicata con il termine di big data.

3.2 Tipologie di algoritmi di apprendimento

Quasi tutti, per non dire tutti gli algoritmi di machine Learning operano splittando la loro esecuzione in due processi, il primo di training (apprendimento), e il secondo di predizione. In letteratura esistono molti tipi di algoritmi di machine learning che solitamente vengono divisi in 3 macro-categorie così come indicato in Figura 3.1: algoritmi di tipo supervisionato, non supervisionato, e attivi.

Un algoritmo si definisce supervisionato quando a partire da un dataset che contiene almeno due o più proprietà, dove una delle quali è un'etichetta, tenta di

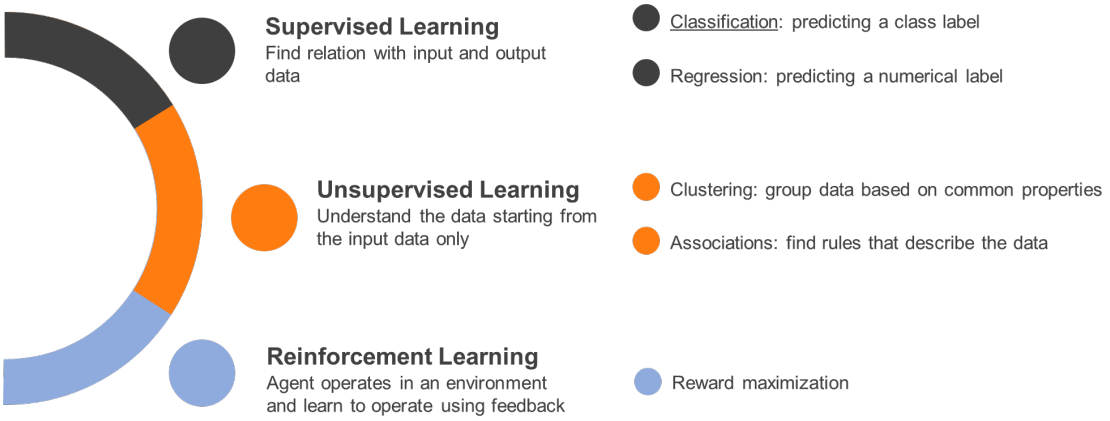

Figura 3.1: Categorie di algoritmi di Machine Learning.

capire come relazionare le proprietà con l'etichetta. Quando l'etichetta è di tipo testuale si parla di classificazione, quando è di tipo numerico si chiama regressione.

Le tecniche di apprendimento non supervisionato, invece, si basano fondamentalmente sul clustering e sulle regole di associazione. Nel clustering partiamo da un dataset, questa volta non etichettato, nella quale gli item vengono raggruppati in un certo numero di cluster, secondo varie logiche come la similarità o la distanza. Mente gli algoritmi di tipo associativo sono abili a ricavare relazioni comuni tra i dati.

L'ultima macro-categoria riguarda gli algoritmi di Reinforcement Learning, letteralmente: apprendimento per rinforzo. Questa è la tecnica di machine learning che rappresenta la versione computerizzata dell'apprendimento umano. In realtà si presta ad apprendere e ad adattarsi ai cambiamenti ambientali tramite un sistema di valutazione, che stabilisce una ricompensa quando l'azione compiuta è corretta, oppure una penalità nel caso opposto. L'obiettivo è quello di massimizzare la ricompensa ricevuta.

3.3 Algoritmi di apprendimento

Di seguito verranno illustrati brevemente alcuni degli algoritmi più diffusi nel machine learning, divisi per tipologia.

3.3.1 Algoritmi di apprendimento supervisionato

Linear Regression: questo modello viene utilizzato per stimare i valori reali (costo delle case, numero di abbonamenti, vendite totali per persona, ecc.) in base a variabili continue (metri quadri, sottoscrizioni, istruzione di una persona, ecc.). Qui la relazione tra le variabili indipendenti e le variabili dipendenti è stabilita da una

linea che solitamente rappresenta più o meno bene la relazione tra le due variabili. Si sviluppa la regressione lineare calcolando la funzione "$y = ax + b$" per la quale la differenza tra i valori attesi e quelli reali è minima. Nella formula qui sopra y è la variabile dipendente (output) e x è la variabile indipendente (input).

Logistic Regression: l'obiettivo di questi algoritmi è confrontare l'uscita effettiva con l'uscita desiderata e regolare l'algoritmo di conseguenza. In particolare, le regressioni logistiche sono utili quando la variabile dipendente è binaria (es. vincita o perdita) e possono essere utilizzate per stimare dalle variabili indipendenti come la probabilità di un evento specifico (es. Vincere una partita). Una buona regressione logistica consente di prevedere con precisione i risultati attesi.

k-nearest neighbors: tipicamente abbreviato come: k-NN è un algoritmo di apprendimento automatico utilizzato per la regressione e la classificazione, in cui il risultato determina se l'oggetto analizzato appartiene alla classe più comune dei suoi k vicini (con k intero positivo). Il punto di forza di questo algoritmo è che consente di memorizzare tutte le istanze disponibili e classificarle giudicando la loro distanza dai loro vicini. La distanza calcolata tra i due punti dati è solitamente la distanza euclidea, sebbene a volta viene utilizzata la distanza di Manhattan, Minkowski o Hamming. Le prime tre funzioni sono utilizzate per le variabili continue e la quarta (Hamming) per le variabili categoriali.

Classification and Regression Tree (Cart): conosciuto anche come albero decisionale, è un tipo di algoritmo di apprendimento supervisionato che viene utilizzato principalmente per problemi di classificazione e funziona anche con variabili dipendenti, categoriali e continue. In altre parole, l'albero CART è un albero decisionale binario che si ottiene dividendo più volte un nodo in due nodi figli, a partire dal nodo radice che contiene l'intero esempio di apprendimento. Ogni nodo padre rappresenta una singola variabile di input (x) e il punto di divisione per quella variabile (assumendo che la variabile sia numerica). I nodi foglia dell'albero contengono una o più variabili di output utilizzate per generare la previsione.

Naïve Bayes: attraverso l'utilizzo del teorema di bayes, l'algoritmo permette di assegnare un'etichetta a ogni gruppo di testo, per facilitarne la classificazione. Questo è ciò che fa lo Spam Filtering, una popolare applicazione dell'algoritmo Naïve Bayes. Fondamentalmente, è tra i metodi di apprendimento più popolari che raggruppa i dati analizzati in base alla loro somiglianza.

Random Forest: con il random forest, vengono valutati più alberi decisionali contemporaneamente. Come risultato si ottiene la foresta casuale, essa sceglierà il numero di classe più frequente ottenuta da tutti i modelli analizzati, e proporrà una

risposta (previsione del modello). Ciò aiuta a superare le difficoltà in cui si trova l'albero decisionale puntando a migliorare le prestazioni del modello.

Support Vector Machines (SVM): Viene utilizzato in vari ambiti, tra cui il riconoscimento facciale, classificazione del testo o delle immagini. L'algoritmo funziona dividendo i dati in diverse classi trovando una linea di separazione tra le diverse classi (di solito detta iperpiano). Questa linea non è presa casualmente: è quella che massimizza la distanza tra le varie classi nel caso fosse più di una. In questo modo, maggiore è la distanza maggiore sarà l'accuratezza del modello. Il support vector machine viene utilizzando anche per modelli più complessi (SVM non lineari). In questo caso non è possibile separare i dati di addestramento utilizzando un iperpiano, ma attraverso una funzione kernel, che ci aiuta a modellare modelli non lineari di dimensioni superiori.

Neural Networks: approfondite nella sezione 3.5.

3.3.2 Algoritmi di apprendimento non supervisionato

K-means clustering: il valore "K" viene immesso dall'utente e si riferisce al numero desiderato di cluster (gruppi) nel set di dati. Il clustering è definito come "raggruppamento di un insieme di campioni in modo che quelli appartenenti ad un gruppo (o un cluster) siano più simili (secondo alcuni criteri) di quelli di altri gruppi". L'algoritmo considera un campione di dati e lo separa al meglio in un numero K di cluster. L'onere di selezionare il numero appropriato di cluster è completamente a carico del data scientist. Rispetto ad altri algoritmi non supervisionati K-Means è incredibilmente veloce e potrebbe, ad esempio, essere utilizzato per identificare diversi segmenti di clienti in un mercato e, poi, per classificare i nuovi clienti come appartenenti ad una di queste categorie. Raccogliendo ulteriori dati su come si comportano i diversi segmenti di clienti le aziende possono generare previsioni accurate sui guadagni futuri, sul potenziale di crescita, sulla composizione demografica dei clienti, ecc.

Hierarchical clustering: le strategie per il clustering gerarchico sono tipicamente di due tipi: Agglomerativo: si tratta di un approccio "bottom up" (dal basso verso l'alto) in cui si parte dall'inserimento di ciascun elemento in un cluster differente e si procede quindi all'accorpamento graduale di cluster a due a due. Divisivo: si tratta di un approccio "top down" (dall'alto verso il basso) in cui tutti gli elementi si trovano inizialmente in un singolo cluster che viene via via suddiviso ricorsivamente in sotto-cluster. In entrambi i casi, quindi, si esplorano tutte le possibili combinazioni di cluster, da un estremo (un unico cluster) all'altro (un cluster per dato) e viceversa.

Neural Network: approfondite nella sezione 3.5.

3.3.3 Reinforcement Learning

Ecco alcuni termini importanti usati nel Reinforcement Learning:

- Agente: È un'entità presunta che esegue azioni in un ambiente per ottenere una ricompensa.
- Ambiente (e): uno scenario che un agente deve affrontare.
- Ricompensa (R): un ritorno immediato dato a un agente quando esegue un'azione o un'attività specifica.
- Stato (i): lo stato si riferisce alla situazione attuale restituita dall'ambiente.
- Policy (Π): è una strategia applicata dall'agente per decidere l'azione successiva in base allo stato corrente.

Esistono tre approcci per implementare un algoritmo di apprendimento per rinforzo:

- Basato sul valore: in questa tecnica si cerca di massimizzare la funzione V (s). In questo metodo, l'agente si aspetta un ritorno a lungo termine dagli stati attuali seguendo la politica scelta a priori.
- Basato sulla politica: si tenta di elaborare una policy tale che l'azione eseguita in ogni stato aiuti da ottenere la massima ricompensa in futuro.
- Basato sul modello: è necessario creare un modello virtuale per ogni ambiente. L'agente impara a prendere decisioni in quell'ambiente specifico.

3.3.4 Come scegliere il giusto algoritmo

Una volta introdotti tutti questi algoritmi, la vera domanda è come capire quale di questi utilizzare. In primo luogo è possibile restringere il campo di applicazione determinando se è necessario un algoritmo di apprendimento supervisionato o non supervisionato. Quindi se si hanno a disposizione gli output (etichette) per ciascuno dei valori di input, allora conviene usare un algoritmo di apprendimento supervisionato. I fattori più importanti da considerare quando si sceglie un algoritmo sono:

- La dimensione, la qualità e la natura dei dati
- Il tempo di calcolo disponibile
- L'urgenza del compito

- Cosa si vuole fare con i dati
- Precisione, tempo di training del modello, facilità d'uso

3.4 Come validare i modelli di classificazione

L'ultima considerazione da fare in merito alla classificazione riguarda il metodo di validazione dei modelli. La validazione di un classificatore è una procedura attraverso la quale si misura la qualità delle previsioni ottenute e si valuta se il modello soddisfa i requisiti del problema in esame. Per valutare un modello di machine learning che analizza grandi dataset si può pensare di usare la matrice di confusione (meglio conosciuta come confusion matrix). La matrice di confusione viene utilizzata per prevedere il risultato di un classificatore binario. Un classificatore binario produce come output due valori ad esempio Si/No e 1/0. La classe di interesse viene solitamente indicata come "positiva" e l'altra come "negativa". Per comprendere meglio le relazioni tra valori effettivi e previsti, la matrice di confusione è composta da ulteriori valori, come mostrato in Figura 3.2:

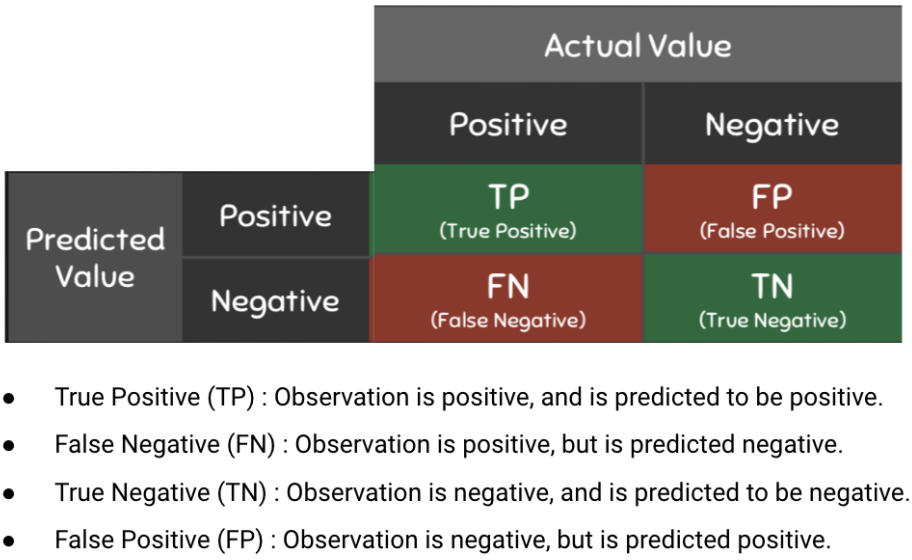

Figura 3.2: Matrice di confusione e parametri fondamentali.

La qualità delle previsioni dipende da quante e da quali classi sono state assegnate correttamente e viene misurata attraverso diverse metriche ricavate dai valori della matrice di confusione [31].

Error Rate (o Tasso di errore): è il tasso di errore (ERR) e viene calcolato come il numero di tutte le previsioni errate diviso per il numero totale del set di dati. Il miglior tasso di errore è 0, mentre il peggiore è 1.

$$ERR = \frac{FP+FN}{TN+FP+FN+TP}$$

Accuracy (o accuratezza): indica l'accuratezza del modello come dice il nome. Pertanto, la migliore accuratezza è 1, mentre la peggiore è 0. Può anche essere calcolato da 1 - ERR, o dalla seguente formula:

$$\text{Accuracy} = \frac{TP+TN}{TN+FP+FN+TP}$$

Precision (o precisione): La precisione è l'abilità di un classificatore di non etichettare un'istanza positiva che è in realtà negativa. Per ogni classe è definito come il rapporto tra veri positivi e la somma di veri e falsi positivi. Detto in un altro modo, "per tutte le istanze classificate come positive, quale di queste in percentuale erano corrette?". La formula che si ottiene è la seguente:

$$\text{Precision} = \frac{TP}{TP+FP}$$

Tecniche di cross validation La cross validation è un metodo in grado di stimare l'abilità dei modelli di apprendimento automatico.

Il concetto fondamentale su cui si fonda è quello di non utilizzare l'intero set di dati durante l'addestramento di un modello: una parte di essi vengono rimossi prima dell'inizio dell'allenamento. Al termine della formazione, i dati rimossi possono essere utilizzati per testare le prestazioni del modello appreso su "nuovi" dati. In altre parole, il set di dati di test è una parte separata dello stesso set di dati da cui deriva il set di training. Lo scopo principale dell'utilizzo del set di dati di test è testare la capacità di generalizzazione di un modello addestrato.

In definitiva, il modello viene valutato solamente dopo che è stato addestrato. Insieme alla formazione del modello, la cross validation mira a trovare un modello ottimale con le migliori prestazioni [32].

3.5 Reti Neurali e deep learning

Le tecniche basate su reti neurali, chiamate anche Neural Networks o *Artificial Neural Networks* (ANNs), definiscono un modello di classificazione ispirato ad alcuni elementi dell'apprendimento del cervello umano. Tale modello prevede, infatti, la classificazione di ogni esempio del dataset attraverso una rete di neuroni, connessi tra di loro.

Le reti neurali sono in grado di operare bene anche quando la mole dei dati è notevole e la loro qualità non è elevata (ad esempio sono presenti diversi dati

mancanti), senza richiedere all'utente di effettuare in partenza delle ipotesi di carattere statistico-probabilistico.

Ogni neurone riceve un'informazione dal livello sottostante, la elabora, e la trasmette ai neuroni dello strato superiore. Fanno eccezione i neuroni d'input, che ricevono l'informazione direttamente dall'esterno del sistema, e quelli d'output che la inoltrano al di fuori della rete. Ogni neurone comunica con gli altri attraverso connessioni pesate: legami a cui è associato un valore, un peso, che indica la forza della connessione.

I nodi intermedi della rete sono chiamati neuroni nascosti. Ogni neurone nascosto definisce un proprio un vettore di pesi ed un valore di offset, che usa per trasformare gli attributi in ingresso in un risultato intermedio normalizzato nell'intervallo [0,1]. I valori ottenuti dallo strato nascosto della rete (hidden layer) convergono infine sui neuroni di uscita, che determinano la classe da assegnare al campione.

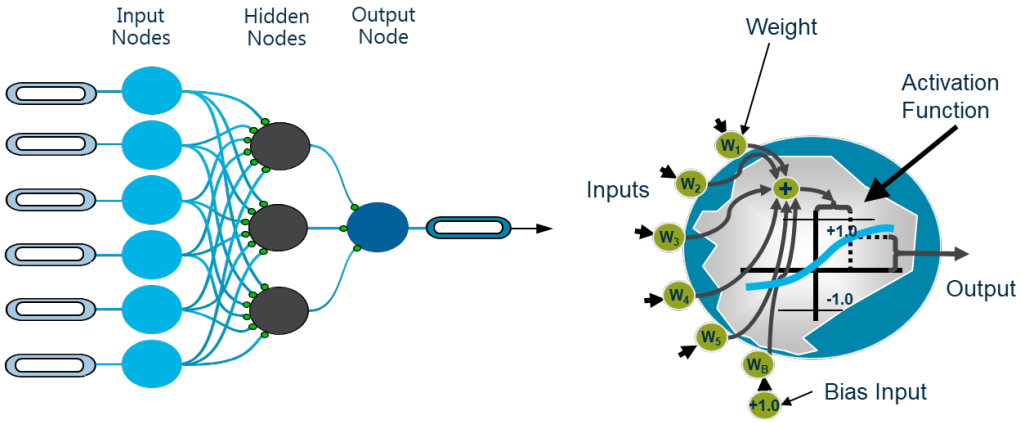

Figura 3.3: Rappresentazione di una rete neurale

In fase di generazione del modello, il processo con cui viene classificato un campione, attraverso la rete, è eseguito per tutte le istanze del training set in modo iterativo e, ad ogni iterazione, si calcola l'errore commesso sulla previsione delle classi. Il valore dell'errore di classificazione è propagato dall'uscita all'ingresso della rete, attraverso un processo definito back propagation, il cui scopo è quello di correggere i pesi e gli offset di ogni neurone e ottimizzare l'accuratezza del modello di classificazione. Dopo un numero massimo di iterazioni, se non è stato raggiunto il livello di accuratezza desiderato, oppure i pesi e gli offset dei neuroni non devono essere ulteriormente modificati, il processo iterativo si arresta e la configurazione dei neuroni raggiunta definisce il modello di classificazione.

L'addestramento di una rete neurale viene normalmente eseguita cercando di minimizzare una certa funzione di perdita (o di costo) $J(w)$. Solitamente questa minimizzazione viene effettuata attraverso un'ottimizzazione con un gradiente discendente (Gradient Descent), graficamente è ciò che avviene in Figura 3.4.

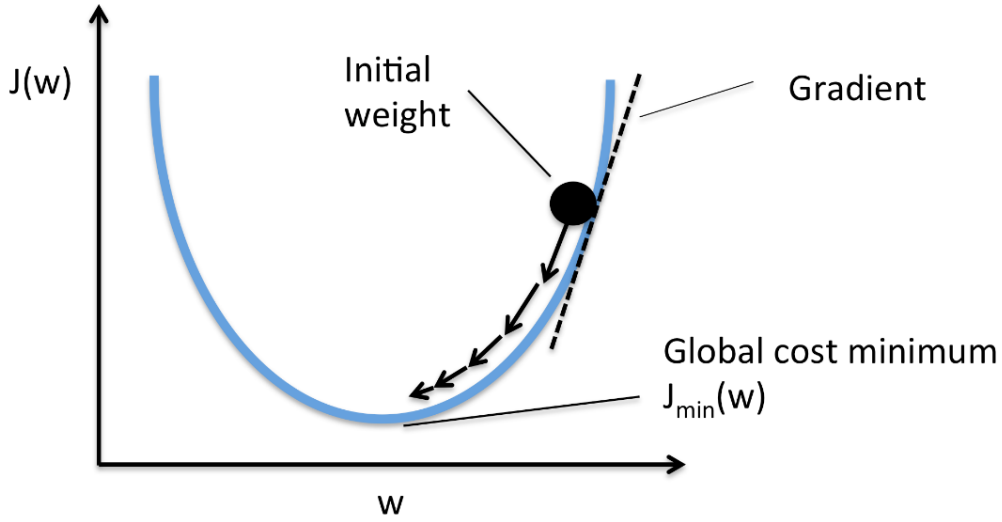

Figura 3.4: Rappresentazione di alto livello del Gradient Descent.

3.5.1 Difetti

I modelli creati dalle reti neurali, anche se molto efficienti, non possono essere spiegati con il linguaggio simbolico umano: i risultati devono essere accettati "così come sono", da qui la definizione inglese di reti neurali come "scatole nere".

Richiedono una fase di addestramento del sistema che stabilisca i pesi dei singoli neuroni e questa fase può richiedere molto tempo se il numero di record e variabili analizzati è molto elevato. Non esistono teoremi o modelli che ci consentano di definire una rete ottimale, quindi il successo della rete dipende in gran parte dall'esperienza del suo sviluppatore.

3.6 Reti Neurali Ricorrenti

Le architetture di rete neurale discusse nelle sezioni precedenti accettano input di dimensioni fisse e forniscono output di dimensioni fisse. Questa sezione eliminerà questo vincolo introducendo le reti neurali ricorrenti (RNN). Le RNN ci aiutano a gestire sequenze di lunghezza variabile definendo una relazione di ricorrenza su queste sequenze (da cui il nome). Esempi di tali dati includono le parole di una frase o il prezzo di un'azione variabile nel tempo. La parola sequenziale, implica che gli elementi della sequenza sono correlati tra loro e il loro ordine è importante. Ad esempio, se si prende un libro e si mescolano casualmente tutte le parole in esso contenute, il testo perderà il suo significato, nonostante sarà ancora possibile leggere le singole parole.

In effetti, le RNN possono simulare qualsiasi procedura/programma che un normale computer non sarebbe in grado di calcolare. Ad esempio, DeepMind di Google ha proposto un modello chiamato Differentiable Neural Computer, che può imparare come eseguire semplici algoritmi, come l'ordinamento. La relazione di ricorrenza definisce il modo in cui lo stato si evolve passo dopo passo nella sequenza attraverso un ciclo di feedback sugli stati precedenti, come illustrato nella figura seguente Figura 3.5:

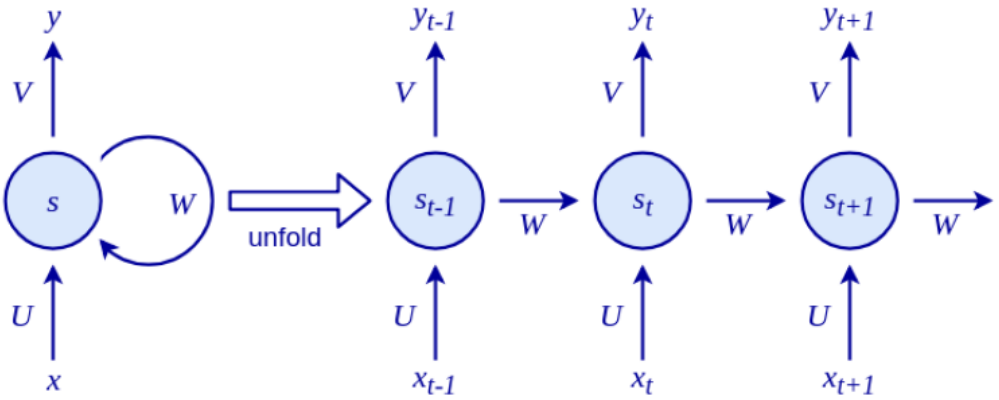

Figura 3.5: Evoluzione di un nodo ricorrente. [33]

Le RNN hanno tre serie di parametri (o pesi):

- U trasforma l'ingresso $x(t)$ nello stato $s(t)$
- W trasforma lo stato precedente $s(t-1)$ nello stato corrente $s(t)$
- V mappa lo stato interno appena calcolato $s(t)$ sull'uscita $y(t)$

U, V e W possono essere legati da un'espressione lineare definendo lo stato interno e l'uscita della rete come segue (f corrisponde ad una funzione di attivazione non lineare):

$$s_t = f\left(s_{t-1} * W + x_t * U\right)$$
$$y_t = s_t * V$$

Da questa relazione si nota che in una RNN, ogni strato dipende da tutti i calcoli precedenti tramite appunto questa relazione di ricorrenza. Un'importante implicazione di ciò è che le RNN hanno memoria nel tempo, poiché gli stati contengono informazioni basate sui passaggi precedenti. In teoria, le RNN possono ricordare le informazioni per un periodo di tempo arbitrariamente lungo, ma in pratica si limitano a guardare indietro solo di pochi passaggi.

Poiché le RNN non si limitano all'elaborazione di input di dimensioni fisse, alcune combinazioni di input-output sono le seguenti:

- **one-to-one**: si tratta di un'elaborazione non sequenziale, come se si avesse a che fare con reti neurali feedforward e convoluzionali. Un esempio è la classificazione delle immagini.
- **one-to-many**: questa elaborazione genera una sequenza basata su un singolo input, ad esempio la generazione di didascalie da un'immagine.
- **many-to-one**: questa elaborazione restituisce un singolo risultato in base a una sequenza, ad esempio, la classificazione di un testo (se spam o meno, se positivo o negativo, ecc. . .).

Quanto segue è una rappresentazione grafica delle precedenti combinazioni input-output, più alcune combinazioni complesse di queste tre tipologie Figura 3.6.

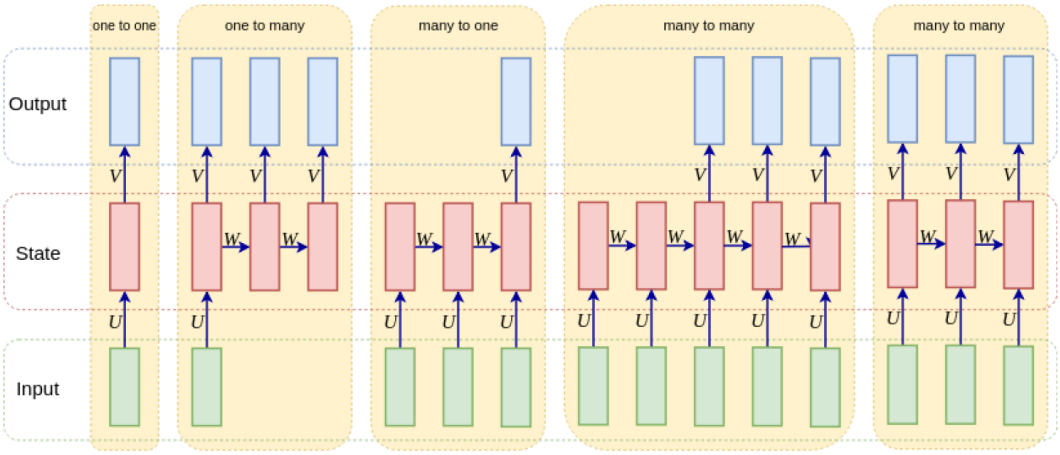

Figura 3.6: Tipologie più comuni di utilizzo delle reti ricorrenti.

Le RNN sono utili sotto molti aspetti, soprattutto perché sono applicabili in qualsiasi scenario, ma se lasciate nella loro versione classica soffrono del cosiddetto "problema del gradiente discendente" [34], per poter gestire questa problematica, le RNN devono essere in grado di mantenere al loro interno dati già processati, aggiungendo un feedback dagli stati più a valle, verso gli stati di ingresso.

Le celle base RNN hanno difficoltà a ricordare/sfruttare input di step lontani: la memoria dei primi input tende a svanire. D'altro canto sappiamo che in una frase anche le prime parole possono avere un'importanza molto rilevante. Per risolvere questo problema e facilitare la convergenza in applicazioni complesse, sono state proposte celle più evolute dotate di un effetto memoria a lungo termine: LSTM e GRU sono le più note tipologie.

Le LSTM possono gestire le dipendenze a lungo termine grazie a una cella di memoria appositamente predisposta. In realtà, funzionano così bene che la maggior parte dei risultati ottenuti nell'addestramento di RNN su una varietà di problemi sono dovuti all'uso di LSTM.

L'idea chiave delle LSTM è lo stato della cella, in cui le informazioni possono essere scritte o rimosse solo esplicitamente in modo che lo stato rimanga costante fino a quando non ci sono interferenze esterne. Questa cella può essere modificata solo da ingressi specifici chiamate porte. Queste porte sono composte da una funzione sigmoidea logistica e da una moltiplicazione. Poiché la funzione logistica restituisce solo valori compresi tra 0 e 1, la moltiplicazione può solo ridurre il valore che attraversa il "cancello". Una tipica cella LSTM è composta da tre porte: una porta chiamata "forget gate", una porta di ingresso e una porta di uscita come mostrato in Figura 3.7. Lo stato della cella, l'input e l'output sono tutti vettori, quindi le LSTM possono contenere una combinazione di diversi blocchi di informazioni in ogni fase temporale.

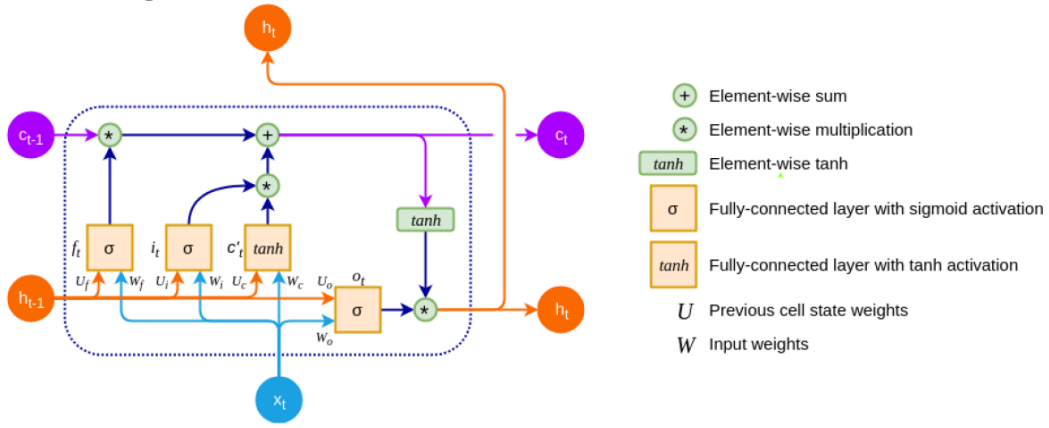

Figura 3.7: Tipica cella LSTM [35].

Una Gated Recurrent Unit (GRU) è un tipo di blocco ricorrente introdotto nel 2014 da Kyunghyun Cho al. [36] [37], come miglioramento rispetto a LSTM. Un'unità GRU di solito ha prestazioni simili o migliori a un LSTM, ma lo fa con meno parametri e operazioni (come intuibile dalla Figura 3.8):

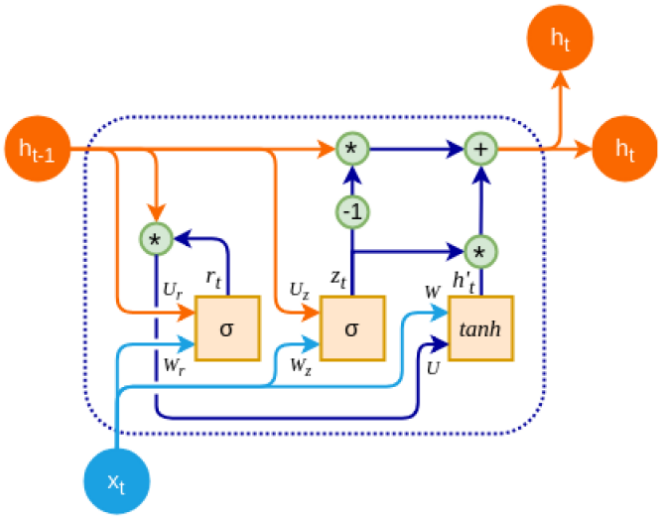

Figura 3.8: Tipica cella GRU.

Capitolo 4

Stato dell'arte e lavori correlati

4.1 Introduzione

Negli ultimi anni, si ha avuto una crescita esponenziale degli articoli di ricerca che trattano argomenti come il CACC e l'ACC analizzando molti aspetti legati sia alla comunicazione che alla sicurezza. Nel Giugno 2019 è stato pubblicato un articolo nel quale si metteva in evidenza proprio questa crescita [38], dalla quale è stato possibile estrapolare la Figura 4.1.

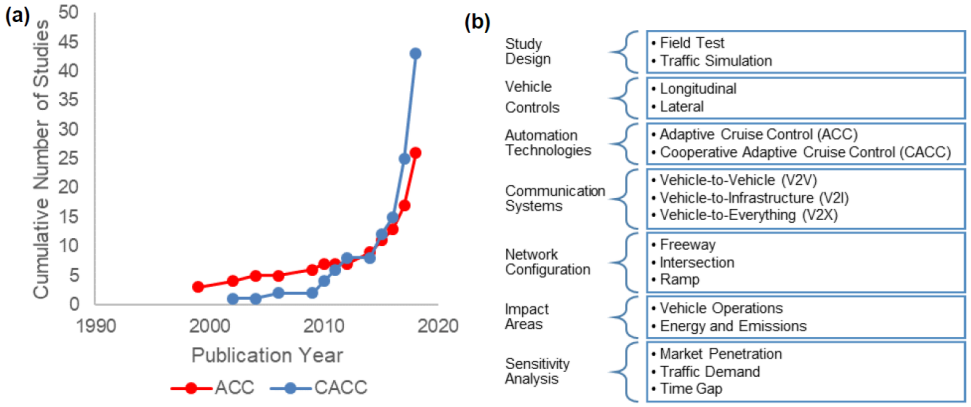

Figura 4.1: Curva di crescita del numero di studi sul C/ACC.

La Figura 4.1(a) mette in luce in quale quantità la ricerca è cresciuta negli ultimi anni, nel dettaglio i dati fanno riferimento al range temporale che parte dai primi anni 2000, fino al primo trimestre del 2019; nella Figura 4.1(b) sono elencati i principali aspetti trattati in questi articoli.

4.2 Origine dell'elaborato

Il lavoro svolto in questa tesi pone le sue basi a partire dall'articolo pubblicato dal M. Iorio et al. nel [3], nella quale l'obbiettivo prefissato era pressoché il medesimo di questa tesi, ovvero realizzare una logica di detection per attacchi di tipo injection in algoritmi di Cooperative adaptive cruise control. In particolare la logica proposta nell'articolo sfrutta le peculiarità del plotone dei veicoli e si basa su due intuizioni principali.

- Innanzitutto, riconosce che le diverse variabili fisiche riguardanti un singolo veicolo (cioè posizione, velocità e accelerazione) non possono evolvere indipendentemente: sono invece legate dalle ben note equazioni cinematiche.

- In secondo luogo, muovendosi in un convoglio, i veicoli consecutivi tendono a reagire agli stessi stimoli, mostrando così schemi di movimento molto simili: pertanto le incoerenze potrebbero essere un chiaro indizio di un attacco di falsificazione.

In una prima fase l'articolo analizza il comportamento di diversi algoritmi di CACC nel momento in cui si presenta un attacco; in particolare viene eseguito un confronto tra gli attacchi perpetrati da un veicolo leader, e da uno no-leader, si conclude questa prima analisi constatando che gli attacchi proveniente da un leader, in uno scenario cooperativo sono più efficaci.

Mentre nella seconda fase avviene il test vero e proprio degli attacchi; quest'ultimi sono di vario genere, infatti si prova ad alterare una variabile per volta per poi combinarle tutte insieme in attacchi coordinati, nella quale le ipotesi del platooning e le equazioni del moto sono rispettate, rendendoli i più difficili da rilevare. Questa è una caratteristica peculiare della ricerca perché come si vedrà nei successivi articoli inerenti all'argomento, presentati in questa sezione, allo stato dell'arte si tende a realizzare degli attacchi impulsivi che non rispettano nessuna legge ed ipotesi rendendo questo tipo di attacchi rilevabili quasi istantaneamente.

Per poter validare la coerenza dei messaggi ricevuti, come anticipato precedentemente, oltre alle informazioni V2V vengono considerate anche quelle locali, quindi per poter rilevare un comportamento scorretto si sono confrontati i valori ricevuti attraverso il V2V e i radar con quelli stimati, dove per stimati si intendono quei valori che ci si aspetta nel momento in cui le leggi del moto fossero rispettate anche se di fatto la stima viene ottenuta mediante un filtro di Kalman [29]. Il filtro permette di stimare lo stato di un processo combinando previsioni teoriche e misurazioni provenienti da sensori rumorosi, per correggere le derive del mondo reale che non possono essere catturate da modelli semplificati. In altre parole, il filtro tende a mediare tra la stima e le misurazioni effettive a seconda degli intervalli di confidenza associati, convergendo molto rapidamente verso i valori reali.

L'algoritmo qui presentato è basato su un set di disequazioni, nelle quali si confronta la differenza tra i valori ricevuti e quelli stimati con delle soglie ricavate attraverso una procedura trial and error.

Le equazioni considerate sono:

$$|d_i^{\text{EST}} - \delta| < \alpha_1 \cdot \delta \tag{4.1}$$

$$|\overline{d_i^{\text{V2V}} - d_i^{\text{EST}}}| < \alpha_2 \cdot 3\sigma_{d_i}^{\text{EST}} \tag{4.2}$$

$$|\overline{v_i^{\text{V2V}} - v_i^{\text{EST}}}| < \alpha_3 \cdot (\epsilon_v^{\text{V2V}} + 3\sigma_{v_i}^{\text{EST}}) \cdot (1 + \alpha_8 |a_i|) \tag{4.3}$$

$$|d_i^{\text{RAD}} - \delta| < \alpha_4 \cdot \delta \tag{4.4}$$

$$|\overline{d_i^{\text{RAD}} - d_i^{\text{EST}}}| < \alpha_5 \cdot (\epsilon_d^{\text{RAD}} + 3\sigma_{d_i}^{\text{EST}}) \tag{4.5}$$

$$|\overline{\Delta v_i^{\text{RAD}} - \Delta v_i^{\text{V2V}}}| < \alpha_6 \cdot (\epsilon_{\Delta v}^{\text{RAD}} + 2\epsilon_v^{\text{V2V}}) \cdot (1 + \alpha_8 |a_i|) \tag{4.6}$$

$$|\overline{\Delta v_i^{\text{RAD}} - \Delta v_i^{\text{EST}}}| < \alpha_7 \cdot (\epsilon_{\Delta v}^{\text{RAD}} + 3\sigma_{\Delta v_i}^{\text{EST}}) \cdot (1 + \alpha_8 |a_i|) \tag{4.7}$$

Dove, α_k per $k = 1 \dots 8$ sono tutte costanti definite dall'utente non negative, $\alpha_1 < 1$ e $\alpha_4 < 1$.

La l_i, p_i, v_i e a_i rappresentano rispettivamente lunghezza, posizione, velocità e accelerazione del i^{th} veicolo nel plotone, con $d_i = p_{i-1} - p_i - l_{i-1}$ e $\Delta v_i = v_{i-1} - v_i$ che indicano la distanza e la velocità relativa rispetto al predecessore; inoltre, gli apici indicano se i valori sono ottenuti attraverso il V2V ($\cdot^{\text{V2V}}$), il filtro di Kalman ($\cdot^{\text{EST}}$) o mediante una misurazione radar ($\cdot^{\text{RAD}}$).

Quindi, sia δ la distanza attesa tra due veicoli, ϵ_x l'incertezza associata al sensore x e $\sigma_{d_i}^{\text{EST}} = \sigma_{p_i}^{\text{EST}} + \sigma_{p_{i-1}}^{\text{EST}}$, $\sigma_{\Delta v_i}^{\text{EST}} = \sigma_{v_i}^{\text{EST}} + \sigma_{v_{i-1}}^{\text{EST}}$ rappresentano la deviazione standard fornita in output dal filtro di Kalman insieme alla stima.

Infine, con $|\cdot|$ indichiamo il valore assoluto e con $\bar{\cdot}$ la media mobile dei precedenti n dati, con n inpostato dall'utente. Quindi, per ogni veicolo i, viene rilevato un comportamento scorretto se viene violata una delle precedenti disuguaglianze per un intervallo di tolleranza superiore a quello specificato ΔT_{th}.

La parte sinistra di ogni disuguaglianza è una differenza, usata per valutare quanto simili siano le quantità di interesse. Nella maggior parte dei casi, viene sfruttata una media mobile per mediare tra valori consecutivi e attenuare le oscillazioni causate da letture rumorose: maggiore è la dimensione della finestra, maggiore è la media ottenuta, a costo di un aumento del ritardo di rilevazione. Diversamente, il lato destro rappresenta una soglia, a seconda dei parametri definiti dall'utente e delle incertezze associate alle misurazioni.

Anche in questo articolo, come il presente documento è stato utilizzato come ambiente simulativo per le VANET, OMNeT++, VEINS e SUMO per il controllo dei veicoli.

Successivamente, una volta che l'attacco viene rilevato, attualmente la contromisura adottata è eseguire lo switch all'ACC classico, fisicamente tradotto vuol dire aumentare la distanza dal predecessore ed utilizzare solo i sensori locali.

I vari motivi che hanno portato ad uno step successivo, presentati in questa tesi, sono stati:

- la presenza di soglie statiche, di fatto queste non sono in grado di comprendere se il veicolo è in una fase di accelerazione o decelerazione, ma assumono sempre lo stesso valore;
- molti attacchi non sono stati rilevati, soprattutto quando il radar non è accessibile ed in particolare negli attacchi coordinati;
- infine si nota la presenza di falsi positivi e di tempi di rilevamento, che anche se tutto sommato bassi, offrono un margine di miglioramento.

4.3 Evoluzione delle tecniche di rilevamento

In questa sezione si andrà ad analizzare l'evoluzione nel tempo di queste tecniche di rilevamento, da quelle più semplici a quelle più complesse e moderne che fanno uso del Machine Learning.

Nel 2015, Amoozadeh et al. [39] ha presentato una delle prime indagini su una serie di possibili attacchi contro flussi di veicoli connessi, sia a livello di rete che di applicazione. Le loro simulazioni, eseguite usando la piattaforma VENTOS [40], hanno mostrato come falsificando i messaggi l'instabilità del plotone viene compromessa, mentre gli attacchi di disturbo di radio jamming sono riusciti a forzare l'algoritmo CACC a degradare in ACC eliminando la componente cooperativa. Infine, hanno sottolineato la necessità di tecniche di rilevamento per identificare i veicoli compromessi e proteggere i controllori CACC. Successivamente nel 2017, van der Heijden et al. [41] ha continuato lungo questa linea di ricerca simulando con maggior dettaglio gli effetti sia degli attacchi di disturbo che di iniezioni di dati dannosi. Hanno utilizzato il simulatore Plexe, e hanno concluso che quando gli algoritmi cooperativi sono sotto attacco, portano in molti casi a incidenti potenzialmente fatali (in particolare la loro ricerca era focalizzata su 3 tipi di algoritmi di controllo cooperativo). Pur essendo già piuttosto completa come analisi, e bene precisare alcune limitazioni. In particolare, il loro lavoro ha trascurato la possibilità di raccogliere alcune delle misurazioni fisiche attraverso un radar, intrinsecamente più difficile da ingannare, piuttosto che utilizzare quelle ottenute tramite V2V. Inoltre, hanno confrontato il comportamento dei controller utilizzando diverse impostazioni di distanza intra-veicolare e causato cambiamenti bruschi sui valori considerati, facilitandone la loro rilevazione, mentre in questa tesi così come in [3], le variazioni considerate sono più fluide con il fine di rendere l'attacco molto più difficile da rilevare.

A dimostrazione del fatto che questo campo di ricerca riscuote sempre più successo, nel primo trimestre del 2018 un gruppo di ricercatori hanno realizzato un

database pubblico, chiamato VeReMi (Vehicular Reference Misbehavior Dataset) [42]. VeReMi, infatti, è stato il primo dataset pubblico estensibile, che consente a chiunque di riprodurre il processo di generazione degli attacchi, per confrontare nuovi meccanismi di rilevamento con quelli esistenti. Il set di dati consiste di 225 simulazioni individuali, con 5 diversi attacchi, 3 diverse densità degli attaccanti, 3 diverse densità di traffico e 5 ripetizioni per ogni set di parametri. Anche loro, come il presente documento hanno utilizzato come ambiente simulativo per le VANET, OMNeT++, VEINS e SUMO per il controllo dei veicoli. Uno dei suoi vantaggi è proprio quello di essere aggiornabile di volta in volta da tutti coloro che ne fanno uso. Infatti in questa prima versione il numero di simulazioni considerate sono molto poche, soprattutto per il numero di scenari considerati, e in ogni caso gli attacchi considerati sono di facile rilevazione perché impulsivi. Di questo database esiste anche una versione estesa rilasciata nel 2020 [43].

Nel 2017 Sun et al.[44] hanno esplorato la validazione della posizione un veicolo malintenzionato utilizzando le funzionalità disponibili al livello fisico della comunicazione Vehicle-to-Vehicle (V2V). Nel loro modello l'attaccante trasmette posizioni dubbie all'interno dei cosiddetti messaggi di sicurezza di base (BSMs – Basic Safety Messages). Gli autori considerano uno scenario autostradale con almeno un veicolo onesto nel raggio di comunicazione supportato dal modello. Verificano la posizione dell'aggressore e i suoi movimenti utilizzando la stima AoA - Angle of Arrival , la misurazione della velocità dell'effetto Doppler (DS), il filtro di Kalman esteso (EKF) e l'input dai veicoli vicini.

La vulnerabilità di questi sistemi è dovuta a soluzioni crittografiche (ad es. le PKI) in grado di fornire solo l'integrità del messaggio e non la sua correttezza. Perciò, occorre trovare un modo per rilevare questi comportamenti scorretti, in letteratura questa procedura prende il nome di "misbehavior detection (MBD)", l'obiettivo di questa tecnica è analizzare i dati delle applicazioni e i processi fisici che in questi sistemi devono essere rispettati, e sfruttare queste peculiarità a loro vantaggio per rilevare messaggi errati, consentendo la revoca locale dei veicoli che trasmettono messaggi dannosi. Per messaggi errati si intendono sia i messaggi manipolati da un attaccante sia quei messaggi che entità malfunzionanti (ad es. sensori) trasmettono involontariamente agli altri veicoli.

Per quanto riguarda il rilevamento di comportamenti scorretti nelle VANET che sfrutta direttamente le peculiarità dei plotoni è il documento recentemente presentato nel 2018 [45]. Nel loro lavoro, gli autori hanno sfruttato le relazioni spaziali tra più membri per ottenere un'informazione aggiuntiva che può essere valutata per il rilevamento di possibili attacchi di iniezione. Tuttavia, non hanno tratto vantaggio dalle correlazioni esistenti tra le diverse misurazioni fisiche presenti nei beacon V2V.

Più recentemente (02/2020) è stato rilasciato un articolo nella quale viene effettuata la detection degli attacchi sfruttando la distanza reciproca tra due veicoli, attraverso la localizzazione mediante onde 5G [46]. In particolare anche loro

utilizzano il Kalman Filter per stimare in quale stato dovrà trovarsi il veicolo nella prossima misurazione, riprendendo parte del lavoro svolto in [44]. Questo algoritmo è stato testato in vari scenari e densità di traffico (quindi, non solo il caso CACC), ed i risultati della valutazione effettuata hanno mostrato che il sistema può raggiungere un'accuratezza superiore al 97% nel rilevare le false segnalazioni da vari tipi di veicoli che si comportano male. Gli attacchi considerati anche se molto semplici da rilevare sono stati costruiti rispettando le leggi del moto, infatti posizione e velocità sono state alterate rispettando l'accelerazione, così come realizzato in questa tesi. I test da loro effettuati hanno mostrato un tempo di reazione di circa 0.2 s. Concludendo, questo approccio potrebbe essere utilizzato nella realtà, non nell'immediato ma in un vicino futuro, perché richiede una densità di antenne 5G molto elevata e si basa su una triangolazione della posizione dei veicoli accuratissima.

4.4 Detection con Machine Learning

Negli articoli appena presentati le tecniche di detection sfruttavano perlopiù sistemi ben noti come il Kalman-Filter, GPS..., ma nessuno utilizzava algoritmi di Machine Learning, la motivazione di ciò potrebbe risiedere nella potenza di calcolo richiesta da queste procedure, che in ambiente embedded non trovano molte applicazione. Tuttavia, recentemente la tecnologia si è evoluta tanto che a bordo veicolo è più facile trovare architetture elaborative più performati, come le GPU. Motivazione per la quale negli ultimissimi anni la loro applicabilità è cresciuta notevolmente, come dimostrano i successivi articoli pubblicati tra il 2018 e il 2020.

Nel 2018 So et al. [47] ha proposto un framework per effettuare i controlli di plausibilità valutando due modelli di apprendimento automatico le SVM e le K-NN. Le prestazioni di questi modelli sono state calcolate utilizzando l'accuratezza e la precisione della classificazione. Gli autori hanno considerato gli attacchi di spoofing della posizione dal set di dati VeReMi [42]. Gli attacchi e il rilevamento sono stati eseguiti in uno scenario combinato, cittadino ed extra-urbano. I loro dati di simulazione sono stati generati utilizzando VEINS e la valutazione è stata eseguita in MATLAB. Sebbene il loro studio fosse completo, il tipo di attacco era limitato agli attacchi di spoofing della posizione.

Un articolo che più si avvicina al presente testo è quello pubblicato nell'aprile 2020, da Kamel et al. [48]; in questo documento, viene presentato un framework di simulazione di MisBehavior Detection (MBD) che consente alla comunità di ricercatori di sviluppare, testare e confrontare diversi algoritmi MBD seguendo il lavoro presentato in precedenza sul VeReMi [42]. Una caratteristica importante è che per validare il framework realizzato utilizzano diversi scenari di esempio e testano molti algoritmi tra cui quelli ad apprendimento automatico, interessante è

anche la strategia di fall-back utilizzata. Il framework è open-source ed è disponibile a tutti su github[1]. In questa architettura è presente un'entità in grado di investigare e decidere se revocare o meno il mittente dei messaggi contraffatti, essa prende il nome di Misbehavior Authority (MA). Gli attacchi considerati nello studio che sono particolarmente efficaci sono i seguenti:

- Attaccante interno, in grado di superare le PKI;
- Attaccante attivo, partecipa alla comunicazione inviando dati contraffatti;
- Modifica del payload, è previsto un man-in-the-middle che altera i messaggi che viaggiano nel bus CAN;
- Alterazione del rate di trasmissione;

Il framework presenta più tecniche di detection, a partire dalle più semplici fino alle più avanzate, in totale sono 3:

1. Controllo della plausibilità (Plausibility Checks): alle variabili come posizione, velocità e accelerazione li viene assegnato un range di plausibilità dalla quale non possono uscire, mentre ogni volta che si riceve un nuovo pacchetto (beacon) viene eseguito un controllo di consistenza con i precedenti. In questa categoria sono compresi anche i check mediante Kalman-filter;
2. Advanced MBD: qui viene già implementato un concetto di apprendimento, infatti gli algoritmi facente parte di questa tipologia in una prima fase addestrano i loro valori analizzandone il comportamento per poi settare ad esempio delle soglie;
3. Machine Learning Algorithm: lo sviluppo di questi algoritmi è diviso in due fasi, offline ed online. Nella fase offline, e possibile addestrare e salvare il modello ML. Utilizzano le librerie scikit-learn[2] per la modellazione degli algoritmi e la libreria joblib[3] per il salvataggio dei modelli. Nella fase online i modelli precedentemente salvati e caricati su un server vengono scaricati e testati a run-time; Per poter classificare gli attacchi sono stati utilizzati i seguenti algoritmi di ML:
    - SVM: Support Vector Machines;
    - MLP: Multi Layer Perceptron, basata sulle reti neurali;

[1] F2MD: `https://github.com/josephkamel/F2MD`

[2] scikit-learn: `https://scikit-learn.org/stable/`

[3] joblib: `https://joblib.readthedocs.io/en/latest/`

- LSTM: Long Short-Term Memory, algoritmi facente parte delle reti neurali ricorrenti, ampiamente discussi nel capitolo 2.

La Misbehavior Authority (MA) prima di poter escludere un veicolo dalla comunicazione doveva essere sicura che questo stava seguendo un comportamento anomalo, pertanto ad ogni messaggio ricevuto dalla MA è stato assegnato un livello da 0 a 5, dove il livello 0 corrisponde a nessun intervento, i livelli intermedi prevedono dei messaggi di warning che richiedono sempre più dettagli, mentre il livello 5 rappresenta la revoca del certificato del veicolo mittente; Come si evince da questa ricerca le tecniche di ML sono più efficaci delle altre, ma richiedono una prima fase di training molto lunga, mentre tecniche più statiche sono da subito disponibili e non richiedono molte risorse.

Capitolo 5

Architettura e implementazione degli algoritmi

5.1 Introduzione

In questo capitolo, verranno descritti principalmente i due algoritmi sviluppati in questo lavoro. In primo luogo, verrà presentata una panoramica che inquadra il problema da analizzare e le motivazioni alla base delle scelte effettuate. Saranno definite delle metriche in grado di confrontare i diversi approcci tale da confrontarne le prestazioni. Quindi, sarà presentato il primo algoritmo basato su una tecnica statica e successivamente l'algoritmo vero e proprio basato sul Machine Learning nonché le reti neurali ricorrenti; poi saranno definiti i componenti principali e l'architettura di entrambi i metodi descritti.

5.2 Il punto di partenza

Questo elaborato nasce dalla necessità di rendere i sistemi cooperativi più sicuri al fronte di attaccanti interni in grado di superare i protocolli di sicurezza implementati in questi algoritmi. Come anticipato nel capitolo 4 la tesi in oggetto può essere vista come uno step successivo all'articolo [3]. Prima di analizzare nel dettaglio il suo funzionamento e la sua architettura riprendiamo alcuni concetti fondamentali utili per comprendere sia l'algoritmo sviluppato nell'articolo [3] sia gli step futuri realizzati in questo elaborato. In particolare la logica proposta nell'articolo sfrutta le seguenti intuizioni:

- riconosce che le diverse variabili fisiche riguardanti un singolo veicolo (cioè posizione, velocità e accelerazione) non possono evolvere indipendentemente: sono invece legate dalle ben note equazioni cinematiche.

- I veicoli muovendosi in un convoglio tendono a reagire agli stessi stimoli, mostrando così schemi di movimento molto simili: pertanto le incoerenze potrebbero essere un chiaro indizio di un attacco di falsificazione.

- Per poter rilevare gli attacchi possono essere utilizzate congiuntamente sia le informazioni provenienti dal radar sia quelle provenienti dalle comunicazioni V2V.

- Gli attacchi più efficaci e difficilmente rilevabili sono quelli di iniezione, e come dimostrato nell'articolo (dove sono stati confrontanti più algoritmi di CACC), se questi provengono da un veicolo leader è più facile generare incidenti.

Gli attacchi di iniezione considerati sono di vario genere, infatti si prova ad alterare una variabile per volta (posizione, velocità e accelerazione) per poi combinarle tutte insieme in attacchi coordinati, nella quale le ipotesi del platooning e le equazioni del moto sono rispettate, rendendoli i più difficili da rilevare.

5.2.1 Architettura di alto livello

L'algoritmo presentato nell'articolo [3] è basato su un set di disequazioni (1-7), nella quale si confronta la differenza dei valori ricevuti, misurati o stimati, con delle soglie statiche ricavate attraverso una procedura "trial and error". Di seguito verrano elencate le disequazioni presentate nella sezione 4.2

$$|d_i^{\mathrm{EST}} - \delta| < \alpha_1 \cdot \delta \tag{5.1}$$
$$|\overline{d_i^{\mathrm{V2V}} - d_i^{\mathrm{EST}}}| < \alpha_2 \cdot 3\sigma_{d_i}^{\mathrm{EST}} \tag{5.2}$$
$$|\overline{v_i^{\mathrm{V2V}} - v_i^{\mathrm{EST}}}| < \alpha_3 \cdot (\epsilon_v^{\mathrm{V2V}} + 3\sigma_{v_i}^{\mathrm{EST}}) \cdot (1 + \alpha_8|a_i|) \tag{5.3}$$
$$|d_i^{\mathrm{RAD}} - \delta| < \alpha_4 \cdot \delta \tag{5.4}$$
$$|\overline{d_i^{\mathrm{RAD}} - d_i^{\mathrm{EST}}}| < \alpha_5 \cdot (\epsilon_d^{\mathrm{RAD}} + 3\sigma_{d_i}^{\mathrm{EST}}) \tag{5.5}$$
$$|\overline{\Delta v_i^{\mathrm{RAD}} - \Delta v_i^{\mathrm{V2V}}}| < \alpha_6 \cdot (\epsilon_{\Delta v}^{\mathrm{RAD}} + 2\epsilon_v^{\mathrm{V2V}}) \cdot (1 + \alpha_8|a_i|) \tag{5.6}$$
$$|\overline{\Delta v_i^{\mathrm{RAD}} - \Delta v_i^{\mathrm{EST}}}| < \alpha_7 \cdot (\epsilon_{\Delta v}^{\mathrm{RAD}} + 3\sigma_{\Delta v_i}^{\mathrm{EST}}) \cdot (1 + \alpha_8|a_i|) \tag{5.7}$$

Quindi per ogni veicolo i, viene rilevato un comportamento scorretto se viene violata una delle disuguaglianze per un intervallo di tolleranza superiore a quello specificato in ΔT_{th} (default 1 s). Questo viene visto come un ritardo intrinseco dell'algoritmo che non può essere bypassato, infatti la detection non potrà mai avvenire prima di questo ΔT_{th}.

Le disuguaglianze da (5.1) a (5.3) si basano solo sulle informazioni ricevute attraverso la comunicazione V2V e valori stimati tramite i filtri di Kalman; quindi, possono essere valutati indipendentemente da e per ciascun membro del plotone.

Invece le disuguaglianze da (5.4) a (5.7) richiedono ulteriori misurazioni ottenute attraverso il radar e, pertanto, sono idonee a essere verificate solo dal follower di ciascun veicolo. Il loro scopo, è quello di confrontare le informazioni in arrivo da fonti diverse, per verificarne la plausibilità. La parte sinistra di ogni disuguaglianza è una differenza, usata per valutare quanto simili siano le quantità di interesse. Diversamente, il lato destro rappresenta una soglia, a seconda dei parametri definiti dall'utente e delle incertezze associate alle misurazioni. Inoltre, le disuguaglianze (5.3), (5.6) e (5.7) comprendono anche un fattore di correzione basato sull'accelerazione attuale: si prevede infatti che i cambiamenti di velocità aumentino temporaneamente le oscillazioni anche durante il normale funzionamento.

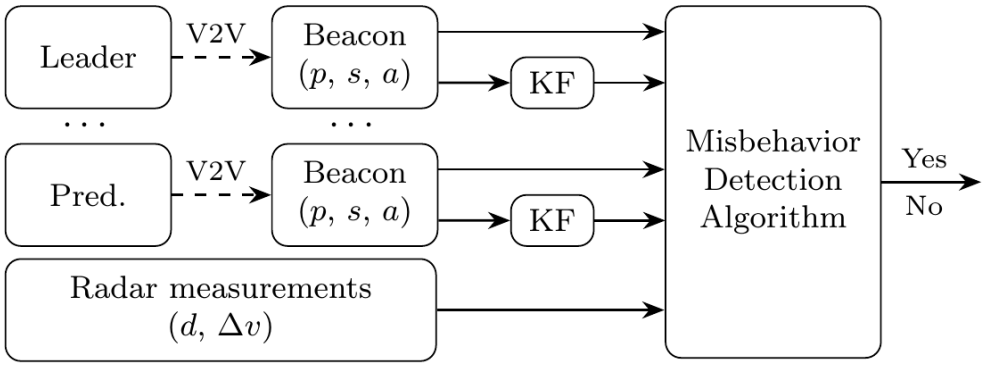

Figura 5.1: Schematizzazione dei dati necessari per effettuare la detection.

Nella Figura 5.1, si vuole rappresentare una schematizzazione ad alto livello, evidenziando quali sono le informazioni richieste dall'algoritmo di rilevamento. In particolare, mette in luce come ogni veicolo può eseguire autonomamente l'algoritmo di rilevamento da solo, sfruttando solo i messaggi ricevuti e le stime locali calcolate attraverso i filtri Kalman. Tuttavia, ogni veicolo può utilizzare le misurazioni del proprio radar per raccogliere informazioni privilegiate sul proprio predecessore. Pertanto, confrontando e combinando le misurazioni raccolte in modo indipendente, ogni veicolo è in grado di arricchire le proprie conoscenze, rendendo più difficile l'inganno dal suo predecessore e, di conseguenza, aumentando la sicurezza dell'intero plotone.

Mentre, nella Figura 5.2 è rappresentato l'andamento nel tempo di uno di questi valori (quindi il lato sinistro della disuguaglianza) e il funzionamento del filtro di Kalman utile per la detection; si nota una linea di separazione verticale, essa rappresenta l'istante in cui parte l'attacco, nel dettaglio, sul lato sinistro di questa linea avviene l'invio dei valori legittimi, ed è evidente come il filtro Kalman è perfettamente in grado di seguire le misurazioni ricevute tramite V2V, anche se caratterizzato da un rumore evidente. Poco dopo l'inizio dell'attacco di iniezione, nella parte destra, i valori ricevuti e la stima iniziano a divergere, evidenziando immediatamente l'incoerenza. Quindi introducendo una soglia orizzontale posta ad

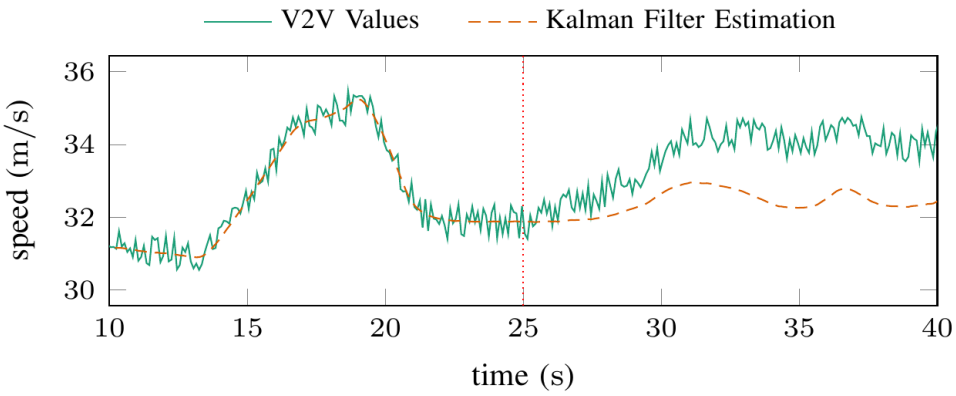

Figura 5.2: Applicazione del filtro di Kalman e detection dell'attacco

Tabella 5.1: Risultati paper [3]

Injection Attack	w/o radar		w/ radar	
	Detect. [%]	Delay [s]	Detect. [%]	Delay [s]
None	1.2	-	0.6	-
Position	99.8	1.75	100.0	1.75
Speed	99.8	2.81	100.0	2.81
Acceleration	99.8	3.80	100.0	3.79
All	99.8	1.76	100.0	1.75
Coordinated	4.4	6.00	100.0	3.90

una determinata altezza è possibile identificare questo andamento. Di fatto questo è rappresentato dalla parte destra delle disuguaglianze.

5.2.2 Pregi, difetti e risultati ottenuti

I risultati ottenuti da questa tecnica sono stati raccolti in una tabella (Tabella 5.1) che riassume la risposta di questo algoritmo di fronte a 1000 simulazioni. Essa in particolare mette in evidenza la percentuale di attacchi di iniezione identificati correttamente insieme al ritardo medio tra l'inizio dell'attacco e l'istante di rilevamento effettivo.

E' bene distinguere i risultati in base alla presenza o meno del radar, in particolare, quando è possibile utilizzare il radar vuol dire che chi sta rilevando l'attacco è il diretto inseguitore dell'attaccante, quindi dato che lo scenario utilizzato è: attacco eseguito dal leader, questo è riferito al secondo veicolo, mentre quando non è possibile utilizzare il radar, ma ci si affida alle sole informazioni V2V, si stanno considerando

tutti gli altri veicoli (dal terzo in poi), anche se questi in minima parte percepiscono l'influenza del veicolo che li precede.

La prima riga di questa tabella indica il numero di falsi positivi, ovvero la percentuale di attacchi rilevati che in realtà non avrebbero dovuto essere classificati come tali. Le successive mostrano gli altri tipi di attacchi analizzati; cioè relativi alle singole variabili e quando sono combinate. Nel complesso, è evidente l'efficacia della tecnica di rilevazione: tutti i tentativi di falsificazione sono stati individuati correttamente in pochissimi secondi. Come previsto, tuttavia, gli attacchi coordinati sono stati i più difficili da rilevare. Ed è su questo tipo di attacco, e sui falsi positivi, che si concentreranno i futuri sviluppi e adeguamenti.

Per quanto riguarda le diverse disuguaglianze presentate, (5.2) e (5.3) hanno rappresentato il rilevamento di quasi tutti gli attacchi, motivo per la quale il ritardo di rilevamento è pressoché lo stesso in entrambi gli scenari Radar-NoRadar dato che questi sono valori V2V disponibili per tutti i veicoli; tuttavia negli attacchi coordinati è stato richiesto il contributo della (5.6) e della (5.7), valori utilizzabili solo dal diretto inseguitore dell'attaccante, quindi, considerando i soli valori V2V la logica non è in grado di rilevare questo tipo di attacchi.

La contromisura adottata in questo algoritmo è disabilitare immediatamente il CACC nel momento in cui viene rilevato un attacco, e abilitare il classico ACC, utilizzando solamente le informazioni locali e quelle ottenute mediante radar; in ogni caso dopo essere stato abilitato, nessun attacco è riuscito a provocare un incidente, ma in un paio di casi molto specifici questi accadono. Banalmente, di per sé gli attacchi quando rilevati non causano mai un tamponamento, ma lo è la contromisura, questo perché nel momento in cui si degrada ad ACC il radar legge una distanza di pochi metri (5-10 m) dal veicolo precedente, molto inferiore ai metri di sicurezza richiesti da questi algoritmi (in base alla velocità di crociera), quindi il veicolo frena bruscamente, e se gli altri veicoli del plotone non eseguono la stessa manovra in maniera cooperativa, ma ognuno attende il riscontro del radar, prima o poi l'effetto molla si fa sentire, è più un veicolo è in coda al plotone più e probabile che questo accada. Immaginando uno scenario eterogeneo dove nel plotone sono presenti veicoli con caratteristiche diverse occorre considerare anche la potenza frenante di ognuno di questi.

Riassumendo:

- Le soglie assumono sempre lo stesso valore e non sono in grado di comprendere se il veicolo è in una fase di accelerazione o decelerazione;
- quando il radar non è accessibile ed in particolare negli attacchi coordinati molti di questi non sono stati rilevati;
- il numero di falsi positivi e i tempi di rilevamento anche se tutto sommato bassi, possono essere migliorati;

Le caratteristiche che distinguono questo algoritmo rispetto a molti altri esposti nel capitolo 4 ed in generale presenti in letteratura sono due:

- gli scenari di attacco considerati sono generati casualmente, tuttavia sussistono all'interno di range di validità, quindi non si tratta di attacchi impulsivi facilmente rilevabili in qualche decimo di secondo, ma di attacchi ben studiati mirati a compromettere il più possibile gli algoritmi cooperativi;

- escludendo gli attacchi di tipo coordinato, gli altri sono sempre stati individuati, questo perché, ad esempio, variare solo la posizione senza rispettare le equazioni del moto, può essere rilevato con un semplice check; di fatto questo tipo di attacchi invece di essere considerati tali sono stati studiati per simulare eventuali malfunzionamenti dei sensori, mentre gli attacchi coordinati rappresentano una vera e propria minaccia proveniente da un attaccante malevolo intento a generare appositamente degli incidenti.

Queste peculiarità sono posi state mantenute per tutti gli sviluppi successivi presenti in questo elaborato.

5.3 Primo approccio: thresholds reduction

Un primo miglioramento della tecnica appena descritta potrebbe essere la sostituzione del meccanismo "try-and-error" manuale con uno automatizzato, eseguito dall'elaboratore, e poi utilizzare questi nuovi valori nelle nuove simulazioni. Di fatto, questa soluzione rappresenta un upgrade della precedente, tanto che i concetti su cui si basa sono gli stessi. Gli obbiettivi di questo nuovo approccio sono: aumentare il numero di attacchi rilevati, e diminuire il tempo di rilevamento.

La realizzazione di questa soluzione è divisa sperimentalmente in due fasi, chiamate rispettivamente di train e di test, presenti anche nell'approccio successivo.

In una prima fase si sono analizzati i risultati ottenuti con le vecchie soglie, e sono stati calcolati i valori ottimali (gli α_i delle disuguaglianze (4.1)-(4.7)), che massimizzavano la percentuale di detection, e il tempo medio di rilevamento, mentre in una seconda fase è stata eseguita la validazione utilizzando delle nuove simulazioni confrontando l'output con l'approccio originale. Quest'ultima parte, oltre ad essere realizzata dal simulatore stesso, è stata implementata in python, mediante uno script realizzato appositamente per replicare fedelmente la logica del simulatore.

5.3.1 Architettura di funzionamento (fase di Train)

Osservando la Figura 5.3 (presa da una vera e propria simulazione) si è in grado di analizzare il funzionamento della logica presente nell'articolo [3] al fine di comprendere al meglio questo primo approccio, che come anticipato, sostituisce il

metodo try-and-error manuale con uno automatizzato, in pratica questa rappresenta la prima fase, quella di Train.

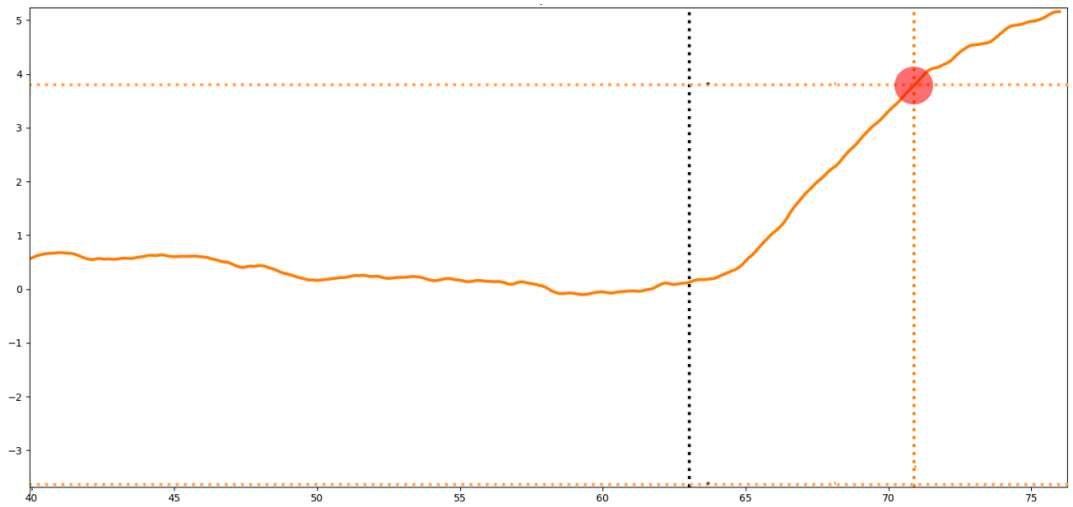

Figura 5.3: Andamento nel tempo discreto di uno dei valori considerati, con soglie e fase di detection.

Nella Figura 5.3 viene mostrato l'andamento nel tempo di uno dei sette valori sulla quale si basa la logica di detection, in particolare la linea arancione continua rappresenta l'evoluzione della disequazione (4.1), quindi il lato sinistro di essa ($|d_i^{\text{EST}} - \delta|$). Questo ragionamento può essere esteso facilmente a tutti gli altri valori, qui per semplicità si è deciso di analizzarne solo uno. La linea nera verticale tratteggiata evidenzia l'istante di tempo in qui parte l'attacco (circa al 63° secondo, valore estratto casualmente), mentre quella verticale arancione l'istante in cui viene eseguita la detection (oltre il 70° secondo), localizzata precisamente dal cerchietto rosso. Le due linee orizzontali tratteggiate rappresentano la safe zone, ovvero, la zona dentro la quale non è possibile rilevare un attacco, in particolare queste linee rappresentano la parte destra della disuguaglianza ($\alpha_1 \cdot \delta$), quest'ultimo risulta così ampio perché nella prima tecnica si è seguito un approccio più conservativo attribuendo alle soglie valori elevati, scelta guidata dalla necessità di ridurre il più possibile il numero di falsi positivi; infatti restringendo di molto questo range si rischia di rilevare un attacco ancor prima che questo venga attuato.

Tuttavia, come è evidente, tale intervallo può essere ampiamente ridotto, come mostrato in Figura 5.4. Prima di introdurre l'algoritmo è bene generalizzare questa logica per tutti i valori trattati, quindi per tutte le 7 disequazioni, in particolare quest'ultime possono essere sintetizzate come segue:

$$V_i(z) < \alpha_i \cdot Th_i(z)$$

dove:

- i : indica l'i-esima disequazione di quelle presentate in 5.2.1;
- $V_i(z)$: rappresenta l'andamento nel tempo discreto dell'i-esimo valore, quindi sono dei campioni equi spaziati ogni 0.1 secondi, valore imposto dalle simulazioni;
- α_i : è il parametro da determinare, colui il quale nella precedente tecnica veniva settato manualmente dall'utente;
- $Th_i(z)$: come per $V_i(z)$, ma è riferito all'andamento della soglia, per semplicità, ma come avviene nella maggior parte delle disequazioni, questa può essere considerata costante;

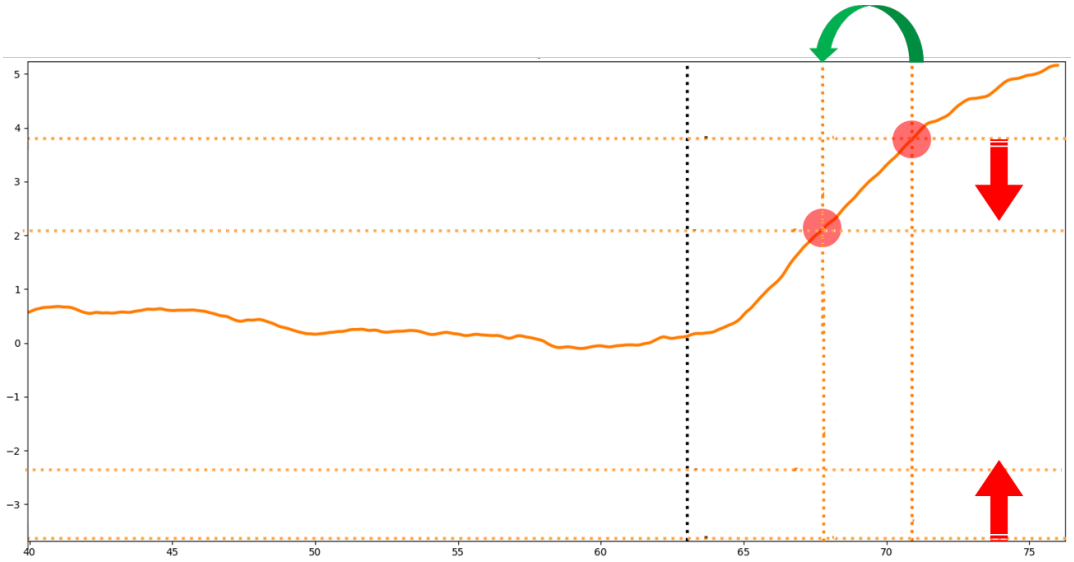

Figura 5.4: Rappresentazione grafica del funzionamento dell'algoritmo statico aggiornato.

L'algoritmo di riduzione è composto dai seguenti step:

1. Si considera ogni campione dall'istante in cui parte l'attacco:
2. Per ognuno di questi campioni viene calcolato α_i tale da garantirne la detection nel successivo ΔT_{th}; di fatto si trasforma la disequazione generale in una equazione e attraverso una formula inversa si ottiene l'α_i;
3. Se il nuovo valore di α_i non genera un falso positivo, quindi non rileva l'attacco prima che venga compiuto allora viene memorizzato in un vettore (Vec);
4. Se la condizione 3 è rispettata si procede con la simulazione successiva, altrimenti si ritorna la punto 2 e si prova con il campione successivo;

5. Terminate le simulazioni nel dataset di train il massimo valore memorizzato in Vec rappresenterà il nuovo α_i da testare nelle simulazioni di test;

Al termine di questo processo, graficamente avviene il restringimento segnalato dalle frecce in rosso, questo porta ad anticipare la linea di detection di qualche secondo (in questo caso circa 3 s), miglioramento indicato dalla freccia di colore verde.

5.3.2 Pregi e difetti

Nel capitolo successivo verranno analizzati nel dettaglio i risultati ottenuti da questa logica, in questa sezione verrà discusso invece il guadagno complessivo ricavato.

Tabella 5.2: Risultati con valori aggiornati.

Attacchi di Iniezione	w/o radar		w/ radar	
	Detect. [%]	Delay-Reduction [s]	Detect. [%]	Delay-Reduction [s]
None	1.0	-	1.0	-
Position	100.0	0.09	100.0	0.09
Speed	100.0	0.26	100.0	0.26
Acceleration	100.0	0.26	100.0	0.25
All	100.0	0.09	100.0	0.08
Coordinated	93	+1.32	100.0	0.48

Nella Tabella 5.2 sono riassunti i miglioramenti ottenuti utilizzando le soglie calcolate con la tecnica precedente. In prima analisi si nota come in alcuni casi la riduzione è risultata irrisoria, tale da riportare dei minimi vantaggi in termini di tempi di detection e numero di falsi positivi, che in alcuni casi sono risultati gli stessi, tranne per gli attacchi di tipo coordinato (questo perché negli altri casi si aveva un 100% di detection, che non è possibile migliorare).

Di fatto, questo comportamento era prevedibile, perché, proprio come indicato al quinto passaggio, prendendo il massimo tra i possibili valori di α_i è facile intuire come in alcuni casi, pur di evitare i falsi positivi, paradossalmente, la soglia ottenuta poteva essere addirittura maggiore di quella staticamente inserita dall'utente nell'approccio originale.

Le motivazioni per le quali questo algoritmo non presenta la soluzione al problema proposto, sono così riassunte:

- le soglie, anche se determinate automaticamente, sono comunque statiche;
- il tempo di detection per gli attacchi coordinati e troppo elevato,

- soprattutto la probabilità che si presentino dei falsi positivi può solo aumentare e non diminuire, per questa ragione durante la fase di restringimento si è tenuto conto di questo fenomeno, ma in ogni caso non può essere ridotto, proprio perché soglie più piccole vuol dire meno range di oscillazione, quindi, poco margine di sicurezza.

Quindi, concludendo, sia la logica originale, sia quella aggiornata, permettono di ottenere dei risultati discreti, ma non sono in grado di raggiungere gli obbietti prefissati (100% di detection e pochissimi falsi positivi), difficoltà guidata soprattutto da un approccio per nulla adattivo. Quindi occorre pensare ad una soluzione più generale, che permetta l'applicazione di una logica di detection, ottimale per tutti i vari scenari, compreso il coordinato. Per far ciò oggigiorno si hanno a disposizione degli algoritmi in grado di imparare e poi lavorare autonomamente senza l'intervento dell'uomo, in particolare si fa riferimento agli algoritmi di Machine Learning, settore sempre più in evoluzione che non conosce limiti di applicabilità.

5.4 Detection attraverso le reti neurali ricorrenti

5.4.1 Introduzione

Prima di analizzare nel dettaglio l'architettura del modello di rete ricorrente utilizzato nell'algoritmo finale presentato in questo elaborato, verrà effettuata una breve introduzione in modo tale da introdurre i vari step che si sono resi necessari prima di raggiungere la versione definitiva. I valori utilizzati in tutti gli algoritmi che sono stati testati da questo momento in poi, interessano esclusivamente la parte sinistra delle disequazioni dell'approccio originale mostato in Sezione 4.2.

In primo luogo sono stati testati i principali algoritmi di Machine Learning, tra qui:

- Logistic Regression
- Linear Discriminant Analysis
- K-Nearest Neighbors
- Decision Trees
- Naive Bayes
- Support Vector Machines

Successivamente le reti neurali ed infine le reti ricorrenti.

Per poter essere utilizzati, questi algoritmi necessitano di un dataset ben strutturato, maggiori dettagli sulla loro configurazione saranno forniti nella Sezione 6.6.1, ma per il momento è bene indicare le loro caratteristiche principali:

- Negli algoritmi standard di Machine Learning è stato utilizzato un unico dataset per tutti e sette i valori $V_i(z)$, dove per ognuno di questi veniva considerato il trend (o andamento) dei dieci campioni precedenti, quindi ad esempio, in una simulazione da 900 campioni (90secondi) vi erano 90 righe e 7 colonne;

- Per quanto riguarda le reti neurali, poi ricorrenti, è stato realizzato un modello per ognuno dei 7 valori, di conseguenza disponevano di un dataset separato e venivano considerati tutti i campioni in una finestra sperimentalmente fissata a 10 campioni, quindi, utilizzando l'esempio precedente, ogni simulazione veniva scomposta in 7 dataset, uno per ogni valore, ed in ognuno di questi vi erano 900 righe e 10 colonne.

Il motivo per la quale si è passati dagli algoritmi standard alle reti neurali è dovuto agli scarsi risultati che questi metodi hanno fornito, infatti, per quanto riguarda i tempi di detection, questi erano vicini all'approccio originale, ma, i falsi positivi, dopo aver configurato gli iperparametri in modo opportuno, erano di gran lunga superiori ai precedenti.

Mentre il passaggio da rete neurale standard a rete ricorrente è stato quasi immediato, infatti, le informazioni considerate in questi algoritmi sono molto dipendenti l'una dall'altra, dato che i valori in esame sono delle successioni numeriche, quindi introdurre un feedback forzato all'interno dell'algoritmo avrebbe portato sicuramente un vantaggio. Vantaggio evidenziato in Figura 5.5 dove viene analizzato, sulla sinistra l'andamento dell'accuratezza del modello addestrato con la sola rete neurale, mentre sulla destra il training del modello con la rete ricorrente più alcuni strati standard, infatti si nota come all'aumentare delle epoche questa aumenta in tutti e due i casi, ma nella figura di destra già a partire dalla prima epoca si nota un'accuratezza maggiore. Di fatto, aggiungere la ricorrenza permette di ottenere a parità di epoche una maggiore accuratezza.

5.4.2 Iperparametri rete neurale

Nella pratica dell'apprendimento automatico e del Deep Learning, i parametri del modello sono le proprietà dei dati che il modello apprende da solo durante l'addestramento dal classificatore. Ad esempio, pesi e bias.

Gli iperparametri del modello sono invece proprietà che governano l'intero processo di addestramento. Includono variabili che determinano la struttura della rete (ad esempio, Numero di unità nascoste) e le variabili che determinano il modo in cui la rete viene addestrata (ad esempio, Tasso di apprendimento). Gli iperparametri del modello vengono impostati prima dell'allenamento (prima dell'ottimizzazione dei pesi e dei bias). Ad esempio, ecco alcune variabili di configurazione integrate nel modello: tasso di apprendimento, numero di epoche, livelli nascosti, unità nascoste, funzioni di attivazione ecc...

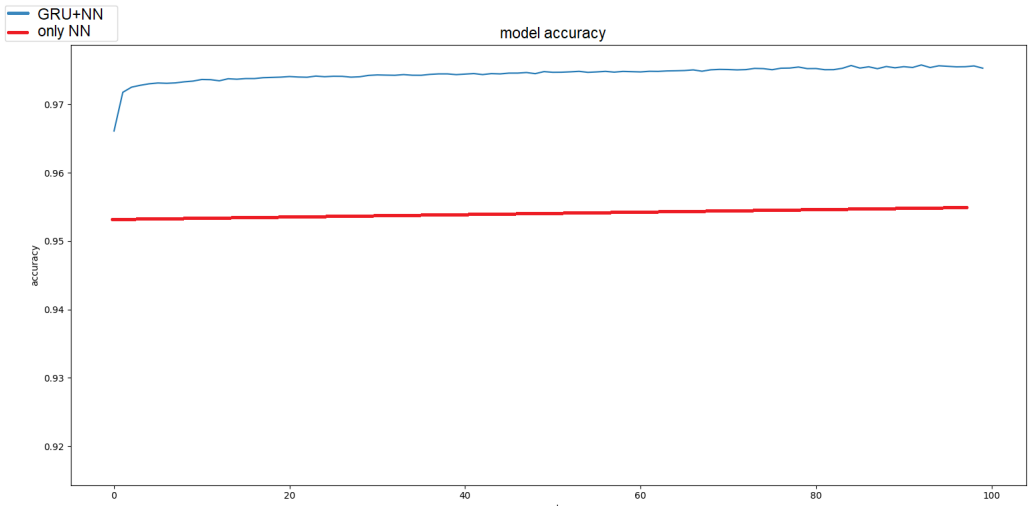

Figura 5.5: Confronto solo rete neurale con rete neurale più rete ricorrente

La scelta di iperparametri appropriati gioca un ruolo chiave nelle architetture di rete neurale, dato l'impatto che questi hanno sul modello.

Numero di unità nascoste: il numero di unità nascoste è uno degli iperparametri più misteriosi. Le reti neurali sono approssimatori di funzioni e per imparare a stimarle, devono avere una "capacità" sufficiente per apprendere la funzione stessa. Il numero di unità nascoste è la misura principale della capacità di apprendimento del modello. Per una funzione semplice, potrebbe essere necessario un numero inferiore di unità nascoste. Più complessa è la funzione, maggiore sarà la capacità di apprendimento di cui avrà bisogno il modello.

Un numero di unità leggermente superiore al numero ottimale non è un problema, ma un numero molto maggiore porterà ad un overfitting. Settando nel primo livello nascosto un numero di unità superiore al numero dei dati di input consente di ottenere dei risultati migliori in termini di accuratezza del modello.

Numero di livelli nascosti: i problemi che richiedono più di due livelli nascosti erano rari prima del deep learning. Due o meno livelli sono spesso sufficienti con semplici set di dati. Tuttavia, con set di dati complessi che coinvolgono serie temporali o visione artificiale, possono essere utili livelli aggiuntivi.

Numero di epoche: per scegliere il giusto numero di epoche per la fase di addestramento, la metrica a cui si deve prestare attenzione è l'errore di validazione e/o l'accuratezza.

Intuitivamente si potrebbe fare in modo tale che il modello venga addestrato per il maggior numero di iterazioni fino a quando l'errore di validazione continua

a diminuire. Esiste una tecnica che denominata Early stopping per determinare quando interrompere l'addestramento del modello; si tratta di interrompere il processo di addestramento nel caso in cui l'errore di validazione non sia migliorato nelle ultime 10 o 20 epoche.

L'arresto anticipato è una delle tecniche più popolari e anche efficaci per prevenire l'overfitting. La giustificazione di questa regola è abbastanza semplice: il punto in cui l'accuratezza inizia a diminuire è quando il modello inizia a sovra-adattare i dati di addestramento, poiché da questo punto in poi la sua capacità di generalizzazione inizia a diminuire. L'arresto anticipato può essere utilizzato da solo o in combinazione con altre tecniche di regolarizzazione.

Batch size: la dimensione del batch ha un effetto non banale sui requisiti di risorse del processo di addestramento, sulla velocità e sul numero di iterazioni. La tecnica comunemente utilizzata oggi consiste nell'impostare una dimensione del mini-batch. Valori di partenza consigliati per la sperimentazione: 1, 2, 4, 8, 16, 32, 64, 128, 256.

Una dimensione del mini-batch più grande consente miglioramenti computazionali, tuttavia, ha bisogno di più memoria per il processo di formazione. Una dimensione di mini-batch più piccola richiede meno memoria, ma più tempo di elaborazione.

Tasso di apprendimento (Learning rate): se il tasso di apprendimento del modello è troppo inferiore ai valori ottimali, ci vorrà un tempo molto più lungo (centinaia o migliaia) di epoche per raggiungere uno stato ideale. D'altra parte, se il tasso di apprendimento è molto maggiore del valore ottimale, allora supererebbe lo stato ideale e l'algoritmo potrebbe non convergere. Un ragionevole tasso di apprendimento iniziale = 0,001.

Dropout: il dropout è una tecnica di regolarizzazione per evitare l'overfitting (adattarsi ai dati di train) aumentando così il potere generalizzante.

Con il Dropout, alcuni neuroni vengono "eliminati" casualmente dalla rete ad ogni iterazione dell'addestramento (Figura 5.6). In generale, è consigliato utilizzare un valore di dropout del 20% -50%. Un valore troppo basso ha un effetto minimo mentre uno troppo alto comporta un sotto apprendimento da parte della rete. Si ottengono prestazioni migliori quando il dropout viene utilizzato su una rete più ampia, dando al modello più opportunità di apprendere rappresentazioni indipendenti.

Funzioni di attivazione: le funzioni di attivazione vengono utilizzate per introdurre la non linearità nei modelli. Queste hanno un effetto importante sulla capacità della rete neurale di convergere e sulla velocità di convergenza, o in alcuni casi, le funzioni di attivazione potrebbero impedirne la convergenza.

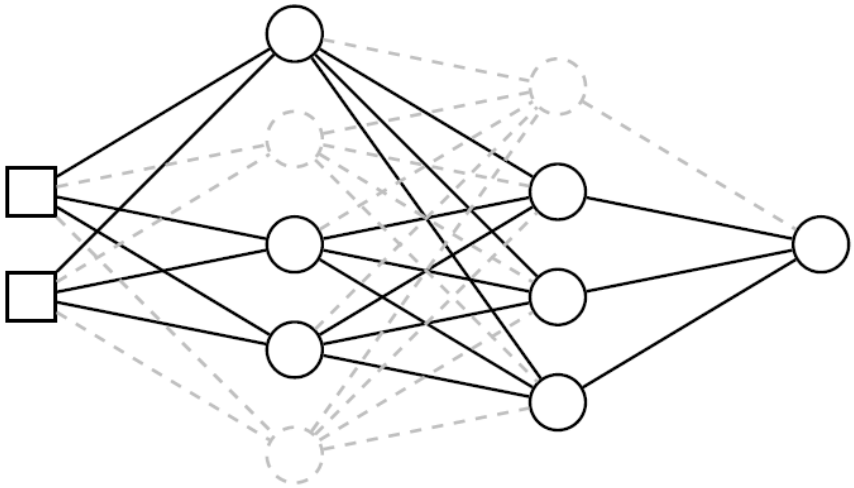

Figura 5.6: Come si comporta il livello di Dropout nelle reti neurali.

Le funzioni di attivazione sono equazioni matematiche che determinano l'output di una rete neurale. La funzione è collegata a ciascun neurone nella rete e determina se deve essere attivata o meno, in base al fatto che l'input di ciascun neurone sia rilevante per la previsione del modello. Le funzioni di attivazione aiutano anche a normalizzare l'uscita di ciascun neurone in un intervallo compreso tra 1 e 0 o tra -1 e 1. Esempi di funzioni di attivazione possono essere riassunti nella Figura 5.7.

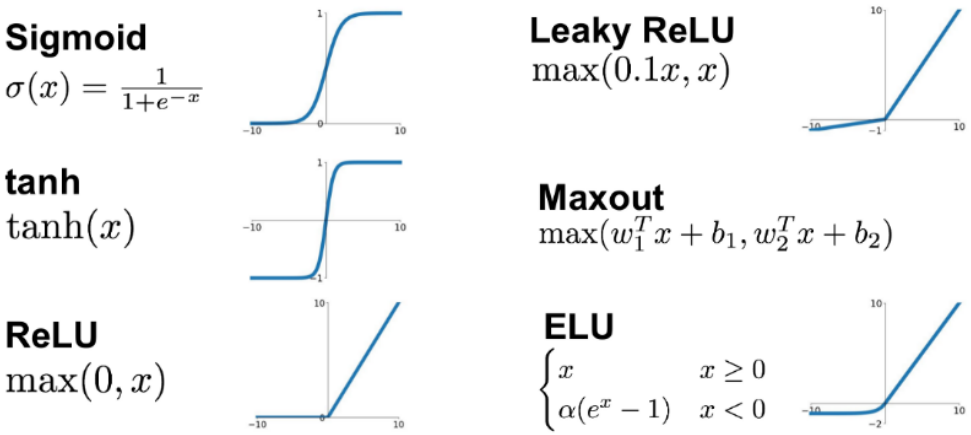

Figura 5.7: Tipiche funzioni di attivazione.

Un aspetto aggiuntivo delle funzioni di attivazione è che devono essere efficienti dal punto di vista computazionale perché vengono calcolate su migliaia o addirittura milioni di neuroni per ogni campione di dati.

Algoritmi di ottimizzazione: il modo in cui è possibile modificare i pesi e/o le velocità di apprendimento della rete neurale per aumentarne l'accuratezza è definito dall'algoritmo di ottimizzazione utilizzato. Di seguito verranno analizzati alcuni di questi evidenziandone i loro vantaggi:

- Gradient Descent: è l'algoritmo di ottimizzazione più semplice ma anche il più utilizzato. È ampiamente impiegato nella regressione lineare e negli algoritmi di classificazione. Anche la backpropagation nelle reti neurali utilizza un algoritmo di discesa del gradiente (per più informazioni a riguardo, si rimanda nella sezione background capitolo 2).

- Stochastic Gradient Descent (SGD): è una variante del Gradient Descent. Prova ad aggiornare i parametri del modello più frequentemente. In questo, i parametri del modello vengono modificati dopo il calcolo della 'loss function' su ogni esempio di addestramento. Quindi, se il set di dati contiene 1000 righe, SGD aggiornerà i parametri del modello 1000 volte in un ciclo del set di dati invece di una volta come nell'algoritmo precedente.

- AdaGrad: uno degli svantaggi di tutti gli ottimizzatori spiegati è che la velocità di apprendimento è costante per tutti i parametri e per ogni ciclo. Questo ottimizzatore cambia la velocità di apprendimento. Cambia il tasso di apprendimento "η" per ogni parametro e ad ogni fase temporale "t". È un algoritmo di ottimizzazione del secondo ordine.

- AdaDelta: è un'estensione di AdaGrad che tende a rimuovere il problema del decadimento del tasso di apprendimento. Invece di accumulare tutti i gradienti precedentemente calcolati, AdaDelta limita la finestra dei gradienti passati accumulati a una dimensione fissa w. In questa viene utilizzata la media mobile esponenziale piuttosto che la somma di tutti i gradienti.

- Adam (Adaptive Moment Estimation): si comporta come il precedente, solo che esegue quelle operazioni più lentamente, per una ricerca più attenta.

Il default, per le reti neurali, e l'algoritmo Adam, perché si presta a qualsiasi problema, l'impego degli altri algoritmi dipendono molto da come sono strutturati i dati di input e da come è formata la rete.

Procedura di normalizzazione: la normalizzazione è una tecnica spesso applicata come parte della preparazione dei dati per l'apprendimento. L'obiettivo della normalizzazione è modificare i valori delle colonne numeriche nel set di dati in una scala comune, senza distorcere le differenze negli intervalli di valori. Per l'apprendimento automatico, ogni set di dati non richiede la normalizzazione. È richiesto solo quando i dati hanno intervalli diversi.

Le due tecniche più popolari sono la normalizzazione e la standardizzazione. La normalizzazione ridimensiona ciascuna variabile di input separatamente all'intervallo 0-1. La standardizzazione scala ogni variabile di input separatamente sottraendo la media (chiamata centratura) dividendo per la deviazione standard per spostare la distribuzione in modo che abbia una media nulla e una deviazione standard di uno.

Inizializzare i pesi: nelle reti neurali, i pesi rappresentano la forza delle connessioni tra i nodi negli strati nascosti. La trasformazione lineare di questi pesi e dei valori del livello precedente passa attraverso una funzione di attivazione non lineare per produrre i valori per il livello successivo. Questo processo avviene da strato a strato durante la propagazione in avanti; attraverso la retro propagazione, i valori ottimali di questi pesi possono essere trovati in modo da produrre output accurati dato un input.

I pesi inizializzati in modo errato possono influire negativamente sul processo di addestramento e possono contribuire all'esplosione del problema del gradiente. Inizializzare i pesi a zero per la prima epoca è la soluzione più semplice, ma anche la meno efficace.

Ad esempio, con valori piccoli la situazione è analoga alla precedente, solo che è più rallentata, quello che accade e che la varianza del valore di input diventerà gradualmente sempre più piccola fino a quando non ha attraversato tutti gli strati e alla fine è svanita. Un piccolo valore di input continuerà a ridursi fino a quando il suo valore sarà così piccolo da non essere più considerato.

Un grosso problema che deriva da pesi molto piccoli è che rendono difficile introdurre la non linearità, che è vitale per gli algoritmi di machine learning più complessi. Partendo con pesi elevati la rete neurale diventa molto più sensibili a piccoli rumori nei dati. Rumore nei dati significa ulteriori informazioni inutili che non contribuiscono all'apprendimento e contribuiscono a cattivi risultati. Pertanto non è banale determinare questo tipo di valori, ed è per questo che in letteratura esistono molti inizializzatori, come ad esempio: uniform, glorot_uniform, he_normal, he_uniform, glorot_normal ecc [49].

5.4.3 Architettura del modello

Dopo aver analizzato a grandi linee il percorso seguito in questo elaborato analizziamo nel dettaglio l'algoritmo finale, colui il quale, rispetto agli approcci precedenti ci ha permesso di ottenere dei notevoli miglioramenti. Il punto di partenza, come anticipato nelle battute conclusive dell'introduzione del capitolo corrente, è localizzato in quelle che sono le reti neurali, da queste è stato poi possibile ottenere l'architettura finale. In particolare, dopo una fase di test e di documentazione la topologia della rete scelta è stata realizzata combinando i due approcci (RNN + NN), ovvero una prima parte nella quale sono stati inseriti dei livelli nascosti ricorrenti, mentre una seconda parte nella quale è stata caricata la rete neurale classica, questo sia per

velocizzare la fase di training sia per compattare i dati facendo riconoscere alla rete le sequenze simili. In Figura 5.8 si vuole rappresentare un'architettura di alto livello del modello utilizzato.

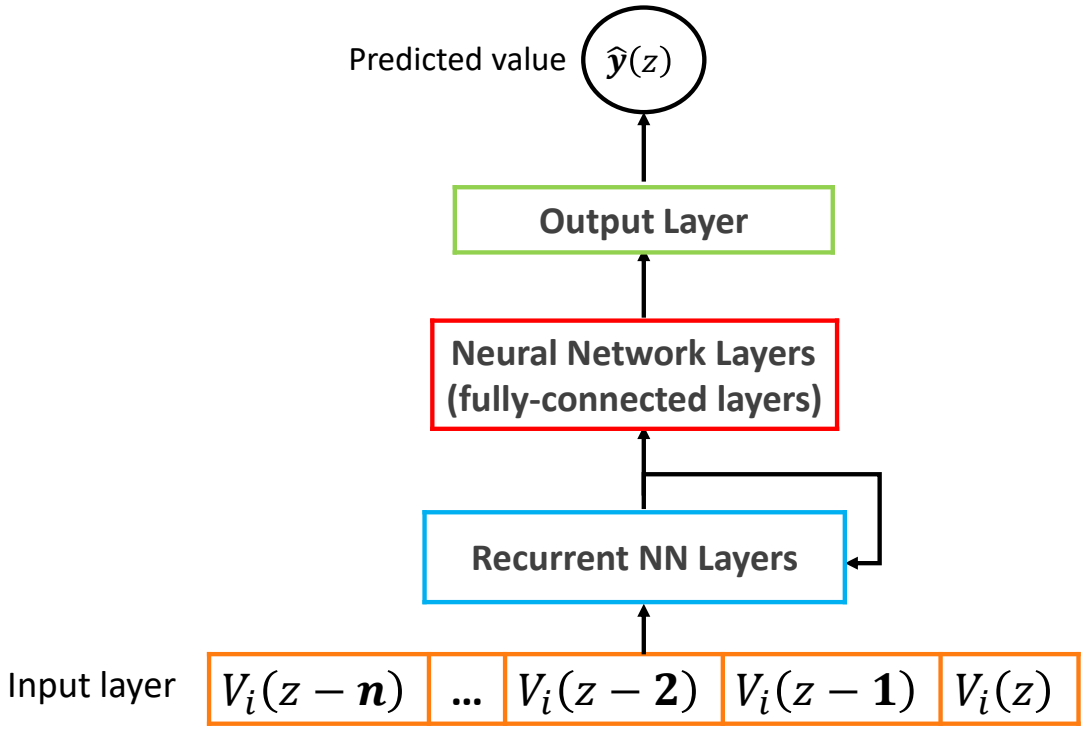

Figura 5.8: Architettura rete neurale ricorrente utilizzata

Osservando la Figura 5.8 andiamo ad analizzare passo passo tutti i suoi livelli, partendo dal basso verso l'alto:

- **Input layer**: per ogni time-step l'input del modello è ottenuto da sequenze di dati di lunghezza "n", dove "n" rappresenta la dimensione della finestra di campioni da considerare, questo può essere scelto in base a dei vincoli posti all'applicazione, in questo caso non ci sono dei veri e proprio vincoli, tuttavia, il suo valore può essere regolato per migliorare l'accuratezza della previsione. Istintivamente più "n" è alto più le previsioni migliorano, ma indirettamente aumenta il tempo di detection, perché sono richiesti più campioni. Nei nostri esperimenti è stato impostato n = 10 in modo tale che sia coerente con gli step precedenti;
- **RNN** layers + **NN** layers, data la loro struttura complessa saranno ampiamente discussi successivamente;
- **Output layer**: viene aggiunto un livello di output per riportare la previsione nel dominio dei dati originali. Questo strato lineare utilizza le funzioni di

attivazione "sigmoide" per scalare i valori tra 0 e 1, nel contesto in esame 0 ed 1 vogliono dire rispettivamente nessun attacco o attacco.

- L'output finale è una previsione $\hat{y}$, questa può essere interpretata come una misura di certezza anche se si presenta come una vera e propria probabilità. Ad esempio, un valore previsto di 0,9 indica che la rete neurale prevede con elevata certezza che il valore reale è 1, quindi si tratta di un attacco. Mentre valori intorno a 0,5 indicano che il modello non è sicuro della sua previsione.

Di conseguenza, una volta eseguita la predizione assume molta importanza la funzione di approssimazione utilizzata, ovvero, trovare quel valore che sia in grado di stabilire se data la sequenza di "n" valori in input, una predizione maggiore di "x" sia un attacco o meno. Inizialmente questo valore era stato settato a 0,8, poi successivamente portato a 0,95. La modifica di questo valore di soglia, impostato una sola volta manualmente, principalmente scatena il verificarsi di due eventi:

- In primis, un valore di "x" troppo elevato fa sì che il numero di falsi positivi sia pressoché nullo;
- Al contrario, valori troppo bassi, < 0.7, aumentano il numero di falsi positivi, ma anticipa di molto la detection, dato che in questa situazione, bastano solo pochi campioni per dire che si tratta di un attacco;

In realtà, oltre alle motivazione tecniche per la quale è stata scelta questa architettura (Figura 5.5), ve ne è anche una più logistica, ovvero, le reti neurali sono molto più semplici delle ricorrenti, il loro tempo di addestramento (come è stato riscontrato nei vari test effettuati), è di carica 5 volte inferiore, quindi, dato che entrambe le reti condividono gli stessi iperparametri (fondamentali per poter adattare il modello al caso specifico in esame) si è optato per un approccio più conservativo, infatti, per le NN sono state testate tutte le configurazioni possibili dei principali parametri, mentre per le RNN sono stati eseguiti dei test più mirati e specifici, dato che in questo caso testare tutte le configurazioni sarebbe stato logisticamente impossibile. Questi test effettuati in due fasi distinte ha permesso di specializzare il più possibile la seconda parte della topologia. Maggiori informazione per quanto riguarda le tempistiche e i testi effettuati sono fornite nella sezione apposita sezione (6.6.3).

5.4.4 Training e performance

Per le reti neurali la fase più importate tra quella di training e di test è senz'altro la prima, infatti, è in questa fase che il modello leggendo i dati da un dataset appositamente costruito è in grado di riconoscere dalle più semplici alle più complesse relazioni che sussistono tra essi. Ragion per cui questo step assume un notevole prestigio, soprattutto quando il modello, nonché la sua topologia, viene costruita

passo passo, partendo da livelli semplici, fino a combinazioni di parametri e sottolivelli sempre più sofisticati.

Il capitolo di background (Capitolo 2), e il capitolo 6, descrivono teoricamente (il primo) e praticamente (il secondo) in che modo le reti andrebbero costruite, evidenziando quelli che sono gli aspetti fondamentali da tenere in conto quando si esegue una progettazione di questo tipo. Soprattutto il capitolo 6 contiene una panoramica completa di tutti gli iperparametri testati indicandone anche una piccola descrizione, pertanto in questa sezione, nella quale sarà presentata la rete completa, questi saranno solamente citati.

L'architettura finale è mostrata in maniera semplice, ma al contempo, arricchita dei vari parametri in Figura 5.9, questa è stata ottenuta dal software di design Netron[1], che dato in input il modello generato dallo script python ne restituisce una sua rappresentazione grafica.

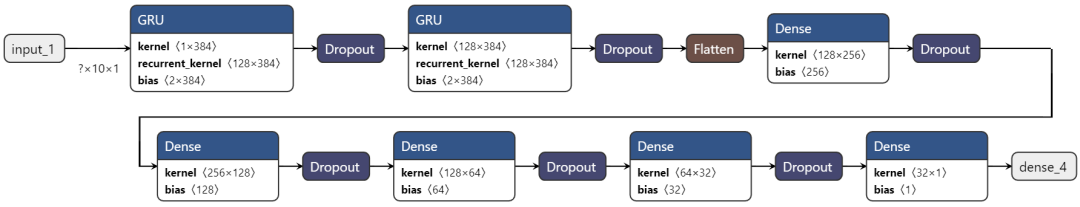

Figura 5.9: Architettura rete utilizzata - Netron

In particolare, il modello è composto da:

- Un InputLayer nella quale sono presenti i dati di input ai successivi livelli, questi rappresentano gli "$n = 10$" campioni da classificare;
- Sezione ricorrente, dove troviamo:
  - Un primo livello nascosto ricorrente composto da 128 nodi GRU, con funzione di attivazione "relu" ed inizializzatore uniforme;
  - Un secondo livello ricorrente composto, ancora una volta, da 128 nodi GRU, con iperparametri analoghi al precedente;
- Livello Flatten: questo è un livello che rende esplicito come serializzare un array multidimensionale (tipicamente quello di input). Ciò consente la mappatura tra l'array di input e il primo layer nascosto. Se il livello successivo è uno strato nascosto ed è "denso", quindi fully-connected (completamente connesso), ogni elemento del vettore di input (serializzato) sarà connesso con ogni elemento dell'array nascosto. Se non si utilizza Flatten, il modo in cui avviene il

[1] Netron: `https://netron.app/`

mapping sul primo livello Dense sarebbe ambiguo. Questo perché i livelli Dense accettano come input un vettore, non una matrice come per i livelli ricorrenti.

- Sezione completamente connessa, dove troviamo:
  - Un primo livello "Dense" composto da 256 nodi, con funzione di attivazione "relu" e inizializzatore dei pesi di tipo uniforme (come per i livelli ricorrenti);
  - Successivamente altri 3 livelli "Dense", analoghi al precedenti, composti rispettivamente da 128, 64 e 32 nodi;
- Infine l'ultimo livello "Dense" che rappresenta l'outputLayer precedentemente citato, con funzione di attivazione la "sigmoide" ed inizializzatore analogo ai precedenti livelli connessi. Quest'ultimo livello è quello che poi fornisce l'output del modello.

Tutti i livelli nella rete sono seguiti da uno strato di Dropout di valore 20% . Il dropout è uno strumento utilizzato durante l'addestramento di reti neurali che mira a evitare l'overfitting rimuovendo casualmente alcune unità nella rete.

Tutta la rete viene ogni volta percorsa interamente ad ogni epoca, in modo tale da aggiornare volta per volta i pesi in ogni sottolivello, pesi che assumono molta importanza nei livelli ricorrenti, dato che questi memorizzano al loro interno la storia passata, fondamentale per velocizzare la convergenza dell'interno modello. Terminata l'architettura della rete il modello può essere compilato, la fase di compilazione è caratterizzata da molti parametri, ecco i principali:

- Funzione di perdita o di costo (loss): essa è in grado di riassumere in un unico valore scalare tutti gli aspetti positivi e negativi del modello, in modo tale da semplificarne la comparazione. La funzione selezionata per questa rete è la BinaryCrossEntropy (BCE), scelta dettata dal tipo di classificatore, in questo caso trattasi di classificatore binario, tale funzione assume un valore tra 0 ed 1, e necessita che l'ultimo livello della rete abbia come funzione di attivazione la "sigmoide" e sia composta da un solo nodo.
- Il tipo di ottimizzatore: scelta ricaduta nell'ottimizzatore "Adam";
- Infine la metrica da considerare, trattandosi di classificazione, la metrica selezionata è l'accuratezza, essa assume un valore tra 0 ed 1, e più è vicina ad 1 più il modello si sta addestrando correttamente. In ogni caso questo valore, nella fase di train assume un'importanza minima, la vera accuratezza di un modello deve essere valutata sul dataset di test;

Per quanto riguarda la funzione di perdita e le metriche, queste vengono aggiornate ad ogni epoca. Una volta compilato può partire l'addestramento vero e proprio, nella quale vanno specificate il numero di epoche (50/100) e il batch-size da considerare (32).

Per poter valutare effettivamente le prestazioni del modello su dati non di train durante la fase di addestramento è possibile dividere il dataset di input; dove una porzione (solitamente il 70-80%) viene fornito alla rete, mentre la restante percentuale viene utilizzata per testare a run-time le prestazioni. Esistono comunque tecniche di validazione più sofisticate dove la divisione non è sotto il controllo del progettista ma varia ad ogni iterazione; un esempio di algoritmo di validazione è la cross-validation. Tuttavia, il test vero e proprio è bene che avvenga in uno step successivo a quelli di train, con dati che il modello mentre si stava addestrando non ha mai visto. Infatti, prima di implementare questi algoritmi all'interno del simulatore sono stati eseguiti vari test seguendo proprio questo approccio. Quindi, una volta terminato l'addestramento, realizzato in python, il modello generato da questa fase è stato esportato per poi essere importato, prima in script appositi per eseguire una validazione e confermare le sue prestazioni, e successivamente, se i risultati ottenuti erano abbastanza soddisfacenti veniva importato all'interno del simulatore; dove, di volta in volta ad ogni campione ricevuto sia dai sensori sia dalle comunicazioni V2V si è in grado di effettuare la detection degli attacchi.

Capitolo 6

Validazione

6.1 Introduzione

In questo capitolo verranno presentati gli esperimenti e risultati ottenuti dagli algoritmi di rilevamento testati. Inizialmente verrà data una panoramica generale sull'ambente utilizzato per eseguire le simulazioni, i parametri principali, necessari a poter replicare i risultati ottenuti ed infine come è stato settato l'ambiente nella fase di train delle reti neurali. Successivamente verranno analizzati i primi risultati ricavati dall'aggiornamento dell'approccio statico presentato in 6.5 andando a commentare le motivazioni per la quale si è deciso di non fermarsi davanti a questi esiti ma di procedere con tecniche più moderne.

L'ultima parte di questo capitolo interesserà gli algoritmi di Machine Learning, nella quale, inizialmente verrà fornita una descrizione approfondita di come sono stati elaborati i dati, poi i primi risultati ricavati dagli algoritmi standard di ML, ed infine una panoramica generale dei parametri utilizzati nelle reti neurali; a tal proposito sarà presentata la configurazione della rete neurale ricorrente e la sua implementazione, per poi concludere con i risultati ottenuti da quest'ultima tecnica.

Con i risultati ottenuti nell'approccio ricorrente si vuole dimostrare che:

- Le reti ricorrenti forniscono un buon modello per il rilevamento degli attacchi date le misurazioni dei sensori a bordo veicolo.

- È possibile rilevare gli attacchi, anche se costruiti appositamente per mettere in crisi l'algoritmo di detection;

- Nonostante il tempo di detection supera il secondo è comunque possibile evitare incidenti;

6.2 Configurazione ambiente di prova

Le simulazioni realizzate in questo articolo utilizzano il lavoro presentato in [50] e [51]. Nella quale il suo principale contributo è stata la progettazione di PLEXE, un'estensione per il framework di simulazione veicolare Veins che consente studi di ricerca su vari aspetti del platooning, tra cui la progettazione e la valutazione di algoritmi di controllo, protocolli di comunicazione e applicazioni. Lo stesso lavoro presenta un algoritmo di controllo del platooning che può essere adattato alle condizioni della rete. Inoltre, questo lavoro propone una serie di protocolli di trasmissione di informazioni non orientate (beaconing) che tengono specificamente conto dei requisiti dell'applicazione.

Eseguire le simulazioni è il metodo migliore per valutare le prestazioni di algoritmi di controllo e gestione dei veicoli prima di iniziare a implementare il prototipo reale. Ogni test può essere facilmente eseguito e ripetuto per l'analisi a posteriori. PLEXE fornisce dinamiche del veicolo realistiche e diversi modelli di controllo della velocità di crociera. Questo quadro consente una simulazione dettagliata della comunicazione wireless tra i veicoli, insieme a una mobilità realistica. PLEXE è basato su Veins[1], e accoppiato anche al simulatore di rete OMNeT++ con il simulatore di traffico stradale SUMO, il che significa che gli utenti possono beneficiare di uno stack di rete IEEE 802.11p e IEEE 1609.4 DSCR/WAVE completamente dettagliato per una simulazione realistica delle reti veicolari. Inoltre, estende SUMO implementando diversi modelli di controllo della velocità e dinamiche del motore. Grazie a questi vantaggi, il framework PLEXE è stato scelto per eseguire tutti i seguenti test di simulazione. Per i test è stato adottato il formato comune dei beacon e la procedura di diffusione dei messaggi standard utilizzata in PLEXE. I beacon nell'applicazione ITS sono messaggi di trasmissione periodici single-hop, trasmessi da ogni veicolo ogni 100ms, quindi 10 al secondo. Le informazioni ricevute per mezzo di questi beacon sono accelerazione, posizione e velocità.

Per quanto riguarda lo scenario generale, si è optato per simulare un plotone di otto veicoli che percorrono un'autostrada dritta ad una velocità costante desiderata variabile nel tempo, infatti all'interno delle simulazioni sono previste delle variazioni randomiche della velocità di crociera, che può sia diminuire che aumentare, ereditando il pattern di mobilità del [3]. Sebbene sia molto semplice, questo scenario è considerato la condizione operativa più frequente in situazioni realistiche. Inoltre, mira a evidenziare il più possibile l'effetto degli attacchi stessi, senza l'interferenza causata da perturbazioni esterne. Si suppone che il CACC venga utilizzato principalmente al di fuori delle strade cittadine, quindi gli scenari urbani non sono stati considerati.

[1] Veins: `http://veins.car2x.org`

La Figura 6.1 mostra una parte dell'autostrada con un plotone di otto veicoli. Ogni veicolo ha un ID assegnato, dove l'ID del comandante di plotone è uguale a 0, il primo veicolo che segue ha l'ID uguale ad 1 e così via. Per questo studio è stato ipotizzato un plotone omogeneo, supponendo che tutti i veicoli nella stringa mostrino un comportamento dinamico identico e implementino lo stesso modello di motore. Il simulatore di rete OMNeT++ rappresenta i veicoli sotto forma di nodi di comunicazione, come mostrato nella Figura 6.1. Tutti i nodi inviano segnali periodici per eseguire lo scambio di informazioni.

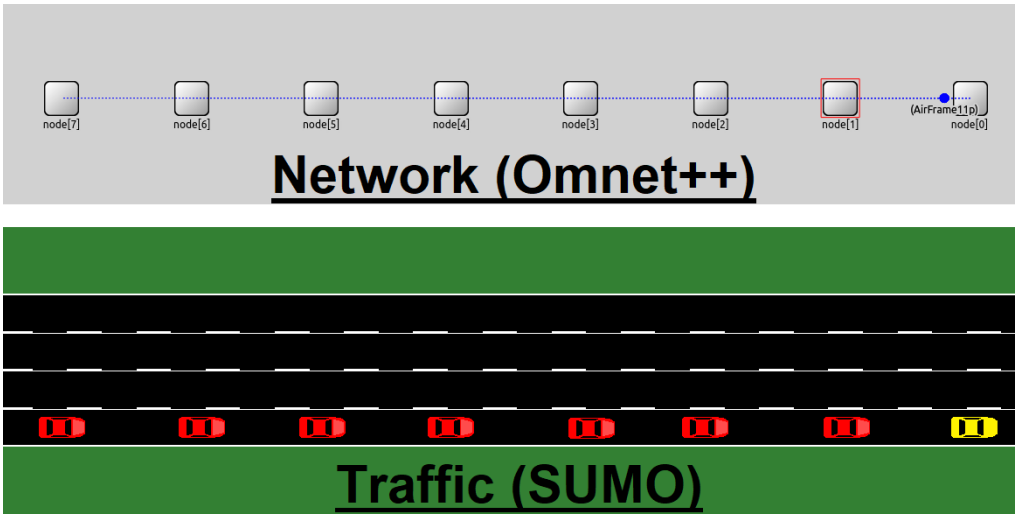

Figura 6.1: Comunicazione tra i veicoli, più illustrazione 2D del simulatore.

In ogni fase della simulazione, il nodo OMNeT++ passa i dati beacon ricevuti a SUMO. Quindi, SUMO utilizza queste informazioni come input per il controller CACC per calcolare la velocità e l'accelerazione. I valori risultanti vengono utilizzati per stimare la posizione del veicolo. SUMO restituisce questa posizione a OMNeT, che sposta i suoi nodi di conseguenza.

Prima di simulare gli attacchi di iniezione, sono state apportate alcune modifiche preliminari a Plexe, al fine di rendere più uniformi l'implementazione e il comportamento dei diversi controller. In particolare, è stata introdotta la possibilità di selezionare se utilizzare le misurazioni ottenute dal radar o i valori raccolti dalle comunicazioni V2V, nel caso siano disponibili entrambi. Inoltre, è stato implementato un modello per simulare le incertezze associate a ciascun sensore: una volta ottenuto un valore esatto da SUMO, viene sostituito da un altro estratto in modo uniforme da un intervallo di incertezza rendendo le misurazioni di questi sensori più realistiche. Queste modifiche sono disponibili in [3].

Tutti gli attacchi di iniezione sono stati simulati sostituendo il contenuto dei beacon legittimi con valori errati prima della loro trasmissione. Le diverse variabili considerate durante l'analisi comprendevano quelle richieste dai controllori

longitudinali (cioè posizione, velocità e accelerazione) sia individualmente che contemporaneamente. Per rendere gli attacchi più difficili da rilevare, si è optato per variazioni regolari nei valori iniettati, anziché cambiamenti bruschi e istantanei.

Pur mantenendo i parametri del controller principale ai loro valori predefiniti, come indicato precedentemente, e come in [3] è stata scelta una distanza intra-veicolare fissa per tutte le simulazioni, il valore scelto è 10 m (corrispondente a un gap di 0.36 s a 100 km/h), considerando un equilibrio tra una configurazione troppo compatta o troppo eccessivamente grande.

Si presume che ogni veicolo sia dotato di tutti i sensori necessari per raccogliere le misure richieste dagli algoritmi CACC, incluso un GPS, che oggi è considerato per lo più onnipresente nelle reti veicolari e un radar. Per quanto riguarda quest'ultimo, è sempre sfruttato per misurare la distanza relativa dal veicolo precedente, grazie alla sua precisione e affidabilità molto più elevate rispetto alla triangolazione GPS. Per quanto riguarda la valutazione della velocità del predecessore, d'altra parte, sono possibili due strade, la prima è utilizzare le sole informazioni V2V, mentre la seconda è utilizzare sia il radar che il V2V; in generale in un plotone il radar ha una visione completa del solo veicolo che lo precede, mentre degli altri non conosce nulla, pertanto in uno scenario cooperativo le V2V sono sempre considerate quando ricevute da veicoli diversi dal predecessore.

In linea con l'analisi effettuata capitolo 4 gli attacchi considerati sono quelli pervenuti dal veicolo leader, dato che questi risultane essere i più difficili da rilevare e più efficaci, dove per efficaci si intende che riescono in breve tempo a causare un incidente.

6.3 Parametri delle simulazioni

Nella Tabella 6.1 sono riassunti i parametri di simulazione utilizzati, riguardanti sia la generazione del modello di movimento, il comportamento dell'attaccante, l'algoritmo di rilevamento e alcuni valori dell'ambiente riprodotto.

Analizzando la tabella, nella sezione "Simulation" e "Communication" sono stati riportati i valori discussi poco fa, con qualche dettaglio in più sul protocollo di comunicazione, seguendo comunque i valori di default dell'ambiente PLEXE [3] [52].

In "Mobility" viene indicato il comportamento del veicolo controllato, i valori selezionati sono scelti in maniera random per ogni simulazione, rispettando il range configurato. A partire da una velocità iniziale viene indicata la probabilità con la quale all'interno della simulazione ci sia una frenata (quindi una decelerazione), oppure un aumento di velocità (accelerazione), il tutto rispettando le linee guida del plotone, ovvero non sono previsti eventi bruschi. La durata di questo evento (step duration) è anch'esso selezionato da un range di valori uniformi, oppure arbitrariamente assume un valore fisso molto piccolo per affrontare quelle situazioni dove la velocità varia in maniera impulsiva.

In "Attack", è indicato in quale maniera i valori iniettati dall'attaccante (il leader) vengono alterati. Anch'essi come in Mobility sono valori che variano di simulazione in simulazione, selezionati casualmente all'interno di un range, inoltre ad ogni test il momento in cui parte l'attacco è sempre diverso, ma non avviene mai prima dei 15 s e mai dopo il settantacinquesimo secondo, in simulazioni che durano 90 s. Nello specifico, ad ogni variabile è stato assegnato un rate di attacco, ed un valore massimo oltre il quale l'attacco dovrà arrestarsi.

Trattandosi di valori random, è stata adottata una tecnica per il quale, i valori sono comunque random, ma uguali rispetto all'esecuzione precedente, al fine di ottenere una ripetibilità delle simulazioni. Ad esempio, la decima simulazione è

Tabella 6.1: Parametri delle simulazioni.

	Parameter	Value
Simulation	Duration	90 s
	Controller	CACC
	Platoon-size	8
	Car lenght	5 m
	Intra-platoon time-gap	0.36 s (start)
	Intra-platoon distance-gap	10 m (start)
Mobility	Scenario	Random Value
	Initial speed	90 ÷ 110 km/h
	Maximum speed	130 ÷ 150 km/h
	Acceleration (min, avg, max)	0.1 m/s^2, 0.5m/s^{2}2.0m/s^2
	Acceleration probability	0.15 ÷ 0.25
	Deceleration (min, avg, max)	0.1 m/s^2, 0.75 m/s^2, 4.0 m/s^2
	Deceleration probability	0.15 ÷ 0.25
	Step duration	0.5s, 1.5 ÷ 3.0 s
Attack	Attacker	Leader vehicle
	Attack start	15 ÷ 75 s
	Position inj. (rate, limit)	1 ÷ 5 m/s, 25 ÷ 75 m
	Speed inj. (rate, limit)	0.05 ÷ 0.25 m/s^2, 1 ÷ 5 m/s
	Acceleration inj. (rate, limit)	0.025 ÷ 0.1 m/s^3, 0.25 ÷ 1 m/s^2
Communication	PHY/MAC model	IEEE 802.11p / 1609.4
	Frequency	5.89 GHz
	Bitrate	6 Mbits/
	Beacon size	150 Byte
	Beacon frequency	10 Hz
	Trasmit power	20 dBm

uguale ogni qualvolta la si esegue. Così i confronti con le varie tecniche di detection sviluppate sono più accurati e facilmente comparabili.

6.4 Ambiente di Training

Per l'esecuzione degli algoritmi di Machine Learning ci si è serviti del linguaggio di programmazione python, essendo il più adatto in queste circostanze, soprattutto perché offre un numero notevole di librerie in grado di ottimizzare al massimo l'esecuzione di algoritmi complessi e laboriosi. In particolare le scelte effettuate sono ricadute nei seguenti software e librerie:

- Python 3.6[2],
- Pip 20.1.1, Python Package Index (PyPI) è un repository di software per il linguaggio di programmazione Python.
- Tensorflow 2.2 con Python API[3], libreria software open source per l'apprendimento automatico.
- Keras 2.4.3 con back-end Tensorflow[4], come la precedente, è libreria open source per l'apprendimento automatico e le reti neurali, scritta in Python.
- Pandas 1.0.5 (Libreria di Python)[5], libreria per la manipolazione e l'analisi dei dati.
- Numpy 1.19.0 (Libreria di Python)[6], libreria che aggiunge un supporto di calcolo a grandi matrici e array multidimensionali offrendo metodi ottimizzati.
- scikit-learn 0.23.1 (Libreria di Python)[7], libreria che contiene al suo interno i principali algoritmi di Machine Learning.
- frugally-deep 0.15.1 libreria per importare il modello in C++

Configurazione del server sulla quale sono state eseguite le simulazioni:

- SO: Ubuntu 18.04.5 LTS;

[2] Python: `https://www.python.org/`

[3] Tensorflow: `https://www.tensorflow.org/`

[4] Keras: `https://www.keras.io/`

[5] Pandas: `https://pandas.pydata.org/`

[6] Numpy: `https://www.numpy.org/`

[7] scikit-learn: `https://scikit-learn.org/stable/`

- CPU: 2x16 core, Intel Xeon E5-2660;
- RAM: 128 GB;
- HD: 3TB;

6.5 Risultati primo approccio: thresholds reduction

Prima di analizzare le tecniche in grado di sfruttare gli algoritmi di Machine Learning, analizziamo i risultati del primissimo approccio, ricordando che questo è un primo miglioramento della tecnica descritta in [3] nella quale si vuole sostituire il meccanismo try-and-error manuale con uno automatizzato, al fine di poter ricavare valori di soglia meno conservativi, da testare nelle nuove simulazioni.

Tabella 6.2: Miglioramenti ottenuti per i rispettivi α_i.

α_i	Vecchio valore	Nuovo valore
α_1	0.33	0.13
α_2	1	0.83
α_3	1	0.84
α_4	0.25	0.11
α_5	1	0.84
α_6	1	0.72
α_7	1	0.97

Nella Tabella 6.2 sono riassunti i miglioramenti ottenuti per i rispettivi α_i. In particolare questi sono stati ottenuti considerando le prime 100 simulazioni come simulazioni di train. Mentre nelle tabelle successive, sono riportati i risultati ottenuti dalla fase di test, nella quale sono state considerate le simulazioni dalla 100° alle 199°. Nella Tabella 6.3 sono stati riportati gli esiti ricavati dai vecchi valori di α_i, mentre nella Tabella 6.4 quelli con i nuovi valori. In prima analisi si nota come in quei casi dove la riduzione è risultata irrisoria, ci sono stati dei minimi vantaggi in termini di tempi di detection, mentre i falsi positivi restano invariati (riga None). Tuttavia negli attacchi di tipo coordinato c'è stato un netto miglioramento degli attacchi rilevati, passati dal 3 al 93%, ma con un aumento considerevole del tempo di rilevamento che supera i 7 s.

Tabella 6.3: Risultati approccio orginale.

Injection Attack	w/o radar		w/ radar	
	Detect. [%]	Delay [s]	Detect. [%]	Delay [s]
None	1.0	-	1.0	-
Position	99.0	1.74	99.0	1.74
Speed	99.0	2.77	99.0	2.77
Acceleration	99.0	3.76	99.0	3.73
All	99.0	1.75	99.0	1.75
Coordinated	3	1.82	99.0	3.82

Tabella 6.4: Risultati con valori α_i aggiornati.

Injection Attack	w/o radar		w/ radar	
	Detect. [%]	Delay [s]	Detect. [%]	Delay [s]
None	1.0	-	1.0	-
Position	99.0	1.66	99.0	1.66
Speed	99.0	2.55	99.0	2.55
Acceleration	99.0	3.54	99.0	3.53
All	99.0	1.67	99.0	1.67
Coordinated	93	7.32	99.0	3.42

6.6 Algoritmi di Machine Learning

6.6.1 Data–preprocessing

Per poter essere utilizzati, gli algoritmi di Machine Learning necessitano di un dataset ben strutturato.

In particolare, i dati utilizzati in questi algoritmi sono gli stessi utilizzati nei precedenti approcci, ovvero i valori attesi che venivano confrontati con le soglie (la parte sinistra delle disequazioni 4.2), un problema che questo comporta è il fatto che questo tipo di informazioni sono variabili nel tempo (comunque tempo discreto), e riuscirle a tabellare non è banale, soprattutto perchè questi valori non sono affatto indipendenti l'uno dall'altro.

Per superare questo ostacolo esistono alcune metriche in grado di schematizzare queste informazioni. Le cosiddette metriche per le serie temporali. Le comuni metriche che possono essere estratte da una serie temporale sono moltissime, le principali sono: minimo, massimo, media, deviazione standard, trend ecc... Ma per poterle tabellare si rendono necessarie le cosiddette tecniche di ingegnerizzazione delle caratteristiche, tra qui troviamo:

- *Lag Feature*, tecnica di ritardo che permette di tabellare in una riga i precedenti "lags" value, ad esempio con un lag di 10, vuol dire che in ogni riga, oltre al valore corrente, sono presenti i precedenti 10.
- *Le Rolling/step windows*, viene calcolata una delle metriche classiche prendendo in input i valori compresi nella finestra, dove quest'ultima può spostarsi di un campione alla volta (rolling), o più di uno (step).
- Ed infine l'*Expanding window*, dove la finestra aumenta la sua ampiezza di volta in volta.

Tra le precedenti metriche elencate, una scelta iniziale è ricaduta in una tecnica a finestra, con un incremento fisso, fissato ad 1 secondo, e come metrica considerata il trend.

La Figura 6.2 rappresenta il comportamento di questa soluzione.

Questa tecnica è stata selezionata dopo un'attenta analisi dell'andamento di questi valori nel tempo, è si è visto che, quando questi iniziavano a divergere, o verso l'alto o verso il basso significava che in quel momento era in corso un attacco, quindi un dato come il trend che uno step prima è pressoché orizzontale mentre uno step successivo tende al verticale è stato considerato come un ottimo campanello di allarme.

Mentre la scelta di una soluzione a step, e non rolling è scaturito dal fatto che si voleva ridurre drasticamente il tempo di learning e la dimensione dei dataset da considerare, diminuendo sia l'uno che l'altro di un fattore 10 (proprio perché in un secondo abbiamo 10 campioni).

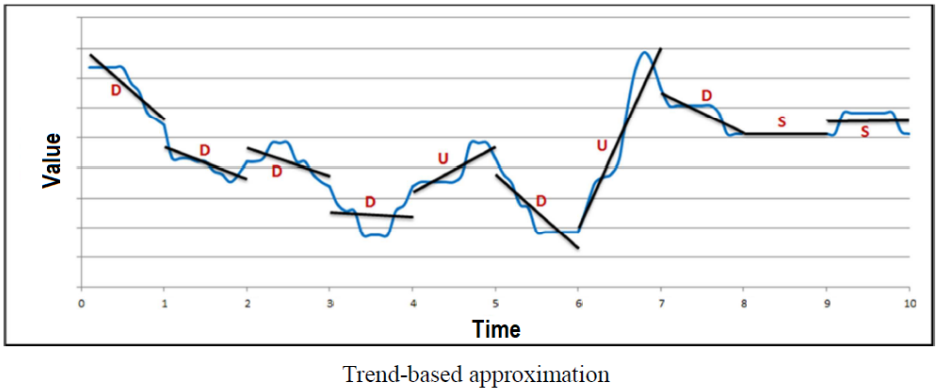

Figura 6.2: Approssimazione trend-based di una serie temporale.

Ad esempio, ipotizzando un dataset di train composto da 10 simulazioni da 90 secondi l'una, avrebbe assunto le seguenti caratteristiche:

- 90 righe per ogni simulazione;
- 7 colonne, che rappresentano le cosiddette "feature", ovvero i 7 V_i di $V_i(z) < \alpha_i \cdot Th_i(z)$;
- 1 colonna aggiuntiva per il "target", ovvero un valore binario che assumeva 1 se le "feature" rappresentavano un attacco, altrimenti 0;
- In totale: 90x10 righe + 8 colonne = 900x8

La scelta appena descritta è stata utilizzata con gli algoritmi standard di ML, mente con le reti neurali ci si è accorti subito che la metrica selezionata era troppo approssimativa, ragion per cui, invece di considerare un'approssimazione dei valori (trend), si è deciso di prenderli interamente, anche perché questi erano già mediati, ed un ulteriore generalizzazione portava ad un incremento del ritardo di detection. Media tra l'altro modificata in direzione di una lineare pesata, dove l'ultimo campione assume più importanza dei precedenti.

Successivamente, invece di considerare i valori all'interno di una finestra a gradino (cioè che si sposta di uno step per volta, step fissato ad 1 secondo), si è optato per una finestra scorrevole con incremento campione per campione (quindi ogni 0.1 s), aumentando sensibilmente il numero dei dati da analizzare, ma, anche se di poco ci sono stati dei miglioramenti, in pratica è stata realizzata la "Lag-feature" in una rolling-window.

Il dataset così costruito, rispetto al precedente risulta molto più grande, soprattutto se si considera che per le reti neurali è stato realizzato un modello per ogni valore; quindi riprendendo l'esempio precedente, con 10 simulazioni da 90 s, il dataset di train si sarebbe presentato in questo modo:

- 900 righe per ogni simulazione;
- 10 colonne di "feature", dove oltre al valore appena ricevuto dall'esterno (sensori o V2V) vi sono anche i precedenti 9;
- 1 colonna di "target" analoga alla precedente;
- In totale: 900x10 e 11 colonne = 9000x11;

Il totale appena calcolato si riferisce al singolo valore, quindi se si considerano 7 valori allora ci sono 7 dataset di train con queste caratteristiche.

6.6.2 Risultati algoritmi standard ML

Analizziamo i risultati ottenuti dai primi test effettuati con gli algoritmi di Machine Learning.

Nello specifico sono state utilizzate 100 simulazioni di train, e altrettante di test, per ogni tipo di attacco, così come per l'approccio statico precedentemente descritto. Osserviamo innanzitutto cosa accade quando è disponibile il radar[8] , esaminando il grafico a barre in Figura 6.3, dove sull'asse delle y troviamo il ritardo medio di detection espresso in secondi dei vari algoritmi testati, mentre sull'asse delle x i diversi attacchi. Questi risultati possono essere confrontati con l'approccio statico analizzato in precedenza, marcato da un contorno rosso molto spesso.

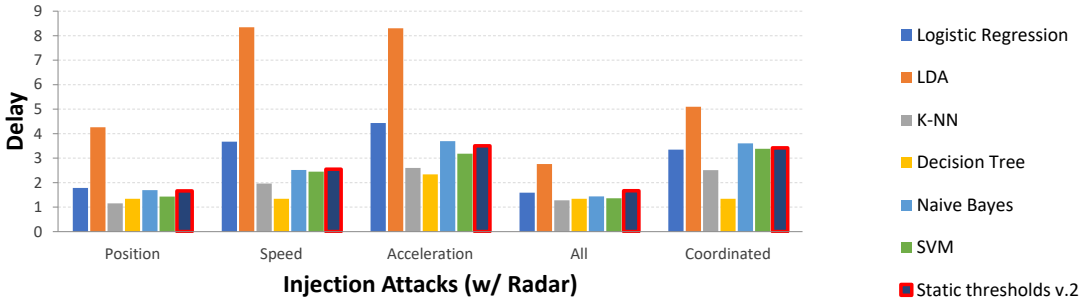

Figura 6.3: Confronto tra diversi algoritmi di Machine Learning in base all'attacco considerato

In una prima analisi, considerando solo i ritardi medi di rilevamento, sembra che l'algoritmo da escludere sia LDA, ma se andassimo ad analizzare anche il numero di

[8] opzione valida solo per il diretto inseguitore dell'attaccante, quindi il secondo veicolo

falsi positivi, esposti nella Tabella 6.5, si nota come in realtà sembra sia la scelta migliore. Oltre al LDA i risultati ottenuti dall'SVM sono tutto sommato positivi. In questo scenario la percentuale di detection è sempre stata del 100% per tutti i tipi di attacchi. Ovviamente quando vi è un falso positivo, questo viene registrato, ma, ipotizzando una logica più raffinata in grado di rilevare queste situazioni si è scelto di ignorare questo riscontro procedendo con la simulazione, in modo tale da verificare se effettivamente l'algoritmo è in grado di rilevare gli attacchi.

Tabella 6.5: Falsi positivi con algoritmi sstandard di ML

	Falsi positivi	
	(w/ Radar)	(w/o Radar)
LDA	0	0
Thresholds reduction	1	1
SVM	10	32
Logistic Regression	42	9
K-NN	78	94
Naive Bayes	88	60
Decision Tree	99	99

Prima di mostrare i risultati ottenuti quando il radar è disabilitato, analizziamo un aspetto tecnico fino ad ora non trattato, dei sette valori utilizzati per la detection, 3 sono V2V (Figura 6.4 in verde) e 4 sono con il radar (Figura 6.4 in blu), graficamente questo diagramma a barre rappresenta l'importanza media che ogni algoritmo attribuisce ad ogni proprietà per effettuare la classificazione; quindi nel momento in cui non è possibile utilizzare le ultime quattro perché il radar non è disponibile, questi algoritmi per forza di cose sono meno precisi, e come evidenziano le barre, in realtà, solo una di queste contribuirà effettivamente alla classificazione, mentre le altre solo in una minuscola percentuale.

Per gli algoritmi testati poco fa, questo si traduce in una percentuale media di detection scesa del 14%, quindi alcuni attacchi non sono stati per nulla rilevati, risultato poco promettente, ma in linea con le aspettative; dal punto di vista dei falsi positivi alcuni algoritmi sono migliorati, altri peggiorati, come mostrato nella tabella 6.5. In particolare si nota che i falsi positivi dell'LDA sono rimasti invariati, a scapito di una percentuale di detection comunque peggiorata (anch'essa scesa all'80%). L'unico algoritmo che è riuscito a rilevare tutti gli attacchi è stato l'SVM, ma con il 32% di dei falsi positivi.

Gli esiti dettagliati di queste sperimentazioni sono stati appositamente omessi perché esenti dallo scopo di questa ricerca, infatti, vedendo questi risultati il passaggio a metriche più dettagliate e l'implementazione delle reti neurali è stato quasi immediato.

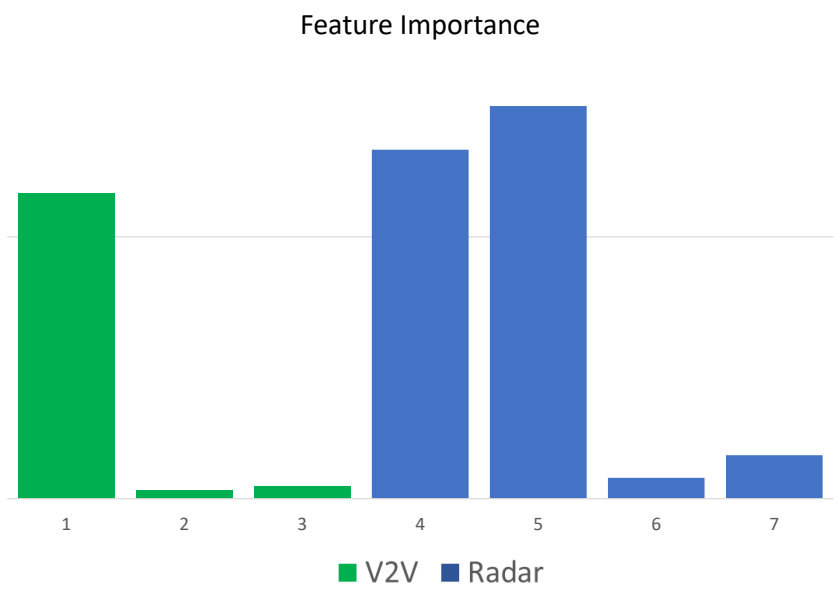

Figura 6.4: Livello di importanza dei valori considerati per effettuare la detection e come sono distribuiti

6.6.3 Ottimizzazione e configurazione degli iperparametri

Una volta presentati parte dei parametri testati nella sezione 5.4.2, in questa parte verrà discussa la procedura con la quale questi sono stati ricavati. Il processo per trovare gli iperparametri nel Machine Learning è chiamato ottimizzazione degli iperparametri. I due algoritmi più comuni sono:

- Grid Search
- Random Search

La **Grid Search** è una tecnica più tradizionale per l'implementazione di iperparametri. Si tratta di una tecnica 'a forza bruta' di tutte le combinazioni. La Grid Search addestra l'algoritmo per tutte le combinazioni utilizzando i set di iperparametri di input (velocità di apprendimento, numero di livelli, numero di nodi, epoche ecc...) e misura le prestazioni utilizzando una tecnica di cross-validation. Questa tecnica di convalida garantisce che il modello addestrato ottenga la maggior performance dei modelli dal set di dati (uno dei metodi migliori per eseguire la convalida è il "K-Fold Cross Validation" che aiuta a fornire ampi dati sia per l'addestramento del modello sia per la convalida).

Tuttavia, un tale approccio basato sulla forza bruta è computazionalmente costoso. Supponiamo che il modello richieda 3 ore per l'addestramento e abbia 2 iperparametri. Per testare 5 valori per ogni iperparametro, la Grid Search impiegherà $(5^2) \times 3 = 75\,\text{h}$ per valutare tutte le combinazioni.

Se il modello è veloce da addestrare, e ha pochi iperparametri o se sono disponibili grandi quantità di potenza di calcolo, Grid Search è una soluzione facile da implementare e contemporaneamente molto esaustiva.

Nel caso in cui non si disponga di tali quantità di tempo o risorse di calcolo, un'alternativa è la ricerca casuale. Invece di testare tutte le combinazioni, la ricerca casuale consiste nel campionare valori casuali nello spazio degli iperparametri. Sebbene bilancia il costo computazionale della Grid Search, questo approccio è inefficiente nel modo in cui esplora le combinazioni degli iperparametri. In effetti, l'algoritmo non impara dalle combinazioni precedentemente testate, svantaggio presente anche nella tecnica precedente, infatti con la Grid Search, si testeranno molte combinazioni equivalenti, sprecando quindi risorse e tempo di elaborazione.

Tutto sommato dato che la rete da addestrare in una prima fase processava una piccola quantità di dati, e all'incirca ogni epoca richiedeva meno di un minuto si è deciso di utilizzare la Grid Search; gli iperparametri testati sono riassunti nella tabella seguente.

Tabella 6.6: Iperparametri testati.

Iperparametro	Valori o Range
#livelli nascosti	2,3,4
#nodi livelli nascosti (Dense)	[128,32], [256,128,64], [512,256,128,64], [256,128,128,64]
Epoche	30, 50
Batch-size	16, 32, 64, 128
Funzioni di attivazione	Relu, tanh, sigmoid, softsign
Learning rate	0.001(default) , 0.0001
Dropout	0.2, 0.4, 0.5, 0.6, 0.8
Weight initializer	uniform, normal, glorot_normal, glorot_uniform, he_uniform
Optimizer	SGD, Adamax, Adam (default), Adagrad

Il motivo per cui i parametri testati sono "solo" questi è dovuto al fatto che prima di avviare la Grid search, sono stati eseguiti dei test locali, prendendo la combinazione di sottolivelli più complessa ([512,256,128,64]) e provando a variare un parametro alla volta, di fatto questa procedura è stata eseguita solo per diminuire il numero di funzioni di attivazione, di inizializzatori e di ottimizzatori utilizzati, dato che Keras è in grado gestirne molti di più (senza importare ulteriori librerie).

In particolare questi test hanno interessato 8 funzioni di attivazioni, 8 inizializzatori e 7 ottimizzatori. Quindi, terminata questa procedura, sono stati esclusi solo quei parametri che restituivano dei pessimi risultati, in particolare quelli esclusi per ogni iperparametro sono:

- Funzioni di attivazione escluse: softplus, hard_sigmoid, linear, softmax;
- Inizializzatori esclusi: lecun_uniform, zero, he_normal;

- Ottimizzatori esclusi: RMSprop, Adadelta, Nadam;

Da questa tabella si evince innanzitutto come il numero di epoche è stato mantenuto basso, questa scelta è stata guidata, sia per motivi logistici temporali, sia perché nelle combinazioni dove il Dropout era molto basso, si è osservato sperimentalmente che all'incirca, intorno alla trentesima epoca l'accuratezza del modello o si stabilizzava per le successive epoche, oppure iniziava a diminuire, con un corrispondente aumento della loss-function, quindi per evitare uno spreco di risorse sono stati utilizzati solo due valori.

6.7 Configurazione rete ricorrente

6.7.1 Introduzione

Nel capitolo 5 è stata presentata l'architettura dell'approccio relativo all'utilizzo della rete neurale ricorrente, specificando quali parametri sono stati utilizzati e la topologia della rete stessa. In questa sezione invece, verranno esibiti i risultati ottenuti evidenziando i vantaggi che questa tecnica possiede rispetto ai precedenti algoritmi testati e presentati in questo elaborato; tuttavia, prima che questo avvenga, verrà dedicata una sezione nella quale si è voluto mettere in risalto una particolare scelta di progettazione, ovvero utilizzare nodi ricorrenti di tipo GRU, rispetto ai nodi LSTM; questo non è altro che un piccolo perfezionamento guidato principalmente dalla letteratura, infatti il confronto tra queste due tecniche non è tra gli obbiettivi di questa ricerca, mentre lo è effettuare la detection il prima possibile con metodi poco dispendiosi in termini di risorse di calcolo e di tempo, vantaggi ottenibili con le GRU.

Ciononostante gli ci si è comunque voluto dedicare un'intera sezione, questo perché gran parte dei test effettuati prima di realizzare quelli definitivi facevano uso di questi nodi (LSTM), infatti solo in una seconda fase di sviluppo, dopo un'attenta documentazione, si è deciso di aggiornare i modelli utilizzando tecniche più recenti, per l'appunto le GRU.

6.7.2 LSTM vs GRU

In questo capitolo verranno presentate le motivazioni che hanno portato ad una variazione del modello di machine Learning considerato, variando i nodi dei livelli intermedi ricorrenti da LSTM a GRU.

Innanzitutto, in una prima fase sperimentale, i livelli ricorrenti erano stati realizzati utilizzando nodi LSTM, ma dopo aver analizzato lo stato dell'arte, ed i vantaggi che le GRU hanno rispetto alle LSTM, questi livelli sono stati variati seguendo l'approccio più moderno, appunto le GRU.

In parole semplici, l'unità GRU non deve utilizzare un'unità di memoria per controllare il flusso di informazioni come l'unità LSTM. Può utilizzare direttamente tutti gli stati nascosti senza alcun controllo. Le GRU hanno meno parametri e quindi possono addestrarsi più velocemente o richiedere meno dati per generalizzare. Ma, con dati di grandi dimensioni, le LSTM possono portare a risultati migliori.

Le GRU sono quasi simili alle LSTM tranne per il fatto che hanno due porte: reset gate e update gate. Reset gate determina come combinare il nuovo input con la memoria precedente e update gate determina quanto dello stato precedente mantenere. Update gate in GRU è ciò che in LSTM erano il gate di input e il forget gate.

Prove sperimentali [53] [54] [55], in cui i due metodi sono stati pesantemente testati hanno portato ai seguenti risultati:

- Le GRU si allenano più velocemente e hanno prestazioni migliori delle LSTM quando si hanno pochi dati di addestramento;
- Le GRU e le LSTM a livello computazionale richiedono le stesse risorse;
- Le GRU sono più semplici e quindi più facili da modificare, hanno pochi parametri e sono facilmente estendibili. Quindi meno codice.
- Le LSTM dovrebbero in teoria memorizzare sequenze più lunghe delle GRU e superarle in compiti che richiedono la modellazione di relazioni a lunga distanza. Il "dovrebbe" è giustificato da un numero di test in questa direzione ancora troppo basso, data la modernità delle GRU.

6.7.3 Implementazione

Prima di addentrarci nell'implementazione vera e propria del modello, nel seguente elenco numerato verranno indicati, brevemente, tutti gli aspetti fondamentali necessari per comprendere al meglio il percorso effettuato e soprattutto come mai il modello finale ha queste determinate caratteristiche:

1. I valori considerati per fare la detection sono gli stessi utilizzati negli algoritmi standard di ML, quindi la parte sinistra delle formule presentate in 4.2;
2. Il dataset di train considerato è quello descritto nell'ultima del paragrafo 6.6.1, quindi: metrica lag-feature di 1 secondo e un dataset per ogni valore;
3. La rete realizzata è composta da due parti ben distinta, una prima parte ricorrente ed una seconda parte completamente connessa priva di feedback (sezione 5.4.3);

4. I parametri di cui è composta la seconda parte della rete sono stati frutto di innumerevoli test, nella quale ognuno di questi variava un parametro alla volta e ne valutava le sue prestazioni, così come descritto poco fa in 6.6.3 dove viene introdotta la Grid Search;

5. Mentre gli iperparametri della parte ricorrente sono stati ricavati seguendo uno stile RandomSearch, anche se effettivamente i test non sono stati casuali, anzi, derivano da simulazioni mirate e specifiche, onde evitare lunghe attese addestrando modelli inutilizzabili.

Ai valori del punto 1, ne è stato aggiunto un altro che tiene conto della differenza di velocità stimata mediante il filtro di Kalman ottenuta attraverso la V2V ($|\overline{\Delta v_i^{\text{EST}}}|$). La ragione per la quale si è reso necessario utilizzare un'informazione in più è stata frutto dei risultati ottenuti con i precedenti algoritmi (maggiori dettagli in 6.6.2), infatti quest'ultimi negli attacchi coordinati soffrivano (e non poco) quando il radar era disabilitato, pertanto per ovviare a questa problematica è stato aggiunto un ulteriore dato indipendente, ottenibile dalla comunicazione V2V; la scelta è ricaduta in un valore che tenesse conto della velocità, perché questo tipo di dato è molto lineare, in un tipico scenario normale non dovrebbe avere forti variazioni, né tantomeno variare impulsivamente; così come è accaduto quando si è provato ad aggiungere l'accelerazione, essa infatti in molte simulazioni variava in modo repentino per poi stabilizzarsi.

In Figura 6.5 sono riportati i passaggi essenziali per ottenere una previsione accurata in real-time all'interno di PLEXE.

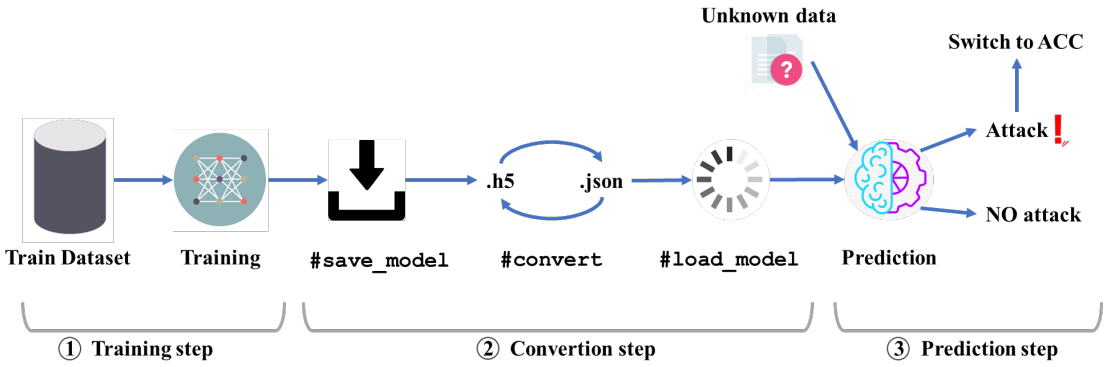

Figura 6.5: Step per ottenere la rilevazione a run-time nel simulatore.

Terminato l'addestramento (step 1 – Figura 6.5), attraverso la libreria joblib[9] è possibile salvare su disco il modello appena costruito, successivamente questo viene

[9] joblib: `https://github.com/joblib/joblib`

fornito in input alla libreria frugrally-deep[10] in grado di interpretare il modello ed i relativi pesi aggiornati all'ultima epoca, per poi generare un file con estensione .json da fornire in input al simulatore c++ mediante la funzione load_model messa a disposizione dalla libreria stessa (step 2 – Figura 6.5).

Infine, per ogni simulazione vengono caricati i modelli generati all'interno del simulatore, costui effettua la predizione per ogni nuovo valore ricevuto dall'esterno, radar o V2V che sia, se questa supera la soglia di affidabilità, fissata a 0.95, allora viene considerato come attacco, di conseguenza scatta il meccanismo di fallback, ovvero "switchare" da CACC ad ACC (step 3 – Figura 6.5).

Di seguito è riportato il codice in grado di realizzare i modelli necessari per effettuare la detection.

```
inputs = Input(shape=(10,1))
L1r = GRU(128, activation='relu', return_sequences=True, kernel_initializer='uniform')(inputs)
L1rd = Dropout(0.2)(L1r)
L2r = GRU(128, activation='relu', return_sequences=False, kernel_initializer='uniform')(L1rd)
L2rd = Dropout(0.2)(L2r)
Fl = Flatten()(L2rd)
L1 = Dense(256, activation='relu',  kernel_initializer ='uniform')(Fl)
L1d = Dropout(0.2)(L1)
L2 = Dense(128, activation='relu',  kernel_initializer ='uniform')(L1d)
L2d = Dropout(0.2)(L2)
L3 = Dense(64, activation='relu',  kernel_initializer ='uniform')(L2d)
L3d = Dropout(0.2)(L3)
L4 = Dense(32, activation='relu',  kernel_initializer ='uniform')(L3d)
L4d = Dropout(0.2)(L4)
predictions = Dense(1, activation='sigmoid',  kernel_initializer ='uniform')(L4d)
model = Model(inputs=inputs, outputs=predictions)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
es = EarlyStopping(monitor='loss', mode='min', verbose=0, patience=49)
model.fit(X_train,y_train, epochs=50, batch_size=32, verbose=0, callbacks=[es])
```

[10] frugrally-deep: `https://github.com/Dobiasd/frugally-deep`

Tabella 6.7: Risultati approccio statico con 1000 Simulazioni.

Injection Attack	w/o radar		w/ radar	
	Detect. [%]	Delay [s]	Detect. [%]	Delay [s]
None	1.2	-	0.6	-
Position	98.8	1.75	99.4	1.75
Speed	98.8	2.81	99.4	2.81
Acceleration	98.8	3.80	99.4	3.79
All	98.8	1.76	99.4	1.75
Coordinated	4.4	6.00	99.4	3.90

Tabella 6.8: Risultati finali (incidenti 0.2%).

Injection Attack	w/o radar		w/ radar	
	Detect. [%]	Delay [s]	Detect. [%]	Delay [s]
None	0.9	-	0.9	-
Position	99.1	0.9	99.1	0.9
Speed	99.1	2.1	99.1	2.0
Acceleration	99.1	3.5	99.1	3.4
All	99.1	0.9	99.1	0.9
Coordinated	99.1	6.50	99.1	3.5

6.7.4 Risultati e considerazioni

Una volta analizzata l'architettura dell'approccio relativo all'utilizzo della rete neurale ricorrente, specificato quali parametri sono stati utilizzati e la topologia della rete scelta, vediamo i risultati ottenuti. In particolare, gli esiti qui presentati fanno riferimento ad un dataset di train composto da 200 simulazioni di train, e da 1000 di test. Per poter apprezzare i vantaggi della nuova tecnica di rilevamento introdotta in questo elaborato verrà fatto un confronto con la tecnica statica utilizzando le stesse simulazioni; in particolare nella tabella 6.7 sono riportati i risultati con le soglie statiche, mentre nella tabella 6.8 gli esiti ricavati dall'algoritmo ricorrente.

Osservando questi risultati si nota come il tempo complessivo di detection è sceso in tutti gli attacchi, ma soprattutto è stato possibile scendere sotto il secondo in due di questi; situazione che per come era stata implementata la logica statica non poteva mai accadere.

Da notare come solamente 2 simulazioni su 1000 hanno causato un incidente;

tuttavia quest'ultimi necessitano di un'ulteriore investigazione, dato che apparentemente sembrano siano legati a problemi interni del simulatore stesso, infatti queste venivano arrestate e catalogate come incidente anche se questo non fosse mai accaduto.

Mentre nel momento in cui il radar non è più disponibile i risultati ottenuti sono gli stessi, tranne per gli attacchi coordinati, dove le reti ricorrenti hanno mantenuto la stessa percentuale di detection degli scenari precedenti, ma il tempo di detection per quanto possa sembrare alto, e comunque il valore più basso ottenuto fino a questo momento. La motivazione per cui questo accade è che questi tipi di attacchi sono proprio difficili da rilevare dato che i valori seguono le informazioni contraffatte, quindi anche se con un po' di ritardo gli attacchi sono comunque rilevati, almeno rispetto alla tecnica statica che ne rilevava solo il 4,4%. Infatti Position, Speed, Acceleration e All non andrebbero considerati come dei veri e propri attacchi, ma possono essere visti come degli eventuali malfunzionamenti dei sensori, mentre quelli coordinati rappresentano una vera e propria minaccia proveniente da un attaccante malevolo intento a generare appositamente degli incidenti.

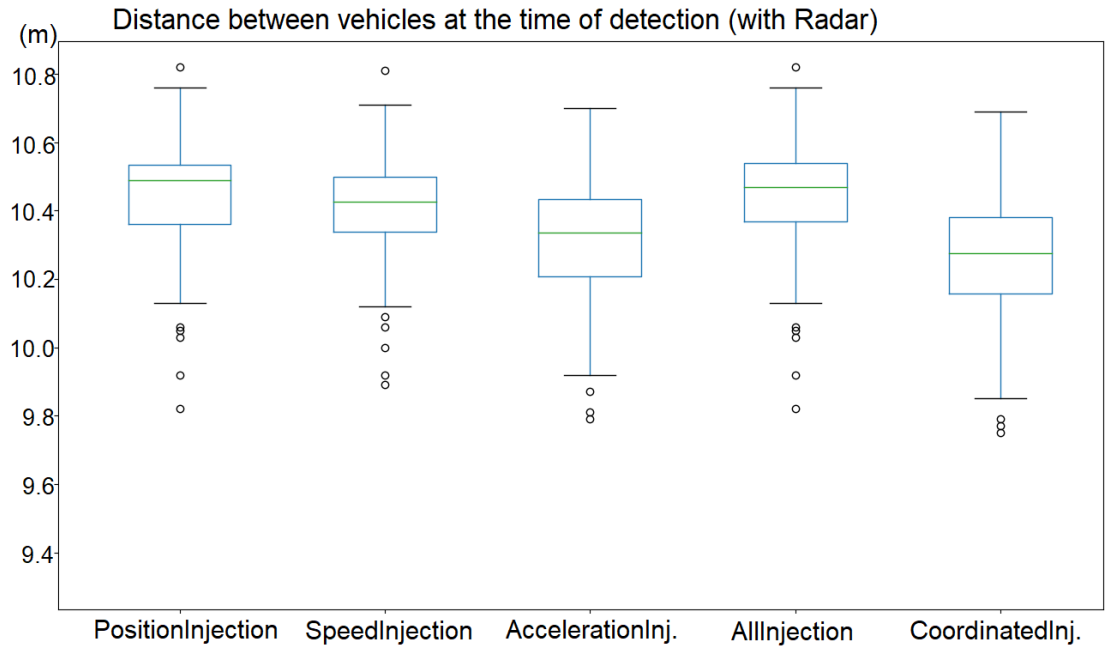

Figura 6.6: Distanza tra due veicoli nel momento della detection (con radar).

Lo scopo di questa ricerca era sì ottenere dei risultati migliori, ma parallelamente, evitare il più possibile il numero di incidenti, pertanto i grafici in Figura 6.6 e Figura 6.7 ritraggono due box-plot in grado di sottolineare la distribuzione della distanza che sussiste tra due veicoli (rispettivamente nel caso in cui c'è o meno il radar). In particolare l'asse delle ordinate denota la distanza in metri, mentre sull'asse delle ascisse, gli attacchi considerati; partendo da un setup delle simulazioni

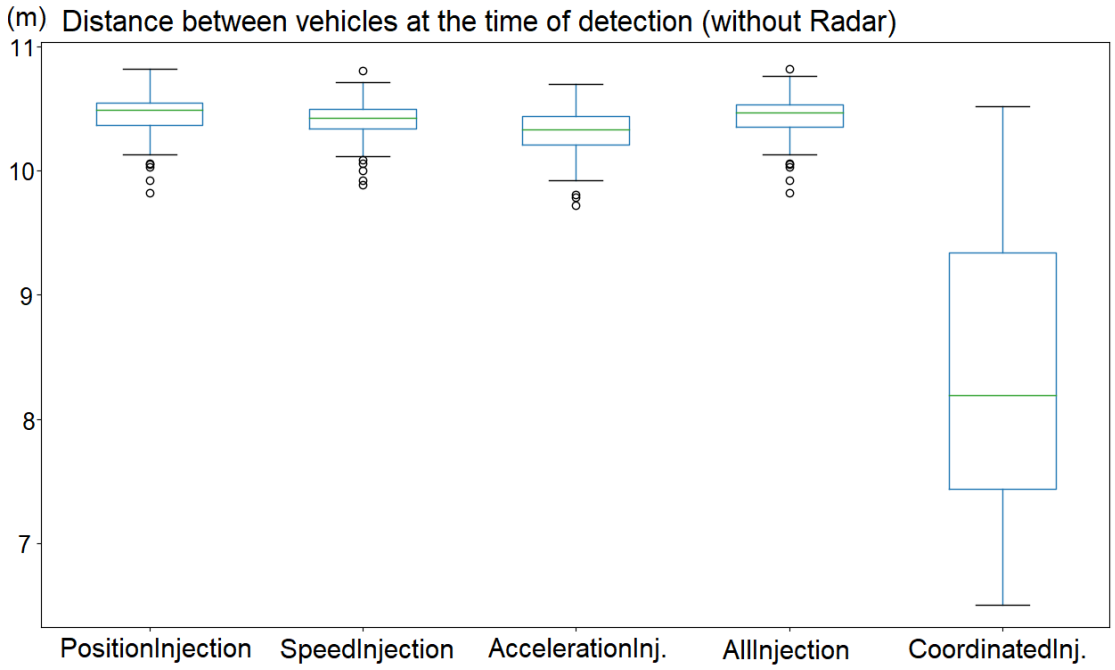

Figura 6.7: Distanza tra due veicoli nel momento della detection (senza radar).

di un plotone di 8 veicoli distanziati tra di loro di 10 metri; distanza che sia il CACC che l'ACC in condizioni normali si impegnano a mantenere.

Il grafico appena descritto assume molta importanza, perché mette in risalto un aspetto fondamentale, ovvero, fa emergere come nonostante la detection sia avvenuta dopo qualche secondo il divario tra i veicoli del plotone non ne ha ancora risentito, quindi si ha ancora molto margine di manovra per poter gestire al meglio diverse strategie di recupero ed evitare così il verificarsi di situazioni spiacevoli.

Capitolo 7

Conclusioni e sviluppi futuri

Il controllo cooperativo dei veicoli è una delle applicazioni più promettenti nell'ambito dei sistemi di trasporto intelligenti (ITS) e la cooperazione è stata ampiamente riconosciuta come una caratteristica chiave per trarne ulteriori vantaggi. Tuttavia, gli algoritmi CACC possono essere facilmente ingannati da vari attacchi informatici. A tal proposito, è stata condotta un'indagine testando diversi approcci in grado di sfruttare altrettanti algoritmi di machine learning, quest'ultimi utilizzati per determinare l'influenza di un attacco sul comportamento dei veicoli. Questi test hanno interessato 1000 simulazioni distribuite per ogni tipo di attacco considerato, da quelli più semplici da rilevare, a quelli studiati appositamente per mettere in difficoltà l'intero sistema. I risultati ottenuti da questi test hanno dimostrato sia l'esistenza di un problema di sicurezza legato al meccanismo di funzionamento degli algoritmi cooperativi, ma hanno anche stabilito come questi attacchi possono essere rilevati per tempo, evitando situazioni critiche, come incidenti o collisioni.

Il presente studio integra la letteratura già esistente riguardante le nuove tecnologie di detection e le conseguenze che queste avranno sui futuri sistemi autonomi. Studi precedenti avevano già fornito delle soluzioni efficaci, ma Sulla base di questo studio, è possibile estendere queste tecniche approcciandosi verso modelli di rilevamento più moderni.

Tuttavia, è importante tener presente che questa ricerca si è concentrata esclusivamente su scenari autostradali, quindi monodirezionali, dove non sono possibili variazioni di corsie, oppure ingresso ed uscita da parte di un veicolo all'interno di un plotone. L'ultima soluzione proposta sfruttava una delle più moderne tecnologie di addestramento, le reti neurali ricorrenti, queste hanno permesso di ottenere gli obbiettivi prefissati ma con un costo computazionale ben più alto rispetto a quello richiesto dall'approccio statico. Quest'ultimo infatti, una volta fissati i valori di soglia è già disponibile per la detection, mentre utilizzando approcci di machine learning questo non è possibile, considerando il tempo necessario per addestrare il modello. Ciò nonostante, terminata la prima fase di addestramento, realizzata

rigorosamente offline, il tempo richiesto per effettuare la detection può essere paragonato ad un lookup di una tabella, quindi è praticamente istantanea, facilmente applicabile in tempo reale.

Gli algoritmi presentati in questo elaborato rappresentano solo una parte delle possibili strategie di rilevamento, una raccomandazione per ulteriori sviluppi futuri potrebbe essere quella di realizzare uno studio simile utilizzando diversi scenari di attacco e tecniche più moderne, in grado, chissà, di ridurre notevolmente il tempo di detection e rendere questi sistemi applicabili nel mondo reale senza correre alcun rischio. Ulteriori aggiornamenti possibili sono, l'utilizzo di una strategia di fallback più sofisticata in grado di coinvolgere anche in questa fase algoritmi di machine learning, così come sostituire le stime effettuate da filtro di Kalman con sistemi predittivi. Entrambi gli sviluppi proposti attualmente richiedono un'infrastruttura di calcolo molto complessa e costosa difficilmente integrabile a bordo veicolo, ma in un futuro non molto lontano questo sarà sicuramente possibile.

Bibliografia

[1] A. Vahidi e A. Eskandarian. «Research advances in intelligent collision avoidance and adaptive cruise control». In: *IEEE Transactions on Intelligent Transportation Systems* 4.3 (2003), pp. 143–153. DOI: 10.1109/TITS.2003.821292.

[2] Christoph Sommer e Falko Dressler. *Vehicular Networking*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107110649.

[3] M. Iorio et al. «Detecting Injection Attacks on Cooperative Adaptive Cruise Control». In: *2019 IEEE Vehicular Networking Conference (VNC)*. 2019, pp. 1–8. DOI: 10.1109/VNC48660.2019.9062798.

[4] Steven Shladover. «Automated vehicles for highway operations (automated highway systems)». In: *Proceedings of The Institution of Mechanical Engineers Part I-journal of Systems and Control Engineering - PROC INST MECH ENG I-J SYST C* 219 (feb. 2005), pp. 53–75. DOI: 10.1243/095440705X9407.

[5] Marc Green. «How Long Does It Take to Stop? Methodological Analysis of Driver Perception-Brake Times». In: *Transportation Human Factors* 2 (set. 2000), pp. 195–216. DOI: 10.1207/STHF0203_1.

[6] Pravin Varaiya. «Smart Cars on Smart Roads: Problems of Control». In: *IEEE Transactions on Automatic Control* 38 (feb. 1993), pp. 195–207.

[7] A. Eskandarian. *Handbook of Intelligent Vehicles*. Handbook of Intelligent Vehicles. Springer London, 2012. ISBN: 9780857290847. URL: https://books.google.it/books?id=01-wAQAACAAJ.

[8] J. Piao e M. McDonald. «Advanced Driver Assistance Systems from Autonomous to Cooperative Approach». In: *Transport Reviews* 28.5 (2008), pp. 659–684. DOI: 10.1080/01441640801987825.

[9] Daniel Sperling e Deborah Gordon. *Two Billion Cars: Driving Toward Sustainability*. Gen. 2009.

[10] Steven Shladover, D. Su e Xiao-Yun Lu. «Cooperative Adaptive Cruise Control Impact on freeway traffic flow». In: *Transportation Research Record Journal of the Transportation Research Board* 2423 (gen. 2012).

[11] R. Bishop. *Intelligent Vehicle Technology and Trends*. Artech House ITS library. Artech House, 2005. ISBN: 9781580539111. URL: https://books.google.it/books?id=%5C_t0mAQAAMAAJ.

[12] G. J. L. Naus et al. «String-Stable CACC Design and Experimental Validation: A Frequency-Domain Approach». In: *IEEE Transactions on Vehicular Technology* 59.9 (2010), pp. 4268–4279. DOI: 10.1109/TVT.2010.2076320.

[13] P. Krzesinski. «Decentralized CACC controllers for platoons of heterogeneous vehicles with uncertain dynamics». In: 2017. URL: http://resolver.tudelft.nl/uuid:13a80af8-14cc-4a79-894e-a50d40e9870b.

[14] Vadym Hapanchak et al. «Simulation and Testing of a Platooning Cooperative Longitudinal Controller: Second EAI International Conference, INTSYS 2018, Guimarães, Portugal, November 21–23, 2018, Proceedings». In: gen. 2019, pp. 66–80. ISBN: 978-3-030-14756-3. DOI: 10.1007/978-3-030-14757-0_6.

[15] «IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments». In: *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, and IEEE Std 802.11w-2009)* (2010), pp. 1–51. DOI: 10.1109/IEEESTD.2010.5514475.

[16] Z. Mahmood. *Connected Vehicles in the Internet of Things: Concepts, Technologies and Frameworks for the IoV*. Springer International Publishing, 2020. ISBN: 9783030361679. URL: https://books.google.it/books?id=Yi3KDwAAQBAJ.

[17] Marc Bechler, Jochen Schiller e Lars Wolf. «In-car communication using wireless technology». In: *8th World Congress on Intelligent Transport Systems* (gen. 2001).

[18] Gaurang Naik, Biplav Choudhury e Jung-Min Park. «IEEE 802.11bd & 5G NR V2X: Evolution of Radio Access Technologies for V2X Communications». In: *CoRR* abs/1903.08391 (2019). arXiv: 1903.08391. URL: http://arxiv.org/abs/1903.08391.

[19] Muawia Abdelmagid Elsadig e Yahia A Fadlalla. «VANETs security issues and challenges: A survey». In: *Indian Journal of Science and Technology* 9.28 (2016), pp. 1–8.

[20] Sudeep Tanwar et al. «A systematic review on security issues in vehicular ad hoc network». In: *Security and Privacy* 1.5 (2018).

[21] Abdeldime MS Abdelgader et al. «Security challenges and trends in vehicular communications». In: *2017 IEEE Conference on Systems, Process and Control (ICSPC)*. IEEE. 2017, pp. 105–110.

[22] Manar Abu Talib et al. «Systematic literature review on Internet-of-Vehicles communication security». In: *International Journal of Distributed Sensor Networks* 14.12 (2018), p. 1550147718815054.

[23] Maria Azees, Pandi Vijayakumar e Lazarus Jegatha Deborah. «Comprehensive survey on security services in vehicular ad-hoc networks». In: *IET Intelligent Transport Systems* 10.6 (2016), pp. 379–388.

[24] Ankit Kumar e Madhavi Sinha. «Overview on vehicular ad hoc network and its security issues». In: *2014 International conference on computing for sustainable global development (INDIACom)*. IEEE. 2014, pp. 792–797.

[25] Huu Phuoc Dai Nguyen e Rajnai Zoltán. «The current security challenges of vehicle communication in the future transportation system». In: *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*. IEEE. 2018, pp. 000161–000166.

[26] Rashmi Mishra, Akhilesh Singh e Rakesh Kumar. «VANET security: Issues, challenges and solutions». In: *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE. 2016, pp. 1050–1055.

[27] Michael Feiri, Jonathan Petit e Frank Kargl. «The case for announcing pseudonym changes». In: *Ulmer Informatik-Berichte* (2015), p. 31.

[28] . *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Giu. 2018. DOI: https://doi.org/10.4271/J3016_201806. URL: https://doi.org/10.4271/J3016_201806.

[29] Greg Welch. «An Introduction to the Kalman Filter, Univ. of North Carolina». In: *http://www. cs. unc. edu/˜ welch/media/pdf/kalman_intro. pdf* (2006).

[30] O. Calzone. *Machine Learning: Introduzione All'apprendimento Automatico*. Independently Published, 2018. ISBN: 9781726646048. URL: https://books.google.it/books?id=AIvVwQEACAAJ.

[31] Lorenzo Govoni. *Matrice di confuzione*. 2020. URL: https://lorenzogovoni.com/matrice-di-confusione/.

[32] Lorenzo Govoni. *5 Tecniche di cross-validation*. 2020. URL: https://lorenzogovoni.com/5-tecniche-di-cross-validation/.

[33] Ivan Vasilev et al. *Python Deep Learning: Exploring deep learning techniques and neural network architectures with Pytorch, Keras, and TensorFlow*. Packt Publishing Ltd, 2019.

[34] Sepp Hochreiter e Jürgen Schmidhuber. «Long Short-term Memory». In: *Neural computation* 9 (dic. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[35] Christopher Olah. *Understanding LSTM Networks*. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[36] Kyunghyun Cho et al. «Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation». In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[37] Junyoung Chung et al. «Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling». In: *CoRR* abs/1412.3555 (2014). arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555.

[38] Andrew Eilbert, Ian Berg, Scott B Smith et al. *Meta-Analysis of Adaptive Cruise Control Applications: Operational and Environmental Benefits*. Rapp. tecn. United States. Department of Transportation. Intelligent Transportation, 2019.

[39] M. Amoozadeh et al. «Security vulnerabilities of connected vehicle streams and their impact on cooperative driving». In: *IEEE Communications Magazine* 53.6 (giu. 2015), pp. 126–132. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7120028.

[40] Mani Amoozadeh et al. «VENTOS: Vehicular Network Open Simulator with Hardware-in-the-Loop Support». In: *Procedia Computer Science* 151 (2019), pp. 61–68. ISSN: 1877-0509. DOI: 10.1016/j.procs.2019.04.012.

[41] R. van der Heijden, T. Lukaseder e F. Kargl. «Analyzing attacks on cooperative adaptive cruise control (CACC)». In: *Proc. IEEE Vehicular Networking Conference (VNC)*. Nov. 2017, pp. 45–52.

[42] Rens W van der Heijden, Thomas Lukaseder e Frank Kargl. «Veremi: A dataset for comparable evaluation of misbehavior detection in vanets». In: *International Conference on Security and Privacy in Communication Systems*. Springer. 2018, pp. 318–337.

[43] Joseph Kamel et al. «VeReMi Extension: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs». In: *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE. 2020, pp. 1–6.

[44] Mingshun Sun, Ming Li e Ryan Gerdes. «A data trust framework for VANETs enabling false data detection and secure vehicle tracking». In: *2017 IEEE Conference on Communications and Network Security (CNS)*. IEEE. 2017, pp. 1–9.

[45] Pengyuan Lu et al. «Attack-resilient sensor fusion for cooperative adaptive cruise control». In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 3955–3960.

[46] Van-Linh Nguyen, Po-Ching Lin e Ren-Hung Hwang. «Enhancing misbehavior detection in 5g vehicle-to-vehicle communications». In: *IEEE Transactions on Vehicular Technology* 69.9 (2020), pp. 9417–9430.

[47] Steven So, Prinkle Sharma e Jonathan Petit. «Integrating plausibility checks and machine learning for misbehavior detection in VANET». In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2018, pp. 564–571.

[48] Joseph Kamel et al. «Simulation Framework for Misbehavior Detection in Vehicular Networks». In: *IEEE Transactions on Vehicular Technology* (2020).

[49] Huimin Li, Marina Krček e Guilherme Perin. «A Comparison of Weight Initializers in Deep Learning-Based Side-Channel Analysis». In: *International Conference on Applied Cryptography and Network Security*. Springer. 2020, pp. 126–143.

[50] Michele Segata. «Safe and efficient communication protocols for platooning control». Tesi di dott. University of Trento, 2016.

[51] Michele Segata et al. «Plexe: A platooning extension for Veins». In: vol. 2015. Dic. 2014. DOI: 10.1109/VNC.2014.7013309.

[52] Vadym Hapanchak et al. «Simulation and testing of a platooning cooperative longitudinal controller». In: *First International Conference on Intelligent Transport Systems*. Springer. 2018, pp. 66–80.

[53] Junyoung Chung et al. «Empirical evaluation of gated recurrent neural networks on sequence modeling». In: *arXiv preprint arXiv:1412.3555* (2014).

[54] Wenpeng Yin et al. «Comparative study of cnn and rnn for natural language processing». In: *arXiv preprint arXiv:1702.01923* (2017).

[55] Sanidhya Mangal, Poorva Joshi e Rahul Modak. «Lstm vs. gru vs. bidirectional rnn for script generation». In: *arXiv preprint arXiv:1908.04332* (2019).