



Università di Genova

DIPARTIMENTO DI INGEGNERIA NAVALE, ELETTRICA,
ELETTRONICA E DELLE TELECOMUNICAZIONI

Tesi Magistrale in Ingegneria Elettronica

Ottobre 2026

Ottimizzazione di un framework di simulazione orientato al deployment per la predizione dei cambi di corsia mediante tecniche di deep learning

*Optimizing a Deployment-Oriented Simulation Framework for Deep
Learning-based Lane Change Prediction*

Relatori:

Prof. Francesco Bellotti, Prof. Riccardo Berta

Correlatori:

Prof. Luca Lazzaroni, Dott. Matteo Fresta

Candidati:

Matteo Ghiglia, Dario Giardini

Sommario

La predizione del cambio di corsia è una funzione chiave per i sistemi avanzati di assistenza alla guida: riconoscere in anticipo l'intenzione del conducente consente interventi di sicurezza tempestivi e un supporto più fluido alla manovra. Questa tesi parte da un framework basato sul simulatore CARLA, che comprende un dataset di manovre eseguite da 50 conducenti su un'autostrada simulata di 60 km. Il framework stima il tempo al cambio di corsia (TTLC) con un unico regressore e ne valuta anche l'implementabilità su microcontrollori.

Il lavoro interviene sulla pipeline del framework in tre direzioni. In primo luogo, introduce una procedura di filtraggio basata su vincoli di fattibilità cinematica (velocità di imbardata, accelerazione laterale, coerenza tra le due grandezze e *jerk* laterale e longitudinale), che elimina le finestre fisicamente non plausibili dovute ad artefatti o incidenti della simulazione: vengono rimosse circa il 17% delle finestre di cambio di corsia e il 33% di quelle di marcia in corsia. In secondo luogo, scompone il problema in una classificazione a tre classi (marcia in corsia, cambio a sinistra, cambio a destra) e in due regressori del TTLC specializzati per direzione, che operano nell'intervallo 0–4 s. Infine, confronta quattro architetture per serie temporali (CNN1D, Transformer, BI-GRU e CNN1D-LSTM) su entrambi i compiti ed esegue un'analisi accurata degli errori.

Tutti i classificatori raggiungono un'accuratezza di circa il 92%. Gli errori si concentrano nelle fasi iniziali della manovra, quando il cambio di corsia è ancora lontano nel tempo. Escludendo i campioni con TTLC superiore a 3.1 s, l'accuratezza sui restanti campioni sale al 95%. Nella regressione il miglior modello ottiene un RMSE di 0.423 s per i cambi a destra e di 0.438 s per quelli a sinistra, entrambi inferiori a quello di 0.510 s del miglior modello del framework di partenza. Sui campioni di marcia in corsia erroneamente classificati come cambio di corsia, inoltre, i regressori stimano TTLC elevati (in media 3.2–3.4 s): questa informazione può essere sfruttata per limitare i falsi allarmi.

Indice

Sommario	i
Elenco degli acronimi	vii
1 Introduzione	1
1.1 Contesto e motivazioni	1
1.2 Definizione del problema	4
1.3 Obiettivi e contributi	7
1.4 Struttura della tesi	8
2 Stato dell'arte	9
2.1 Riconoscimento dell'intenzione di cambio corsia	9
2.1.1 Approcci di classificazione	10
2.1.2 Regressione del tempo al cambio corsia	11
2.2 Dataset per lo studio del cambio corsia	12
2.2.1 Dataset reali	12
2.2.2 Dataset sintetici e basati su simulazione	13
2.3 Modelli di deep learning per serie temporali	15
2.3.1 Reti convoluzionali monodimensionali	15
	iii

2.3.2	Reti ricorrenti	16
2.3.3	Architetture ibride	18
2.3.4	Transformer	18
2.4	Pulizia dei dati e rilevamento di outlier nelle traiettorie	19
2.5	Il framework di partenza	20
2.5.1	Descrizione	20
2.5.2	Risultati principali	21
2.5.3	Limiti e spunti di miglioramento	21
3	Architettura del sistema	23
3.1	Panoramica del framework	23
3.2	Ambiente di simulazione	24
3.2.1	Mappa autostradale in CARLA	24
3.2.2	Modellazione del traffico	25
3.2.3	Modellazione del veicolo e registrazione dei dati	25
3.3	Descrizione del dataset	26
3.4	Pipeline di predizione proposta	28
3.4.1	Filtro di fattibilità cinematica	28
3.4.2	Classificatore della manovra	29
3.4.3	Regressori del TTLC per direzione	29
4	Metodologia	30
4.1	Filtraggio delle traiettorie	30
4.1.1	Vincoli cinematici	31
4.1.2	Scelta delle soglie	32
4.2	Preparazione dei dati	34

4.2.1	Creazione delle finestre ed etichettatura	34
4.2.2	Suddivisione e bilanciamento del dataset	35
4.3	Architetture dei modelli	36
4.3.1	CNN1D	37
4.3.2	Transformer	38
4.3.3	BI-GRU	39
4.3.4	CNN1D-LSTM	40
4.4	Procedura di addestramento	41
4.5	Metriche di valutazione	42
4.5.1	Metriche di classificazione	42
4.5.2	Metriche di regressione	43
5	Risultati sperimentali	44
5.1	Effetto del filtraggio delle traiettorie	44
5.2	Classificazione	50
5.2.1	Classificazione della manovra	50
5.2.2	Caratteristiche di deployment	51
5.2.3	Configurazione selezionata	53
5.2.4	Confronto con lo stato dell'arte	53
5.3	Regressione del TTLC	54
5.3.1	Cambio corsia verso sinistra	54
5.3.2	Caratteristiche di deployment	55
5.3.3	Configurazione selezionata	56
5.3.4	Cambio corsia verso destra	56
5.3.5	Caratteristiche di deployment	57

5.3.6	Configurazione selezionata	58
5.3.7	Confronto con lo stato dell'arte	59
5.4	Analisi degli errori	59
5.4.1	Errori di regressione	59
5.4.1.1	Distribuzione dell'errore	59
5.4.1.2	Analisi qualitativa	64
5.4.2	Errori di classificazione	67
5.4.2.1	Campioni di marcia in corsia classificati come cam- bio corsia	67
5.4.2.2	Cambi di corsia classificati come marcia in corsia	69
5.5	Confronto con il framework di partenza	73
5.6	Discussione	75
6	Conclusioni	77
6.1	Sintesi dei contributi	77
6.2	Limiti	79
6.3	Sviluppi futuri	80
	Bibliografia	82

Elenco degli acronimi

ADAS	Advanced Driver Assistance Systems	1
ADF	Automated Driving Functions	1
BACC	Balanced Accuracy	53
BI-GRU	Bidirectional Gated Recurrent Unit	56
CNN1D	1D Convolutional Neural Network	57
DL	Deep Learning	20
FLW	Follow (marcia in corsia)	75
FP32	32-bit Floating Point	51
GRU	Gated Recurrent Unit	39
LCL	Lane Change Left	79
LCR	Lane Change Right	79
LSTM	Long Short-Term Memory	59
MAE	Mean Absolute Error	73
MSE	Mean Squared Error	56
RMSE	Root Mean Squared Error	78

TTLC Time To Lane Change 80

TTLCL Time To Lane Change Left 34

TTLCR Time To Lane Change Right 34

Capitolo 1

Introduzione

1.1 Contesto e motivazioni

Negli ultimi anni i sistemi avanzati di assistenza alla guida (ADAS) e le funzioni di guida automatizzata (ADF) hanno assunto un ruolo centrale nell'evoluzione del settore automobilistico. Il passaggio verso i livelli più elevati di guida autonoma richiede che il veicolo sia in grado non solo di percepire l'ambiente circostante, ma anche di interpretare e anticipare il comportamento del conducente e degli altri utenti della strada. Le soluzioni più promettenti in questo ambito si basano su tecniche di deep learning (DL), la cui efficacia dipende in larga misura dalla disponibilità di dataset ampi, annotati e rappresentativi. La raccolta di tali dati, tuttavia, comporta costi elevati, tempi lunghi e infrastrutture complesse [1].

Tra le manovre di guida, il cambio di corsia è al tempo stesso una delle più frequenti e più critiche per la sicurezza, poiché richiede al conducente di valutare contemporaneamente velocità, distanze e intenzioni dei veicoli circostanti [2]. Le statistiche sugli incidenti ne confermano la rilevanza: i cambi di corsia sono coinvolti nel 12.6% degli incidenti nei Paesi Bassi [3], nel 9.8% degli incidenti mortali

in Canada [4] e nel 4.9% degli incidenti in Cina [2]. Dal punto di vista della guida autonoma, la gestione di questa manovra richiede una stretta integrazione tra percezione, decisione e controllo [5], includendo pianificazione della traiettoria e algoritmi di controllo adattivo [6, 7].

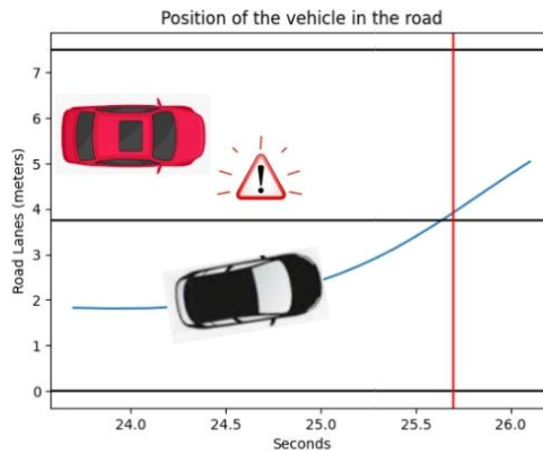


Figura 1.1: Esempio di situazione critica di cambio corsia

Un elemento chiave è il riconoscimento dell'intenzione del conducente, ossia la capacità di prevedere, a partire dall'analisi del comportamento di guida, che una manovra di cambio corsia sta per essere eseguita [7, 8, 9]. Riconoscere l'intenzione con qualche secondo di anticipo consente di attivare interventi di sicurezza proattivi e di supportare la manovra in modo più fluido. In letteratura il problema viene spesso affrontato come una classificazione binaria (cambio corsia imminente oppure no) [8, 9]. Un'alternativa più informativa consiste nello stimare direttamente il tempo mancante al cambio di corsia (TTLC) [10, 11]: sapere che la manovra avverrà tra 3 s oppure tra 0.5 s porta infatti a decisioni molto diverse da parte

del sistema di assistenza.

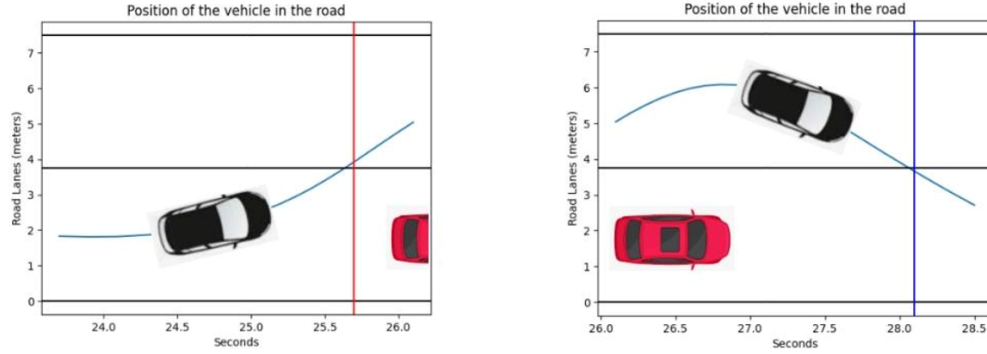


Figura 1.2: Esempio di dinamica di cambio corsia

Lo sviluppo di modelli di questo tipo richiede dati che riproducano comportamenti di guida realistici. Esistono diversi dataset reali disponibili pubblicamente, come NGSIM [12], HDD [13], Brain4Cars [14] e highD [15], che tuttavia presentano limiti significativi per lo sviluppo di funzioni di bordo: molti sono ricavati da riprese aeree su tratti autostradali brevi e rettilinei e non contengono i segnali interni del veicolo, come l'azione sui pedali o l'attivazione degli indicatori di direzione. I simulatori di guida rappresentano un supporto efficace alla sperimentazione su strada [1, 16, 17]: consentono di riprodurre scenari ad alta fedeltà, di variare in modo controllato le condizioni dell'esperimento e di raccogliere dati da conducenti reali in completa sicurezza. In questo contesto si colloca il simulatore open-source CARLA [18], ampiamente utilizzato nella ricerca sulla guida autonoma.

1.2 Definizione del problema

Il punto di partenza di questa tesi è il framework di simulazione proposto da Forneris et al. [19]. Il framework, basato su CARLA, comprende una mappa autostradale ad anello di 60 km con due corsie, un dataset di manovre eseguite in simulatore da 50 conducenti, con oltre 3400 cambi di corsia annotati [20, 21] e una pipeline per la preparazione dei dati e l'addestramento di modelli di DL. Il problema è formulato come una regressione del TTLC nell'intervallo 0–4 s, in cui le finestre prive di un cambio di corsia imminente vengono etichettate con il valore convenzionale di 4.1 s. Il miglior modello, un Transformer, ottiene una radice dell'errore quadratico medio (RMSE) di 0.510 s.

La struttura di questo framework presenta margini di miglioramento, che costituiscono il punto di partenza di questo lavoro:

- **Qualità dei dati simulati:** le traiettorie generate in simulazione risentono dei limiti del simulatore, come la calibrazione del modello dinamico del veicolo, la sensibilità dei comandi e la gestione delle collisioni. Ne derivano traiettorie con dinamiche fisicamente non plausibili, ad esempio accelerazioni laterali o variazioni di imbardata incompatibili con un veicolo reale, che vengono comunque utilizzati per l'addestramento dei modelli.
- **Un unico modello per due compiti:** il regressore deve al tempo stesso riconoscere se un cambio di corsia sta per avvenire e stimare con precisione quando avverrà. Gli autori osservano che il modello tende a sovrastimare il TTLC nell'intervallo 1.2–3 s producendo una stima poco informativa che si avvicina al valore medio dell'intervallo. Nella fascia 3–4 s il modello riesce

a rilevare l'intenzione, ma non a stimarne con precisione i tempi. Gli stessi autori suggeriscono, come sviluppo futuro, di affiancare al regressore un classificatore dedicato al riconoscimento dell'intenzione [19].

- **Nessuna distinzione tra le direzioni della manovra:** un unico modello tratta allo stesso modo i cambi di corsia verso sinistra e verso destra. Nello scenario considerato, le due manovre hanno natura diversa: il cambio verso sinistra corrisponde tipicamente all'inizio di un sorpasso, quello verso destra al rientro in corsia al termine della manovra.

Più formalmente, i dati vengono acquisiti con frequenza di campionamento $f_s = 10$ Hz ($\Delta t = 0.1$ s) e suddivisi in finestre temporali di durata $T_w = 5$ s. Sia quindi:

$$\mathbf{X}_t = [\mathbf{x}_{t-N+1}, \dots, \mathbf{x}_t] \in \mathbb{R}^{N \times F}, \quad N = T_w * f_s = 50, \quad F = 31, \quad (1.1)$$

una finestra di N campioni consecutivi, ciascuno composto da F grandezze misurate sul veicolo ego relative alla sua dinamica, alla geometria della strada e ai veicoli circostanti.

$$X_t \in \mathbb{R}^{N \times F}$$

t	x_1	x_2	x_3	$\dots$	x_F
$t - N + 1$	0.12	-0.03	0.07	$\dots$	0.05
$t - N + 2$	0.11	-0.01	0.08	$\dots$	0.12
$t - N + 3$	0.09	0.02	-0.04	$\dots$	0.07
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
t	0.08	0.01	-0.02	$\dots$	0.11

$F = 31$
feature

$N = 50$
time step

Figura 1.3: Esempio di finestra temporale di 5 s

A ogni finestra viene associata un'etichetta, che indica la classe della manovra e il valore del TTLC.

In questa tesi il problema viene scomposto in due sottoproblemi:

1. **Classificazione:** stimare la classe della manovra

$$\hat{c}_t = f_{\theta}(\mathbf{X}_t), \quad \hat{c}_t \in \{\text{FLW}, \text{LCL}, \text{LCR}\}, \quad (1.2)$$

dove FLW indica la marcia in corsia, mentre LCL e LCR indicano rispettivamente il cambio corsia verso sinistra e verso destra.

2. **Regressione:** se $\hat{c}_t \neq \text{FLW}$, stimare il TTLC con un regressore dedicato alla direzione della manovra

$$\hat{y}_t = g_{\phi}^{(\hat{c}_t)}(\mathbf{X}_t), \quad \hat{y}_t \in [0, 4] \text{ s}. \quad (1.3)$$

1.3 Obiettivi e contributi

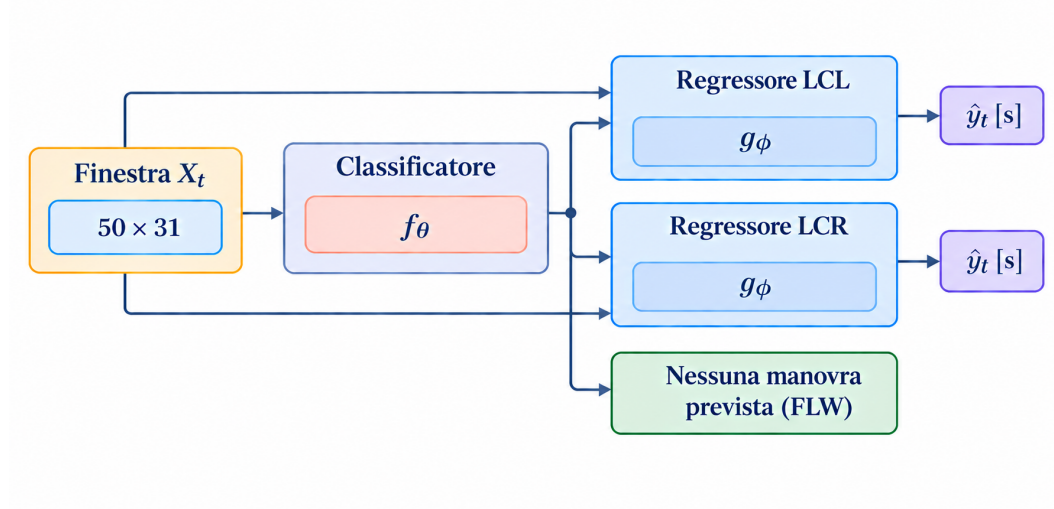


Figura 1.4: Architettura finale proposta

L'obiettivo di questa tesi è migliorare la qualità dei dati e le prestazioni complessive del sistema di riconoscimento dell'intenzione di cambio corsia, usando come riferimento il framework di partenza e intervenendo sia sulla pulizia del dataset sia sulla struttura della pipeline di addestramento. I principali contributi sono i seguenti:

- **Filtraggio delle traiettorie basato su vincoli di fattibilità cinematica:** viene introdotta una procedura che verifica, per ogni finestra, il rispetto di limiti fisici su velocità di imbardata, accelerazione laterale, coerenza tra imbardata e accelerazione laterale, jerk laterale e longitudinale, con soglie differenziate a seconda del tipo di manovra.

- **Scomposizione del problema in classificazione e regressione per direzione:** al singolo regressore viene sostituita una pipeline a due stadi, composta da un classificatore ternario e da due regressori del TTLC specializzati rispettivamente sui cambi di corsia verso sinistra e verso destra.
- **Confronto tra architetture per serie temporali:** vengono addestrati e confrontati quattro modelli (CNN1D, Transformer, BI-GRU e CNN1D-LSTM) su entrambi i compiti.
- **Analisi degli errori:** viene condotta un'analisi esplorativa dei campioni classificati erroneamente per indagare l'eventuale presenza di pattern sistematici, in particolare rispetto alla fase della manovra in cui si verificano.

1.4 Struttura della tesi

Il resto della tesi è organizzato come segue: il capitolo 2 presenta lo stato dell'arte sul riconoscimento dell'intenzione di cambio corsia, sui dataset disponibili e sui modelli per serie temporali, descrivendo in dettaglio il framework di partenza. Il capitolo 3 descrive l'architettura del sistema, dal simulatore alla pipeline di predizione proposta. Il capitolo 4 illustra la metodologia seguita per il filtraggio delle traiettorie, la preparazione del dataset e l'addestramento dei modelli. Il capitolo 5 riporta e discute i risultati sperimentali. Infine, il capitolo 6 riassume il lavoro svolto, ne discute i limiti e delinea i possibili sviluppi futuri.

Capitolo 2

Stato dell'arte

Questo capitolo inquadra il lavoro di tesi nella letteratura esistente. La Sezione 2.1 esamina gli approcci al riconoscimento dell'intenzione di cambio corsia, distinguendo tra formulazioni di classificazione e di regressione. La Sezione 2.2 confronta i dataset disponibili, reali e sintetici. La Sezione 2.3 richiama le architetture di deep learning per serie temporali utilizzate nel seguito, mentre la Sezione 2.4 tratta le tecniche di pulizia dei dati di traiettoria. Infine, la Sezione 2.5 descrive il framework di partenza e i limiti che questa tesi intende affrontare.

2.1 Riconoscimento dell'intenzione di cambio corsia

I lavori sulla predizione del cambio di corsia si distinguono anzitutto per la sorgente dei dati. Gli approcci *basati su infrastruttura* osservano il traffico dall'esterno, tipicamente da telecamere fisse o da droni e ricostruiscono le traiettorie dei veicoli come sequenze di posizioni e velocità. Gli approcci *ego-centrici*, al contrario, utilizzano le grandezze misurate a bordo del veicolo di interesse, ad esempio: accelerazioni, angolo del volante, azione sui pedali e stato degli indicatori di direzio-

ne. Poiché i secondi utilizzano segnali effettivamente disponibili per ogni singolo veicolo, risultano essere realisticamente applicabili a una funzione di bordo.

2.1.1 Approcci di classificazione

L'approccio più diffuso tratta il riconoscimento dell'intenzione come un problema di classificazione, in cui il modello stabilisce se una manovra di cambio corsia sia imminente e, nei casi multiclasse, in quale direzione. La rassegna di Xing et al. [7] raccoglie le principali famiglie di metodi, dai modelli probabilistici alle reti neurali profonde e ne discute limiti e prospettive.

Jeong [8] impiega una rete Long Short-Term Memory bidirezionale per decidere se eseguire un cambio di corsia in scenari autostradali, mostrando che l'informazione contenuta nella sequenza temporale migliora sensibilmente le prestazioni rispetto a un'analisi istantanea. Manzour et al. [9] adottano una rappresentazione a grafo della conoscenza combinata con inferenza bayesiana, con l'obiettivo di rendere la predizione interpretabile; lo stesso gruppo ha successivamente esteso l'approccio agli scenari di quasi-incidente [22]. Liang et al. [23] affrontano invece il problema con reti per il riconoscimento di azioni, trattando la sequenza di immagini come un video da classificare.

Il limite comune a questi lavori è che l'uscita del modello indica *se* la manovra avverrà, ma non *quando*. Per una funzione di assistenza alla guida, questa informazione è insufficiente poichè la strategia di intervento è diversa se il cambio di corsia è previsto tra tre secondi oppure tra mezzo secondo.

2.1.2 Regressione del tempo al cambio corsia

Un filone più recente formula il problema come regressione del TTLC, ossia della differenza tra l'istante corrente e l'istante in cui il veicolo attraverserà la linea di corsia [23]. La predizione diventa così un valore continuo, che porta con sé sia il riconoscimento dell'intenzione sia una stima del suo orizzonte temporale.

Wirthmüller et al. [11] utilizzano reti ricorrenti basate su LSTM per stimare il tempo residuo alla manovra. Mozaffari et al. [10] propongono una rete convoluzionale con meccanismo di attenzione addestrata in modalità multi-task, mentre Yang et al. [24] combinano convoluzioni e ricorrenza per predire congiuntamente la manovra e il rischio di collisione associato. I risultati riportati in questi lavori sono compresi tra 0.629 s e 0.7 s di RMSE, con anticipi fino a circa 0.52 s.

I tre lavori citati utilizzano il dataset highD [15] costituito da traiettorie estratte tramite drone su tratti autostradali brevi e rettilinei e privo di segnali interni del veicolo. Questo approccio introduce limiti significativi in termini di realizzabilità. Nessuno di questi lavori, inoltre, combina un classificatore della manovra con regressori specializzati, che equivale alla struttura proposta nel Capitolo 3.

2.2 Dataset per lo studio del cambio corsia

2.2.1 Dataset reali

Diversi dataset di guida sono disponibili pubblicamente. NGSIM [12] raccoglie traiettorie veicolari rilevate da telecamere installate su edifici lungo alcune arterie statunitensi. È stato a lungo il riferimento per lo studio del comportamento del traffico. highD [15] contiene traiettorie di veicoli su autostrade tedesche riprese da drone, con elevata precisione di posizionamento. HDD [13] e Brain4Cars [14] adottano invece un punto di vista ego-centrico, combinando telecamere interne ed esterne per studiare il comportamento del conducente.

Dal punto di vista dello sviluppo di funzioni di bordo, come già descritto in precedenza, questi dataset presentano due limiti ricorrenti. Il primo è l'assenza dei segnali interni del veicolo: le traiettorie estratte dall'alto descrivono la cinematica osservata, ma non l'azione del conducente sui comandi. Il secondo è la scarsa varietà geometrica dei tratti considerati, spesso brevi e rettilinei. A ciò si aggiunge il fatto che la raccolta di dati reali è costosa, richiede tempi lunghi e non consente di riprodurre a piacimento condizioni rare o pericolose [1].

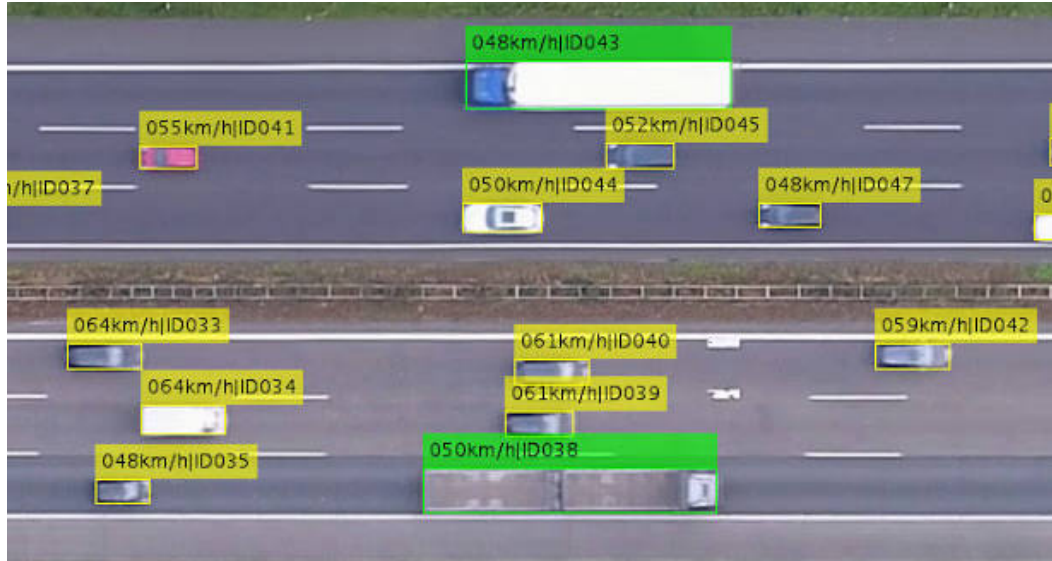


Figura 2.1: Frame del dataset highD

2.2.2 Dataset sintetici e basati su simulazione

La simulazione permette di superare parte di questi limiti, consentendo di generare dati in condizioni controllate e ripetibili [16]. Il simulatore open-source CARLA [18] è oggi uno degli strumenti più utilizzati in questo ambito.

La Tabella 2.1 confronta i principali dataset sintetici che supportano lo studio del cambio di corsia. Si può notare che emergono alcune differenze rilevanti: molti dataset contengono un numero limitato di manovre, non tutti sono distribuiti con licenza aperta e in diversi casi i dati non sono stati raccolti da persone che guidano in simulatore, ma generati replicando manovre reali o simulando interamente il comportamento dei veicoli [25, 26]. SRLCD [27], il lavoro più vicino a quello di riferimento, coinvolge un numero inferiore di partecipanti e registra le sessioni su una mappa a forma di otto, caratterizzata da curve strette che possono limitare

la generalizzazione dei modelli.



Figura 2.2: Frame del dataset CARLA

Dataset	Open source	Segnali veicolo	Cambi corsia	Partecipanti	Ore di guida
CRASH [22]	no	sì	50	n.d.	0.15
Reasonable Crowd [26]	sì	no	<100	n.d.	0.08
CARLA RTS [25]	sì	no	<2300	n.d.	1.5
SRLCD [27]	no	sì	600	30	2.5
Framework di partenza [19]	sì	sì	3405	50	12.5

Tabella 2.1: Confronto tra i principali dataset sintetici per lo studio del cambio di corsia

2.3 Modelli di deep learning per serie temporali

Il problema affrontato in questa tesi consiste nell'elaborare una finestra multivariata $\mathbf{X}_t \in \mathbb{R}^{N \times F}$, definita nell'Equazione (1.1), per produrre un'etichetta di classe o un valore continuo. Si richiamano di seguito le quattro famiglie di architetture confrontate nel Capitolo 5; le configurazioni specifiche adottate sono descritte nella Sezione 4.3.

2.3.1 Reti convoluzionali monodimensionali

Le reti convoluzionali monodimensionali applicano filtri che scorrono lungo l'asse temporale. Indicando con $\mathbf{w}^{(k)} \in \mathbb{R}^{K \times F}$ il k -esimo filtro di ampiezza K e con $b^{(k)}$ il relativo termine di bias, l'uscita in posizione i è

$$z_i^{(k)} = \varphi \left(\sum_{j=0}^{K-1} \sum_{f=1}^F w_{j,f}^{(k)} x_{i+j,f} + b^{(k)} \right), \quad (2.1)$$

dove $\varphi(\cdot)$ è una funzione di attivazione non lineare. Impilando più strati convoluzionali il campo recettivo si allarga progressivamente, consentendo di cogliere andamenti su scale temporali diverse.

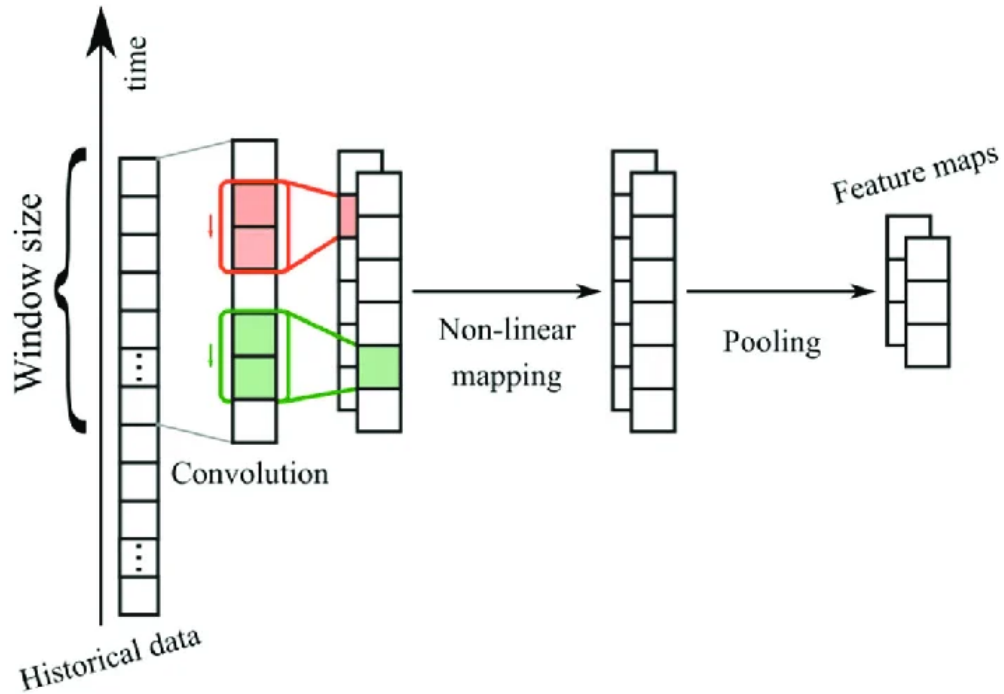


Figura 2.3: Schema CNN1D

Il vantaggio principale di questa famiglia è l'efficienza: il numero di parametri è contenuto e le operazioni parallelizzabili con facilità [28], caratteristiche particolarmente interessanti in prospettiva di un'implementazione a bordo veicolo.

2.3.2 Reti ricorrenti

Le reti ricorrenti elaborano la sequenza un campione alla volta, mantenendo uno stato interno che riassume la storia passata. Le celle LSTM, introdotte da Hochreiter e Schmidhuber [29], regolano il flusso dell'informazione attraverso tre

porte:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \quad (2.2)$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \quad (2.3)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \quad (2.4)$$

rispettivamente di dimenticanza (forget), ingresso (input) e uscita (output), che aggiornano lo stato della cella $\mathbf{c}_t$ e lo stato nascosto $\mathbf{h}_t$. Questa struttura attenua il problema della scomparsa del gradiente (vanishing gradient) tipico delle reti ricorrenti semplici.

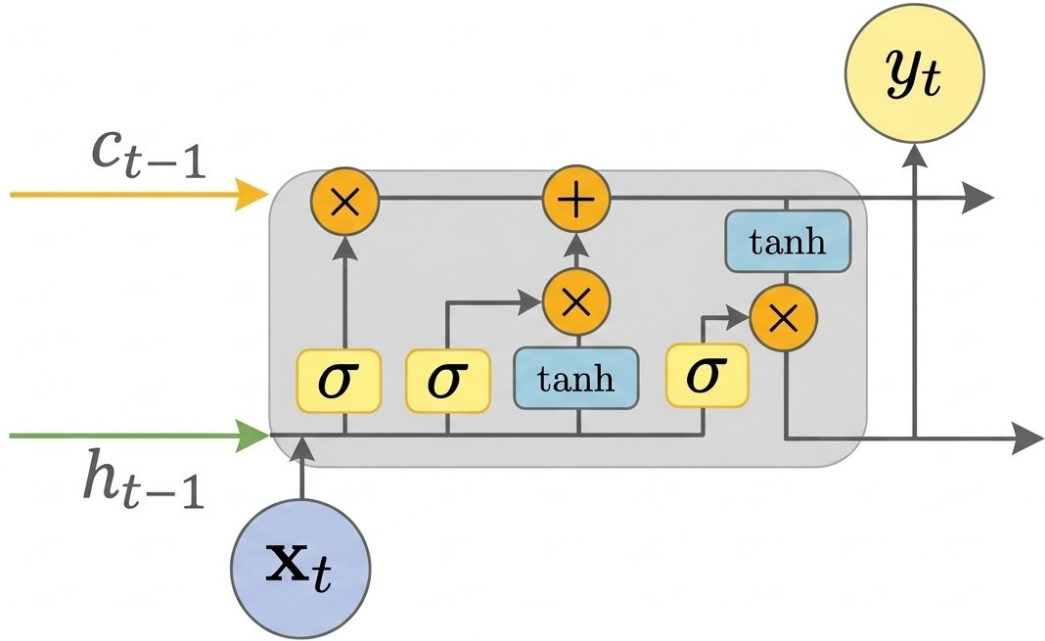


Figura 2.4: Schema dell'architettura LSTM

Le celle GRU [30] perseguono lo stesso obiettivo con due sole porte, di aggior-

namento (update) e di azzeramento (reset), riducendo il numero di parametri a parità di capacità di modellazione. Nelle varianti bidirezionali [31] la sequenza viene elaborata in entrambi i versi e i due stati nascosti vengono concatenati. Poiché in questo lavoro la finestra si conclude nell'istante corrente, la bidirezionalità non introduce alcuna dipendenza da campioni futuri rispetto al momento della predizione: semplicemente, la rete elabora due volte la stessa finestra già acquisita, al prezzo di un maggiore costo computazionale.

2.3.3 Architetture ibride

Le architetture ibride combinano i due approcci precedenti: uno o più strati convoluzionali estraggono caratteristiche locali dalla serie, riducendone al tempo stesso la lunghezza e uno strato ricorrente ne modella l'evoluzione temporale. Soluzioni di questo tipo sono state applicate con successo sia alla predizione del cambio di corsia [24] sia ad altri compiti di analisi del comportamento del conducente [32].

2.3.4 Transformer

I Transformer [33] sostituiscono la ricorrenza con il meccanismo di self-attention, che mette in relazione ogni istante della sequenza con tutti gli altri:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) \mathbf{V}, \quad (2.5)$$

dove $\mathbf{Q}$, $\mathbf{K}$ e $\mathbf{V}$ sono rispettivamente le matrici di query, chiave (key) e valore (value), e d_k la dimensione delle chiavi. Poiché l'operazione è invariante rispetto all'ordine dei campioni, l'informazione temporale viene reintrodotta tramite una codifica posizionale sommata agli ingressi. Il principale punto di forza di

questa famiglia è la capacità di modellare dipendenze a lungo raggio senza propagazione sequenziale dello stato, a fronte di un costo computazionale che cresce quadraticamente con la lunghezza della finestra.

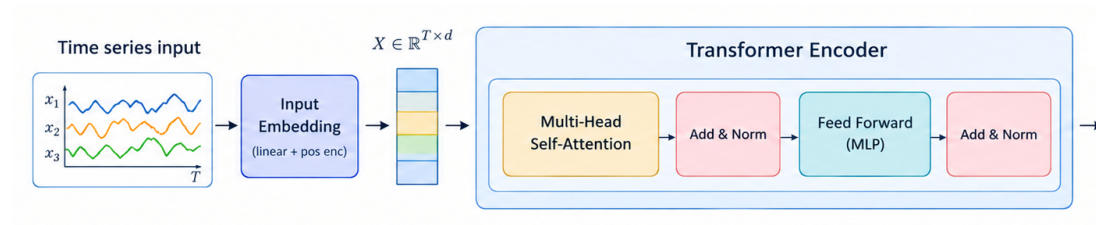


Figura 2.5: Schema Transformer (encoder only)

2.4 Pulizia dei dati e rilevamento di outlier nelle traiettorie

I dati di traiettoria, soprattutto se generati in simulazione, contengono inevitabilmente campioni anomali. Nel caso della simulazione le cause principali sono la calibrazione del modello dinamico del veicolo, la sensibilità dei comandi e la gestione delle collisioni, che possono produrre segmenti con dinamiche non realizzabili da un veicolo reale.

La rassegna sperimentale di Duarte e Sakr [34] distingue due famiglie di metodi. I metodi *statistici* apprendono dai dati una nozione di normalità, ad esempio tramite soglie sulla deviazione standard, tecniche di clustering o modelli di ricostruzione e segnalano come anomali i campioni che se ne discostano. I metodi *euristici*, invece, verificano che ogni campione rispetti un insieme di limiti rigidi derivati dalla fisica del sistema osservato.

In questa tesi si adotta il secondo approccio, per tre motivi. In primo luogo, le soglie hanno un significato fisico diretto e possono essere giustificate a partire dai limiti di aderenza e di dinamica del veicolo, mentre una soglia statistica dipende dalla distribuzione del particolare dataset. In secondo luogo, il criterio è interpretabile: per ogni finestra scartata è possibile sapere quale vincolo è stato violato e in quale istante. Infine, un metodo puramente statistico rischia di eliminare manovre rare ma legittime, escludendo così potenziali campioni preziosi per l'addestramento, poiché aggiungono variabilità.

2.5 Il framework di partenza

2.5.1 Descrizione

Il framework proposto da Forneris et al. [19] è il primo strumento open-source che integra un dataset di manovre di cambio corsia eseguite da conducenti reali in simulatore, comprensivo dei segnali del veicolo ego, con una pipeline di generazione dei campioni pronta per l'addestramento di modelli di DL.

L'ambiente di simulazione è costruito su CARLA e comprende una mappa autostradale ad anello di 60 km con due corsie da 3.75 m, progettata con curve di raggio variabile per evitare che la geometria della strada introduca distorsioni sistematiche nei dati. Il traffico è generato proceduralmente mediante due tipologie di veicoli non controllati dall'utente, che inducono naturalmente situazioni di sorpasso. Il simulatore è stato inoltre esteso con un abitacolo realistico, dotato di specchietti [21].

Il dataset [20] raccoglie le sessioni di 50 partecipanti, per un totale di oltre 3400

manovre annotate. I log sono campionati a 10 Hz e convertiti in serie temporali con 31 grandezze, relative alla dinamica del veicolo ego, alla geometria della strada e ai veicoli circostanti. Il framework include infine un insieme di modelli pre-addestrati e la valutazione della loro deployabilità su microcontrollori, aspetto non trattato in questa tesi.



Figura 2.6: Schema del framework di partenza

2.5.2 Risultati principali

Il problema è formulato come regressione del TTLC nell'intervallo 0–4 s, con le finestre prive di intenzione di cambio corsia etichettate al valore convenzionale di 4.1 s. Cinque modelli sono stati confrontati: un modello a gradient boosting e quattro reti neurali per serie temporali. Il Transformer ottiene i risultati migliori su tutte le metriche di apprendimento, con un RMSE di 0.510 s e un MAE di 0.298 s, mentre le soluzioni più compatte risultano preferibili quando si considerano i vincoli di esecuzione su hardware embedded.

2.5.3 Limiti e spunti di miglioramento

L'analisi dei risultati mette in luce tre aspetti, che motivano i contributi di questa tesi.

Il primo riguarda la **qualità dei dati**. Le finestre utilizzate per l'addestramento non sono sottoposte ad alcun controllo di plausibilità fisica: eventuali artefatti della simulazione, comprese le collisioni, entrano nel dataset al pari delle manovre realistiche.

Il secondo riguarda la **struttura del problema**. Un unico regressore deve sia riconoscere l'intenzione sia stimarne l'orizzonte temporale. Gli autori osservano che il modello tende a sovrastimare il TTLC nell'intervallo 1.2–3.0 s, con la massima variabilità attorno a 2.4 s, e che nell'intervallo 3.0–4.0 s restituisce un valore prossimo alla media riuscendo a rilevare l'intenzione, ma non a collocarla nel tempo. Nelle conclusioni viene esplicitamente suggerito di affiancare al regressore un classificatore dedicato al riconoscimento dell'intenzione, restringendo il compito di regressione a un intervallo temporale più breve [19].

Il terzo aspetto riguarda la **direzione della manovra**. Un solo modello tratta allo stesso modo i cambi di corsia verso sinistra e verso destra, benché nello scenario considerato le due manovre abbiano un significato diverso.

A partire da queste osservazioni, il capitolo seguente descrive l'architettura del sistema e la pipeline di predizione proposta.

Capitolo 3

Architettura del sistema

Questo capitolo descrive il sistema nel suo complesso: l'ambiente di simulazione da cui provengono i dati (Sezione 3.2), la struttura del dataset (Sezione 3.3) e la pipeline di predizione proposta (Sezione 3.4). Le scelte implementative e i dettagli procedurali sono rimandati al Capitolo 4.

3.1 Panoramica del framework

Il sistema si articola in tre fasi. La *preparazione dei dati*, già svolta nel framework di partenza, comprende la raccolta delle sessioni in simulatore, la loro conversione in serie temporali e l'estrazione delle finestre. Il *filtraggio* scarta le traiettorie che violano i vincoli di fattibilità cinematica, così da impedire che i modelli apprendano da dinamiche non realizzabili. La *predizione* costituisce infine il sistema vero e proprio, quello destinato a operare a bordo veicolo: la finestra corrente viene classificata ed eventualmente inoltrata al regressore corrispondente alla direzione stimata.

Le prime due fasi sono eseguite una sola volta e non fanno parte della pipeline, mentre la terza opera sulla singola finestra e, in un'applicazione di bordo, verrebbe eseguita in tempo reale. Il filtro non fa quindi parte del flusso di predizione, in cui il sistema deve trattare tutte le finestre che riceve.

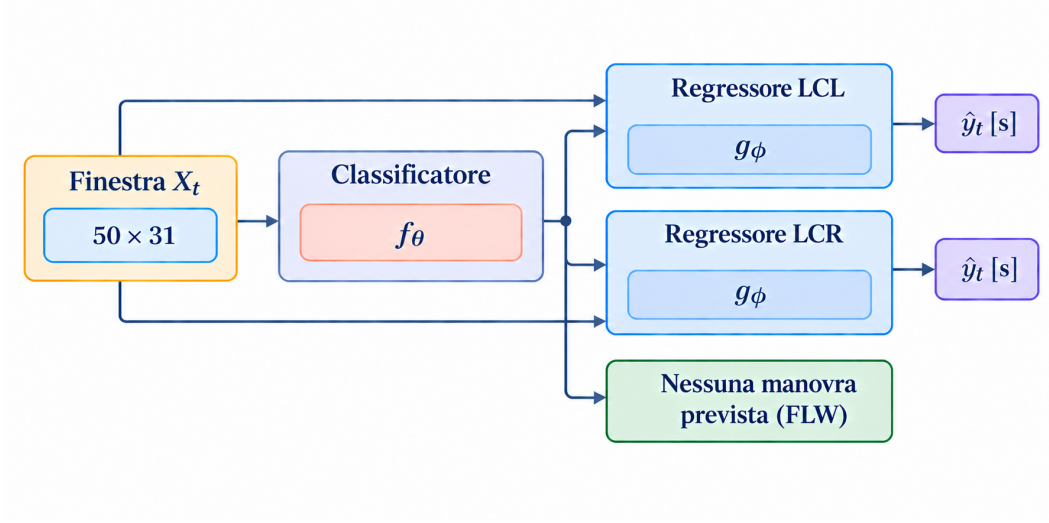


Figura 3.1: Architettura della pipeline

3.2 Ambiente di simulazione

L'ambiente di simulazione è quello sviluppato nel framework di partenza [19, 21] e non è stato modificato in questo lavoro. Se ne richiamano di seguito le caratteristiche rilevanti per interpretare i dati.

3.2.1 Mappa autostradale in CARLA

Il tracciato è un anello autostradale di 60 km a senso unico, con due corsie di larghezza 3.75 m. Il percorso alterna tratti rettilinei e curve con raggi diversi,

bilanciate tra destra e sinistra in modo che la geometria della strada non introduca distorsioni sistematiche nel comportamento registrato.

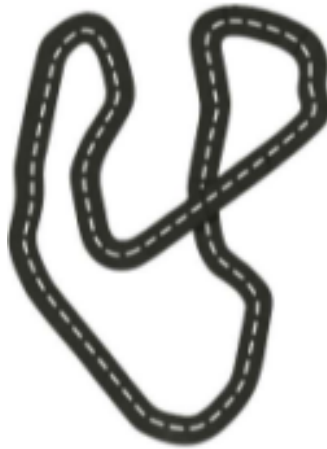


Figura 3.2: Mappa autostradale in CARLA

3.2.2 Modellazione del traffico

Il traffico è composto da veicoli non controllati dall'utente di due tipologie. I veicoli standard percorrono la corsia di destra a velocità costante di 100 km/h. I veicoli pericolosi, caratterizzati da una velocità massima di 180 km/h, vengono introdotti nella corsia di sinistra per creare situazioni di sorpasso più impegnative e rappresentano un elemento di rischio stocastico, ma realistico.

3.2.3 Modellazione del veicolo e registrazione dei dati

Per rendere l'esperienza di guida realistica, il simulatore è stato esteso con gli elementi tipici dell'abitacolo: specchietto retrovisore centrale e laterale sinistro, cruscotto con tachimetro, contagiri, contachilometri e indicatori di direzione. Al

contempo, viene effettuata la registrazione dei log per consentire successivamente la loro conversione in finestre temporali.

3.3 Descrizione del dataset

Il dataset raccoglie le sessioni di guida di 50 partecipanti, ciascuna della durata di circa 30 km, con posizioni di partenza variare in modo sistematico [20].

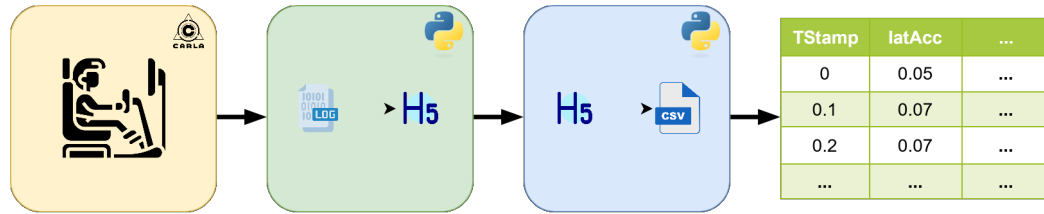


Figura 3.3: Workflow di generazione del dataset

I log prodotti da CARLA sono campionati a 10 Hz e convertiti in file CSV contenenti $F = 31$ grandezze per ciascun istante di campionamento, riconducibili a tre tipologie.

- **Dinamica del veicolo ego:** velocità, accelerazioni longitudinale e laterale, velocità di imbardata, angolo del volante, azione sui pedali di acceleratore e freno, stato degli indicatori di direzione.
- **Geometria della strada:** curvatura del tracciato e distanze dalle linee di delimitazione della corsia.
- **Contesto di traffico:** distanze e velocità relative rispetto ai veicoli standard e pericolosi più vicini.

Colonna	Unità	Descrizione
Riferimento temporale		
fileTime	s	Istante di campionamento all'interno della sessione
Dinamica veicolo ego		
vehicleSpeed	m/s	Velocità longitudinale
longAcceleration	m/s ²	Accelerazione longitudinale
latAcceleration	m/s ²	Accelerazione laterale
yawRate	rad/s	Velocità di imbardata
steeringAngle	rad	Angolo del volante
throttlePedalPos	–	Posizione pedale dell'acceleratore (normalizzata)
brakePedalPos	–	Posizione pedale del freno (normalizzata)
leftDirectionIndicator	–	Stato dell'indicatore di direzione sinistro
RightDirectionIndicator	–	Stato dell'indicatore di direzione destro
Geometria stradale		
egoLaneWidth	m	Larghezza della corsia occupata
rightLaneLine	m	Distanza laterale dalla linea di corsia destra
leftLaneLine	m	Distanza laterale dalla linea di corsia sinistra
yawAngleRightLine	rad	Angolo tra l'asse del veicolo e la linea destra
yawAngleLeftLine	rad	Angolo tra l'asse del veicolo e la linea sinistra
curvatureRightLine	m ⁻¹	Curvatura della linea destra
curvatureLeftLine	m ⁻¹	Curvatura della linea sinistra
curvatureDxRightLine	m ⁻²	Variazione della curvatura della linea destra
curvatureDxLeftLine	m ⁻²	Variazione della curvatura della linea sinistra
Contesto di traffico		
car1LatPosition	m	Posizione laterale del primo veicolo
car1LongPosition	m	Posizione longitudinale del primo veicolo
car1LatVelocity	m/s	Velocità laterale del primo veicolo
car1LongVelocity	m/s	Velocità longitudinale del primo veicolo
car1YawAngle	rad	Angolo di imbardata del primo veicolo
car1YawRate	rad/s	Velocità di imbardata del primo veicolo
car2LatPosition	m	Posizione laterale del secondo veicolo
car2LongPosition	m	Posizione longitudinale del secondo veicolo

Colonna	Unità	Descrizione
car2LatVelocity	m/s	Velocità laterale del secondo veicolo
car2LongVelocity	m/s	Velocità longitudinale del secondo veicolo
car2YawAngle	rad	Angolo di imbardata del secondo veicolo
car2YawRate	rad/s	Velocità di imbardata del secondo veicolo

Tabella 3.1: Grandezze contenute nei file CSV del dataset.

3.4 Pipeline di predizione proposta

La pipeline proposta sostituisce il regressore unico del framework di partenza con tre modelli distinti, preceduti da una fase di validazione automatica dei dati di addestramento.

3.4.1 Filtro di fattibilità cinematica

Il filtro verifica, per ogni finestra, il rispetto di un insieme di vincoli derivati dalla dinamica del veicolo. Una finestra viene scartata se minimo un indicatore supera la soglia corrispondente in almeno un istante. Le soglie sono differenziate in base al tipo di finestra: durante un cambio di corsia sono fisiologicamente ammessi valori più elevati di accelerazione laterale e di jerk, mentre un tratto di marcia in corsia dovrebbe presentare una dinamica più regolare. La formulazione dei vincoli e i valori adottati sono riportati nella Sezione 4.1.

Come anticipato, il filtro appartiene alla fase di preparazione del dataset, quindi il suo effetto è quello di impedire che i modelli apprendano da traiettorie fisicamente non realizzabili.

3.4.2 Classificatore della manovra

Il primo stadio della pipeline è un classificatore a tre classi che, data la finestra $\mathbf{X}_t$, ne stima la classe secondo l'Equazione (1.2). Le classi considerate sono la marcia in corsia (FLW), il cambio di corsia verso sinistra (LCL) e il cambio di corsia verso destra (LCR).

Rispetto alla formulazione originale, in cui l'assenza di intenzione era codificata con il valore convenzionale di 4.1 s all'interno della regressione, il riconoscimento dell'intenzione diventa un compito affidato a un modello dedicato.

3.4.3 Regressori del TTLC per direzione

Il secondo stadio è costituito da due regressori addestrati separatamente sulle manovre verso sinistra e verso destra, che stimano il TTLC nell'intervallo $[0, 4]$ s secondo l'Equazione (1.3). La separazione per direzione è motivata dal fatto che, nello scenario considerato, le due manovre hanno un contesto differente. Tipicamente, il cambio verso sinistra corrisponde all'inizio di un sorpasso e quello verso destra al rientro.

Ciascun regressore opera quindi su una distribuzione di dati più omogenea rispetto al modello unico del framework di partenza, non dovendo farsi carico di distinguere le finestre prive di intenzione, compito già assolto dal classificatore.

Capitolo 4

Metodologia

Questo capitolo descrive la procedura seguita per ottenere i risultati del capitolo 5: il filtraggio delle traiettorie (sezione 4.1), la costruzione dei campioni (sezione 4.2), le architetture confrontate (sezione 4.3), la procedura di addestramento (sezione 4.4) e le metriche adottate (sezione 4.5).

4.1 Filtraggio delle traiettorie

Le traiettorie raccolte in simulazione risentono dei limiti intrinseci del simulatore: la calibrazione del modello dinamico del veicolo, la sensibilità e i tempi di reazione del sistema di guida. Inoltre, si aggiungono eventi anomali come le uscite di strada e le collisioni, che producono segmenti di traiettoria non riconducibili a una manovra di guida, causando artefatti nei dati.

Per garantire che l'addestramento utilizzi soltanto traiettorie fisicamente e cinematicamente plausibili, viene applicata una procedura di filtraggio euristica basata su vincoli di fattibilità, secondo l'approccio discusso nella sezione 2.4. Ogni campione della finestra viene confrontato con un insieme di limiti rigidi derivati dalla

dinamica del veicolo. Se anche solo un vincolo risulta violato in almeno un istante, l'intera finestra viene scartata.

4.1.1 Vincoli cinematici

Per ciascun istante t della finestra vengono valutati cinque vincoli: la velocità di imbardata $\dot{\psi}(t)$, la velocità longitudinale $v(t)$, le accelerazioni longitudinale e laterale $a_x(t)$ e $a_y(t)$. Fissato il tempo di campionamento $\Delta t = 0.1$ s.

Velocità di imbardata

$$|\dot{\psi}(t)| \leq \dot{\psi}_{\max} \quad (4.1)$$

Impone un limite alla dinamica rotazionale del veicolo attorno all'asse verticale, escludendo transizioni angolari troppo brusche tra campioni consecutivi.

Accelerazione laterale

$$|a_y(t)| \leq a_{y,\max} \quad (4.2)$$

Riflette i limiti fisici di aderenza tra veicolo e strada, consentendo di eliminare traiettorie caratterizzate da accelerazioni trasversali non realizzabili.

Coerenza tra imbardata e accelerazione laterale

$$|\dot{\psi}(t) v(t) - a_y(t)| \leq \varepsilon \quad (4.3)$$

Sfrutta l'identità approssimata $a_y \approx \dot{\psi}v$. Un residuo elevato segnala un'incoerenza tra i segnali registrati, tipicamente dovuta a un artefatto della simulazione.

Jerk laterale

$$j_y(t) = \frac{a_y(t) - a_y(t-1)}{\Delta t}, \quad |j_y(t)| \leq j_{y,\max} \quad (4.4)$$

Limita la rapidità di variazione dell'accelerazione laterale, penalizzando oscillazioni incompatibili con una guida realistica o con misure sensoristiche stabili.

Jerk longitudinale

$$j_x(t) = \frac{a_x(t) - a_x(t-1)}{\Delta t}, \quad |j_x(t)| \leq j_{x,\max} \quad (4.5)$$

Impone un vincolo analogo alla variazione dell'accelerazione lungo la direzione di marcia.

4.1.2 Scelta delle soglie

Le soglie sono state fissate in base al tipo di manovra. Durante un cambio di corsia sono ammessi valori più elevati di accelerazione laterale e di jerk, poiché comportano un maggiore spostamento trasversale; un tratto di marcia in corsia, al contrario, dovrebbe presentare una dinamica più regolare. I valori adottati sono riportati nella Tabella 4.1.

Indicatore	LC	FLW
Velocità di imbardata massima [rad/s]	0.3	0.2
Accelerazione laterale massima [m/s ²]	4.0	2.5
Residuo di coerenza [m/s ²]	2.0	1.0
Jerk laterale massimo [m/s ³]	5.0	3.0
Jerk longitudinale massimo [m/s ³]	8.0	5.0

Tabella 4.1: Soglie di fattibilità cinematica adottate per le finestre di cambio corsia (LC) e di marcia in corsia (FLW)

A titolo di riferimento, l'accelerazione laterale di 4.0 m/s² imposta alle finestre di cambio corsia corrisponde a circa 0.41 g ed è in accordo con il limite di preferenza del conducente nominale indicato in [35], pari a 0.4 g. Nelle finestre di marcia in corsia, in cui per definizione non viene eseguita alcuna manovra, l'unica fonte ammissibile di accelerazione laterale è la percorrenza delle curve del tracciato: la soglia è stata quindi fissata a 2.5 m/s², valore sufficiente a non scartare le curve effettivamente presenti nel circuito ma tale da segnalare qualunque dinamica laterale più marcata.

Il residuo di coerenza confronta l'accelerazione laterale misurata con quella attesa dalla relazione cinematica $a_y \approx \dot{\psi} v$, che vale esattamente quando il veicolo si muove nella direzione in cui è orientato [36]. Il residuo misura quindi lo scostamento da tale condizione: è nullo in marcia rettilinea e in curva stazionaria, e cresce quanto più il veicolo ruota su sé stesso rispetto alla propria traiettoria.

4.2 Preparazione dei dati

4.2.1 Creazione delle finestre ed etichettatura

Le serie temporali vengono suddivise in finestre di $N = 50$ campioni consecutivi, corrispondenti a $T_w = 5$ s alla frequenza di campionamento di 10 Hz, secondo la definizione dell'Equazione (1.1). A ciascuna finestra viene assegnata l'etichetta relativa al suo ultimo campione, ossia all'istante corrente dal punto di vista della predizione.

Le finestre vengono estratte con una scansione scorrevole dell'intera sessione, facendo avanzare di un campione alla volta l'istante finale t . Una finestra è accettata come campione di cambio corsia se l'etichetta del suo ultimo campione risulta essere minore o uguale a 4 s, ossia se la manovra si colloca entro l'orizzonte di predizione considerato ed è assegnata alla classe LCL o LCR in base alla colonna di riferimento (TTLCL oppure TTLCR). Una finestra è invece accettata come campione di marcia in corsia solo se tutte le etichette al suo interno corrispondono alla classe FLW.

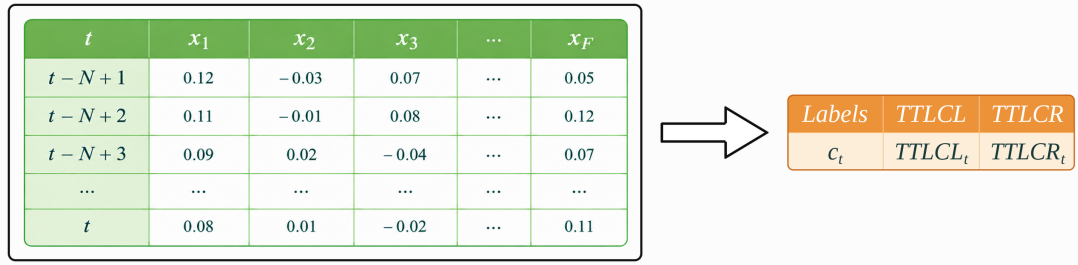


Figura 4.1: Etichettatura della finestra

A titolo di esempio, si consideri una finestra il cui ultimo campione cade all'istante

$t = 120.0$ s e una manovra di cambio corsia verso sinistra che si completa a $t_{LC} = 122.3$ s. La finestra copre l'intervallo $[115.1, 120.0]$ s e le viene assegnata l'etichetta

$$y_t = t_{LC} - t = 122.3 - 120.0 = 2.3 \text{ s}, \quad c_t = \text{LCL}. \quad (4.6)$$

La finestra successiva, che termina a $t = 120.1$ s, riceve l'etichetta $y_t = 2.2$ s: scorrendo di un campione alla volta, il valore decresce di 0.1 s fino ad annullarsi nell'istante in cui il veicolo intercetta la linea di mezzzeria.

4.2.2 Suddivisione e bilanciamento del dataset

La suddivisione del dataset è effettuata per partecipante: i campioni di uno stesso conducente appartengono interamente a uno solo tra gli insiemi di training, validation e test. I 50 partecipanti sono stati ripartiti assegnando 36 conducenti all'insieme di addestramento e 7 a ciascuno degli insiemi di validazione e test. Questa scelta, adottata anche nel framework di partenza [19], evita stime ottimistiche delle prestazioni dovute alla presenza dello stesso soggetto in addestramento e in test, condizione poco realistica dal punto di vista industriale [32].

Per il compito di classificazione, i campioni delle tre classi sono stati bilanciati in numero, così da evitare che il modello privilegi la classe più rappresentata.

Insieme	Partecipanti	FLW	LCL	LCR
Training	36	31 540	31 540	31 540
Validation	7	5 444	5 444	5 444
Test	7	6 574	6 574	6 574

Tabella 4.2: Composizione del dataset utilizzato per la classificazione

4.3 Architetture dei modelli

Sono state confrontate quattro architetture per serie temporali, già introdotte nella Sezione 2.3. Per ciascun modello è stato definito uno spazio di ricerca degli iperparametri ed è stata condotta una ricerca automatica mediante ottimizzazione bayesiana.

L'ottimizzazione bayesiana costruisce un modello surrogato della funzione obiettivo, ossia della metrica di validazione in funzione degli iperparametri, e lo sfrutta per selezionare la configurazione successiva da valutare. Ogni tentativo aggiorna il surrogato, orientando progressivamente la ricerca verso le regioni più promettenti dello spazio e realizzando un compromesso tra esplorazione di configurazioni ancora poco note e sfruttamento di quelle che hanno già dato buoni risultati. Si raggiungono così configurazioni competitive con un numero di addestramenti sensibilmente inferiore rispetto a una ricerca esaustiva, aspetto rilevante data la dimensione degli spazi considerati.

Per ciascuna architettura sono stati addestrati tre modelli: uno per la classificazione della manovra e due per la regressione del TTLC, uno per ciascuna direzione. Le due varianti condividono l'impianto generale e si distinguono per lo strato di uscita, costituito da tre unità con attivazione softmax nel caso della classificazione e da una singola unità con attivazione lineare nel caso della regressione. Le sottosezioni seguenti descrivono ciascuna architettura e le differenze tra le due varianti; i dettagli della procedura di ricerca sono riportati nella Sezione 4.4.

4.3.1 CNN1D

Il modello è costituito da una sequenza di blocchi convoluzionali, ciascuno composto da uno strato Conv1D con padding *same*, normalizzazione del batch, attivazione ReLU e dropout. Il padding *same* preserva la lunghezza della sequenza lungo tutta la parte convoluzionale, cosicché la risoluzione temporale viene ridotta soltanto dagli eventuali strati di max pooling, ognuno dei quali la dimezza. Segue uno stadio di aggregazione e una testa densa che porta allo strato di uscita.

Lo stadio di aggregazione può essere realizzato con un global average pooling, che media le attivazioni lungo l'asse temporale producendo un vettore di dimensione pari al numero di filtri, oppure con un appiattimento dell'intera mappa di attivazione. La prima alternativa riduce il numero di parametri dello strato successivo, la seconda conserva l'informazione delle caratteristiche estratte.

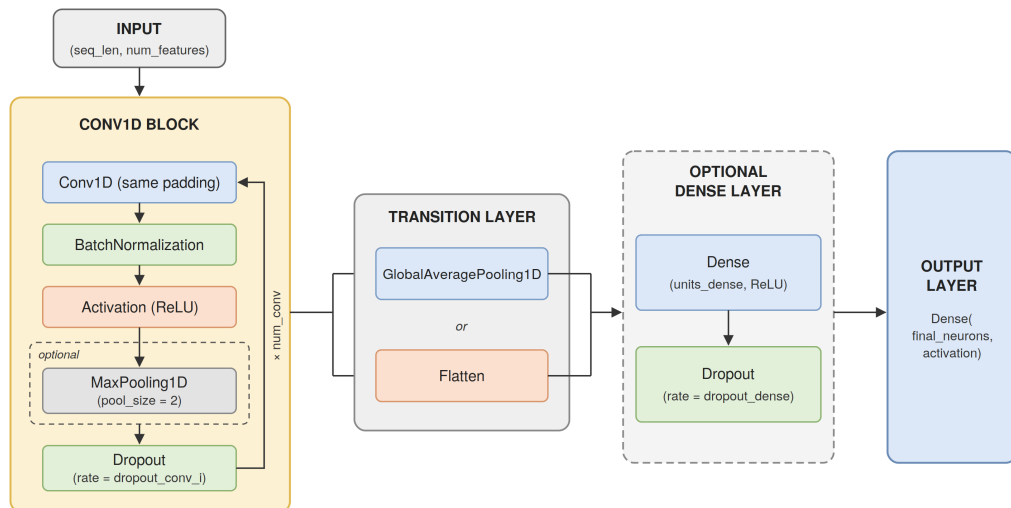


Figura 4.2: Architettura del modello CNN1D

4.3.2 Transformer

La finestra in ingresso viene arricchita con una codifica posizionale appresa, realizzata mediante uno strato di embedding indicizzato dalla posizione del campione all'interno della finestra. La codifica è necessaria perché il meccanismo di self-attention descritto dall'equazione (2.5) è invariante rispetto all'ordine degli elementi della sequenza.

La sequenza attraversa quindi uno stack di blocchi encoder, ciascuno composto da uno strato di multi-head self-attention e da una rete feed-forward a due strati densi, entrambi seguiti da dropout e connessione residua e accompagnati da normalizzazione di strato.

La rappresentazione prodotta dall'ultimo blocco viene aggregata lungo l'asse temporale, mediante media globale oppure selezionando lo stato corrispondente all'ultimo istante della finestra ed infine connessa a una testa densa con attivazione ReLU e dropout.

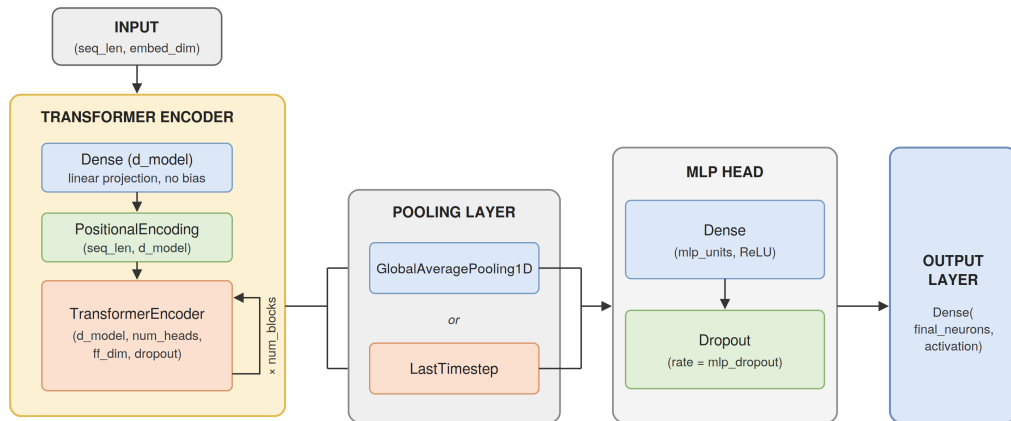


Figura 4.3: Architettura del modello Transformer

4.3.3 BI-GRU

Il modello elabora la finestra attraverso uno o più strati di celle GRU bidirezionali: la sequenza viene percorsa in entrambi i versi e i due stati nascosti vengono concatenati, cosicché la rappresentazione di ogni istante tiene conto sia dei campioni precedenti sia di quelli successivi all'interno della finestra. Come osservato nella Sezione 2.3.2, ciò non introduce alcuna dipendenza da informazioni non ancora disponibili al momento della predizione, poiché la finestra si conclude nell'istante corrente.

Gli strati ricorrenti impiegano la tangente iperbolica come funzione di attivazione e sono ciascuno seguito da uno strato di dropout. Tutti gli strati intermedi restituiscono l'intera sequenza di stati nascosti, mentre l'ultimo restituisce il solo stato finale, che costituisce la rappresentazione compatta della finestra: l'aggregazione temporale è quindi implicita nella ricorrenza e non richiede uno stadio di pooling. Segue una testa densa opzionale con attivazione ReLU e dropout. La profondità della rete è controllata da due variabili booleane annidate: il secondo strato ricorrente può essere inserito o meno e, solo se presente, può esserlo anche il terzo; il numero di unità ammesso decresce con la profondità, secondo uno schema a imbuto che concentra la capacità del modello negli strati iniziali. In entrambe le varianti, l'addestramento impiega il clipping della norma del gradiente al valore 1.0.

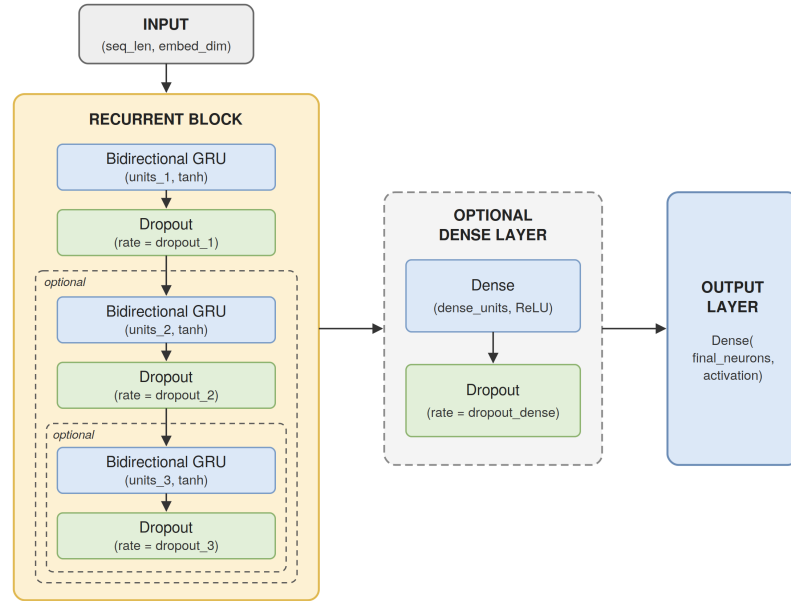


Figura 4.4: Architettura del modello BI-GRU

4.3.4 CNN1D-LSTM

Il modello combina i due approcci precedenti: un blocco convoluzionale estrae le caratteristiche locali della serie e uno o più strati LSTM ne modellano l'evoluzione temporale. La parte convoluzionale è analoga a quella descritta nella Sezione 4.3.1, con strati `Conv1D` a padding *same*, normalizzazione del batch, attivazione ReLU, max pooling opzionale e dropout. La sequenza di caratteristiche così ottenuta alimenta gli strati ricorrenti: gli strati intermedi restituiscono l'intera sequenza, mentre l'ultimo restituisce il solo stato finale, che viene inoltrato alla testa densa opzionale e quindi allo strato di uscita.

Il beneficio principale di questa combinazione è la riduzione del numero di passi che la ricorrenza deve percorrere: ogni strato di max pooling dimezza la lunghezza

della sequenza, per cui con due blocchi dotati di pooling i 50 campioni della finestra si riducono a 13 prima di raggiungere la parte ricorrente, con un vantaggio sia sul costo computazionale sia sulla stabilità dell'addestramento.

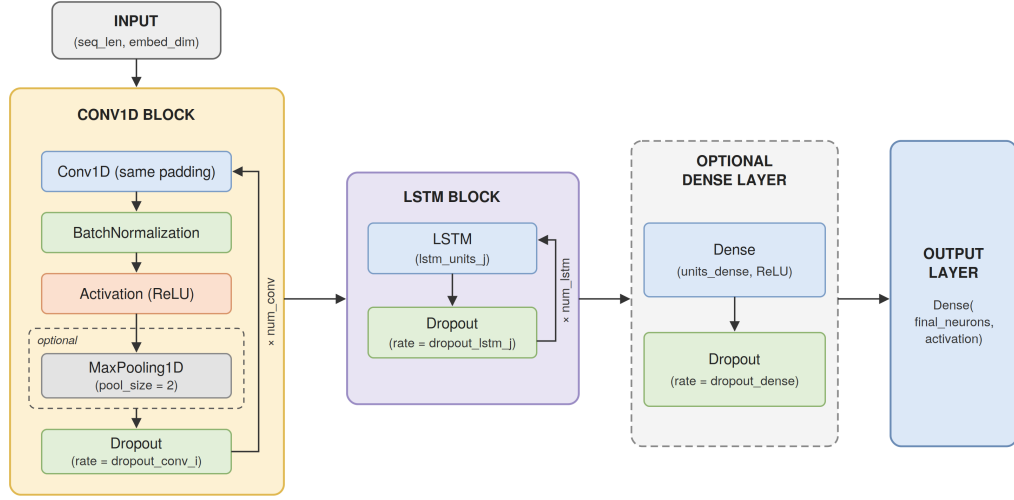


Figura 4.5: Architettura del modello CNN1D-LSTM

4.4 Procedura di addestramento

I regressori sono addestrati minimizzando l'errore quadratico medio tra il TTLC predetto e quello reale,

$$\mathcal{L}_{\text{reg}} = \frac{1}{M} \sum_{m=1}^M (y_m - \hat{y}_m)^2, \quad (4.7)$$

mentre il classificatore è addestrato minimizzando l'entropia incrociata categorica

$$\mathcal{L}_{\text{cls}} = -\frac{1}{M} \sum_{m=1}^M \sum_{k=1}^3 \mathbf{1}\{c_m = k\} \log \hat{p}_{m,k}, \quad (4.8)$$

dove M è il numero di campioni del batch, $c_m \in \{\text{FLW}, \text{LCL}, \text{LCR}\}$ la classe corretta del campione m , $\hat{p}_{m,k}$ la probabilità assegnata dal modello alla classe k per lo stesso campione, e $\mathbf{1}\{\cdot\}$ la funzione indicatrice, pari a 1 se la condizione è verificata e 0 altrimenti.

4.5 Metriche di valutazione

4.5.1 Metriche di classificazione

Le prestazioni del classificatore vengono valutate separatamente per ciascuna classe: fissata la classe k si definiscono veri positivi TP_k (i campioni della classe k correttamente riconosciuti), falsi positivi FP_k (i campioni di altre classi assegnati erroneamente a k) e falsi negativi FN_k (i campioni della classe k assegnati a un'altra classe).

$$\text{Precision}_k = \frac{TP_k}{TP_k + FP_k}, \quad \text{Recall}_k = \frac{TP_k}{TP_k + FN_k} \quad (4.9)$$

e la loro media armonica

$$F_{1,k} = 2 \frac{\text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k} \quad (4.10)$$

I due tipi di errore hanno conseguenze diverse. Una recall bassa su una classe di cambio corsia indica che una parte delle manovre non viene riconosciuta, mentre una precision bassa indica che il modello segnala manovre che non avvengono. Poiché il framework è finalizzato a supportare attivamente il cambio di corsia [19], entrambe le grandezze meritano un'attenzione particolare in fase di analisi dei

risultati.

4.5.2 Metriche di regressione

Le prestazioni dei regressori sono valutate con l'errore assoluto medio (MAE), l'errore quadratico medio (MSE) e la sua radice (RMSE):

$$\text{MSE} = \frac{1}{M} \sum_{m=1}^M (y_m - \hat{y}_m)^2 \quad (4.11)$$

$$\text{MAE} = \frac{1}{M} \sum_{m=1}^M |y_m - \hat{y}_m| \quad (4.12)$$

$$\text{RMSE} = \sqrt{\text{MSE}} \quad (4.13)$$

Le tre metriche forniscono informazioni complementari. L'MSE, e di conseguenza l'RMSE, eleva al quadrato gli scostamenti penalizzando in misura maggiore gli errori grandi, mentre il MAE pesa allo stesso modo tutti gli scostamenti. MAE e RMSE sono espressi nelle unità del target, ossia in secondi risultando direttamente interpretabili come scostamento temporale della predizione; l'MSE, essendo espresso in secondi al quadrato, viene riportato per completezza, ma non ammette un'interpretazione altrettanto immediata.

Capitolo 5

Risultati sperimentali

Questo capitolo presenta i risultati ottenuti applicando la metodologia descritta nel capitolo 4. La sezione 5.1 quantifica l'effetto del filtraggio delle traiettorie, le sezioni 5.3 e 5.2 riportano le prestazioni dei modelli sui due compiti, la sezione 5.4 analizza gli errori commessi e la sezione 5.5 confronta i risultati con quelli del framework di partenza.

5.1 Effetto del filtraggio delle traiettorie

L'applicazione del controllo di fattibilità cinematica riduce il numero di traiettorie disponibili, come riportato nella Tabella 5.1. I valori si riferiscono alle intere manovre sottoposte al controllo, lo scarto di una traiettoria comporta quindi la perdita di tutti i campioni che ne sarebbero derivati (sezione 4.2.1)

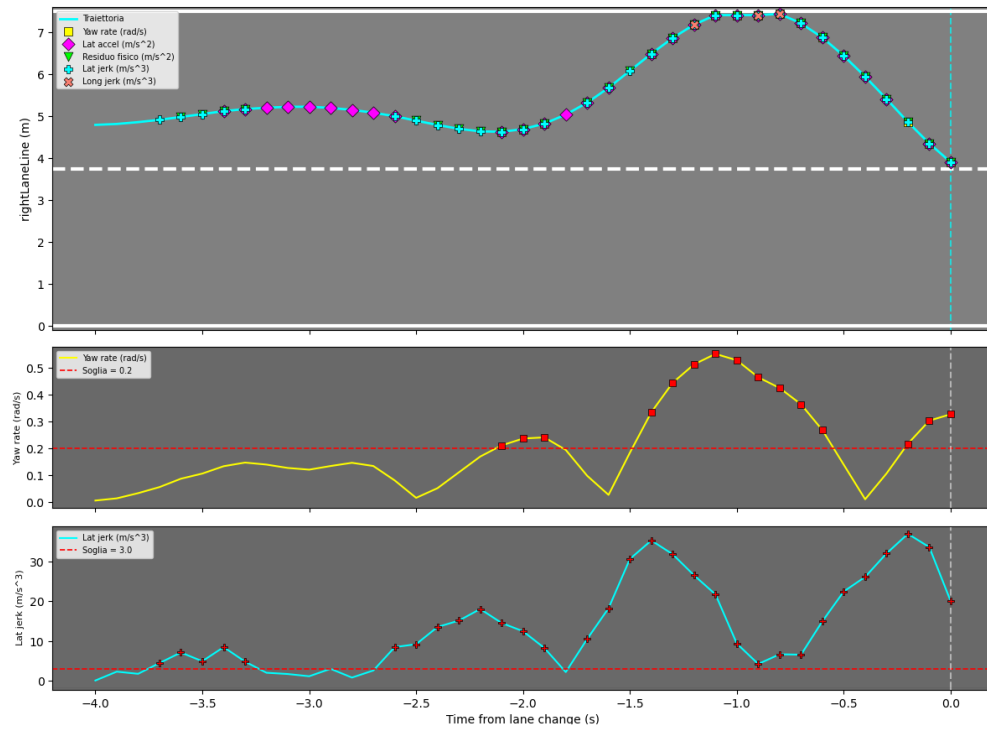
Classe	Originali	Dopo filtraggio	Rimosse	Rimosse [%]
LCL	1304	1103	201	15.4
LCR	1133	922	211	18.6
FLW	3500	2333	1167	33.3

Tabella 5.1: Effetto del filtraggio sul numero di traiettorie disponibili

Per le manovre di cambio corsia la riduzione complessiva è del 16.9%, distribuita in modo sostanzialmente uniforme tra le due direzioni. Le finestre di marcia in corsia risultano invece penalizzate in misura molto maggiore, con il 33.3% dei campioni scartati. Il risultato è coerente con la scelta di adottare soglie più restrittive per questa classe: un tratto di marcia stazionaria che presenti accelerazioni laterali superiori a 2.5 m/s^2 o jerk elevati non descrive un comportamento di guida regolare, ma quasi sempre un artefatto della simulazione o una perdita di controllo del veicolo.

Un ulteriore elemento di interesse è rappresentato dal confronto tra le due direzioni del cambio corsia. Le percentuali di finestre rimosse per LCL e LCR, pari rispettivamente al 15.4% e al 18.6%, risultano relativamente vicine. Il filtraggio produce quindi un effetto complessivamente simile sulle due tipologie di cambio corsia, senza evidenziare una marcata dipendenza dalla direzione della manovra. La differenza diventa invece più evidente nel confronto con la classe FLW, per la quale viene scartato il 33.3% delle finestre disponibili. Questo risultato è coerente con l'adozione di soglie più restrittive per la marcia in corsia, per la quale ci si attende una dinamica più contenuta rispetto a una manovra di cambio corsia.

L'ispezione visiva delle finestre scartate conferma quanto detto sopra. Il superamento delle soglie coincide con un evento anomalo osservabile nella simulazione, come un urto contro la barriera laterale o una collisione con un altro veicolo, in cui gli indicatori cinematici raggiungono valori di uno o due ordini di grandezza superiori ai limiti imposti.



(a) Indicatori cinematici

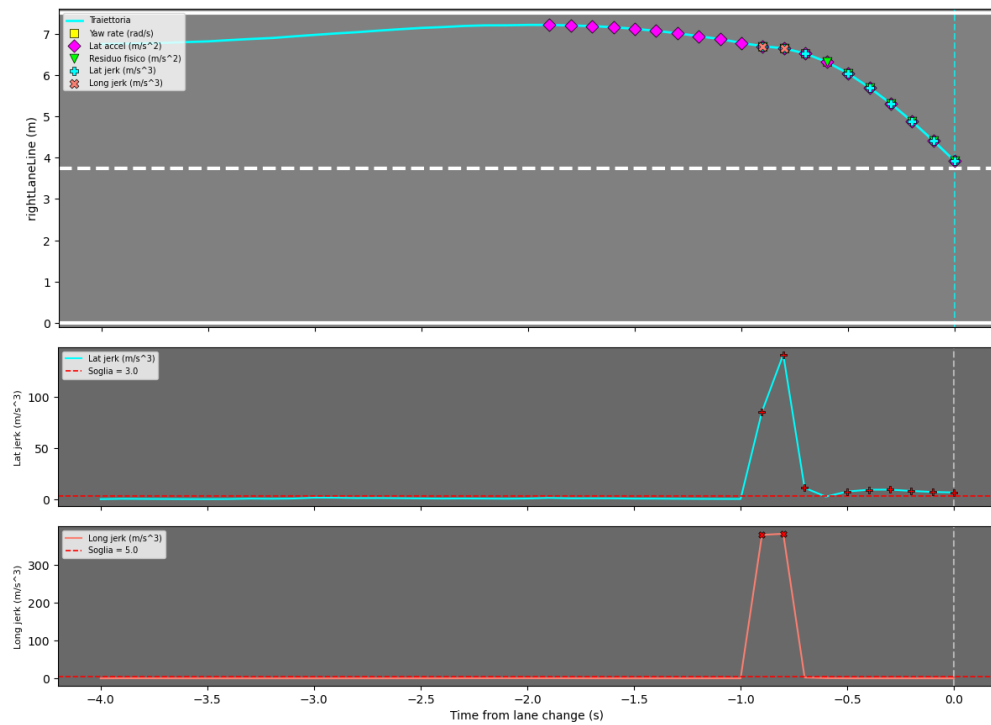


(b) Vista della simulazione

Figura 5.1: Finestra scartata dal filtro ed evento corrispondente

La Figura 5.1 mostra un caso rappresentativo. La traiettoria del veicolo raggiunge 7.5 m, ossia il bordo sinistro della carreggiata, per poi attraversarla in senso opposto fino a ritornare in prossimità della linea di mezzzeria. La vista della simulazione nell'istante corrispondente conferma l'interpretazione, mostrando il veicolo a contatto con la barriera laterale.

L'andamento degli indicatori cinematici rende evidente perché la finestra venga scartata. La velocità di imbardata raggiunge un massimo di circa 0.55 rad/s con due distinti superamenti in corrispondenza dell'uscita e del successivo rientro, facendo intuire che il conducente abbia controsterzato bruscamente tentando di correggere la traiettoria.



(a) Indicatori cinematici



(b) Vista della simulazione

Figura 5.2: Finestra scartata dal filtro ed evento corrispondente

La Figura 5.2 mostra un caso di natura diversa, riconducibile a una collisione. Fino a circa 1 s prima dell'istante di attraversamento, tutti gli indicatori si mantengono su valori ordinari: la velocità di imbardata resta attorno a 0.1 rad/s, l'accelerazione laterale a 1.8 m/s², mentre il residuo di coerenza e i due jerk sono prossimi a zero. A $t \simeq -0.8$ s si verifica una discontinuità simultanea su tutte le grandezze: il jerk longitudinale raggiunge 350 m/s³, quasi sessanta volte la soglia, il jerk laterale 120 m/s³ e il residuo di coerenza 20 m/s², un ordine di grandezza oltre il limite ammesso. La velocità di imbardata sale a 0.40 rad/s e l'accelerazione laterale a 9.5 m/s², per poi rientrare gradualmente.

L'andamento definito da un impulso concentrato in uno o due campioni, seguito da un transitorio di assestamento è compatibile con un impatto. La vista della simulazione nell'istante corrispondente conferma l'interpretazione, mostrando il veicolo in collisione con un altro veicolo del traffico durante lo spostamento verso la corsia di destra.

Casi come questi, in assenza del controllo di fattibilità, verrebbero utilizzati per l'addestramento al pari delle manovre corrette. Così facendo, il modello imparebbe ad associare all'imminenza di un cambio di corsia una dinamica che nessun conducente produrrebbe su strada.

5.2 Classificazione

5.2.1 Classificazione della manovra

Il classificatore è stato addestrato sulle tre classi FLW, LCL e LCR, con campioni bilanciati secondo quanto riportato nella Tabella 4.2. La Tabella 5.2 riassume le prestazioni delle quattro architetture sull'insieme di test.

Modello	F_1 FLW	F_1 LCR	F_1 LCL	Accuratezza
CNN1D	0.88	0.96	0.90	0.92
Transformer	0.89	0.96	0.91	0.92
BI-GRU	0.88	0.96	0.92	0.92
CNN1D-LSTM	0.89	0.97	0.91	0.92

Tabella 5.2: Prestazioni di classificazione sull'insieme di test: F_1 -score per classe e accuratezza complessiva

Un aspetto particolarmente evidente è la stabilità delle prestazioni complessive tra le quattro architetture considerate. Tutti i modelli raggiungono un'accuratezza pari al 92%. Analogamente, i valori di F_1 per le singole classi mostrano variazioni contenute. Il valore più elevato è ottenuto sistematicamente per la classe LCR, con F_1 compreso tra 0.96 e 0.97, la classe FLW presenta valori compresi tra 0.88 e 0.89 e la classe LCL tra 0.90 e 0.92. Le differenze tra le architetture risultano quindi maggiormente visibili a livello delle singole classi, piuttosto che nella prestazione complessiva. In particolare, il BI-GRU raggiunge il valore di F_1 più elevato per LCL, pari a 0.92, mentre la CNN1D-LSTM raggiunge 0.97 per LCR; anche per FLW i valori rimangono compresi in un intervallo molto ristretto. Nel complesso, i risultati mostrano che nessuna delle architetture considerate produce un cambiamento marcato nell'accuratezza globale del classificatore.

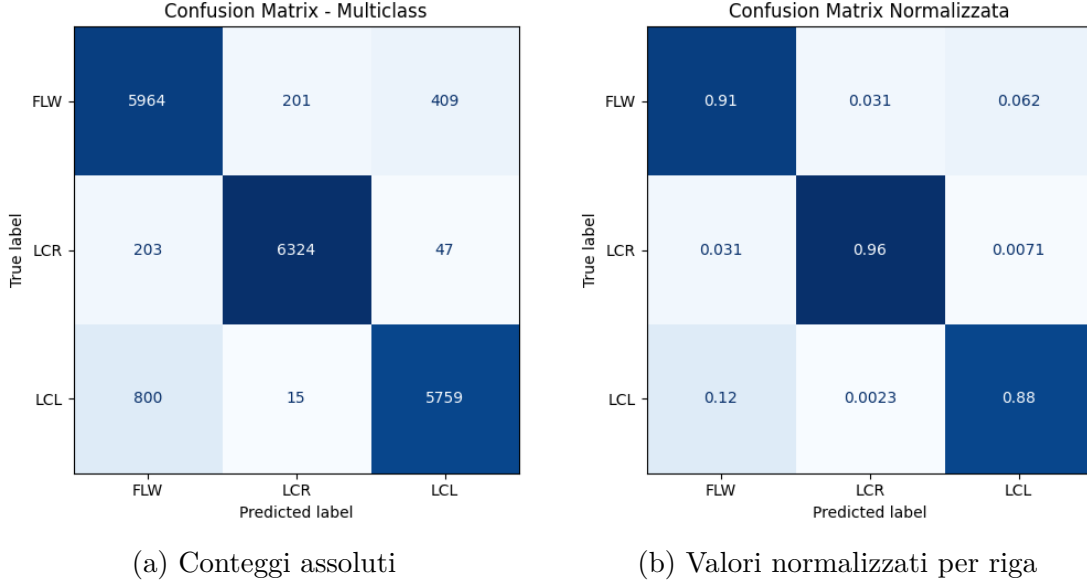


Figura 5.3: Matrice di confusione del modello CNN1D sull'insieme di test.

L'analisi della matrice di confusione mostra che l'errore dominante è la confusione tra le classi FLW e LCL: circa il 10–12% dei cambi corsia verso sinistra viene classificato come marcia in corsia, mentre la confusione tra le due direzioni di manovra è praticamente assente, inferiore all'1% in tutti i modelli.

5.2.2 Caratteristiche di deployment

Le caratteristiche di deployment delle quattro architetture sono state valutate mediante il tool *ST Edge AI Developer Cloud*, che consente di stimare occupazione di memoria e tempo di inferenza su una libreria di microcontrollori, senza disporre fisicamente del dispositivo. I modelli sono stati esportati in precisione singola (FP32), senza applicare alcuna tecnica di quantizzazione.

Parametro	CNN1D	BI-GRU	Transformer	CNN1D-LSTM
Flash [MiB]	1.69	0.53	0.31	2.42
RAM [KiB]	38.61	280.29	283.91	148.85
<i>Tempo di inferenza [ms]</i>				
STM32H7B3 (280 MHz)	33.7	61.7	213.4	391.5
STM32F469 (180 MHz)	92.6	149.3	505.9	1569
STM32U3C5 (96 MHz)	140.5	348.9	624.4	×
Raspberry Pi 5 (2.4 GHz)	0.13	0.71	2.55	4.35

Tabella 5.3: Caratteristiche di deployment dei classificatori in FP32. Il simbolo $\times$ indica che il modello non risulta installabile sul dispositivo considerato

Per il classificatore si è adottata la CNN1D. Poiché esso costituisce lo stadio sempre attivo della pipeline, deve completare l’inferenza entro il periodo di campionamento, pari a 100 ms. Tale vincolo è soddisfatto su STM32H7B3 dalla sola CNN1D e dalla BI-GRU, e su STM32F469 dalla sola CNN1D, mentre il Transformer e la CNN1D-LSTM lo superano su tutte le piattaforme considerate. A questo si aggiunge l’occupazione di memoria volatile, pari a circa 38 KiB per la CNN1D, contro i 149–299 KiB delle altre architetture. La CNN1D richiede per contro una maggiore occupazione di memoria di programma rispetto al Transformer e alla BI-GRU, principalmente a causa del numero di parametri dello strato denso finale. Tale occupazione costituisce tuttavia un costo statico, in quanto la memoria di programma viene allocata una volta in fase di programmazione e non deve essere nuovamente allocata a ogni inferenza. Al contrario, il tempo di esecuzione e le operazioni di accesso alla memoria rappresentano costi ricorrenti a ogni inferenza e possono quindi incidere direttamente sul consumo energetico complessivo del sistema.

5.2.3 Configurazione selezionata

La configurazione individuata dalla ricerca degli iperparametri per questo modello è riportata nella Tabella 5.4.

Iperparametro	Valore
Blocchi convoluzionali	2
Filtri per blocco	64, 128
Ampiezza del kernel	3
Dropout	0.3
Aggregazione	flatten
Unità dello strato denso	64
Dropout dello strato denso	0.4
Learning rate	10^{-4}

Tabella 5.4: Configurazione del classificatore CNN1D selezionata dalla ricerca degli iperparametri

5.2.4 Confronto con lo stato dell'arte

Lavoro	Modello	BACC	Dataset
Wirthmuller et al. [37]	Random Forest	0.84	highD
Mozaffari et al. [10]	CNN + Attention	0.83	highD
Zhang et al. [38]	Stacking Ensemble	0.92	Simulatore
Questo lavoro	CNN1D	0.92	CARLA

Tabella 5.5: Confronto delle prestazioni con alcuni lavori presenti nello stato dell'arte

Viene usata l'accuratezza bilanciata (BACC), ossia la media delle recall sulle tre classi, in quanto è l'unica metrica riportata o ricavabile per tutti i lavori considerati. A differenza dell'accuratezza semplice, la BACC non dipende dalla composizione dell'insieme di valutazione, che varia sensibilmente tra i riferimenti.

I risultati riportati non sono direttamente confrontabili, in quanto ottenuti mediante modelli differenti e su dataset, scenari e condizioni sperimentali non omogenei. La tabella fornisce pertanto unicamente un inquadramento generale delle prestazioni del modello proposto rispetto ad alcuni lavori presenti nello stato dell'arte.

5.3 Regressione del TTLC

I quattro modelli sono stati addestrati separatamente sulle manovre verso sinistra e verso destra, con target il TTLC nell'intervallo $[0, 4]$ s.

5.3.1 Cambio corsia verso sinistra

	Transformer	CNN1D	BI-GRU	CNN1D-LSTM
MSE [s ²]	0.194	0.238	0.191	0.192
MAE [s]	0.297	0.330	0.305	0.300
RMSE [s]	0.440	0.489	0.437	0.438

Tabella 5.6: Risultati di regressione per il cambio corsia verso sinistra (LCL)

Le quattro architetture ottengono prestazioni molto ravvicinate, con l'eccezione della CNN1D, che risulta la meno accurata su tutte le metriche. Il BI-GRU raggiunge il valore migliore di MSE e di RMSE, mentre il Transformer ottiene il MAE più basso. La combinazione è significativa: il Transformer commette mediamente errori più piccoli, ma il BI-GRU contiene meglio gli errori di entità elevata, che l'MSE penalizza in misura maggiore.

Gli scarti tra i tre modelli migliori sono tuttavia dell'ordine di pochi millesimi di secondo e vanno interpretati con cautela: differenze di questa entità rientrano

nella variabilità dovuta all’inizializzazione casuale dei pesi e non consentono di stabilire una gerarchia netta tra le architetture.

5.3.2 Caratteristiche di deployment

Parametro	CNN1D	BI-GRU	Transformer	CNN1D-LSTM
Flash [MiB]	0.46	1.02	0.17	1.98
RAM [KiB]	23.34	86.09	125.2	127.53
<i>Tempo di inferenza [ms]</i>				
STM32H7B3 (280 MHz)	26.96	315.2	55.92	304.3
STM32F469 (180 MHz)	77.9	698.1	144.3	677.4
STM32U3C5 (96 MHz)	118.5	1130.0	198.1	×
Raspberry Pi 5 (2.4 GHz)	0.08	2.54	0.35	3.25

Tabella 5.7: Caratteristiche di deployment del regressore LCL in FP32. Il simbolo × indica che il modello non risulta installabile sul dispositivo considerato

A differenza del classificatore, i regressori vengono invocati esclusivamente in seguito al riconoscimento di una manovra; di conseguenza, il vincolo di tempo reale associato al periodo di campionamento del classificatore non si applica al regressore nei medesimi termini.

Il Transformer risulta il modello con la minore occupazione di memoria di programma fra le quattro architetture considerate, pari a 0.17 MiB, contro gli 0.46 MiB della CNN1D e gli 1.98 MiB della CNN1D-LSTM. Fra i modelli installabili su tutte e tre le piattaforme, condizione non soddisfatta dalla CNN1D-LSTM che risulta esclusa dal dispositivo più vincolato, esso presenta un tempo di inferenza inferiore di oltre cinque volte a quello della BI-GRU, risultando secondo soltanto alla CNN1D. Su un dispositivo di classe superiore, rappresentato in questo caso dal Raspberry Pi 5, il tempo di inferenza del Transformer si riduce a 0.35 ms. In

queste condizioni, il costo computazionale dell’inferenza risulta sufficientemente contenuto da rendere meno rilevanti le differenze di latenza tra le architetture e la scelta del modello può quindi essere guidata principalmente dall’errore di predizione.

5.3.3 Configurazione selezionata

Iperparametro	Valore
Dimensione interna d_{model}	64
Teste di attenzione	2
Unità della rete feed-forward	128
Blocchi encoder	1
Dropout nei blocchi encoder	0.1
Aggregazione	ultimo istante
Unità dello strato denso	160
Dropout dello strato denso	0.2
Learning rate	10^{-4}

Tabella 5.8: Configurazione del regressore Transformer per le manovre verso sinistra, selezionata dalla ricerca degli iperparametri

5.3.4 Cambio corsia verso destra

	Transformer	CNN1D	BI-GRU	CNN1D-LSTM
MSE [s ²]	0.179	0.237	0.190	0.190
MAE [s]	0.290	0.327	0.308	0.309
RMSE [s]	0.423	0.487	0.436	0.436

Tabella 5.9: Risultati di regressione per il cambio corsia verso destra (LCR)

Sulle manovre verso destra il Transformer risulta il migliore su tutte e tre le metriche, con un RMSE di 0.423 s. Si osserva inoltre che tutti i modelli ottengono

risultati leggermente migliori rispetto alle manovre verso sinistra. Una possibile spiegazione risiede nella diversa natura delle due manovre nello scenario considerato: il rientro a destra avviene tipicamente al termine di un sorpasso, mentre lo spostamento a sinistra dipende in misura maggiore dallo stile di guida del conducente e dal momento in cui decide di avviare il sorpasso.

5.3.5 Caratteristiche di deployment

Parametro	CNN1D	BI-GRU	Transformer	CNN1D-LSTM
Flash [MiB]	0.77	0.93	0.17	0.71
RAM [KiB]	13.19	111.84	125.04	711.44
<i>Tempo di inferenza [ms]</i>				
STM32H7B3 (280 MHz)	11.90	286.0	55.75	156.60
STM32F469 (180 MHz)	31.95	633.7	144.40	350.90
STM32U3C5 (96 MHz)	50.18	1083.0	198.10	603.80
Raspberry Pi 5 (2.4 GHz)	0.05	6.38	0.37	2.35

Tabella 5.10: Caratteristiche di deployment del regressore LCR in FP32

Per la direzione LCR tutte e quattro le architetture risultano installabili su ogni piattaforma considerata, e i discriminanti sono pertanto l'occupazione di memoria di programma e il tempo di esecuzione. Il Transformer risulta il più compatto dei quattro, con 0.17 MiB contro gli 0.77 MiB della CNN1D e gli 0.93 MiB della BI-GRU, e presenta un tempo di inferenza circa cinque volte inferiore a quello della BI-GRU e tre volte inferiore a quello della CNN1D-LSTM, risultando secondo soltanto alla CNN1D. Su un dispositivo di classe superiore, rappresentato in questo caso dal Raspberry Pi 5, il tempo di inferenza si riduce a 0.37 ms e

valgono pertanto le medesime considerazioni svolte per la direzione LCL nella Sezione 5.3.2.

5.3.6 Configurazione selezionata

Iperparametro	Valore
Dimensione interna d_{model}	64
Teste di attenzione	2
Unità della rete feed-forward	128
Blocchi encoder	1
Dropout nei blocchi encoder	0.1
Aggregazione	ultimo istante
Unità dello strato denso	160
Dropout dello strato denso	0.2
Learning rate	10^{-4}

Tabella 5.11: Configurazione del regressore Transformer per le manovre verso destra, selezionata dalla ricerca degli iperparametri

Per il sistema finale si è quindi scelto di adottare il Transformer per entrambe le direzioni. Impiegare la stessa architettura per i due regressori semplifica la manutenzione della pipeline.

5.3.7 Confronto con lo stato dell'arte

Lavoro	Direzione	Modello	RMSE [s]	Dataset
Wirthmuller et al. [11]	Sx/Dx	LSTM	0.674/0.743	highD
Mozaffari et al. [10]	Non distinta	CNN + Attention	0.629	highD
Yang et al. [24]	Non distinta	CNN + LSTM	0.660	highD
Fornieris et al. [19]	Non distinta	Transformer	0.510	CARLA
Questo lavoro	Sx/Dx	Transformer	0.440/0.423	CARLA

Tabella 5.12: Confronto delle prestazioni con alcuni lavori presenti nello stato dell'arte

Come già spiegato nel caso della classificazione, anche in questo confronto i risultati devono essere interpretati con cautela, poiché sono stati ottenuti in condizioni sperimentali non completamente omogenee. La tabella è pertanto da intendersi come un riferimento generale rispetto ai risultati riportati in letteratura.

5.4 Analisi degli errori

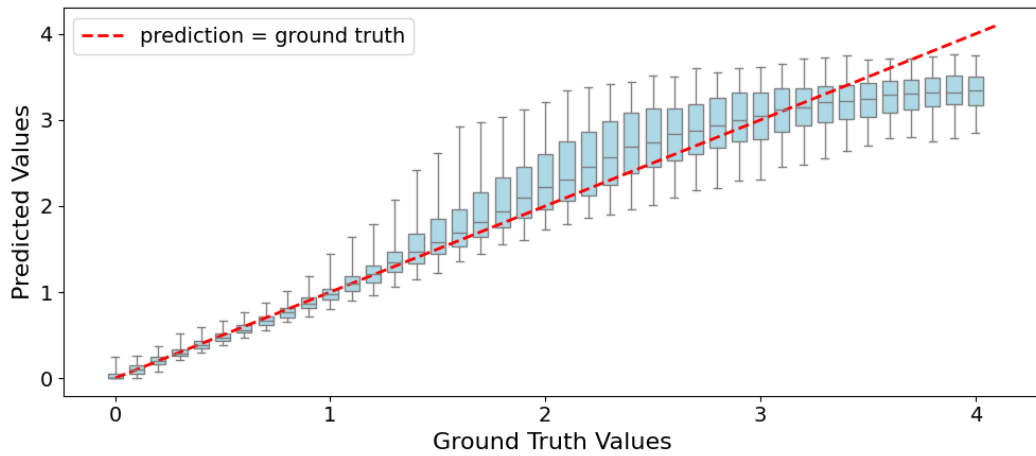
Le metriche aggregate non chiariscono quando i modelli sbagliano né con quale margine. Questa sezione analizza separatamente gli errori dei due stadi della pipeline: prima quelli del regressore, poi quelli del classificatore.

5.4.1 Errori di regressione

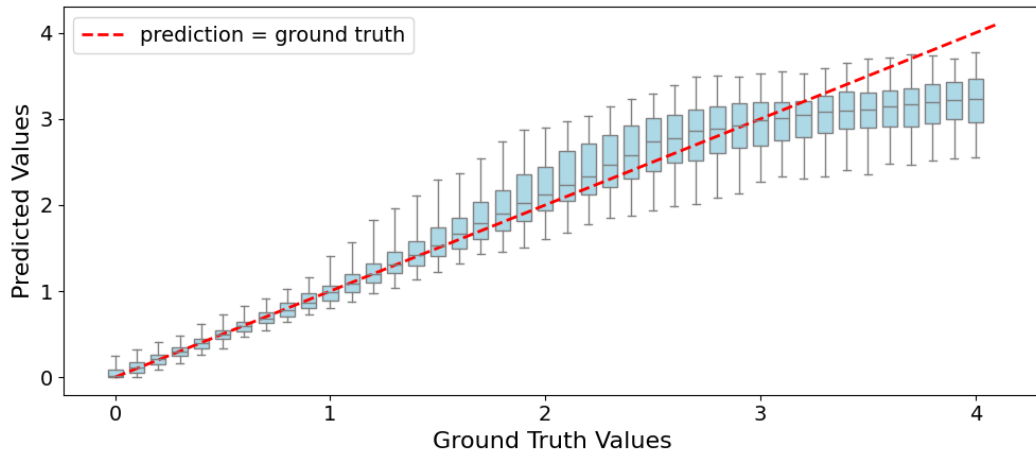
5.4.1.1 Distribuzione dell'errore

Le metriche riportate nella Sezione 5.3 sintetizzano l'errore in un unico valore, senza indicare come esso si distribuisca lungo l'orizzonte di predizione. La Figura 5.4 riporta la distribuzione dei valori predetti in funzione del TTLC reale,

la Figura 5.5 quella dell'errore assoluto: a ciascun intervallo dell'asse orizzontale corrisponde una box costruita sui campioni che vi ricadono.



(a) Cambio corsia verso sinistra

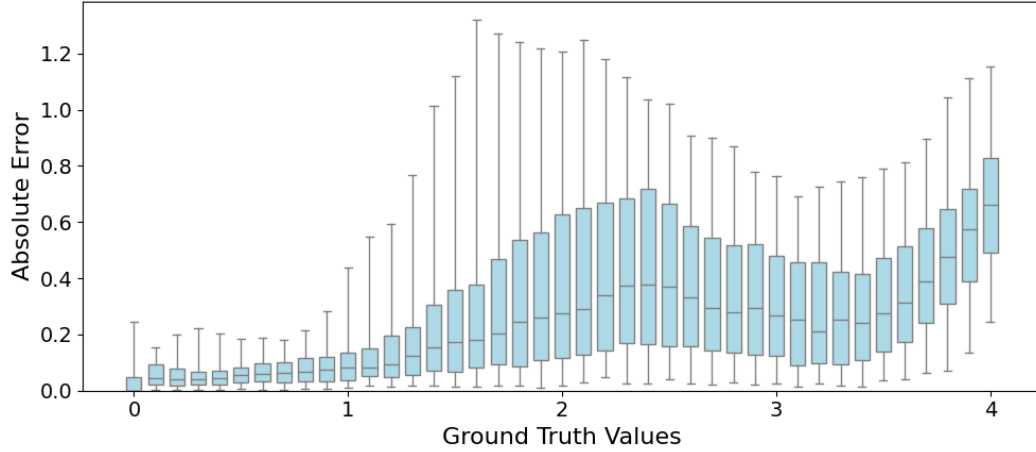


(b) Cambio corsia verso destra

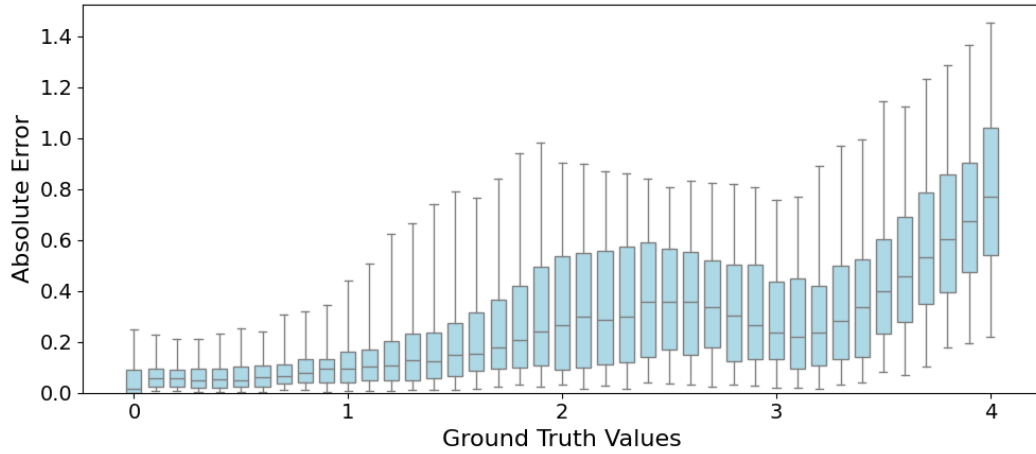
Figura 5.4: Distribuzione TTLC predetti vs valore reale (retta tratteggiata).

Come si può osservare dal box plot della Figura 5.4, al di sotto di circa 1 s le box sono strette e centrate sulla retta di predizione esatta, con un errore mediano

inferiore a 0.1 s in entrambe le direzioni. All'aumentare del TTLC la dispersione cresce e le mediane si collocano leggermente al di sopra della retta, con una tendenza a sovrastimare il tempo residuo. Oltre i 3 s il comportamento si inverte: le mediane scendono sotto la retta e i valori predetti saturano attorno a 3.35 s per le manovre verso sinistra e a 3.15 s per quelle verso destra.



(a) Cambio corsia verso sinistra



(b) Cambio corsia verso destra.

Figura 5.5: Distribuzione errore assoluto in funzione del TTLC reale

L'andamento dell'errore assoluto non è monotono. Il valore mediano cresce fino a un massimo locale in corrispondenza di 2.3–2.5 s, pari a circa 0.37 s per le manovre verso sinistra e 0.30 s per quelle verso destra, per poi diminuire fino a circa 0.22 s nell'intervallo 3.0–3.4 s ed infine raggiungere rispettivamente 0.66 s e 0.77 s in

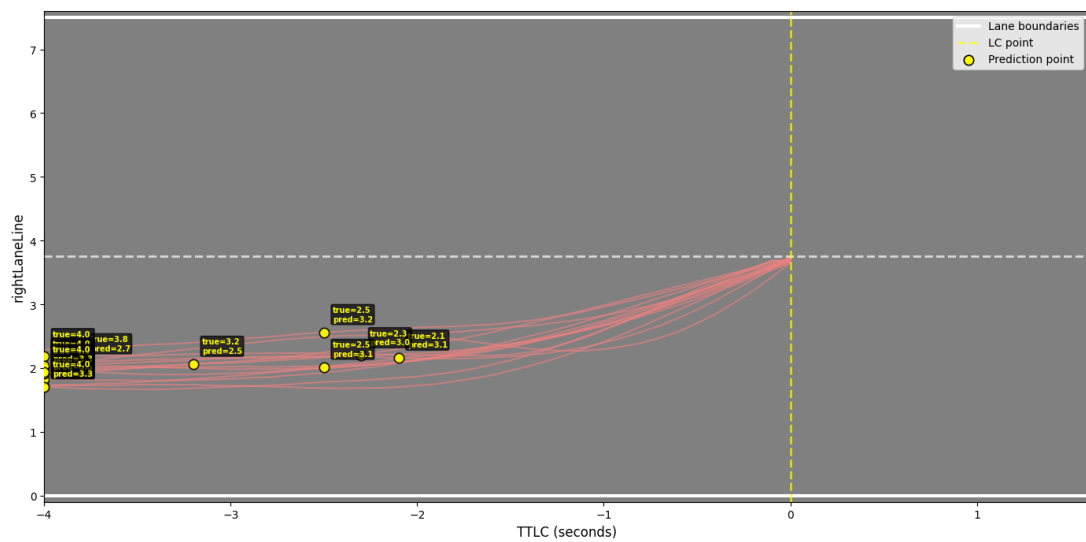
corrispondenza del limite superiore dell'intervallo.

La saturazione indica che, quando la manovra è ancora lontana, il modello riconosce la presenza dell'intenzione, ma non è in grado di collocarla nel tempo, e ripiega su un valore prossimo alla media della distribuzione. Si tratta dello stesso comportamento riscontrato nel framework di partenza [19]. La scomposizione in classificazione e regressione ne riduce l'entità, avendo sollevato il regressore dal compito di riconoscere l'intenzione, ma non lo elimina, poiché la causa risiede nell'informazione disponibile in ingresso piuttosto che nella formulazione del problema.

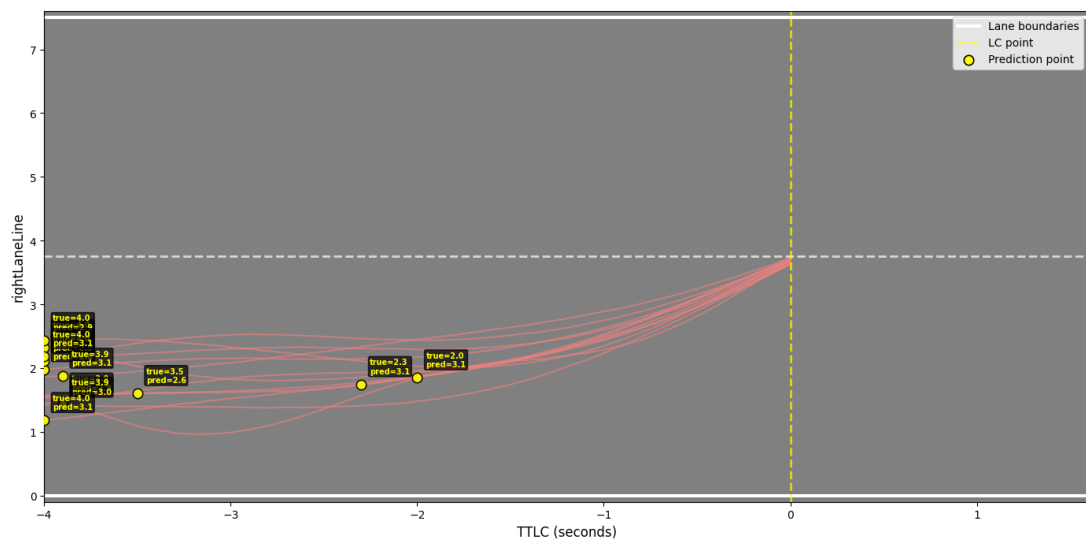
Questo comportamento evidenzia una distinzione importante tra la capacità del modello di riconoscere una manovra e quella di localizzarne con precisione l'istante di completamento. Quando il TTLC è elevato, infatti, la traiettoria contiene già informazioni sufficienti per suggerire la presenza di un futuro cambio corsia, ma non necessariamente elementi cinematici sufficienti per determinare quanto tempo manchi alla sua conclusione. La saturazione osservata nei valori predetti, quindi, può essere interpretata come una conseguenza della limitata osservabilità della manovra nelle sue fasi iniziali.

5.4.1.2 Analisi qualitativa

I diagrammi precedenti mostrano, per ciascun valore di TTLC compreso tra 0 e 4 s, la distribuzione degli errori sull'intero insieme di test e consentono quindi di valutare come l'accuratezza vari in funzione dell'istante considerato. Per osservare invece dove gli errori si collochino all'interno delle singole manovre, le Figure 5.6 e 5.7 riportano le traiettorie di due partecipanti, rispettivamente per le manovre verso sinistra e verso destra, nel piano definito dal tempo rispetto all'istante di attraversamento della linea e dalla distanza dalla linea di corsia destra. La posizione dei punti lungo la curva indica in quale fase della manovra si concentrino le predizioni meno accurate. Vengono riportate le sole manovre con errore assoluto superiore a 0.1 s; il punto evidenziato indica, per ciascuna di esse, l'istante in cui l'errore è massimo, annotato con il valore reale (*true*) e con quello predetto (*pred*).



(a) Utente 25

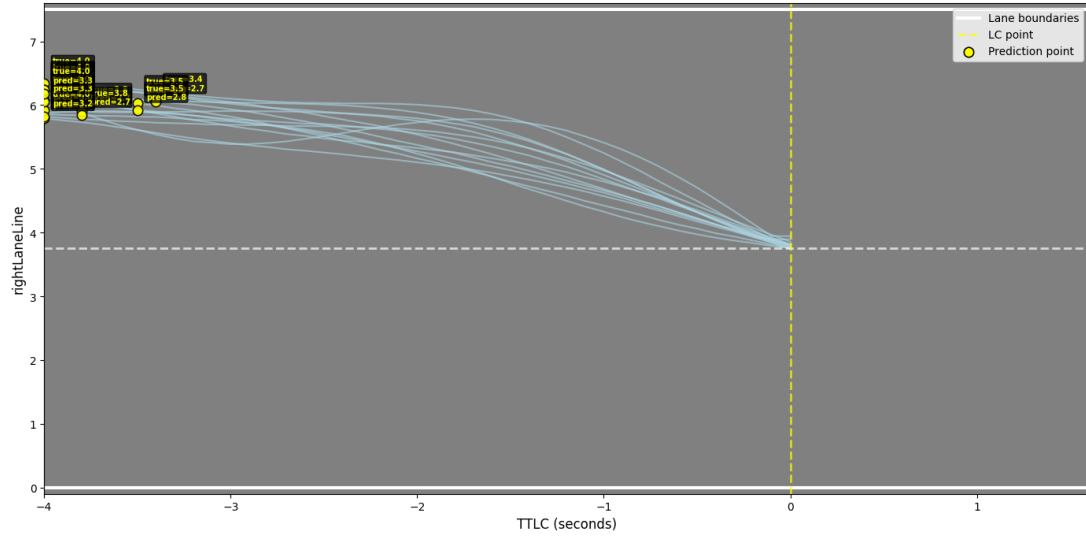


(b) Utente 18

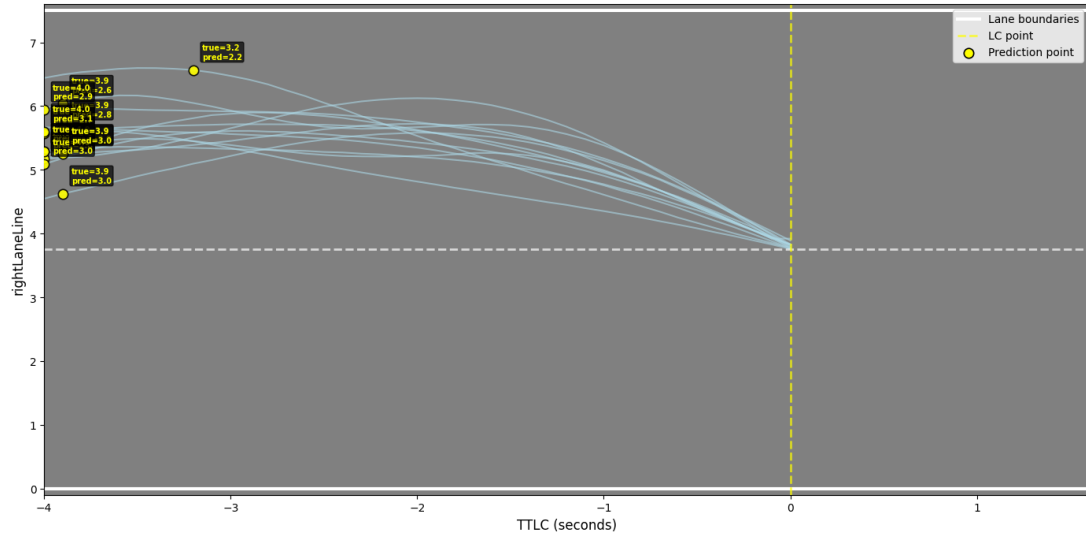
Figura 5.6: Errori di predizione del TTLC per le manovre di cambio corsia verso sinistra.

Le manovre verso sinistra riportate sono 13 per l'utente 25 e 12 per l'utente 18,

con errori massimi medi rispettivamente di 1.15 s e 0.99 s.



(a) Utente 25



(b) Utente 18

Figura 5.7: Errori di predizione del TTLC per le manovre di cambio corsia verso destra

Le manovre verso destra riportate sono 14 per l'utente 25 e 13 per l'utente 18, con errori massimi medi rispettivamente di 0.78 s e 1.15 s.

In tutti i casi, i punti di errore massimo si collocano nella porzione sinistra del grafico, a distanza ancora elevata dall'istante di attraversamento della linea, dove le traiettorie risultano sovrapposte tra loro e sostanzialmente parallele alla linea di corsia. Le predizioni meno accurate corrispondono quindi a configurazioni in cui lo spostamento laterale non si è ancora avviato e il veicolo è indistinguibile da uno che prosegue nella propria corsia. Il comportamento si ripresenta invariato su entrambe le direzioni e su entrambi i partecipanti, segno che non si tratta della peculiarità di un singolo stile di guida, ma di una caratteristica del problema: la difficoltà rilevata dai diagrammi precedenti trova qui la propria spiegazione geometrica.

5.4.2 Errori di classificazione

L'accuratezza aggregata non chiarisce quando e con quale grado di convinzione i modelli sbagliano. Le sottosezioni seguenti analizzano separatamente i due tipi di errore, mettendo in relazione l'uscita del classificatore con quella dei regressori.

5.4.2.1 Campioni di marcia in corsia classificati come cambio corsia

Questo tipo di errore corrisponde a un falso positivo.

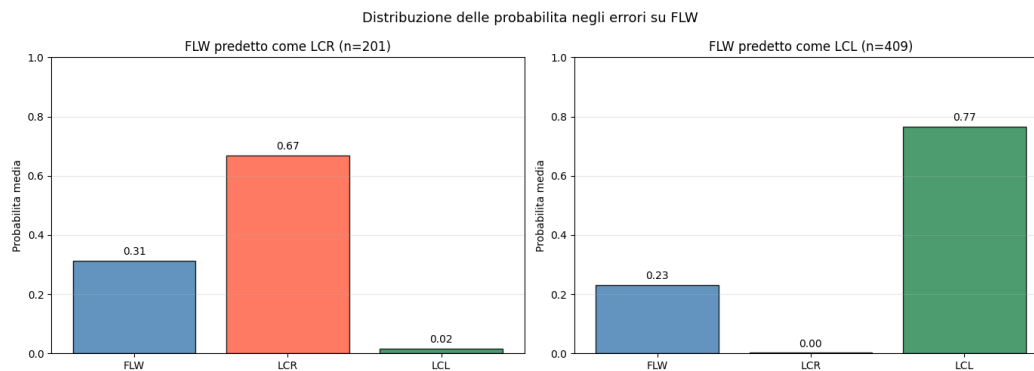


Figura 5.8: Distribuzione delle probabilità assegnate ai campioni FLW erroneamente predetti

L'analisi delle probabilità assegnate dal classificatore mostra che si tratta di errori commessi con una discreta convinzione: nei casi in cui un campione FLW viene predetto come LCR, la classe errata riceve una probabilità media di 0.65, che sale a 0.74 quando la predizione è LCL, mentre alla classe corretta restano rispettivamente 0.35 e 0.26. Non si tratta quindi di predizioni incerte al limite della soglia di decisione.

Dato che queste manovre sono classificate come cambio corsia invece di essere classificate come marcia in corsia, verranno utilizzate dai regressori corrispondenti. Fornendo in ingresso queste finestre ai regressori, ci si aspetterebbe di ottenere valori prossimi al limite superiore dell'intervallo, poiché si tratta di manovre che non si concluderanno mai.

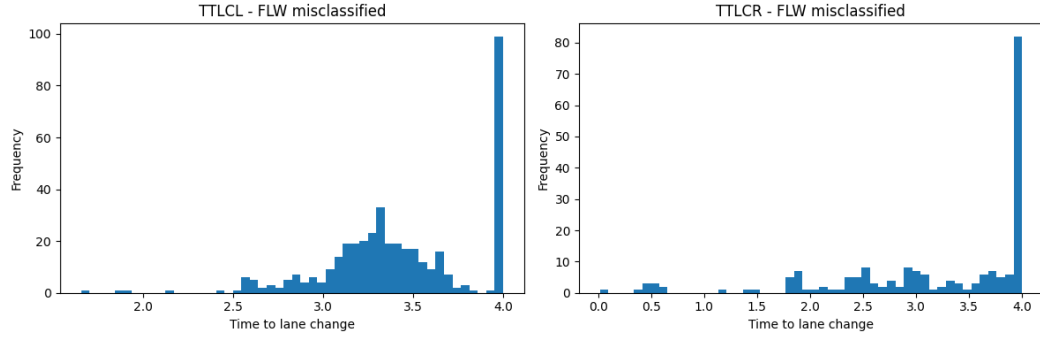


Figura 5.9: Distribuzione del TTLC predetto dai regressori sui campioni classificati

Applicando il regressore corrispondente alla classe erroneamente predetta, il TTLC stimato si distribuisce su valori lontani nel tempo, con una media di 3.43 s per i campioni predetti come LCL e di 3.22 s per quelli predetti come LCR. In entrambe le direzioni, i valori inferiori a 2 s sono rari e costituiscono una coda marginale della distribuzione, come evidenziato dalla Figura 5.9. Questo conferma l'ipotesi iniziale.

5.4.2.2 Cambi di corsia classificati come marcia in corsia

Il secondo tipo di errore corrisponde a una manovra non riconosciuta. Per individuare in quale fase della manovra si concentri, le Figure 5.10 e 5.11 rappresentano le traiettorie delle manovre, con il punto in cui il classificatore si accorge dell'intenzione di cambio corsia, ossia l'istante a partire dal quale la manovra viene riconosciuta. La linea tratteggiata verticale indica il valore medio.

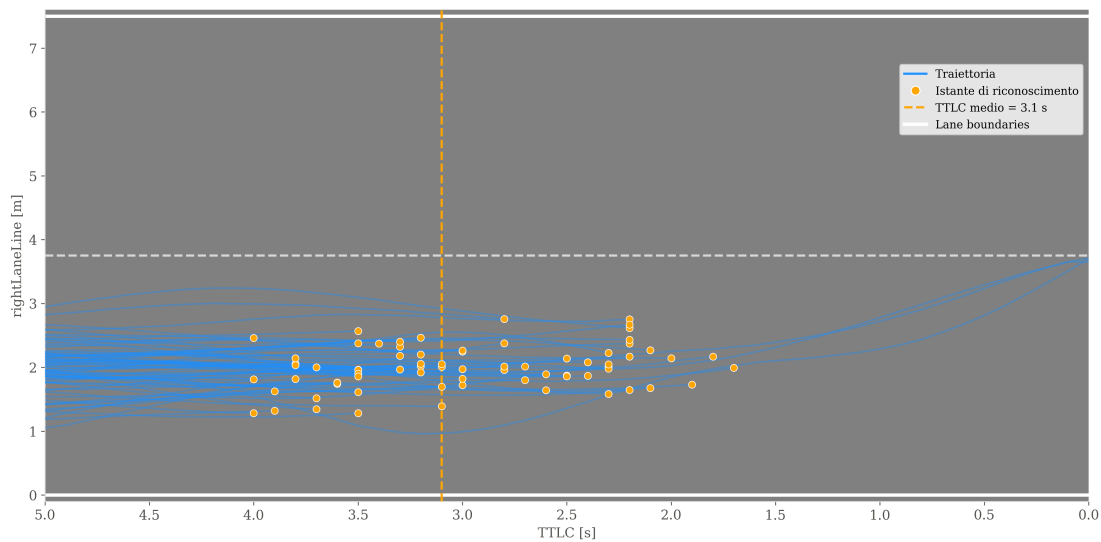


Figura 5.10: Traiettorie dei cambi di corsia verso sinistra non riconosciute

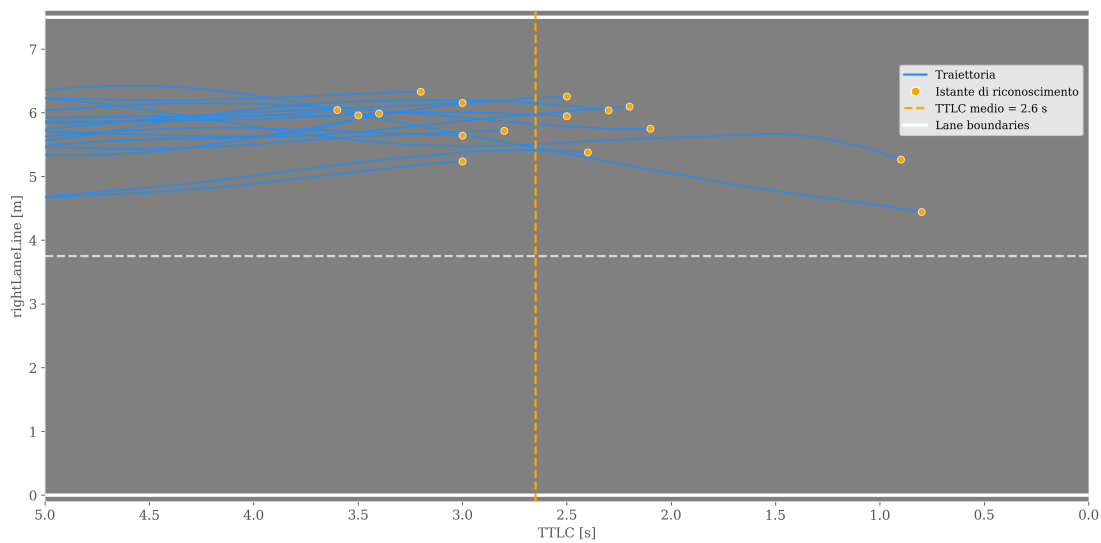


Figura 5.11: Traiettorie dei cambi di corsia verso destra non riconosciute

Il riconoscimento avviene in media a 3.1 s prima del completamento della manovra per i cambi verso sinistra e a 2.6 s per quelli verso destra. Nella maggior parte

dei casi, la natura delle traiettorie non fa presagire ad una possibile intenzione di cambio corsia. L'errore non è quindi imputabile a una limitazione del modello, ma all'assenza di un segnale cinematico che possa aiutare il classificatore.

Per quantificare il peso della componente iniziale, la valutazione è stata ripetuta escludendo i campioni con TTLC compreso tra 3.0 s e 4.0 s, ossia la fascia in cui la manovra è ancora lontana. Se la principale difficoltà del classificatore fosse effettivamente concentrata nelle fasi iniziali, la rimozione di tali campioni dovrebbe produrre un miglioramento delle prestazioni. I risultati sono riportati nella tabella 5.13 e nella figura 5.12.

Classe	Precision	Recall	F_1-score	Campioni
FLW	0.96	0.91	0.93	6087
LCR	0.97	0.98	0.97	6057
LCL	0.93	0.96	0.94	5493
Accuratezza		0.95		17637

Tabella 5.13: Prestazioni di classificazione escludendo i campioni con TTLC compreso tra 3.0 s e 4.0 s

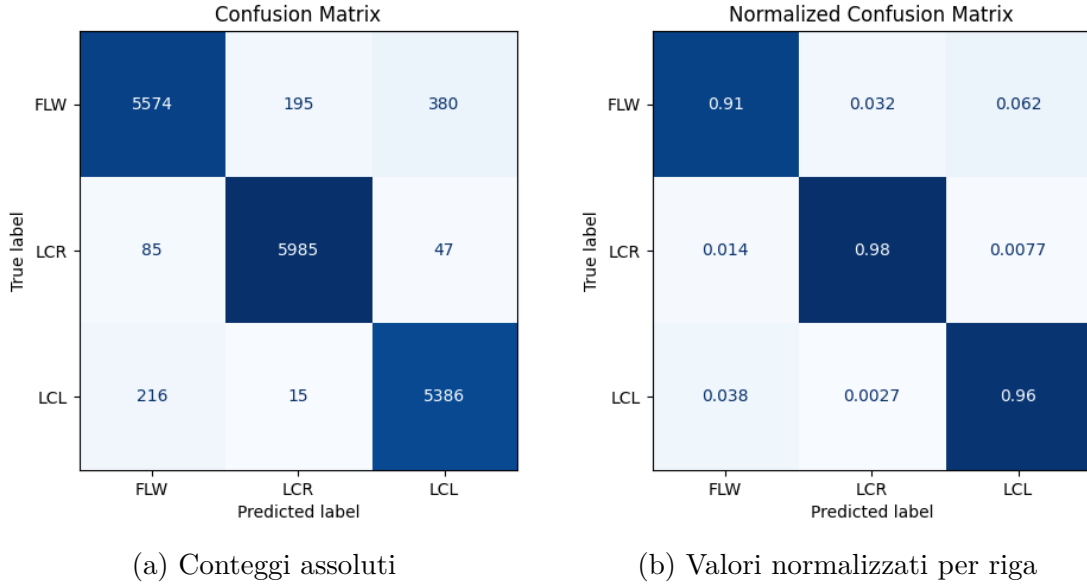


Figura 5.12: Matrice di confusione normalizzata, escludendo i campioni con TTLC tra 3.0 s e 4.0 s

L'esclusione dei campioni con TTLC compreso tra 3.0 s e 4.0 s convalidano l'ipotesi formulata nell'analisi precedente. I risultati confermano questa aspettativa: l'accuratezza passa dal 92% al 95% e il miglioramento risulta particolarmente evidente per la classe LCL. Il confronto mostra quindi che una parte significativa degli errori di classificazione è associata alla fase della manovra, in cui le informazioni cinematiche disponibili non sono ancora sufficientemente discriminanti.

5.5 Confronto con il framework di partenza

Il framework di partenza riporta un RMSE di 0.510 s e un MAE di 0.298 s per il suo modello migliore, un Transformer addestrato su tutte le manovre senza distinzione di direzione [19]. I regressori proposti in questa tesi ottengono valori di RMSE pari a 0.423 s per le manovre verso destra e 0.437 s per quelle verso sinistra, con una riduzione rispettivamente del 17% e del 14%.

Il confronto va però interpretato con prudenza, perché le condizioni sperimentali non sono equivalenti. In particolare:

- i dati utilizzati in questo lavoro sono sottoposti al filtraggio descritto nella Sezione 5.1, che rimuove le finestre fisicamente non plausibili;
- il framework di partenza includeva anche le finestre prive di intenzione di cambio corsia, etichettate al valore convenzionale di 4.1 s, mentre i regressori proposti operano esclusivamente su finestre che il classificatore ha già assegnato a una classe di manovra;
- la specializzazione per direzione rende la distribuzione dei dati di ciascun regressore più omogenea.

Il confronto evidenzia quindi un miglioramento delle prestazioni rispetto al framework di partenza, ma non consente di isolare il contributo dei singoli interventi. Il filtraggio modifica infatti la composizione del dataset, la classificazione introduce una selezione preliminare delle finestre e la separazione per direzione riduce ulteriormente l'eterogeneità del problema affrontato da ciascun regressore. Il risultato ottenuto deve pertanto essere interpretato come effetto complessivo della

nuova pipeline, piuttosto che come misura diretta del solo contributo dell'architettura di regressione. Per stabilire il peso relativo dei singoli fattori sarebbe necessario un confronto controllato, in cui le diverse modifiche vengano introdotte separatamente.

5.6 Discussione

Nel complesso, i risultati suggeriscono che le principali criticità del problema non sono riconducibili esclusivamente alla capacità espressiva delle architetture utilizzate. La sostanziale equivalenza delle prestazioni di classificazione e la ridotta differenza tra i regressori indicano infatti che, una volta definito il problema e fornita una rappresentazione cinematica della finestra, l'aumento della complessità del modello non produce necessariamente un miglioramento significativo.

Un ruolo più rilevante sembra invece essere svolto dalla qualità e dalla struttura dei dati. Il filtraggio elimina una quota consistente delle finestre originali, soprattutto per la classe FLW, mostrando che il dataset contiene dinamiche che non risultano compatibili con il comportamento cinematico atteso. La rimozione di questi campioni non costituisce quindi soltanto una fase di pulizia del dataset, ma modifica direttamente il dominio sul quale i modelli vengono addestrati.

Anche la scomposizione del problema in classificazione e regressione produce un risultato interessante. Il classificatore permette di separare in via preliminare la marcia in corsia dai cambi di corsia e, per questi ultimi, la regressione viene specializzata in funzione della direzione. Il miglioramento rispetto al framework di partenza deve tuttavia essere interpretato considerando congiuntamente queste modifiche, poiché le diverse componenti della pipeline agiscono contemporaneamente sulla distribuzione dei dati e sulla difficoltà del compito.

L'analisi degli errori individua infine un limite più fondamentale, comune ai due stadi della pipeline. Nelle fasi iniziali della manovra, quando il veicolo non ha ancora iniziato uno spostamento laterale evidente, la configurazione osservata può essere compatibile sia con una normale prosecuzione in corsia sia con un

cambio corsia che avrà luogo alcuni secondi più avanti. In questa situazione il problema non è semplicemente migliorare la precisione della rete, ma disporre di informazioni sufficienti a rendere osservabile l'intenzione futura del conducente. Da questo punto di vista, i risultati suggeriscono che un ulteriore miglioramento del sistema potrebbe dipendere più dalla rappresentazione dello stato del veicolo e dalla quantità di informazioni disponibili prima dell'avvio della manovra che dalla sostituzione dell'architettura utilizzata. L'analisi sperimentale indica quindi dove si collochi, per questo problema, il margine di miglioramento. Le quattro architetture confrontate producono risultati statisticamente indistinguibili su entrambi i compiti, mentre la qualità dei dati, la formulazione del problema e l'informazione contenuta nella finestra osservata incidono maggiormente sulle prestazioni ottenibili.

Capitolo 6

Conclusioni

6.1 Sintesi dei contributi

Questa tesi si è basata su un framework di simulazione open-source per la predizione del cambio di corsia [19], nel quale il tempo residuo alla manovra veniva stimato da un unico regressore addestrato su dati non sottoposti ad alcun controllo di plausibilità. Il lavoro è intervenuto su entrambi gli aspetti.

Il primo contributo riguarda la qualità dei dati. La procedura di filtraggio introdotta verifica, per ogni traiettoria, il rispetto di cinque vincoli di fattibilità cinematica derivati dalla dinamica del veicolo e scarta quelle che li violano. Il filtro rimuove circa il 16.9% delle manovre di cambio corsia e il 33.3% dei tratti di marcia in corsia. L'ispezione dei casi scartati ha mostrato che non si tratta di dinamiche semplicemente insolite: le due famiglie ricorrenti sono le perdite di controllo, riconoscibili da un andamento oscillatorio protratto per l'intera finestra, e gli eventi d'urto, in cui tutti gli indicatori presentano una discontinuità concentrata in uno o due campioni. In entrambi i casi, si tratta di traiettorie che nessun conducente produrrebbe su strada e che, in assenza di un controllo, sarebbero

state utilizzate per l'addestramento al pari delle manovre corrette.

Il secondo contributo riguarda la formulazione del problema. Al regressore unico è stata sostituita una pipeline a due stadi, composta da un classificatore a tre classi e da due regressori del TTLC specializzati sulle due direzioni di manovra. La scomposizione risponde a un limite segnalato in modo esplicito nel framework di partenza, in cui un solo modello doveva contemporaneamente riconoscere l'intenzione e stimarne l'orizzonte temporale. Per ciascuno dei tre modelli sono state confrontate quattro architetture per serie temporali, con le configurazioni individuate mediante ottimizzazione bayesiana degli iperparametri. Tutti i classificatori raggiungono un'accuratezza del 92%, con un F_1 -score compreso tra 0.96 e 0.97 sulla classe LCR e tra 0.90 e 0.92 sulla classe LCL; nella regressione il Transformer ottiene un RMSE di 0.423 s sulle manovre verso destra, mentre sulle manovre verso sinistra i migliori modelli si collocano attorno a 0.437 s, a fronte di 0.510 s del framework di partenza. Poiché gli scarti tra le tre architetture migliori sono dell'ordine di pochi millesimi di secondo, per il sistema finale è stato adottato il Transformer in entrambe le direzioni, privilegiando l'uniformità della pipeline rispetto a un vantaggio non significativo.

L'ultimo contributo è l'analisi degli errori. Quelli del regressore non si distribuiscono uniformemente lungo l'orizzonte di predizione, ma si concentrano nella fascia più lontana dalla manovra, dove le predizioni saturano attorno a 3.35 s e 3.15 s a seconda della direzione. Gli errori del classificatore seguono lo stesso andamento: escludendo i campioni con TTLC compreso tra 3.0 s e 4.0 s, l'accuratezza passa dal 92% al 95%. L'esame delle traiettorie individuali ne fornisce la spiegazione geometrica, mostrando che in quella fase lo spostamento laterale non si è ancora avviato e il veicolo è indistinguibile da uno che prosegue nella propria corsia.

6.2 Limiti

I risultati vanno letti alla luce di alcune limitazioni.

Il confronto con il framework di partenza non avviene a parità di condizioni. I dati impiegati in questo lavoro sono filtrati e i regressori sono specializzati per direzione; inoltre, il regressore del framework originale era valutato anche sulle finestre prive di intenzione, etichettate al valore convenzionale di 4.1 s, mentre i regressori proposti operano soltanto su finestre appartenenti a una classe di manovra, essendo il riconoscimento dell'intenzione affidato al classificatore. Il miglioramento osservato è quindi il risultato combinato di più interventi e non può essere attribuito a un singolo fattore. Uno studio di ablazione, in cui filtraggio e separazione per direzione vengono introdotti uno alla volta a parità di insieme di test, sarebbe necessario per quantificarne i contributi individuali.

I due stadi della pipeline, inoltre, sono inoltre stati valutati in modo indipendente. I regressori ricevono in ingresso finestre la cui classe è nota, anziché quella stimata dal classificatore, per cui le metriche riportate non tengono conto della propagazione degli errori tra i due stadi: nel sistema completo, una manovra attribuita alla direzione sbagliata verrebbe inoltrata al regressore non corrispondente. Data la quasi totale assenza di confusione tra LCL e LCR, inferiore all'1% in entrambe le direzioni, l'effetto è presumibilmente contenuto, ma non è stato misurato.

Il filtraggio si basa su soglie fissate a priori a partire da considerazioni sulla dinamica di un generico veicolo. La scelta garantisce interpretabilità, poiché per ogni traiettoria scartata è possibile risalire al vincolo violato.

I dati provengono interamente da simulazione. Per quanto le sessioni siano state registrate da conducenti reali, l'ambiente riproduce uno scenario semplificato, con

una carreggiata a due sole corsie, traffico generato proceduralmente e condizioni ambientali costanti. La trasferibilità dei modelli a dati acquisiti su strada resta da verificare.

6.3 Sviluppi futuri

La direzione più immediata riguarda il limite principale emerso dall'analisi degli errori, ossia la difficoltà nel riconoscere la manovra prima che essa si manifesti nella cinematica del veicolo. Poiché in questa fase l'unica informazione disponibile è il contesto di traffico, una possibile evoluzione consiste nel separare esplicitamente i due compiti: un rilevamento dell'intenzione a lungo termine, basato su posizioni e velocità relative dei veicoli circostanti e su azioni preparatorie del conducente, e un riconoscimento della manovra imminente, basato sulla dinamica del veicolo ego.

L'analisi dei falsi positivi suggerisce inoltre una strategia di mitigazione realizzabile senza modificare i modelli. Sui campioni di marcia in corsia erroneamente classificati come manovra, il TTLC stimato si concentra su valori elevati, con medie di 3.43 s e 3.22 s: una regola a valle della pipeline che tratti come poco affidabili le predizioni prossime al limite superiore dell'intervallo ridurrebbe l'incidenza dell'errore più oneroso per il sistema finale.

Sul fronte dei dati, l'estensione dello scenario a un numero maggiore di corsie e a densità di traffico più elevate permetterebbe di valutare la robustezza dei modelli in condizioni più realistiche, mentre la variabilità osservata tra conducenti suggerisce di esplorare forme di personalizzazione sui diversi stili di guida. La validazione su dati acquisiti su strada resta il passaggio necessario per confermare

i risultati ottenuti in simulazione.

Rimane infine aperta la validazione su piattaforma embedded. Le stime riportate precedentemente sono state ottenute per via simulata e riguardano i modelli presi singolarmente, mentre, un porting effettivo su microcontrollore permetterebbe di misurare la pipeline nel suo complesso. Nello specifico, consentirebbe di quantificare potenza ed energia assorbite, grandezze che lo strumento di stima non fornisce e che dipendono dal ciclo di lavoro effettivo del sistema: il classificatore è sempre attivo, mentre i regressori vengono attivati solo in presenza di una manovra. Il consumo medio viene quindi determinato dalla frequenza con cui i cambi corisa si verificano.

Bibliografia

- [1] J. Guo, U. Kurup, and M. Shah, “Is it safe to drive? An overview of factors, metrics, and datasets for driveability assessment in autonomous driving,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3135–3151, 2020.
- [2] X. Wang, M. Yang, and D. Hurwitz, “Analysis of cut-in behavior based on naturalistic driving data,” *Accident Analysis & Prevention*, vol. 124, pp. 127–137, 2019.
- [3] C. Bax, P. Leroy, and M. P. Hagenzieker, “Road safety knowledge and policy: A historical institutional analysis of The Netherlands,” *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 25, pp. 127–136, 2014.
- [4] M. S. R. Amin, A. Zareie, and L. E. Amador-Jiménez, “Climate change modeling and the weather-related road accidents in Canada,” *Transportation Research Part D: Transport and Environment*, vol. 32, pp. 171–183, 2014.
- [5] W. Zhang, Y. Qin, and X. Tao, “MMPHGCN: A hypergraph convolutional network for detection of driver intention on multimodal physiological signals,” *IEEE Signal Processing Letters*, vol. 31, pp. 1730–1734, 2024.

- [6] J. Cao, S. Liu, C. Wu, Y. Li, and S. Du, “ATHENA – autonomous vehicle trajectory planning considered human action awareness,” *IEEE Signal Processing Letters*, vol. 32, pp. 1845–1849, 2025.
- [7] Y. Xing *et al.*, “Driver lane change intention inference for intelligent vehicles: Framework, survey, and challenges,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4377–4390, 2019.
- [8] Y. Jeong, “Predictive lane change decision making using bidirectional long short-term memory for autonomous driving on highways,” *IEEE Access*, vol. 9, pp. 144985–144998, 2021.
- [9] M. Manzour, A. Ballardini, R. Izquierdo, and M. A. Sotelo, “Vehicle lane change prediction based on knowledge graph embeddings and Bayesian inference,” in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pp. 1893–1900, 2024.
- [10] S. Mozaffari, E. Arnold, M. Dianati, and S. Fallah, “Early lane change prediction for automated driving systems using multi-task attention-based convolutional neural networks,” *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 758–770, 2022.
- [11] F. Wirthmüller, M. Klimke, J. Schlechtriemen, J. Hipp, and M. Reichert, “Predicting the time until a vehicle changes the lane using LSTM-based recurrent neural networks,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2357–2364, 2021.

- [12] V. Alexiadis, J. Colyar, J. Halkias, R. Hranac, and G. McHale, “The next generation simulation program,” *Institute of Transportation Engineers Journal*, vol. 74, pp. 22–26, 2004.
- [13] V. Ramanishka, Y.-T. Chen, T. Misu, and K. Saenko, “Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning,” in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7699–7707, 2018.
- [14] A. Jain, H. S. Koppula, S. Soh, B. Raghavan, A. Singh, and A. Saxena, “Brain4Cars: Car that knows before you do via sensory-fusion deep learning architecture,” 2016. arXiv:1601.00740.
- [15] R. Krajewski, J. Bock, L. Kloecker, and L. Eckstein, “The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems,” in *Proc. 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2118–2125, 2018.
- [16] D. Chen, M. Zhu, H. Yang, X. Wang, and Y. Wang, “Data-driven traffic simulation: A comprehensive review,” *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 4, pp. 4730–4748, 2024.
- [17] Y. Wang, N. Lyu, C. Wu, Z. Du, M. Deng, and H. Wu, “Investigating the impact of HMI on drivers’ merging performance in intelligent connected vehicle environment,” *Accident Analysis & Prevention*, vol. 198, p. 107448, 2024.

- [18] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Proc. 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.
- [19] L. Forneris, R. Berta, M. Fresta, L. Lazzaroni, H. Rojhan, C. Oh, A. Pigghetti, H. Ballout, F. Tango, and F. Bellotti, “A deployment-oriented simulation framework for deep learning-based lane change prediction,” *IEEE Signal Processing Letters*, vol. 33, pp. 136–140, 2026.
- [20] L. Forneris *et al.*, “Lane change intention recognition dataset.” Zenodo, Aug. 2025.
- [21] L. Forneris *et al.*, “Setting up a lightweight simulation environment for automated driving dataset collection,” in *Applications in Electronics Pervading Industry, Environment and Society* (M. Ruo Roch, F. Bellotti, R. Berta, M. Martina, and P. Motto Ros, eds.), pp. 156–163, Cham: Springer Nature Switzerland, 2025.
- [22] M. Manzour, A. Ballardini, R. Izquierdo, and M. A. Sotelo, “Explainable lane change prediction for near-crash scenarios using knowledge graph embeddings and retrieval augmented generation,” in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pp. 1538–1545, 2025.
- [23] K. Liang, J. Wang, and A. Bhalerao, “Lane change classification and prediction with action recognition networks,” in *Computer Vision – ECCV 2022 Workshops*, (Cham), pp. 617–632, Springer Nature, 2023.

- [24] L. Yang, J. Zhang, N. Lyu, and Q. Zhao, “Predicting lane change maneuver and associated collision risks based on multi-task learning,” *Accident Analysis & Prevention*, vol. 209, p. 107830, 2025.
- [25] B. Osiński *et al.*, “CARLA real traffic scenarios – novel training ground and benchmark for autonomous driving,” 2020. arXiv:2012.11329.
- [26] B. Helou *et al.*, “The reasonable crowd: Towards evidence-based and interpretable models of driving behavior,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6708–6715, 2021.
- [27] N. Bao, A. Carballo, and K. Takeda, “Driving risk and intervention: Subjective risk lane change dataset,” in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pp. 556–562, 2022.
- [28] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” 2018. arXiv:1803.01271.
- [29] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [30] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

- [31] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
- [32] M. Fresta *et al.*, “Deep learning-based real-time driver cognitive distraction detection,” *IEEE Access*, vol. 13, pp. 26589–26607, 2025.
- [33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008, 2017.
- [34] M. M. G. Duarte and M. Sakr, “An experimental study of existing tools for outlier detection and cleaning in trajectories,” 2025. arXiv:2511.20139.
- [35] I. Bae, J. Moon, J. Jhung, H. Suk, T. Kim, H. Park, J. Cha, J. Kim, D. Kim, and S. Kim, “Self-driving like a human driver instead of a robocar: Personalized comfortable driving experience for autonomous vehicles,” in *Machine Learning for Autonomous Driving Workshop, 33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019.
- [36] M. Vignati and E. Sabbioni, “A cooperative control strategy for yaw rate and sideslip angle control combining torque vectoring with rear wheel steering,” *Vehicle System Dynamics*, 2022.
- [37] F. Wirthmüller, J. Schlechtriemen, J. Hipp, and M. Reichert, “Teaching vehicles to anticipate: A systematic study on probabilistic behavior prediction using large data sets,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 7129–7144, 2021.

- [38] H. Zhang, S. Gao, and Y. Guo, “Driver lane-changing intention recognition based on stacking ensemble learning in the connected environment: A driving simulator study,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 2, pp. 1503–1518, 2024.
- [39] A. Capello *et al.*, “Exploiting big data for experiment reporting: The Hi-Drive collaborative research project case,” *Sensors*, vol. 23, no. 18, p. 7866, 2023.
- [40] SAE International, “J3016_202104: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles.” https://www.sae.org/standards/content/j3016_202104/, 2021.
- [41] T. H. Itkonen, E. Lehtonen, and Selpi, “Characterisation of motorway driving style using naturalistic driving data,” *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 69, pp. 72–79, 2020.