

UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Ingegneria dell'Informazione ed
Elettrica e Matematica Applicata

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Triennale

**Progettazione di un Sistema V2X per la Gestione
Context-Aware di Dati Meteo e Traffico**

Un Approccio FuzzyCA per la Previsione del Rischio Stradale



Relatore

prof. Diego Gagnaniello

Candidato

Gennaro Francesco Landi

Mat. 0612707526

Tutor Aziendale

MinervaS

Ing. Ennio Andrea Adinolfi

ANNO ACCADEMICO 2024-2025

Ad Elettra

Abstract

Nell'ambito della mobilità intelligente e del supporto alla guida per veicoli industriali, questo elaborato affronta la sfida dell'acquisizione e gestione efficiente di dati provenienti dall'Open Data Hub del Trentino-Alto Adige, incluse condizioni meteo in tempo reale e informazioni dettagliate sul traffico autostradale. Il sistema sviluppato fornisce un flusso dati ottimizzato, integrabile nel co-pilota di MinervaS, con l'obiettivo di potenziare il monitoraggio dinamico dei veicoli ed offrire avvisi tempestivi e localizzati su incidenti, ostacoli e criticità stradali.

L'ampia diffusione dei veicoli connessi sta rivoluzionando i sistemi di assistenza alla guida (ADAS), permettendo di sfruttare dati esterni per elevare la sicurezza stradale. Studi recenti propongono l'impiego di intelligenza artificiale, inclusa la logica fuzzy, per gestire l'incertezza informativa nella valutazione del rischio. Questo lavoro si inserisce in tale contesto, integrando dati meteorologici e di traffico in una piattaforma V2X per la guida sicura, con l'obiettivo di potenziarne le funzionalità di sicurezza proattiva.

Il contributo principale consiste nella progettazione di ODHConnector, un connettore software dotato di un meccanismo di caching con TTL (Time-To-Live) e filtri spazio-temporali che permettono di selezionare eventi per prossimità e pertinenza, con l'obiettivo di ridurre la latenza e il carico sulle API. È stato inoltre concettualizzato l'approccio FuzzyCA (Fuzzy Context-Aware): un framework basato su logica fuzzy che integra variabili endogene (stile e ore di guida) a quelle esogene (meteo, traffico), per una valutazione del rischio più robusta e flessibile rispetto ai metodi tradizionali.

Per validare l'architettura e la pipeline proposta, un prototipo software utilizza l'ODHConnector per interrogare in tempo reale le API Open Data Hub. Le informazioni ottenute vengono filtrate, aggregate e rese disponibili attraverso una dashboard interattiva dotata di mappa. Quest'interfaccia visualizza in modo chiaro le condizioni ambientali correnti e gli eventi rilevanti lungo la tratta autostradale, dimostrando l'efficacia della pipeline dati nel fornire un supporto contestuale e tempestivo al conducente.

Indice

1	Introduzione	6
1.1	Problema affrontato	7
1.2	Obiettivi della tesi	8
2	Stato dell’arte e Analisi comparativa dei co-pilot attuali	10
2.1	Sistemi ADAS avanzati integrati dai produttori	10
2.2	App e piattaforme di navigazione mobile	11
2.3	Sistemi di monitoraggio del conducente (DMS) e supporto cognitivo . .	12
2.4	Sistemi per flotte e soluzioni telematiche	13
2.5	Situazione italiana	14
2.6	Visione MinervaS	14
3	ODHConnector: architettura, progettazione e contributi	16
3.1	Visione d’insieme dell’ecosistema	17
3.2	Requisiti e scelte di system design	19
3.2.1	Requisiti funzionali	19
3.2.2	Requisiti non funzionali	20
3.3	Architettura modulare di ODHConnector	20
3.4	Dataset Open Data Hub	21
3.4.1	Panoramica delle sorgenti meteo	21
3.4.2	Dataset eventi di traffico	23
3.4.3	Meccanismo di caching TTL	24
3.4.4	Filtri spazio-temporali	26
3.5	FuzzyCA: framework <i>Fuzzy Context-Aware</i>	28
3.5.1	Motivazione	28
3.5.2	Variabili linguistiche	29

3.6	Conclusioni del capitolo	30
3.6.1	Esempi applicativi e visualizzazioni	30
4	FuzzyCA: un Modulo Fuzzy Context-Aware per ADAS	33
4.1	Introduzione	33
4.2	Fondamenti della logica fuzzy	34
4.2.1	La pipeline di inferenza fuzzy	35
4.3	Vantaggi e confronti con altri approcci	36
4.4	Architettura e implementazione del modulo FuzzyCA	38
4.4.1	Introduzione	38
4.4.2	Variabili di input e output	38
4.4.3	Regole fuzzy del sistema FuzzyCA	40
4.4.4	Architettura del sistema	41
4.4.5	Vantaggi del modulo	41
4.4.6	Prototipazione e validazione futura	42
5	Conclusioni	44
5.1	Sintesi dei risultati del prototipo con ODHConnector	45
5.2	Analisi critica dei limiti dell'approccio attuale	45
5.3	Proposte di scalabilità nazionale	46
5.4	Confronto con lo stato dell'arte	46
5.5	Prospettive future	47
5.6	Conclusione generale	49
	Riferimenti bibliografici	51
	Ringraziamenti	54

Capitolo 1

Introduzione

Nel passaggio verso forme di mobilità sempre più *sostenibili*, il sistema dei trasporti deve riuscire a coniugare efficienza energetica, competitività economica e tutela dell'ambiente senza sacrificare la sicurezza degli utenti della strada. Sebbene l'evoluzione tecnologica abbia ridotto consumi e emissioni, la sicurezza stradale rimane una priorità irrisolta¹. Le cause di tali sinistri sono molteplici.

Da un lato, vi è il *fattore umano*: errori di guida, distrazioni, stanchezza e malori sono responsabili di gran parte degli incidenti. Secondo l'infografica ISTAT-ACI 2023[11], la *distrazione* del conducente è la principale causa singola, contribuendo a circa il **15 %** degli incidenti totali. La *stanchezza* del conducente è un fattore subdolo ma cruciale: guidare per molto tempo senza pause o in orari notturni aumenta esponenzialmente il rischio di colpi di sonno e riduce i tempi di reazione. Diversi studi indicano che *tra il 10 % e il 22 %* degli incidenti stradali è direttamente correlato alla fatica o al colpo di sonno di chi è al volante, percentuale che cresce sulle tratte autostradali a lungo raggio[2, 7].

Dall'altro lato, vi sono i *fattori ambientali e contestuali*: condizioni meteo avverse (pioggia intensa, neve, ghiaccio, nebbia, vento laterale), insieme a situazioni di traffico critiche (es. traffico intenso o incidenti già avvenuti sulla strada), possono creare pericolo anche per conducenti esperti. La visibilità ridotta, l'asfalto bagnato con minore aderenza e l'imprevedibilità di ostacoli o rallentamenti contribuiscono a incidenti a catena. Secondo la Federal Highway Administration, negli Stati Uniti circa il **47 %** degli incidenti meteo-correlati avviene durante precipitazioni e il **75 %** su asfalto bagnato[18].

¹Nel 2023, in Italia si sono registrati oltre 166 000 incidenti con lesioni e più di 3 000 vittime [10].

1.1 Problema affrontato

Di fronte a queste sfide, le tecnologie di *smart mobility* mirano a rendere il trasporto su strada più sicuro, efficiente e "intelligente" grazie all'uso di sensori, connettività e sistemi automatici di supporto alle decisioni umane. In particolare, i sistemi *ADAS* (Advanced Driver Assistance Systems) stanno diventando sempre più diffusi nei veicoli moderni. Gli ADAS includono funzionalità come il mantenimento automatico della corsia, la frenata d'emergenza assistita, il *cruise control* adattivo, il rilevamento della stanchezza del conducente, i sensori di angolo cieco, il riconoscimento dei segnali stradali, e così via. Queste tecnologie - rese possibili dai progressi in elettronica e IA - hanno l'obiettivo di assistere attivamente il guidatore, riducendo il carico di lavoro mentale e intervenendo in situazioni potenzialmente pericolose. Inoltre, dal 2022 la normativa UE richiede che molte di queste funzioni siano presenti di serie sui nuovi modelli di auto e veicoli commerciali, riconoscendo l'importanza di strumenti di assistenza alla guida per migliorare la sicurezza sulle strade.

Parallelamente alla sicurezza, la sostenibilità ambientale e l'efficienza operativa sono divenute priorità nel settore automotive e dei trasporti. Veicoli più sicuri spesso significano anche veicoli più efficienti: **evitare brusche frenate o incidenti non solo salva vite, ma riduce anche consumi di carburante e congestioni stradali.**

In questo scenario, si delinea la figura del *co-pilota digitale*: un sistema intelligente che, analogamente a un navigatore GPS avanzato, accompagna il conducente durante il viaggio fornendo non solo indicazioni stradali ma anche consigli proattivi su come condurre il mezzo in modo sicuro ed efficiente.

Sebbene l'attenzione dell'industria sia sempre più rivolta verso lo sviluppo di *veicoli a guida autonoma*, la realtà attuale - specie in contesti operativi complessi o in mercati con normative conservative - rende improbabile una diffusione capillare e completa nel breve termine. Inoltre i sistemi attuali risultano spesso incompleti o non del tutto affidabili in assenza di supporto contestuale esterno. In questo contesto, il co-pilota digitale non è una soluzione transitoria, ma un elemento chiave per migliorare l'interazione uomo-macchina, fornire dati ambientali aggiuntivi e contribuire, nel tempo, al perfezionamento dei futuri modelli di guida autonoma.

1.2 Obiettivi della tesi

L'idea del *co-pilot* è quella di avere un "secondo occhio" elettronico che monitori costantemente sia l'ambiente esterno (strada, meteo, traffico, infrastrutture) sia lo stato del veicolo e del conducente, integrando queste informazioni per supportare le decisioni in tempo reale. Un co-pilota efficace dovrebbe, ad esempio, avvisare il conducente di un incidente qualche chilometro più avanti sul percorso (suggerendo eventualmente un itinerario alternativo), raccomandare di rallentare se sta sopraggiungendo un banco di nebbia o se la strada diventa sdrucchiolare, oppure ricordare al conducente di fare una pausa se rileva segnali di affaticamento.

In sostanza, *si tratta di passare da un approccio **reattivo** - in cui l'ADAS interviene per correggere un errore o mitigare le conseguenze di un pericolo imminente - a un approccio **proattivo e predittivo***, in cui il sistema cerca di prevenire le situazioni di rischio adattando in anticipo la condotta di guida. Questo richiede la fusione di dati da molteplici fonti eterogenee e l'uso di algoritmi avanzati di analisi in tempo reale.

Negli ultimi anni, grazie alla diffusione di connessioni mobili veloci e di piattaforme *open data*, è diventato possibile arricchire il veicolo di informazioni provenienti dall'esterno. Il concetto di **V2X** (*Vehicle-to-Everything*) indica la capacità del veicolo di comunicare con l'infrastruttura stradale, con altri veicoli e con il cloud, scambiando dati utili a migliorare la sicurezza e l'efficienza. Ad esempio, tramite V2X un veicolo può ricevere dati sui limiti di velocità, sullo stato dei semafori, su incidenti segnalati qualche chilometro avanti o su condizioni meteo localizzate.

L'*Open Data Hub* dell'Alto Adige utilizzato in questo lavoro è un esempio di piattaforma che rende disponibili liberamente molte informazioni di questo tipo, in formato standard e aggiornate frequentemente. Sfruttando questi dati e integrandoli con quelli interni al veicolo si può realizzare una sorta di *gemello digitale* del viaggio, dove il sistema co-pilota ha una visione complessiva sia del contesto attuale sia di ciò che attende il veicolo sul tragitto programmato, consentendo decisioni più informate. Questa combinazione di ADAS + connettività + IA rientra pienamente nei trend della *smart mobility* del prossimo futuro.

Struttura del lavoro. Per guidare il lettore, di seguito si offre una rapida roadmap dei capitoli che seguono.

- **Capitolo 2 – Stato dell'arte.** Passa in rassegna le soluzioni di co-pilot già presenti

sul mercato (ADAS OEM, app di navigazione, DMS, piattaforme fleet-management) evidenziandone limiti e spazi di miglioramento.

- **Capitolo 3 – *ODHConnector*.** Descrive l’architettura modulare del connettore proposto, le scelte di system-design e i contributi tecnici (ingestion dei dataset Open Data Hub, caching-TTL, filtri spazio-temporali, dashboard prototipale).
- **Capitolo 4 – *FuzzyCA*.** Approfondisce il framework di inferenza fuzzy context-aware pensato per estendere l’intelligenza del co-pilota; vengono formalizzate variabili linguistiche, regole, architettura e potenziali vantaggi rispetto ad approcci differenti.
- **Capitolo 5 – Conclusioni e sviluppi futuri.** Riassume i risultati del prototipo, evidenzia i limiti attuali e traccia le prossime tappe: integrazione V2X, test su flotte reali, adozione di indicizzazioni spaziali (quadtree/R-tree) e computazione granulare per scalare a coperture nazionali.

Capitolo 2

Stato dell'arte e Analisi comparativa dei co-pilot attuali

Il futuro del trasporto su gomma tende verso la guida autonoma *full* (livelli SAE¹ 4-5), ma, nel frattempo, rimane cruciale supportare il conducente con sistemi di assistenza avanzata. Ad esempio, Wickramanayake et al.[21] propongono un framework di monitoraggio in tempo reale del comportamento di guida e feedback diretto al conducente per incentivare stili di guida più efficienti in termini di consumi. Questo sottolinea la necessità attuale di sistemi *co-pilot* che affianchino il guidatore, fornendo consigli o interventi real-time per migliorare sicurezza e comfort.

2.1 Sistemi ADAS avanzati integrati dai produttori

I moderni veicoli di serie integrano numerosi sistemi di assistenza alla guida (ADAS) di livello SAE 1-2, come Adaptive Cruise Control (ACC), mantenimento di corsia, frenata automatica d'emergenza, e integrazioni più evolute (es. lane centering). Esempi concreti includono Tesla Autopilot, Nissan ProPILOT Assist e altri. Questi sistemi utilizzano sensori (radar, lidar, telecamere) montati sul veicolo per effettuare correzioni sterzo-frenata in tempo reale pur richiedendo la supervisione del conducente.

¹SAE International ha definito uno standard[23] con sei livelli (da 0 a 5) per classificare il grado di automazione nei veicoli.

- **Limiti tecnici attuali:** Questi sistemi operano generalmente in condizioni di guida previste (strade ben marcate, traffico moderato, condizioni meteo regolari) e possono essere ingannati da strisce poco visibili o situazioni impreviste. Per esempio, il rapporto IIHS 2024[19] ha classificato "poco sicuri" sistemi come Tesla Autopilot, Full Self-Driving (FSD), Nissan ProPILOT e Volvo Pilot Assist, evidenziando come la tecnologia di automazione parziale debba ancora migliorare molti aspetti (sorveglianza del guidatore, procedure di emergenza) per garantire vera sicurezza.
- **Apertura a dati esterni/interoperabilità:** I sistemi integrati OEM (Original Equipment Manufacturer) sono generalmente soluzioni proprietarie chiuse; non permettono di importare dati esterni facilmente. Alcune case (es. Nissan ProPILOT con Navi-link) integrano dati di mappa per migliorare la guida, ma in linea generale i dati raccolti (dallo stesso veicolo o dalla telematica di bordo) restano confinati al sistema del costruttore.
- **Supporto in tempo reale:** Gli ADAS operano in real-time. Le decisioni di controllo (sterzo, frenata, accelerazione) sono calcolate nel millisecondo in base ai sensori. Il livello di integrazione hardware/software permette reazioni immediate (per es. frenata d'emergenza automatica). Tuttavia, il feedback al guidatore è principalmente di tipo passivo o di allarme (acustico/luminoso) piuttosto che consigli attivi sullo stile di guida; in molti casi il sistema si limita ad eseguire direttamente l'azione di assistenza (es. sterzata assistita), senza fornire spiegazioni attive al conducente.

2.2 App e piattaforme di navigazione mobile

Le applicazioni di navigazione su smartphone (es. Google Maps, Waze, Apple Maps, Here WeGo) costituiscono una forma di copilota digitale *light*, orientata alla navigazione e all'informazione sul traffico. Offrono mappe GPS, calcolo del percorso, informazioni sul traffico in tempo reale e avvisi di incidenti o pericoli.

- **Funzionalità principali:** Queste app usano dati storici e live (traffico, incidenti, sensori urbani) per ottimizzare il percorso. Google Maps raccoglie milioni di posizioni di telefono per stimare il traffico attuale; Waze invece si basa sul contributo degli utenti (crowdsourcing) che segnalano incidenti, ingorghi, autovelox, pericoli

sulla strada. In tempo reale ricalcolano la rotta se cambia la situazione del traffico e mostrano alert visuali e vocali (polizia, pericoli, tutor).

- **Limiti tecnici:** La principale limitazione è la dipendenza dal segnale dati cellulare: senza connettività adeguata le mappe offline sono limitate. Inoltre, la sicurezza attiva è modesta: queste app non possono agire sul veicolo, solo consigliare. Non sono integrate con i sensori del veicolo (come radar o telecamere) e non rilevano lo stile di guida interno. La loro accuratezza dipende dalla qualità dei dati crowdsourced (per Waze) o dalla rete di telecomunicazioni (per Google).
- **Apertura a dati esterni/interoperabilità:** Google Maps e Waze possono importare dati esterni come indirizzi o percorsi; dispongono di API che consentono ad applicazioni terze di ricevere dati di traffico. Tuttavia queste API sono controllate dai fornitori e non completamente aperte né gratuite.

2.3 Sistemi di monitoraggio del conducente (DMS) e supporto cognitivo

I driver monitoring systems (DMS) sono progettati per rilevare lo stato di attenzione, affaticamento o distrazione del conducente. Esempi concreti includono telecamere IR sul cruscotto (es. Toyota Safety Sense, Mercedes Distronic con Driver Monitoring), sensori sul volante (per rilevare la pressione delle mani, come fa Tesla), e sistemi basati su sensori fisiologici (trasformazioni EEG, frequenza respiratoria) sperimentali. Lo scopo è avvertire il guidatore in caso di sonnolenza o disattenzione e, nei casi più avanzati (condizionamento SAE 3), abilitare il takeover automatico.

- **Esempi concreti:** Numerosi veicoli premium includono DMS. Ad esempio Volvo ha sistemi di monitoraggio dello sguardo sul volante, Mercedes integra telecamere cabina in DrivePilot, Nissan implementa telecamere per il monitoraggio dell'attenzione. Esistono anche soluzioni aftermarket (es. assistenti di guida basati su visione artificiale) e ricerca accademica. Ad esempio Ayas et al.[3] riportano che nella letteratura gli interventi più studiati basati su DMS sono avvisi acustici o visivi rivolti al conducente. Alcuni studi sperimentali mostrano come l'allerta del sistema (es. vibrazione del sedile o segnale sonoro) riduca i livelli di sonnolenza e migliora la prestazione alla guida.

- **Limiti tecnici attuali:** I DMS devono bilanciare accuratezza e invasività. Le telecamere possono mal interpretare occhiali scuri o cappellini, e generano falsi allarmi in caso di brevi distrazioni. Inoltre, l'efficacia reale dei DMS nel lungo termine è ancora poco documentata. Problemi di privacy e accettazione da parte dell'utente (sorveglianza continua) possono limitarne la diffusione. Non esiste ancora un consenso sul miglior modo di fornire feedback: se solo avvisi acustici o takeover automatico. Gli studi indicano che sono necessari ulteriori esperimenti sull'*interazione uomo-macchina* per massimizzarne i benefici.
- **Supporto in tempo reale:** I DMS operano in tempo reale: elaborano in streaming le immagini del volto e lo stato degli occhi durante la guida, per inviare istantaneamente avvisi (beep, luci di allerta) o suggerimenti vocali. Sono supporti cognitivi in tempo reale, specialmente per prevenire micro-sonnolenza e mantenere alta l'attenzione. Alcuni sistemi avanzati possono persino preconditionare il veicolo (es. suggerire una pausa o regolare l'aria condizionata), e in caso di conferma del livello di pericolo possono richiamare direttamente il guidatore alla concentrazione.

2.4 Sistemi per flotte e soluzioni telematiche

Per la gestione di flotte aziendali e pubbliche vengono impiegati sistemi telematici dedicati (Fleet Management Systems), come quelli di Geotab con Vodafone Fleet. Questi sistemi combinano hardware installato sui veicoli (GPS e moduli connettività) con piattaforme cloud per monitorare e ottimizzare operazioni e stili di guida.

- **Limiti tecnici attuali:** I sistemi di flotta richiedono installazione di hardware su ogni veicolo (costo iniziale) e abbonamenti dati/canoni licenza. Spesso sono pensati per veicoli commerciali e non per utenti consumer. Possono soffrire di latenze di rete o di copertura ridotta in aree rurali. Dal punto di vista tecnico, il volume di dati trasmessi (funzionamento motore, telecamere video, V2X) è limitato dalla banda disponibile e dai costi di connettività. Inoltre, la standardizzazione dei formati è ancora frammentata: sebbene piattaforme come Geotab offrano API aperte (oltre 200 add-on disponibili), non esiste uno standard unico di interoperabilità tra produttori telematici diversi.

2.5 Situazione italiana

In Italia sono in corso diverse iniziative nel settore della mobilità intelligente. Sul mercato italiano coesistono i medesimi sistemi internazionali di cui sopra, installati su veicoli Fiat, Alfa Romeo, BMW, Mercedes, ecc., con aggiornamenti normativi specifici (per es. guida autonoma sperimentale su strade urbane).

Fra i progetti accademici e aziendali si segnala il progetto *Smart Road* di ENEA[14]: un tratto di strada sperimentale di circa mezzo chilometro a Roma, dotato di lampioni intelligenti interconnessi e un veicolo elettrico autonomo. I "lampioni hi-tech" raccolgono dati ambientali e di traffico e comunicano bidirezionalmente con il veicolo tramite un sistema centralizzato CIPCast, fornendo mappe real-time di rischio e consigli alla vettura su come comportarsi (es. variazioni di traffico, calamità meteorologiche).

Anche enti come ENAV (gestore controllo volo) e altre istituzioni per la mobilità intelligente collaborano a progetti di ricerca su droni o EAV (Advanced Air Mobility), portando competenze sull'integrazione di sensori e telematica. Sul fronte privato, startup come "Social Self Driving" (fondata dall'ex ingegnere Ferrari Luigi Mazzola) stanno sviluppando soluzioni di guida autonoma e personalizzazione dello stile: in questo caso, l'innovazione è un software che registra lo stile di guida di un utente e lo riproduce su veicoli autonomi.

Per quanto riguarda gli open data, la Regione Trentino-Alto Adige promuove l'Open Data Hub (insieme con altre province EU) che mette a disposizione numerosi dataset di mobilità (traffico urbano, corsi TPL, occupazione parcheggi, sensoristica ambientale). In particolare, è possibile ottenere dati in tempo reale o storici su flussi veicolari e velocità medie per tratto urbano. Tali dati sono utilizzati da sperimentazioni per arricchire gli algoritmi di guida con informazioni ambientali e di traffico esterne.

2.6 Visione MinervaS

La proposta innovativa sviluppata dall'azienda MinervaS si fonda su un'architettura di tipo plug&play, pensata per integrare in modo agile e non invasivo i dati provenienti dal veicolo con fonti informative esterne. L'obiettivo è fornire suggerimenti personalizzati per una guida più sicura ed efficiente, sfruttando tecnologie di comunicazione avanzate e logiche fuzzy.

Il sistema si basa su un dispositivo OBD-II², collegato al bus CAN del veicolo. Questo consente l'acquisizione di parametri chiave come velocità, stato del motore, segnali dei sistemi ADAS e dati di accelerazione utili alla classificazione dello stile di guida. A supporto, un modulo di connettività wireless (4G/5G) riceve dati V2X (Vehicle-to-Everything) provenienti da infrastrutture connesse, altri veicoli e sensori ambientali.

La struttura modulare dell'architettura MinervaS consente di implementare questa tecnologia su veicoli di qualsiasi marca senza la necessità di interventi invasivi, garantendo così la massima compatibilità e flessibilità operativa.

L'iniziativa si avvale inoltre di collaborazioni strategiche con realtà di riferimento nella telematica e nella connettività: tramite Geotab, viene utilizzata un'infrastruttura aperta per la raccolta sicura dei dati di flotta; con Vodafone, vengono integrate soluzioni IoT mobili che abilitano un flusso costante di informazioni, sfruttabili per l'analisi dei big data orientata alla sicurezza e all'efficienza dei consumi.

Di particolare rilievo è l'integrazione con l'ecosistema Open Data Hub del Trentino-Alto Adige, che fornisce dataset in tempo reale relativi a condizioni meteorologiche e di traffico, caratterizzati da elevata granularità e frequenti aggiornamenti. Le API REST esposte, insieme a interfacce grafiche accessibili, consentono una rapida incorporazione dei dati nel sistema.

In sintesi, questa visione porta alla creazione di un ecosistema digitale collaborativo per la mobilità intelligente, volto a incrementare la sicurezza su strada e l'efficienza energetica, grazie all'integrazione dinamica di dati e all'uso di tecniche computazionali avanzate.

²OBD-II (On-Board Diagnostics II) è uno standard internazionale per la diagnosi elettronica dei veicoli, che consente di accedere a parametri e codici di errore tramite una porta dedicata.

Capitolo 3

ODHConnector: architettura, progettazione e contributi

In questo capitolo viene presentato in dettaglio il **cuore tecnologico** del lavoro di tesi: *ODHConnector*, un modulo software progettato per abilitare la piattaforma MinervaS all'utilizzo dinamico e intelligente dei dati ambientali e stradali pubblicati in tempo reale dall'infrastruttura *Open Data Hub (ODH)* della *Provincia autonoma di Trento*.

ODHConnector rappresenta il ponte tra l'ecosistema cloud-based di ODH e il sistema embedded di co-pilota digitale sviluppato da MinervaS. La sua funzione principale è quella di estrarre, filtrare, normalizzare e mettere a disposizione in modo efficiente i dati esterni rilevanti - come meteo, traffico, incidenti, cantieri e condizioni di guida - per potenziare la consapevolezza situazionale del veicolo e migliorare la sicurezza operativa in tempo reale.

Dopo una panoramica dell'architettura generale, verranno illustrate le principali scelte di *system design*, con particolare attenzione al:

- disaccoppiamento tra accesso ai dati remoti e logica applicativa, attraverso una struttura modulare a componenti adattatori (ad es. *WeatherAdapter*, *TrafficAdapter*);
- l'adozione di un meccanismo di **caching TTL** per ridurre il carico computazionale e l'uso della rete;
- l'integrazione di **filtri spazio-temporali** per la selezione mirata di eventi rilevanti in prossimità del veicolo.

Un paragrafo specifico è dedicato alla **commentazione dei dataset ODH** più rilevanti per il sistema: saranno analizzati i dati meteorologici in tempo reale, le sorgenti di informazioni stradali (incidenti, cantieri, lavori in corso) e i metadati associati, evidenziando i campi utilizzati, i formati di risposta e le strategie adottate per gestire dati incompleti o non disponibili.

Viene inoltre commentato il modulo *FuzzyCA* (Fuzzy Context-Aware), utilizzato per integrare e interpretare i dati ambientali e comportamentali in un *indice di rischio* continuo. Sebbene la descrizione teorica ed esemplificativa del funzionamento di FuzzyCA sia rinviata al Capitolo 4, in questo viene mostrata l'integrazione diretta tra dati ambientali e inferenza fuzzy nel sistema complessivo.

3.1 Visione d'insieme dell'ecosistema

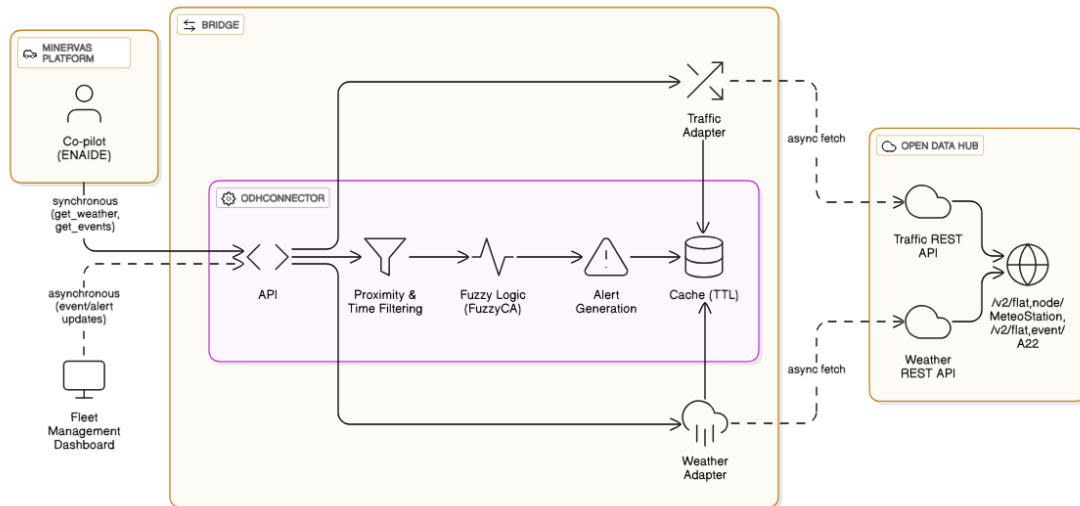


Figura 3.1: Architettura a tre domini: MinervaS Core, ODHConnector e Open Data Hub; con pull sincrono, fetch asincrono, fuzzy logic, alert e cache TTL.

Il sistema progettato si basa su un modulo software denominato **ODHConnector**, concepito come ponte tra il copilota digitale *ENAIDE*, sviluppato da MinervaS, e le API pubbliche di *Open Data Hub* (ODH), focalizzandosi in particolare sul tratto autostradale A22 in Trentino. L'obiettivo principale del modulo è raccogliere, filtrare e integrare dati ambientali e di traffico in tempo quasi reale, per poi fornirli, in forma normalizzata, al sistema di assistenza alla guida.

Dal punto di vista implementativo, ODHConnector è una classe pubblica istanziabile, che funge da punto di accesso centralizzato e da orchestratore dell'intero flusso informativo. Essa si interfaccia con due adapter modulari (WeatherAdapter e TrafficAdapter), incaricati di acquisire i dati dalle rispettive API REST di ODH e di convertirli da formato JSON in oggetti Python tipizzati (ad esempio Incident, WeatherIndex, ecc.). Dopo il recupero, i dati vengono sottoposti a filtri spazio-temporali, elaborati tramite un motore fuzzy (FuzzyCA) e, se necessario, trasformati in alert contestuali. I risultati sono infine memorizzati in una cache con TTL per ottimizzare la latenza delle risposte successive.

L'architettura (Figura 3.1) si articola in tre domini funzionali:

1. MinervaS Core

- *ENAIDE*, il co-pilota a bordo del veicolo, che riceve via pull i dati meteo e gli eventi di rischio e restituisce in tempo reale raccomandazioni di guida.
- Dashboard di *fleet management* per il monitoraggio e la configurazione del servizio da remoto.

2. ODHConnector

- Espone internamente delle API per popolare i moduli previsionali di ENAIDE.
- Applica filtri di prossimità spaziale e temporale sui dati grezzi.
- Esegue il motore di logica fuzzy context-aware (FuzzyCA) per valutare il livello di rischio.
- Notifica gli *alert* al co-pilota non appena emergono condizioni critiche.
- Memorizza i risultati in una cache con TTL per ridurre latenza e carico sulle API esterne.

3. Open Data Hub (Cloud)

- Piattaforma che rende disponibili, tramite un approccio "API-first", dataset eterogenei (*weather*, *traffic*, ...), approfonditi nella sezione 3.4.
- Offre endpoint REST come per il meteo osservato e dati di traffico.

Flussi di dati principali:

- *Pull sincrónico* (linee continue) da ENAIDE verso ODHConnector.

- *Fetch asincrono* (linee tratteggiate) dai due adapter interni (TrafficAdapter, WeatherAdapter) verso le API di ODH, per aggiornamenti in background.
- *Push notifiche* da ODHConnector verso ENAIDE e la dashboard di back-office, nel momento in cui vengono generati nuovi alert.

3.2 Requisiti e scelte di system design

Il sistema sviluppato si configura come un modulo di supporto intelligente alla guida, integrato all'interno di una pipeline software per la gestione e la diffusione di alert contestuali su veicoli industriali. In questa sezione vengono analizzati i requisiti funzionali e non funzionali che hanno guidato la progettazione, nonché le principali scelte architetturali effettuate durante lo sviluppo.

In particolare, è stata adottata una struttura modulare secondo il principio di *package-by-responsibility*¹, suddividendo logicamente i compiti tra adapter specializzati (per traffico e meteo) e il nucleo operativo dell'ODHConnector, responsabile della logica applicativa vera e propria (filtraggio, inferenza fuzzy, caching e generazione degli alert). La separazione delle responsabilità consente di garantire manutenibilità e testabilità del sistema, facilitando al contempo la futura estensione ad altri domini informativi. L'intero flusso dati - dalla raccolta asincrona via API REST, al filtraggio spaziale e temporale, fino alla produzione di raccomandazioni spiegabili - è stato progettato per operare in tempo quasi reale, minimizzando il carico computazionale sui veicoli e massimizzando la qualità delle informazioni distribuite al copilota e alla dashboard di flotta.

3.2.1 Requisiti funzionali

- **Data fusion** traffico-meteo in tempo quasi reale.
- **Filtraggio dinamico** per prossimità ($d \leq r$) e finestra temporale scorrevole (Δt).
- **Caching TTL** configurabile per ridurre chiamate ridondanti verso ODH.
- **Spiegabilità** delle raccomandazioni → introduzione di *FuzzyCA*.

¹Il principio di *package-by-responsibility* applica il *Single Responsibility Principle* ai moduli, organizzando il software in pacchetti ciascuno responsabile di una funzione specifica. Favorisce coesione, basso accoppiamento e facilita manutenzione e test.

3.2.2 Requisiti non funzionali

Scalabilità capacità di servire molte richieste contemporaneamente senza bloccarsi

Robustezza capacità di resistere a errori temporanei e recuperare in autonomia:

- *circuit breaker*: blocca temporaneamente le chiamate a un servizio guasto
- utilizzo di tecniche per gestire errori o mancate risposte, come il *retry con backoff* progressivo (es. exponential backoff) o strategie adattive come *AIMD* (Additive Increase, Multiplicative Decrease)
- unit ed integration test con `pytest`

Observability facilità di monitoraggio e diagnosi in produzione:

- log strutturati in JSON (ci permettono di filtrare e cercare facilmente gli eventi)
- metrica di *cache hit-ratio* (percentuale di volte in cui i dati vengono presi dalla cache anziché rifare la chiamata all'API)

3.3 Architettura modulare di ODHConnector

Per ricapitolare, il modulo ODHConnector è stato rifattorizzato secondo il principio di *package-by-responsibility*, al fine di separare chiaramente l'accesso alle API ODH, la normalizzazione dei dati e la logica applicativa (Figura 3.1).

Di seguito si descrivono i principali moduli:

models.py (strato *Domain*) Contiene tutte le `dataclass` tipizzate che modellano il dominio, come `Event`, `Incident`, `WorkZone`, `WeatherIndex` e `Alert`. Ogni classe è immutabile e comparabile per identità (UUID), il che semplifica le operazioni di caching e deduplicazione.

connector.py (facade *Application*) Espone metodi ad alto livello (`get_incidents()`, `get_weather_index()`, `generate_alerts()`), implementa una cache TTL, integra il motore fuzzy e restituisce oggetti `Alert` completi di rilevanza e spiegazione.

adapters/ Contiene i componenti che interagiscono con Open Data Hub, convertendo le risposte JSON degli endpoint REST nel modello interno.

WeatherAdapter Incapsula la complessità degli endpoint *mobility* e *tourism*:

- **Query parametrica:** costruisce la rotta REST combinando *stationTypes* e *dataTypes* dinamici.
- **Filtro di prossimità:** seleziona a livello applicativo solo le misure entro un raggio configurabile (*radius_m*, default 10 km) dal GPS del veicolo.
- **Aggregazione temporale:** riduce osservazioni multiple a un indice sintetico *WeatherIndex*, semplificando l'uso da parte del motore *FuzzyCA*.
- **Fallback automatico:** tenta prima la misura *real-time* dal dominio *mobility*; se non disponibile effettua fallback sulle previsioni distrettuali (*tourism*).

TrafficAdapter Implementa una strategia *pull* che massimizza copertura e robustezza. Ogni chiamata interroga l'endpoint più aggiornato di Open Data Hub per ottenere eventi di traffico lungo l'A22 (incidenti, lavori in corso, eventi speciali).

risk/ Contiene il motore *FuzzyCA*. Il connettore popola gli *antecedent* e legge l'output *attention_score* per ordinare e filtrare gli alert (sezione 3.5).

utils.py Funzioni di supporto, fra cui *haversine()* per il calcolo della distanza tra coordinate WGS-84², riutilizzata sia negli adapter sia nel connettore.

3.4 Dataset Open Data Hub

In questa sezione vengono illustrati i principali *dataset* messi a disposizione da **Open Data Hub (ODH)** e viene descritto il meccanismo attraverso cui i moduli **WeatherAdapter** e **TrafficAdapter** ne effettuano l'estrazione, la normalizzazione e l'integrazione nel sistema MinervaS tramite l'interfaccia unificata fornita da **ODHConnector**.

3.4.1 Panoramica delle sorgenti meteo

I dati meteorologici utilizzati provengono da due domini informativi distinti all'interno di ODH: *mobility*, che fornisce misure ambientali osservate in tempo reale da sensori

²WGS-84 è lo standard geodetico internazionale per latitudine e longitudine.

distribuiti, e *tourism*, che raccoglie previsioni aggregate su base giornaliera o oraria. La distinzione tra i due domini è rilevante sia a livello semantico che tecnico.

Il dominio *mobility* è caratterizzato da un'elevata flessibilità e granularità. Gli endpoint seguono una struttura parametrica, che consente query altamente personalizzabili:

`/v2/{representation}/{stationTypes}/{dataTypes}/{from}/{to}`

Grazie a questa architettura, è possibile accedere non solo alle osservazioni correnti (*/latest*), ma anche a misure storiche, metadati delle stazioni (*/metadata*), o flussi dati strutturati (formato *flat*, *node*, *series*, ecc.). Questi dataset sono pensati per applicazioni tecniche e analitiche, come sistemi predittivi o dashboard dinamiche. Rientrano in questa categoria i primi due dataset integrati nel sistema: *Weather Department by South Tyrol and Trentino* e *Real-Time Weather*.

Il dominio *tourism*, invece, si rivolge ad applicazioni più orientate all'utente finale (es. turismo, mobilità dolce) e propone dati aggregati e facilmente interpretabili. Gli endpoint sono stabili e non personalizzabili, come */v1/Weather/District* e */v1/Weather/Forecast*, e offrono previsioni localizzate aggiornate una volta al giorno. Ne fanno parte i dataset *Weather Forecast by District* e *Weather Forecast by Municipality*, che forniscono previsioni meteorologiche ufficiali rispettivamente su scala distrettuale e comunale

Indipendentemente dalla sorgente, tutte le informazioni raccolte vengono trasformate in un formato unificato all'interno di ODHConnector. Il modulo **WeatherAdapter** si occupa di mappare i dati grezzi eterogenei in oggetti standardizzati di tipo *WeatherIndex*, che rappresentano un profilo sintetico delle condizioni meteorologiche rilevanti per la guida (temperatura, pioggia, visibilità, rischio gelo). Questo processo si basa su una normalizzazione lineare dei valori fisici secondo soglie prestabilite, riportati in un intervallo $[0,1]$, tramite la formula:

$$x_{\text{norm}} = \min \left(1, \max \left(0, \frac{x - x_{\min}}{x_{\max} - x_{\min}} \right) \right)$$

I range di riferimento codificati sono:

- *air-temperature*: da -20°C a $+40^{\circ}\text{C}$
- *precipitation-rate*: da 0 a 50 mm/h

- **visibility:** da 0 a 10.000 m

Parametri binari, come il rischio gelo, sono derivati simbolicamente (ad esempio: temperatura < 0°C \Rightarrow rischio = 1). Il risultato è un oggetto coerente, interpretabile e utilizzabile dal motore fuzzy per valutazioni in tempo reale.

Di seguito si elencano i principali dataset meteorologici attualmente integrati:

- **Weather Department by South Tyrol and Trentino:** fornisce misure osservate (non previsionali) provenienti da oltre 250 stazioni meteorologiche gestite dalle Province autonome e da EURAC Research. È accessibile tramite API REST modulare ed è altamente personalizzabile. I dati includono temperatura, umidità, vento, pressione, neve e radiazione solare, con aggiornamenti quotidiani.
- **Real-Time Weather:** progettato per la reattività in tempo reale, fornisce telemetrie aggiornate ogni 5-10 minuti da stazioni locali attive nella Provincia di Bolzano. È ideale per scenari predittivi o per modelli AI a bordo veicolo.
- **Weather Forecast by District:** raccoglie previsioni giornaliere aggregate per i 7 distretti dell'Alto Adige, aggiornate dal Servizio Meteo provinciale. Ogni previsione è suddivisa in 4 fasce orarie ed è accessibile tramite endpoint stabile. I dati sono localizzati per lingua e distretto.
- **Weather Forecast by Municipality:** fornisce previsioni dettagliate a intervalli di 3 ore per oltre 110 comuni. È utile per scenari in cui la guida deve adattarsi alle condizioni previste su tratti precisi del percorso. Le previsioni sono espresse in linguaggio naturale, associate a coordinate e compatibili con sistemi GIS.

3.4.2 Dataset eventi di traffico

Per quanto riguarda le condizioni del traffico, Open Data Hub espone un'unica sorgente nel dominio *mobility*, accessibile tramite l'endpoint:

`/v2/flat,event/{origin}`

Il parametro *origin* specifica la provenienza degli eventi stradali. Nell'ambito di questo progetto, le fonti di maggiore interesse sono A22 (che copre l'Autostrada del Brennero,

tratto Brennero-Modena) e PROVINCE_BZ, che raccoglie eventi relativi alla rete provinciale del territorio di Bolzano. Il formato `flat,event` consente la consultazione semplificata dei dati, con supporto alla paginazione e all'endpoint `/latest`, utile per ottenere solo l'ultima versione di ciascun evento.

Il recupero e la gestione dei dati di traffico sono affidati al modulo **TrafficAdapter**, il cui metodo principale `fetch_events()` effettua richieste successive all'API per coprire l'intero dataset disponibile. La logica di deduplicazione si basa sull'identificativo univoco `evuuid`, associato a ogni evento.

```
1 url = f"{base_url}/v2/flat,event/{origin}/latest"
2 while True:
3     rows = requests.get(url,
4                           params=dict(limit=200,
5                                       offset=offset)).json()["data"]
6     if not rows:
7         break
8     for r in rows:                # deduplica su evuuid
9         if r["evuuid"] not in seen:
10            events.append(_row_to_model(r))
11 offset += len(rows)
```

Listing 3.1: Estratto essenziale di `fetch_events()`

La funzione effettua una paginazione incrementale, specificando i parametri `limit` e `offset` a ogni richiesta. La condizione `if r["evuuid"] not in seen` svolge un ruolo cruciale: evita che lo stesso evento venga elaborato più volte, anche in presenza di richieste ripetute o aggiornamenti asincroni dei dati lato server.

Gli eventi validi vengono poi trasformati in istanze di oggetti dominio, come `Incident` o `Event`, tramite la funzione ausiliaria `_row_to_model()`. Questa conversione consente di lavorare con strutture tipizzate e uniformi all'interno del sistema. L'intero processo è concepito per essere robusto, tollerante a errori temporanei di rete e adatto a un'esecuzione periodica o schedulata in ambiente di produzione.

3.4.3 Meccanismo di caching TTL

Per evitare chiamate ridondanti verso gli endpoint REST di Open Data Hub e ridurre la latenza percepita dai moduli che interrogano ODHConnector, il sistema adotta un

meccanismo di caching in-memory basato su time-to-live (TTL). Questo approccio è particolarmente utile per dati che, pur aggiornandosi frequentemente, non richiedono precisione al secondo, come previsioni meteo o segnalazioni stradali.

Il *cache layer* è integrato direttamente nel connector, e mantiene in memoria l'ultima risposta ricevuta da ciascun endpoint, associandola a un timestamp di scadenza. L'implementazione segue il pattern **write-through, lazy-refresh**:

- all'atto della richiesta di un qualunque dato pubblico (`get_events()`, `get_weather()` ...) il connettore invoca la routine privata `_maybe_refresh()`;
- se il flag `auto_refresh` è disabilitato, il metodo esce immediatamente e la logica di aggiornamento rimane in carico allo *scheduler* dell'applicazione;
- altrimenti vengono verificate due condizioni:
 1. cache vuota (`_last_refresh is None`) o
 2. scadenza del TTL (`now() - _last_refresh > _TTL`).

Al primo *boolean true* scatta `refresh_data()`, che scarica simultaneamente *eventi traffico* e *meteo* e li inserisce nel dizionario `self._cache`.

Il timestamp dell'ultimo aggiornamento viene registrato in `_last_refresh`, consentendo di calcolare rapidamente gli hit/miss senza dover ispezionare il contenuto del cache.

```
1 def _maybe_refresh(self) -> None:
2     """Aggiorna la cache se scaduta o assente."""
3     if not self.auto_refresh:      # caching disabilitato ->
4         no-op                      no-op
5         return
6     now = datetime.now(timezone.utc)
7     expired = (
8         self._last_refresh is None or
9         now - self._last_refresh > self._TTL
10    )
11    if expired:                      # TTL superato -> refresh
12        self.refresh_data()
```

Listing 3.2: Verifica del TTL prima di ogni richiesta

Questo schema consente di bilanciare efficienza e freschezza dei dati, riducendo il carico sulle API pubbliche ODH e migliorando la reattività complessiva del sistema, soprattutto in presenza di interrogazioni multiple o di richieste provenienti da moduli diversi (dashboard, co-pilota, sistemi di allerta).

Considerazioni di performance

1. **Riduzione del traffico di rete.** Con una frequenza di polling dei sensori pari a $\tau_{\text{poll}} = 5 \text{ s}$ ($f_{\text{poll}} = 0.2 \text{ Hz}$) e un TTL della cache di $\tau_{\text{TTL}} = 30 \text{ s}$, il numero di richieste "grezze" che arriverebbero al backend ODH in un intervallo pari al TTL sarebbe $N_{\text{raw}} = \tau_{\text{TTL}} / \tau_{\text{poll}} = 30 / 5 = 6$. Grazie alla cache solo la *prima* di queste sei scatena realmente la REST-call, con un fattore di abbattimento:

$$\eta = 1 - \frac{1}{N_{\text{raw}}} = 1 - \frac{1}{6} = \frac{5}{6} \approx 83,3\%.$$

2. **Latenza costante e bassa.** L'accesso al dizionario Python ha complessità $O(1)$ (decine di ns), mentre una round-trip HTTPS varia tipicamente fra 50 e 150 ms; il *cold-start* avviene una sola volta per ciclo TTL.
3. **Protezione da throttling & quote.** Molti servizi *api* impongono limiti di N richieste/minuto. La cache mantiene il connettore ampiamente sotto soglia, lasciando margine per *retry* o chiamate eccezionali.
4. **Configurabilità.** Il TTL è passabile al costruttore (`ttl_seconds`): valori piccoli ($< 5 \text{ s}$) privilegiano la freschezza dei dati; valori maggiori ($\approx 5 \text{ min}$) massimizzano il risparmio di banda, consentendo di adattarsi al caso d'uso (*real-time monitoring* vs. batch analytics).
5. **Osservabilità.** Un'estensione prevede i contatori `_hits` e `_misses` che permettono di stimare l'*hit-ratio* $\text{hit}/(\text{hit} + \text{miss})$ e calibrare empiricamente il TTL ottimale.

3.4.4 Filtri spazio-temporali

Prima di consegnare i dati al motore *FuzzyCA*, ODHConnector applica due semplici, ma efficaci, restrizioni di pre-filtraggio che riducono drasticamente il traffico e la memoria necessari:

$$\text{relevant}(e) = (d(e, p_{\text{veh}}) \leq r) \wedge |t_{\text{now}} - t(e)| \leq \Delta t,$$

dove

- $d(\cdot, \cdot)$ è la distanza *haversine*³ fra l'evento e e la posizione corrente del veicolo p_{veh} ;
- r è il *raggio di prossimità* (default 10 km), configurabile in base al profilo della flotta e al contesto operativo;
- $t(e)$ è il timestamp dell'evento, mentre Δt (default 24 h) è l'ampiezza della finestra temporale di interesse.

Applicando tale maschera si evitano *over-fetch* di migliaia di record "lontani" sia nello spazio sia nel tempo, riducendo il carico sulla rete e sul motore di inferenza senza perdere informazioni realmente utili per il veicolo in marcia.

Oltre a migliorare l'efficienza computazionale, questi filtri hanno anche una funzione semantica importante: permettono di isolare tempestivamente solo gli eventi realmente rilevanti per il contesto di guida attuale, facilitando la logica di generazione degli alert e la successiva attivazione del sistema di raccomandazioni.

Alert & integrazione con FuzzyCA

Il metodo `generate_alerts()` di ODHConnector esegue tre step:

1. **Raggruppamento** Gli eventi filtrati sono suddivisi in categorie omogenee (`incident`, `queue`, `snow`, ...).
2. **Valutazione di pericolo** Per ogni gruppo si applicano soglie di severità o euristiche (es. `severity` ≥ 3 per gli incidenti) producendo oggetti `Alert(msg, relevance, impact)`:
 - $relevance \in [0,1] \rightarrow$ "quanto è vicino / attuale?"
 - $impact \in [0,1] \rightarrow$ "quanto incide sul rischio?"

Le condizioni meteo dall'ultimo `WeatherIndex` sono trattate allo stesso modo (pioggia > 0.5 , visibilità < 0.4 , ...).

³Formula che calcola la distanza ortodromica fra due coppie di coordinate geografiche (ϕ, λ) su una sfera di raggio $R \approx 6371$ km.

3. **Notifica asincrona** Gli *Alert* nuovi o aggiornati vengono inviati via push a **ENAIDE**, il co-pilota di bordo, e persistere nella cache TTL per evitare ripetizioni.
-

```
1 alerts = []
2 for inc in incidents:
3     if inc.severity >= 3:                # soglia critica
4         alerts.append(Alert("Incidente_grave:_rallentare_50%",
5                               0.5, 0.9))
6 ...
7 wx = self.get_weather_index()
8 if wx.visibility < 0.4:
9     alerts.append(Alert("Scarsa_visibilita':_ridurre_del_40%",
10                        0.6, 0.8))
```

Listing 3.3: Estratto semplificato di `generate_alerts()`

In sintesi, i filtri spazio-temporali *potano* i dati grezzi, il generatore di *Alert* li traduce in eventi di rischio leggibili dal conducente, e *FuzzyCA* li integra in una decisione numerica continua, riducendo al minimo falsi positivi e latenza di elaborazione.

3.5 FuzzyCA: framework *Fuzzy Context-Aware*

3.5.1 Motivazione

Nell'ambito della guida assistita, le tecniche statistiche convenzionali (come modelli a soglia fissa o regressione lineare) presentano due limiti principali: bassa interpretabilità e scarsa capacità di rappresentare l'incertezza percepita dal conducente. Ad esempio, stabilire una soglia netta oltre la quale scatta un allarme (*if rain > 2 mm/h then slow down*) non tiene conto della gradualità con cui un conducente umano valuta il rischio.

Per affrontare questo limite, FuzzyCA adotta la logica fuzzy come motore decisionale *context-aware*. Tale approccio permette di combinare, in modo trasparente e linguistico, molteplici fattori eterogenei:

- **Fattori endogeni**, come lo stile di guida, lo stato di affaticamento, le ore consecutive di guida;

- **Fattori esogeni**, come condizioni meteorologiche, traffico, prossimità di incidenti o cantieri.

L'obiettivo è produrre un indice di rischio **continuo, spiegabile e graduale**, utile per generare raccomandazioni contestuali (es. ridurre velocità, suggerire una pausa). L'output finale può essere tradotto in un `speed_factor` oppure usato per attivare segnali visivi e acustici nel veicolo.

Una trattazione più dettagliata della logica fuzzy, del modello Mamdani, e degli scenari di inferenza sarà sviluppata nel Capitolo 4.

3.5.2 Variabili linguistiche

Il sistema fuzzy di FuzzyCA si basa su un insieme di variabili linguistiche, ciascuna definita da un dominio continuo e da un set di *membership functions* che modellano categorie qualitative. La Tabella 3.1 riassume le principali variabili impiegate nella fase di inferenza.

Tabella 3.1: Principali variabili linguistiche e relativi insiemi fuzzy.

Variabile	Domini fuzzy	Descrizione
Velocità veicolo	{Bassa, Moderata, Alta}	Classificata dinamicamente rispetto ai percentili della distribuzione recente del veicolo stesso
Distanza da incidente	{Vicino, Medio, Lontano}	Calcolata e normalizzata rispetto al raggio di visibilità ODH
Ore di guida	{Fresco, Affaticato}	Stimata tramite contatore orario; soglia convenzionale > 4h
Intensità di pioggia	{Assente, Leggera, Forte}	Derivata da Open Data Hub (ODH) Weather, con aggregazione su media temporale mobile

Il motore fuzzy elabora queste variabili attraverso un insieme di regole di inferenza in linguaggio naturale. Un esempio di regola tipica è il seguente:

IF (Pioggia = *Forte*) \wedge (Distanza-Incidente = *Vicino*) \Rightarrow reduce_speed(0.6)

Tale regola indica che, in presenza simultanea di pioggia intensa e pericolo ravvicinato (es. incidente o cantiere), il sistema consiglia una riduzione del 40% della velocità di

crociera attuale. Il valore 0.6 rappresenta un `speed_factor` fuzzy, da intendersi come moltiplicatore prudenziale.

Il vantaggio di questo approccio è la sua continuità: al variare progressivo delle condizioni (pioggia da leggera a forte, distanza da media a vicina), anche l'output si modifica in modo morbido e spiegabile, evitando soglie rigide e comportamenti discontinui.

3.6 Conclusioni del capitolo

Il connettore sviluppato fornisce alla piattaforma un *gateway* affidabile e performante verso i dati open del territorio, valorizzando l'ecosistema ODH e aprendo la strada a nuove funzionalità fondamentali per la mobilità moderna, come l'*eco-driving*, il routing adattivo e l'analisi predittiva. L'approccio *Fuzzy Context-Aware* impiegato garantisce una maggiore trasparenza nelle decisioni prese dal sistema e rende le raccomandazioni più comprensibili all'utente finale. Inoltre, la strategia di caching basata su TTL assicura un equilibrio tra aggiornamenti frequenti e sostenibilità del carico sulle API pubbliche, migliorando al tempo stesso l'esperienza utente in cabina.

Questo modulo rappresenta dunque un tassello fondamentale per il passaggio da una raccolta passiva dei dati a un sistema proattivo, intelligente e scalabile, orientato al supporto decisionale in tempo reale. Le architetture proposte sono state progettate per essere riutilizzabili e facilmente estendibili anche in contesti diversi da quello sperimentale.

3.6.1 Esempi applicativi e visualizzazioni

Per dimostrare l'utilità pratica di *ODHConnector* sono stati sviluppati due prototipi di front-end che sfruttano i dati prelevati dagli Adapter descritti nella sezione 3.3.

1. una **dashboard da linea di comando (CLI)** pensata per il monitoraggio rapido;
2. una **mappa interattiva**, consultabile via web, che presenta gli eventi di traffico, le condizioni meteorologiche e i consigli di velocità sull'intero percorso.

Traffic & Weather CLI Dashboard

La dashboard (Figura 3.2), realizzata con la libreria `rich` in Python, riceve un flusso in tempo reale da `ODHConnector.watch_live()` e mantiene una tabella degli eventi dell'ultima finestra (2200 h nell'esempio mostrato).

Events (last 2200h)	
Type	Count
intralci viabilità in e fuori alto adige_chiusura temporanea	1
situazione attuale_code	3
incident	2

Alerts:

- Incidente grave: rallentare 50% (relevance=0.90)
- Incidente grave: rallentare 50% (relevance=0.90)
- Code sulla tratta: considerare deviazione (-20%) (relevance=0.70)
- Chiusura: In entrambe le direzioni CHIUSURE dello svincolo Trento Sud in entrata e uscita dalle ore 21:00 alle ore 06:00 per lavori di asfaltatura. (relevance=1.00)

Attention score: 0.88

Speed advice per sample

Lat,Lon	Speed factor
46.0701,11.1200	0.38
46.1587,11.0940	0.38

Figura 3.2: Esempio di esecuzione della *Traffic & Weather* dashboard in ambiente CLI. L'interfaccia elenca le tipologie di eventi osservati, gli *alert* generati da FuzzyCA con rispettivo punteggio di rilevanza e l'*attention score* globale.

Per ogni nuovo evento significativo, il sistema presenta:

- la **descrizione** localizzata (IT/DE/EN), recuperata dinamicamente dai metadati ODH;
- il **consiglio di condotta** (es. "rallentare del 50%") generato in base all'indice di rischio fuzzy;
- la **rilevanza** $\rho \in [0,1]$, mostrata attraverso una scala cromatica (verde \rightarrow rosso), che aiuta l'utente a cogliere con immediatezza la criticità degli eventi.

Mappa interattiva con Folium

Grazie a questa vista geospaziale (Figura 3.3) l'operatore di flotta può identificare immediatamente i *colli di bottiglia* lungo la tratta Brennero-Trento, valutare deviazioni alternative e anticipare messaggi di avviso ai conducenti. Questa visualizzazione interattiva si è dimostrata utile anche in fase di validazione del modello, poiché ha permesso di confrontare visivamente le condizioni predette con quelle effettive e di eseguire test di coerenza sui dati provenienti dai sensori.



Figura 3.3: Mappa interattiva generata con *Folium*. Marker rosso/arancio indicano incidenti e code, mentre la color-bar delle temperature (blu → rosso) visualizza il gradiente termico. Infine, nei *popup* sono riportati i dettagli dell'evento.

In prospettiva, entrambi i prototipi potranno essere integrati in un'unica interfaccia multi-dispositivo (tablet, centrale operativa, cruscotto smart), aprendo nuove opportunità per il supporto alla guida contestuale e il coordinamento dinamico della flotta.

Capitolo 4

FuzzyCA: un Modulo Fuzzy Context-Aware per ADAS

4.1 Introduzione

Un moderno sistema di assistenza alla guida (ADAS) deve prendere decisioni in presenza di incertezza e non linearità, combinando molti fattori eterogenei come farebbe un guidatore esperto. La *logica fuzzy*, introdotta da Zadeh nel 1965[24], è uno strumento potente per formalizzare ragionamenti linguistici e qualitativi tipici dell'essere umano nei sistemi di controllo. Essa consente di rappresentare grandezze continue con etichette linguistiche (ad esempio *bassa*, *media*, *alta*), usando funzioni di appartenenza che passano gradualmente da 0 a 1. In questo modo si evitano soglie nette e si riflette naturalmente l'incertezza delle misure. Un sistema fuzzy tratta gli input come valori parzialmente veri: ad esempio, una visibilità di 300 m può avere grado di appartenenza 0.8 nell'insieme *Bassa* e 0.2 nell'insieme *Media*.

Il vantaggio di un approccio fuzzy è duplice. Da un lato, le regole IF-THEN in linguaggio naturale permettono di codificare l'esperienza dei conducenti: per esempio "*SE visibilità è bassa E conducente è affaticato ALLORA rischio è alto*". Dall'altro, l'uscita del sistema è continua: anziché un allarme on/off, il rischio cresce gradualmente al peggiorare delle condizioni. Ad esempio, in un tratto urbano a visibilità moderata e traffico elevato, il controller fuzzy può generare un livello di rischio superiore a quello proporzionale, senza scatti improvvisi. Questo evita oscillazioni brusche tipiche della logica booleana e permette reazioni più morbide e sicure. Studi recenti mostrano che i modelli fuzzy

possono raggiungere prestazioni predittive paragonabili ai modelli di machine learning, ma con il vantaggio della maggiore *interpretabilità*: ogni regola è in linguaggio naturale, il che semplifica la spiegazione delle decisioni[8, 16]. Ciò è un requisito fondamentale in ambito automotive, per motivare le scelte al guidatore e per la certificazione di sicurezza.

Dunque, per ricapitolare, nei contesti ADAS la logica fuzzy è stata ampiamente studiata per unire informazioni eterogenee. Ad esempio, un controller fuzzy può consigliare una velocità sicura basandosi sulla geometria della strada, condizioni meteo, distanza dal veicolo anteriore, pressione degli pneumatici e comfort del conducente, fornendo un output (velocità raccomandata) che equilibra il benessere e la sicurezza di ques'ultimo. *Rispetto ai tradizionali sistemi a soglia, il fuzzy mitiga l'effetto bivalente*: non si limita a frenare bruscamente quando la distanza scende sotto X, ma aumenta gradualmente l'allerta man mano che la distanza diminuisce. Inoltre, rispetto a modelli di apprendimento statistico "black-box", i modelli fuzzy sono interpretabili: ogni regola è in linguaggio naturale, il che semplifica la spiegazione delle decisioni. Questo è un requisito fondamentale in applicazioni automotive, per motivare le scelte al guidatore e per la certificazione di sicurezza[12].

In questo capitolo presentiamo *FuzzyCA* (Fuzzy Context-Aware), un modulo fuzzy progettato come co-pilota intelligente. FuzzyCA valuta il contesto di guida complessivo combinando più variabili di input per calcolare un indice di rischio. Descriviamo le variabili fuzzy scelte, il processo di fuzzificazione, le regole IF-THEN, l'inferenza fuzzy (modello Mamdani[12]) e la defuzzificazione finale. Il tutto è illustrato con esempi (meteo, traffico, ore di guida) e con figure didattiche che mostrano il processo di inferenza fuzzy.

4.2 Fondamenti della logica fuzzy

Logica fuzzy vs. Logica booleana

La logica fuzzy generalizza la logica booleana tradizionale, introducendo la possibilità di esprimere gradi di verità continui nell'intervallo $[0,1]$, anziché limitarsi ai soli valori discreti *vero* (1) e *falso* (0). Un *insieme fuzzy* A definito su un universo X è caratterizzato da una funzione di appartenenza $\mu_A(x) : X \rightarrow [0,1]$, che assegna a ogni elemento un grado di appartenenza.

Ad esempio, per la variabile linguistica "Visibilità" è possibile definire insiemi fuzzy come *Pessima*, *Bassa*, *Media*, *Alta*, ciascuno associato a una propria funzione

di appartenenza. In questo modo si modellano concetti vaghi e continui senza dover introdurre soglie nette, riflettendo meglio la percezione umana.

È importante distinguere tra grado di appartenenza fuzzy e probabilità. Un valore fuzzy $\mu = 0,7$ indica che l'affermazione qualitativa "la visibilità è scarsa" è vera al 70%, non che c'è il 70% di probabilità che la visibilità sia scarsa. La probabilità esprime incertezza statistica; la membership fuzzy esprime quanto un valore aderisce a una categoria qualitativa. Nei moderni sistemi ADAS, questi approcci possono essere complementari: modelli probabilistici stimano la probabilità di un evento (es. incidente), mentre modelli fuzzy valutano la qualità del contesto (es. "conducente stanco", "strada scivolosa").

Uno dei vantaggi della logica fuzzy è la robustezza nei confronti del rumore: grazie alla transizione graduale tra i livelli, variazioni minime nei dati d'ingresso non causano salti bruschi nell'output. Ad esempio, se il sensore di visibilità oscilla attorno a una soglia tra "bassa" e "media", il grado di rischio varierà in modo continuo e prevedibile, evitando comportamenti instabili.

4.2.1 La pipeline di inferenza fuzzy

Un sistema fuzzy tipico segue tre fasi principali:

- **Fuzzificazione:** i valori numerici delle variabili in ingresso vengono convertiti in gradi di appartenenza a insiemi fuzzy. Ad esempio: "Visibilità = 300 m" produce $\mu_{Bassa} = 0,8$ e $\mu_{Media} = 0,2$.
- **Inferenza fuzzy:** vengono applicate le regole IF-THEN. Gli antecedenti delle regole sono valutati combinando i gradi di appartenenza tramite operatori logici fuzzy (AND \rightarrow minimo, OR \rightarrow massimo). Il risultato è un grado di attivazione per ciascuna regola, che "taglia" parzialmente il conseguente (l'insieme fuzzy di uscita).
- **Defuzzificazione:** tutti i contributi fuzzy delle regole attivate vengono aggregati (solitamente tramite operatore di massimo) in un unico insieme fuzzy. Questo viene poi convertito in un valore numerico (crisp) tramite il metodo del centroide¹.

¹Il metodo del *centroide* (o *center-of-gravity*) è una tecnica di defuzzificazione che calcola il baricentro geometrico dell'insieme fuzzy aggregato. È il metodo più comune nei sistemi Mamdani, in quanto fornisce un compromesso bilanciato tra le diverse regole attivate, risultando in una risposta fluida e continua.

Questo processo è illustrato in Figura 4.2, che mostra la transizione da input numerici a valutazioni qualitative, fino alla produzione di un output numerico continuo.

Esempio di inferenza

Consideriamo un caso realistico: un camion viaggia di notte dopo molte ore di guida. Il conducente è affaticato (*Fatica Alta* con grado di appartenenza $\mu \approx 0,8$), c'è nebbia leggera (*Visibilità Media* con $\mu_{Media} = 0,6$ e $\mu_{Bassa} = 0,3$), e una pioggerellina intermittente (*Meteo Instabile*). Il traffico è scarso (*Traffico Scorrevole*) e l'urgenza della consegna è moderata.

In questa configurazione nessuna delle regole principali si attiva completamente: la visibilità non è pessima, il traffico non è congestionato e la fatica non è ancora estrema. Tuttavia, una regola aggiuntiva del tipo:

SE Fatica è Alta E Visibilità è Media ALLORA Rischio è Alto

può essere attivata parzialmente. Supponiamo che l'attivazione di questa regola raggiunga un valore di 0,7. In parallelo, altre regole meno rilevanti (es. la regola di default in assenza di fattori critici) potrebbero contribuire con gradi molto più bassi.

L'aggregazione dei contributi fuzzy genera un insieme di rischio centrato sulla categoria *Alto*, con possibile sovrapposizione minore su *Moderato*. Dopo la defuzzificazione, si ottiene un valore numerico, ad esempio $\text{risk} = 0,70$.

Questo valore può essere interpretato come un *indice di rischio* significativo, e viene convertito in un speed_factor pari a circa 0,6, ovvero una raccomandazione di ridurre la velocità del 40%.

Il sistema può quindi generare un messaggio motivato e comprensibile per il conducente, ad esempio:

Attenzione: visibilità moderata e fatica elevata \rightarrow rischio alto. Si consiglia di rallentare del 40% e pianificare una pausa a breve.

4.3 Vantaggi e confronti con altri approcci

L'impiego della logica fuzzy in un sistema ADAS come quello presentato in questo elaborato, combina diversi vantaggi teorici e pratici:

- **Spiegabilità:** ogni decisione deriva da regole linguistiche note, facilmente comprensibili dall'uomo. A differenza di un modello di deep learning "scatola nera",

possiamo spiegare al conducente che l'allarme è dovuto ad esempio a "visibilità ridotta + fatica". Ciò facilita la fiducia degli utenti, la diagnosi di errori e la certificazione di sicurezza.

- **Robustezza a rumore e incertezza:** input fluttuanti o rumorosi non fanno saltare improvvisamente gli output, perché le membership parziali attenuano le variazioni. Ad esempio, se il sensore di visibilità oscilla attorno a una soglia, il grado di rischio varierà dolcemente anziché passare bruscamente da 0 a 1. Inoltre, si possono variare le funzioni di appartenenza per ridurre il peso di misure meno affidabili.
- **Adattabilità e flessibilità:** aggiungere nuovi sensori o condizioni equivale a definire ulteriori insiemi fuzzy e regole, senza stravolgere l'architettura. Un sistema fuzzy ben progettato generalizza su molte situazioni con poche regole. In scenari rari o imprevisti (es. maltempo estremo), si possono inserire direttamente regole esperte, mentre un modello statistico potrebbe non aver visto quei casi.
- **Efficienza embedded:** il calcolo fuzzy richiede risorse modeste. Anche decine di regole possono essere valutate in tempo reale su un microcontrollore automotive. Esistono chip dedicati a logica fuzzy da decenni. In pratica FuzzyCA può essere implementato completamente on-board (ad es. in codice ottimizzato Rust o WebAssembly) senza bisogno di cloud, con esecuzione deterministica e sicura.

Confrontando questo approccio con altri tipici:

- **Reti neurali profonde:** eccellono nel processare dati grezzi (immagini, segnali) e possono apprendere funzioni complesse. Tuttavia richiedono grandi dati e rimangono opache. In applicazioni critiche come ADAS l'interpretabilità conta molto. I controller fuzzy possono ottenere prestazioni simili (soprattutto nel *situation awareness*) con maggiore spiegabilità e stabilità computazionale, specie quando i dati sono limitati[8].
- **Controllori tradizionali (PID, logiche a soglia):** PID e trigger booleani hanno struttura rigida e non gestiscono bene molteplici fattori. Un PID classico ad esempio regola la velocità usando solo la distanza al veicolo precedente, senza considerare stato del conducente o condizioni stradali. Invece un controller fuzzy estende il comportamento di base: in un'emergenza, un PID forse frena forte solo in base alla distanza, mentre un fuzzy può modulare la frenata anche in base all'aderenza o

all'attenzione del guidatore. La logica fuzzy, infine, elimina le *zone morte* delle soglie on/off: l'allerta cresce progressivamente man mano che il rischio aumenta, evitando falsi allarmi o reazioni tardive.

4.4 Architettura e implementazione del modulo FuzzyCA

4.4.1 Introduzione

Il modulo FuzzyCA integra un sistema di inferenza fuzzy contestuale in un'architettura ADAS, valutando un indice di rischio basato su fattori ambientali e del conducente. Grazie alla logica fuzzy, il sistema può combinare variabili imprecise (meteo, traffico, stanchezza) mediante regole fornite da esperti e produrre un output interpretabile (rischio).

In particolare, il FIS (Fuzzy Inference System) mappa gli ingressi su un output tramite regole fuzzy. L'obiettivo è fornire un modulo che, data l'informazione in tempo reale di condizioni meteo, traffico e livello di fatica, restituisca un fattore di rischio, utilizzabile dal sistema ADAS per adattare le strategie di assistenza.

4.4.2 Variabili di input e output

Nel modulo FuzzyCA si definiscono tre variabili di input (tutte normalizzate nell'intervallo $[0,100]$):

- *weather* - condizioni meteo, con tre insiemi fuzzy: *good*, *moderate*, *bad*, che rappresentano rispettivamente condizioni ottimali, medie e avverse.
- *traffic* - congestione del traffico, con tre insiemi: *light*, *medium*, *heavy* (traffico scarso, medio, intenso).
- *fatigue* - livello di fatica del conducente, con tre insiemi: *fresh*, *tired*, *extreme* (il conducente è riposato, stanco o molto stanco).

L'output del sistema è la variabile *risk* (fattore di rischio, 0=rischio nullo, 100=rischio massimo), suddiviso in quattro insiemi fuzzy: *low*, *moderate*, *high*, *critical* (rischio basso, moderato, alto o critico). Le funzioni di appartenenza per ciascuna variabile sono definite con forme trapezoidali (*trapmf*) o triangolari (*trimf*), come mostrato in Figura 4.1.

Per esempio, per la variabile *weather* si hanno:

$$\begin{aligned}\mu_{\text{good}}(w) &= \text{trapmf}(w; 0, 0, 15, 30), \\ \mu_{\text{moderate}}(w) &= \text{trimf}(w; 20, 45, 70), \\ \mu_{\text{bad}}(w) &= \text{trapmf}(w; 50, 65, 100, 100).\end{aligned}$$

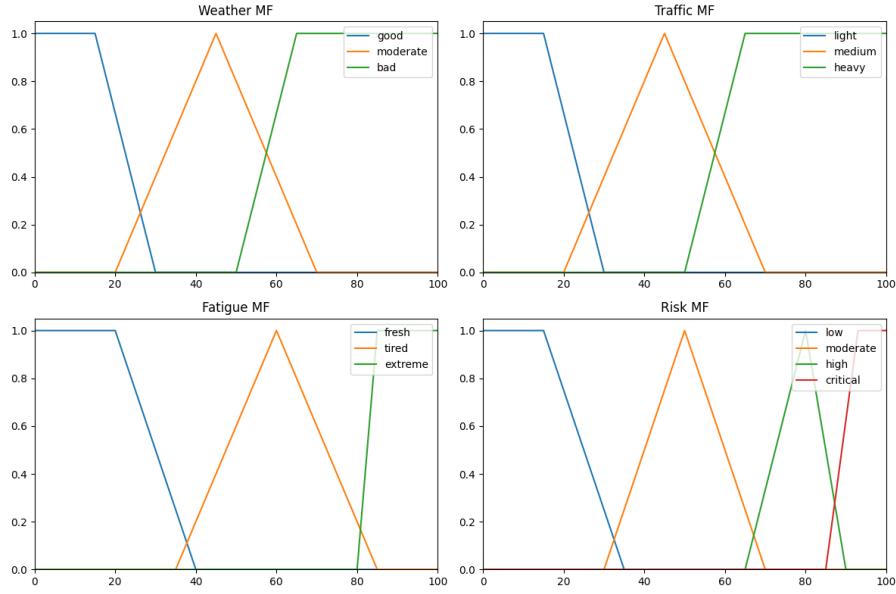


Figura 4.1: Grafici delle funzioni di appartenenza delle variabili fuzzy di ingresso (*weather*, *traffic*, *fatigue*) e di uscita (*risk*).

Analogamente, per le variabili *traffic* e *fatigue* si definiscono:

$$\begin{aligned}\mu_{\text{light}}(t) &= \text{trapmf}(t; 0, 0, 15, 30), \\ \mu_{\text{medium}}(t) &= \text{trimf}(t; 20, 45, 70), \\ \mu_{\text{heavy}}(t) &= \text{trapmf}(t; 50, 65, 100, 100), \\ \mu_{\text{fresh}}(f) &= \text{trapmf}(f; 0, 0, 20, 40), \\ \mu_{\text{tired}}(f) &= \text{trimf}(f; 35, 60, 85), \\ \mu_{\text{extreme}}(f) &= \text{trapmf}(f; 80, 85, 100, 100).\end{aligned}$$

Per l'output *risk* si hanno:

$$\begin{aligned}\mu_{\text{low}}(r) &= \text{trapmf}(r; 0, 0, 15, 30), \\ \mu_{\text{moderate}}(r) &= \text{trimf}(r; 30, 50, 70), \\ \mu_{\text{high}}(r) &= \text{trimf}(r; 65, 80, 90), \\ \mu_{\text{critical}}(r) &= \text{trapmf}(r; 85, 93, 100, 100).\end{aligned}$$

I grafici di queste membership functions sono mostrati in Figura 4.1.

4.4.3 Regole fuzzy del sistema FuzzyCA

Le regole fuzzy sono formulate in linguaggio natural-form utilizzando gli insiemi definiti. Alcuni esempi di regole implementate sono:

- IF *weather* is *bad* AND *traffic* is *heavy*, THEN *risk* is *critical*.
(Condizione gravissima combinata: meteo pessimo e traffico intenso.)
- IF *fatigue* is *extreme*, THEN *risk* is *critical*.
(Anche da sola, una fatica estrema impone un rischio massimo.)
- IF *weather* is *bad* OR *traffic* is *heavy*, THEN *risk* is *high*.
(Una singola variabile molto negativa porta a rischio alto.)
- IF *weather* is *moderate* OR *traffic* is *medium* OR *fatigue* is *tired*, THEN *risk* is *moderate*.
(Condizioni di medio livello producono un rischio moderato.)
- IF *weather* is *good* AND *traffic* is *light* AND *fatigue* is *fresh*, THEN *risk* is *low*.
(Situazione ideale: condizioni favorevoli su tutte le variabili.)

Queste regole (implementate con logica Mamdani) definiscono come combinare le condizioni di contesto per inferire il livello di rischio. Durante l'inferenza, i gradi di appartenenza degli ingressi vengono aggregati secondo le congiunzioni o disgiunzioni logiche delle regole, producendo in output un insieme fuzzy di rischio. Infine la defuzzificazione converte tale insieme nel valore di rischio finale.

4.4.4 Architettura del sistema

L'architettura completa del modulo FuzzyCA preso in esempio prevede il seguente flusso: acquisizione dei tre dati di contesto → fuzzificazione (calcolo delle membership) → inferenza fuzzy secondo le regole → defuzzificazione (metodo LOM) → uscita *risk*. Lo schema a blocchi è illustrato in Figura 4.2. Il modulo è stato implementato utilizzando la libreria `scikit-fuzzy` in Python, che supporta l'approccio Mamdani e la defuzzificazione standard. L'uso di membership continue e regole rende il sistema flessibile e interpretabile, facilitando l'integrazione in architetture ADAS esistenti.

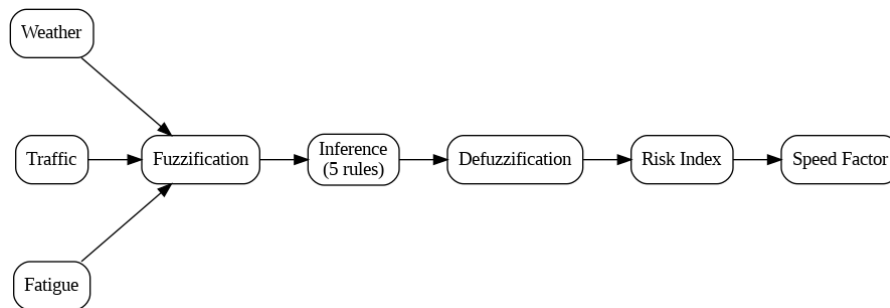


Figura 4.2: Schema a blocchi del modulo FuzzyCA: acquisizione dei dati, fuzzificazione, inferenza Mamdani e defuzzificazione per calcolare l'indice di rischio.

4.4.5 Vantaggi del modulo

Il modulo fuzzy offre numerosi vantaggi in un contesto ADAS. Grazie alla logica fuzzy, è possibile trattare informazioni imprecise (come "traffico moderato" o "leggera pioggia") in modo naturale. Le regole fuzzy possono essere formulate da esperti e facilmente modificate. Inoltre, l'approccio Mamdani produce risultati robusti in presenza di incertezze sensoristiche. In generale, la logica fuzzy è apprezzata per la sua capacità di modellare conoscenza esperta ed effettuare inferenze senza richiedere modelli matematici precisi. Nel nostro contesto, FuzzyCA permette di valutare in tempo reale un indicatore di rischio contestuale che considera congiuntamente meteo, traffico e fatica, migliorando la reattività dell'ADAS rispetto a sistemi basati solo su soglie rigide.

4.4.6 Prototipazione e validazione futura

Per validare il comportamento del modulo FuzzyCA è stata effettuata un'analisi sistematica su una griglia di valori delle variabili di contesto. A tal fine sono state generate mappe e superfici tridimensionali che mostrano l'andamento dell'indice di rischio al variare di due input, mantenendo costante la terza variabile.

La Figura 4.3 mostra le heatmap dell'indice di rischio in tre casi distinti:

- **Caso A:** *fatigue* = 30, variazione su *weather* e *traffic*;
- **Caso B:** *fatigue* = 60, variazione su *weather* e *traffic*;
- **Caso C:** *traffic* = 0, variazione su *weather* e *fatigue*.

Analogamente, la Figura 4.4 mostra le corrispondenti superfici 3D, dove è visibile la forma della funzione di risposta fuzzy: si osservano transizioni nette tra aree di rischio moderato e critico, plateau e interazioni non lineari tra variabili. Queste visualizzazioni confermano il comportamento atteso: il rischio cresce al peggiorare delle condizioni, ed è massimizzato quando più fattori negativi si combinano.

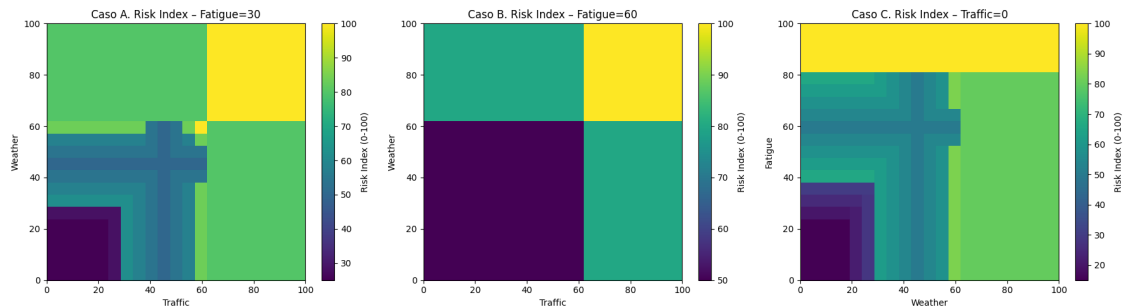


Figura 4.3: Heatmap dell'indice di rischio (0-100) per tre scenari. Caso A: *fatigue* = 30; Caso B: *fatigue* = 60; Caso C: *traffic* = 0.

Lo sviluppo completo di FuzzyCA prevede tre fasi principali:

1. **Prototipo software:** realizzazione di un proof-of-concept² in Python, definizione delle variabili, funzioni di appartenenza e regole di inferenza. In questa fase si testano scenari critici (es. fatica elevata con meteo avverso) per calibrare il sistema.

²Un *proof-of-concept* (PoC) è una dimostrazione preliminare che ha lo scopo di verificare la fattibilità tecnica e concettuale di un sistema. Nel caso di FuzzyCA, consente di validare la correttezza del comportamento fuzzy in scenari simulati prima della fase di sviluppo embedded.

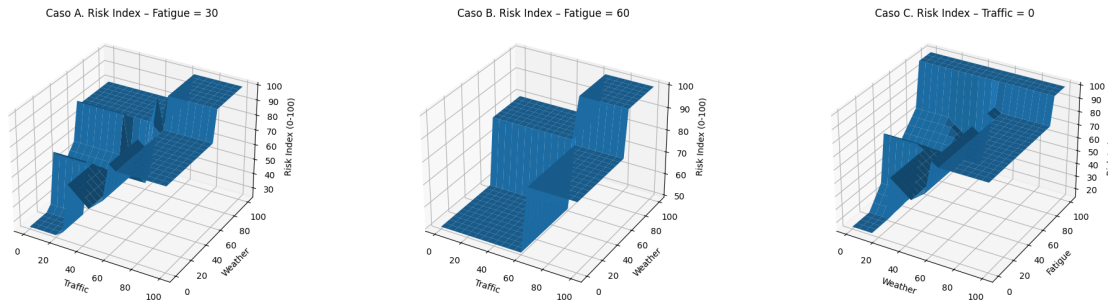


Figura 4.4: Superfici 3D dell'indice di rischio nei tre casi: A) *fatigue* = 30, B) *fatigue* = 60, C) *traffic* = 0.

2. **Benchmark in simulazione:** confronto con modelli alternativi (es. ML supervisionato) in ambienti come CARLA³ o Simulink⁴.
3. **Implementazione embedded e test su veicolo:** porting del motore fuzzy su hardware automotive (es. centralina embedded in Rust/WebAssembly). I test su strada verificheranno la coerenza degli avvisi rispetto alle condizioni reali, con validazione ISO⁵ e analisi FMEA⁶.

³CARLA (Car Learning to Act) è un simulatore open-source per la guida autonoma, basato su Unreal Engine, progettato per supportare lo sviluppo e la validazione di sistemi ADAS e veicoli autonomi in scenari realistici.

⁴Simulink è un ambiente di modellazione e simulazione grafica integrato in MATLAB, ampiamente utilizzato per il design e la verifica di sistemi dinamici, inclusi i controlli fuzzy e i modelli di veicoli.

⁵ISO 26262 è lo standard internazionale per la sicurezza funzionale dei sistemi elettrici ed elettronici nei veicoli. La sua applicazione garantisce che i sistemi ADAS, come FuzzyCA, rispettino requisiti di sicurezza in scenari critici.

⁶FMEA (Failure Mode and Effects Analysis) è una tecnica strutturata per identificare e valutare potenziali modalità di guasto di un sistema, analizzandone le cause, gli effetti e la criticità, allo scopo di prevenirli in fase progettuale.

Capitolo 5

Conclusioni

Il lavoro svolto in questa tesi ha portato alla realizzazione di un modulo software integrabile all'interno di un co-pilota intelligente per veicoli, progettato per raccogliere e filtrare dati eterogenei e fornire in tempo reale raccomandazioni utili al conducente. Attraverso simulazioni su percorsi reali, è stato dimostrato che il sistema è in grado di aumentare la consapevolezza situazionale del guidatore: *ad esempio* segnalando con anticipo la presenza di incidenti o cantieri sul tragitto, avvisando in caso di condizioni meteorologiche pericolose e suggerendo opportuni aggiustamenti nello stile di guida. Integrando la componente di Open Data, il co-pilota può "vedere oltre i sensori di bordo", colmando quel gap informativo che i normali ADAS di serie presentano.

In definitiva, il co-pilota sviluppato da MinervaS incarna il paradigma della *smart mobility* cooperativa, dove il veicolo non è più isolato ma interconnesso con l'ecosistema circostante (infrastrutture, altri utenti, cloud) per offrire un servizio di guida più sicuro, sostenibile e confortevole.

Oltre ai risultati concreti ottenuti con la pipeline implementata, la tesi ha esplorato l'estensione fuzzy (**FuzzyCA**), delineando un percorso di evoluzione verso un'Intelligenza Artificiale spiegabile a bordo veicolo. Questo contributo, seppur teorico, è di notevole rilevanza: in un'industria come quella automotive potersi fidare dell'AI è fondamentale. Un co-pilota che possa giustificare i propri consigli con regole chiare (sul modello "causa-effetto") sarà più facilmente accettato dai conducenti e potrà essere omologato più rapidamente dagli enti regolatori, rispetto a soluzioni *black-box* puramente statistiche.

5.1 Sintesi dei risultati del prototipo con ODHConnector

ODHConnector si è dimostrato un componente chiave per l'integrazione in tempo reale di dati meteo e V2X. Durante la fase di test, l'adozione di ottimizzazioni software dedicate ha migliorato significativamente le prestazioni complessive:

- **Caching TTL** - implementazione di un meccanismo di cache con tempo di vita configurabile che ha ridotto il carico sul servizio ODH, limitando l'accesso ripetuto a dati a bassa variabilità, diminuendo così la latenza.
- **Filtri spazio-temporali** - applicati *a valle* del connettore, hanno permesso di eliminare il rumore e i dati anomali prima dell'inferenza, aumentando la robustezza delle decisioni.
- **Calcolo fuzzy del coefficiente di allerta** - la combinazione graduale di fattori di rischio (precipitazioni, visibilità, congestione traffico) ha prodotto suggerimenti più coerenti con la percezione del conducente.

Questi accorgimenti hanno reso possibile la generazione di consigli di guida efficienti e segnalazioni di pericolo in tempo reale con un carico computazionale sostenibile, dimostrando la validità del prototipo nel contesto sperimentale del Trentino.

5.2 Analisi critica dei limiti dell'approccio attuale

Nonostante i risultati incoraggianti, l'approccio mostra alcuni limiti sostanziali:

1. **Dipendenza geografica** - l'utilizzo esclusivo di OpenDataHub (ODH) per il Trentino limita la generalizzabilità della soluzione proposta.
2. **Scalabilità dei dati** - su scala nazionale il volume di flussi meteo e V2X cresce di ordini di grandezza, rischiando di saturare banda e risorse di calcolo.
3. **Carico computazionale per veicolo** - l'inferenza fuzzy e i filtri devono essere eseguiti per ciascun nodo connesso, ponendo requisiti stringenti per il *real-time* in scenari di traffico intenso.

Questi vincoli vanno affrontati con soluzioni architetturali avanzate prima di un rilascio su larga scala.

5.3 Proposte di scalabilità nazionale

Per superare i limiti evidenziati si propongono tre linee di intervento:

Indicizzazioni spaziali Implementare strutture come *quadtree* o *k-d tree* per ridurre la complessità delle query geografiche da $O(n)$ a $O(\log n)$.

Caching distribuito edge Collocare nodi di cache territoriali che mantengano i dati più recenti e si sincronizzino in differita con la sorgente centrale, diminuendo la latenza percepita dal veicolo.

Rappresentazioni fuzzy granulari Aggregare i dati in insiemi fuzzy spaziali e temporali (es. "zona montana alta", "fascia diurna") per ridurre il numero di punti elaborati senza perdere informazione semantica.

5.4 Confronto con lo stato dell'arte

La seguente sezione è una traduzione adattata (con riformulazione parziale per chiarezza) di alcuni passaggi del lavoro di Wickramanayake et al. (2020)[21], integrata con contributi di altri studi rilevanti.

Numerosi studi recenti sull'eco-driving confermano la validità e l'efficacia dell'approccio adottato in questa tesi, evidenziando l'importanza di soluzioni basate su logica fuzzy e feedback in tempo reale per la riduzione dei consumi. Studi di Aksit, Yavuz e Sen [1], Massoud et al. [13], Wickramanayake, Bandara e Samarasekara [21] e Rolim et al. [15] mostrano come la logica fuzzy, integrata con variabili OBD e feedback in tempo reale, conduca a risparmi tra il 9% e il 20% di carburante.

Ulteriori contributi confermano la centralità del comportamento del conducente nella determinazione dei consumi:

- **Walnum e Simonsen [20]** hanno analizzato dati su camion pesanti in Norvegia, mostrando che, al netto delle condizioni stradali e veicolari, è il comportamento del conducente a incidere maggiormente sull'efficienza energetica.
- **Toledo e Shiftan [17]** e **Ayyildiz et al. [4]** hanno mostrato che il feedback personalizzato - fornito sia in tempo reale che differito - è più efficace dei corsi di formazione collettiva nel promuovere l'adozione di stili di guida ecologici.

- **CGI [5]** ha proposto un sistema di clustering statistico per profilare lo stile di guida e stimare i consumi: i risultati indicano che il comportamento del conducente può spiegare tra il 10% e il 30% della variabilità nel consumo, pur trascurando variabili ambientali complesse.
- **Wickramasinghe et al. [22]** hanno impiegato un controllore fuzzy PD per migliorare i consumi su percorsi noti, evidenziando i limiti di sistemi troppo rigidi che non considerano traffico e condizioni dinamiche.
- **Gilman et al. [9]** hanno sviluppato un'architettura context-aware che fornisce feedback basato su dati ambientali e veicolari; l'efficacia risulta però limitata dall'assenza di aggiornamenti in tempo reale.

In sintesi, l'analisi della letteratura mette in luce tre criticità comuni nei framework esistenti:

1. Il feedback è spesso **off-line** e non permette una correzione immediata dei comportamenti scorretti.
2. Il feedback è **manuale** o supervisionato, quindi costoso e poco adatto a contesti real-time.
3. I modelli considerano un **numero limitato di attributi** (es. solo variabili OBD o meteo semplificato).

Risulta quindi necessario uno sviluppo di sistemi più completi e adattivi, come quello proposto, capaci di integrare più fonti, adattarsi a percorsi variabili e fornire feedback immediato, scalando verso scenari reali di flotta.

5.5 Prospettive future

1. Implementazione embedded e ottimizzazione prestazionale Attualmente il prototipo gira su ambiente PC; un passo successivo sarà il porting su un dispositivo embedded automotive (ad es. un single-board computer tipo Raspberry Pi, o direttamente su un'unità di controllo della cabina). Questo richiederà di ottimizzare il codice e magari rifattorizzarlo in C/C++ per le parti critiche, assicurando tempi di risposta real-time. L'impiego di PostGIS e FastAPI potrebbe essere adattato: su un dispositivo locale con risorse limitate potrebbe

convenire utilizzare librerie geospaziali leggere in-process (es. Shapely in Python) oppure pre-calcolare offline certi percorsi frequenti. Sarà inoltre importante curare la robustezza e la sicurezza software (gestione di eventuali perdite di connessione dati, recovery dopo un reboot, cybersecurity dell'unità di bordo, ecc.) per passare dallo stadio prototipale a un prodotto pronto per test su strada.

2. Estensione a flotte urbane e V2V Finora ci siamo concentrati sul caso di un singolo veicolo che riceve dati dal cloud. Ma un ulteriore livello di efficacia si può ottenere facendo comunicare tra loro più veicoli equipaggiati col co-pilota (Vehicle-to-Vehicle). Immaginiamo una flotta di mezzi pesanti che viaggiano distribuiti su una regione: se il veicolo A incappa in un ingorgo o in condizioni meteo avverse, potrebbe tramite MQTT¹/V2V notificare immediatamente gli altri veicoli B, C, D dietro di lui, i quali adegueranno per tempo il percorso o la velocità. Questo crea un'intelligenza collettiva distribuita. Un caso particolare è il convoglio (platooning) di camion: se dotati di co-pilota comunicante, potrebbero coordinare le frenate e accelerazioni in maniera fuzzy[6] per mantenere distanze minime in sicurezza, riducendo drasticamente la resistenza aerodinamica e quindi il consumo di carburante per l'intero convoglio. Inoltre, l'integrazione con infrastrutture urbane (semafori intelligenti, pannelli a messaggio variabile) potrebbe arricchire ulteriormente i dati a disposizione: la piattaforma ODH era un esempio locale, ma in prospettiva, con le smart city, i veicoli potranno sapere in anticipo anche il timing dei semafori verdi (per ottimizzare l'approccio) o la disponibilità di parcheggi, ecc.

3. Validazione su mezzi reali e studi con utenti Un passo cruciale sarà testare il co-pilota su strada con conducenti veri. MinervaS ha già attivato progetti pilota con partner logistici (ad es. Trans Italia); utilizzare uno o più camion di queste flotte equipaggiandoli col sistema e raccogliere dati durante l'operatività quotidiana sarebbe estremamente utile. Si potrebbero condurre esperimenti controllati: ad esempio, far percorrere a un camion un certo tragitto con e senza il co-pilota attivo, confrontando metriche come il consumo di carburante, il tempo di viaggio, e - tramite telecamere - valutando il comportamento del

¹MQTT (Message Queuing Telemetry Transport) è un protocollo di messaggistica leggero basato su publish/subscribe, progettato per connessioni M2M (Machine-to-Machine) affidabili anche in reti con larghezza di banda ridotta o latenza elevata.

conducente (meno stressato? meno distratto?). Inoltre, interviste e questionari ai driver potrebbero fornire feedback qualitativo su quanto trovano utile e usabile il sistema, e se i consigli fuzzy vengono compresi e seguiti volentieri. Questo ciclo di feedback permetterà di migliorare l'interfaccia utente (es. forse alcuni avvisi andranno resi più chiari o dati con più anticipo) e di tarare eventualmente la base di regole fuzzy. L'obiettivo finale è che il co-pilota diventi un "compagno di fiducia" per chi guida: ciò si ottiene solo iterando con il coinvolgimento diretto degli utenti finali.

4. Integrazione di nuove fonti dati e moduli AI Il sistema è modulare, per cui in futuro si potranno aggiungere altre fonti informative. Ad esempio, l'integrazione con dati in tempo reale sul veicolo stesso: pressione dei pneumatici, temperatura dei freni, carico assiale, etc. - questi fattori possono entrare nelle regole fuzzy (es. "se carico molto elevato e pioggia, rischio aumenta"). Oppure l'uso di telecamere forward-looking per riconoscere oggetti sulla carreggiata (es. rilevare in automatico un ostacolo non segnalato) e integrare anche questo come evento. Un altro modulo potrebbe essere un navigatore evoluto: attualmente ci basiamo su servizi di routing esterni (OSRM²), ma MinervaS potrebbe sviluppare un proprio motore di calcolo percorso che, tenendo conto dei dati in tempo reale (traffico, meteo), proponga itinerari ottimi. Ad esempio, se su una certa autostrada imperversa un temporale, il co-pilota potrebbe suggerire all'autista un percorso alternativo leggermente più lungo ma più sicuro, quantificando anche il consumo extra vs il rischio evitato. Questo implicherebbe risolvere problemi di pathfinding multi-criterio (tempo vs sicurezza vs consumo) - area molto interessante per sviluppi futuri, in cui algoritmi di ottimizzazione e AI possono venire in aiuto.

5.6 Conclusione generale

Questa tesi ha gettato le basi per un co-pilota digitale integrato e intelligente, contribuendo al dominio degli ADAS di nuova generazione - gli **ADAS 2.0** - caratterizzati da cooperazione veicolo-cloud e intelligenza spiegabile. I risultati ottenuti confermano la fattibilità e l'efficacia dell'approccio: unendo dati aperti, modelli AI e conoscenza esperta, è possibile

²OSRM (Open Source Routing Machine) è un motore di routing open-source ad alte prestazioni basato su dati OpenStreetMap, progettato per calcolare percorsi stradali veloci tramite algoritmi di routing come Dijkstra o Contraction Hierarchies.

migliorare sensibilmente sicurezza ed efficienza del trasporto su strada. Le criticità evidenziate sul connettore ODH e le relative proposte di scalabilità delineano con chiarezza il percorso evolutivo necessario per passare dal prototipo locale a una piattaforma nazionale. **MinervaS** intende proseguire su questa strada, puntando a portare *ENAIDE* sul mercato come soluzione innovativa per flotte e costruttori di veicoli. Il riconoscimento nel *Clean Cities ClimAccelerator 2025* testimonia l'interesse verso queste tecnologie. Implementare su larga scala co-pilota intelligenti significherà non solo meno incidenti e consumi ridotti, ma anche un cambiamento culturale: la guida diventerà un'attività collaborativa uomo-macchina, dove l'AI supporta le decisioni e apprende dall'umano, mentre l'umano si fida dell'AI e la supervisiona.

In attesa di un futuro di veicoli totalmente autonomi, soluzioni come quella proposta rappresentano un passo intermedio fondamentale, capace di produrre benefici immediati e di preparare gradualmente l'ecosistema a livelli crescenti di automazione in modo sicuro.

Bibliografia

- [1] Samet Aksit, Akif Yavuz e Osman Taha Sen. «A Fuzzy Logic Approach on the Evaluation of Driving Styles and Investigation of Drivability Calibration Effects». In: *Gazi University Journal of Science* 35.2 (2022), pp. 668–680. DOI: [10.35378/gujs.862867](https://doi.org/10.35378/gujs.862867).
- [2] Automobile Club d'Italia. *Colpo di Sonno, un Killer che non Perdona*. Articolo su *L'Automobile*, 23 marzo 2016. 2016. URL: <https://archiviolautomobile.aci.it/articoli/2016/03/23/colpo-di-sonno-un-killer-che-non-perdona.html>.
- [3] Suzan Ayas, Birsen Donmez e Tang Xing. «Drowsiness Mitigation Through Driver State Monitoring Systems: A Scoping Review». In: *Human factors* 66 (nov. 2023), p. 187208231208523. DOI: [10.1177/00187208231208523](https://doi.org/10.1177/00187208231208523).
- [4] Kemal Ayyildiz et al. «Reducing fuel consumption and carbon emissions through eco-drive training». In: *Transportation Research Part F: Traffic Psychology and Behaviour* 46 (apr. 2017), pp. 96–110. DOI: [10.1016/j.trf.2017.01.006](https://doi.org/10.1016/j.trf.2017.01.006).
- [5] CGI. *Modeling the Relation Between Driving Behavior and Fuel Consumption*. Rapp. tecn. Analisi di clustering comportamentale su dati Scania. CGI White Paper, 2014. URL: https://www.cgi.com/sites/default/files/2022-09/driving_behavior_and_fuel_consumption_white_paper.pdf.
- [6] J. Dickerson, Hyun Mun Kim e B. Kosko. «Fuzzy Control for Platoons of Smart Cars». In: *Proceedings of the 3rd IEEE International Fuzzy Systems Conference*. 1994, pp. 1632–1637. DOI: [10.1109/FUZZY.1994.343940](https://doi.org/10.1109/FUZZY.1994.343940).
- [7] European Commission. *Road Safety Thematic Report: Main Factors Causing Fatal Crashes*. Rapp. tecn. European Road Safety Observatory, DG MOVE, 2024. URL:

- https://road-safety.transport.ec.europa.eu/document/download/a7428369-8eaf-4032-806e-ea08b46028c0_en.
- [8] Goran Ferenc et al. «Towards Enhanced Autonomous Driving Takeovers: Fuzzy Logic Perspective for Predicting Situational Awareness». In: *Applied Sciences* 14.13 (2024), p. 5697. DOI: [10.3390/app14135697](https://doi.org/10.3390/app14135697).
- [9] Ekaterina Gilman et al. «Personalised Assistance for Fuel-Efficient Driving: A Context-Aware Architecture». In: *International Journal of Automotive Technology and Management* 15.2 (2015), pp. 120–136. DOI: [10.1016/j.trc.2015.02.007](https://doi.org/10.1016/j.trc.2015.02.007).
- [10] ISTAT. *Incidenti stradali 2023*. Rapporto. Istituto Nazionale di Statistica, 2024. URL: <https://www.istat.it/it/files/2024/07/REPORT-INCIDENTI-STRADALI-2023.pdf>.
- [11] ISTAT. *Incidenti stradali 2023 — Infografica testuale*. Rapp. tecn. Istituto Nazionale di Statistica, 2024. URL: <https://www.istat.it/wp-content/uploads/2024/07/infografica-incidenti-stradali-2024.pdf>.
- [12] Ebrahim H. Mamdani e Setrag Assilian. «An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller». In: *International Journal of Man-Machine Studies* 7.1 (1975), pp. 1–13. DOI: [10.1016/S0020-7373\(75\)80002-2](https://doi.org/10.1016/S0020-7373(75)80002-2).
- [13] Rana Massoud et al. «A Fuzzy Logic Module to Estimate a Driver’s Fuel Consumption for Reality-Enhanced Serious Games». In: *International Journal of Serious Games* 5.4 (2018), pp. 45–62. DOI: [10.17083/ijsg.v5i4.266](https://doi.org/10.17083/ijsg.v5i4.266).
- [14] Luca Moretti. *Trasporti: Lampioni Intelligenti e Veicoli a Guida Autonoma nella Smart Road di ENEA*. 2023. URL: <https://www.media.enea.it/comunicati-e-news/archivio-anni/anno-2023/trasporti-lampioni-intelligenti-e-veicoli-a-guida-autonoma-nella-smart-road-di-enea.html>.
- [15] Catarina Rolim et al. «Real-Time Feedback Impacts on Eco-Driving Behavior and Influential Variables in Fuel Consumption in a Lisbon Urban Bus Operator». In: *IEEE Transactions on Intelligent Transportation Systems* 18.11 (2017), pp. 3061–3071. DOI: [10.1109/TITS.2017.2657333](https://doi.org/10.1109/TITS.2017.2657333).

- [16] Timothy J. Ross. *Fuzzy Logic with Engineering Applications*. 3^a ed. Chichester, UK: John Wiley & Sons, 2010. ISBN: 978-0-470-74376-8. DOI: [10.1002/9781119994374](https://doi.org/10.1002/9781119994374). URL: <https://onlinelibrary.wiley.com/doi/book/10.1002/9781119994374>.
- [17] Galit Toledo e Yoram Shiftan. «Can feedback from in-vehicle data recorders improve driver behavior and reduce fuel consumption?» In: *Transportation Research Part A: Policy and Practice* 94 (dic. 2016), pp. 194–204. DOI: [10.1016/j.tra.2016.09.001](https://doi.org/10.1016/j.tra.2016.09.001).
- [18] U.S. Federal Highway Administration. *Rain and Flooding — Road Weather Management Program*. https://ops.fhwa.dot.gov/weather/weather_events/rain_flooding.htm. 2023.
- [19] Zoe Visconti. *ADAS Safeguards Are Lacking across Auto Brands: IIHS*. 2024. URL: <https://www.teslarati.com/adas-safeguards-brands-iihs/>.
- [20] Hans Jakob Walnum e Morten Simonsen. «Does driving behavior matter? An analysis of fuel consumption data from heavy-duty trucks». In: *Transportation Research Part D: Transport and Environment* 36 (mag. 2015). DOI: [10.1016/j.trd.2015.02.016](https://doi.org/10.1016/j.trd.2015.02.016).
- [21] S. Wickramanayake, H. M. N. Bandara e N. A. Samarasekara. *Real-Time Monitoring and Driver Feedback to Promote Fuel Efficient Driving*. <https://arxiv.org/abs/2007.02728>. Preprint. 2020. arXiv: [2007.02728](https://arxiv.org/abs/2007.02728).
- [22] Chathurika Wickramasinghe et al. «Intelligent Driver System for Improving Fuel Efficiency in Vehicle Fleets». In: *IEEE International Symposium on Innovations in Intelligent Systems and Applications*. Giu. 2019, pp. 34–40. DOI: [10.1109/HSI47298.2019.8942624](https://doi.org/10.1109/HSI47298.2019.8942624).
- [23] Wikipedia contributors. *Sistema di Guida Autonoma — Livelli di Automazione SAE*. https://it.wikipedia.org/wiki/Sistema_di_guida_autonoma. Ultimo accesso: giugno 2025. 2024.
- [24] Lotfi A. Zadeh. «Fuzzy Sets». In: *Information and Control* 8.3 (1965), pp. 338–353. DOI: [10.1016/S0019-9958\(65\)90241-X](https://doi.org/10.1016/S0019-9958(65)90241-X).

Ringraziamenti

Naturalmente, il primo ringraziamento va alla mia famiglia.

Grazie **papi**, lo so che non ti prendi mai i meriti, ma sappi che è soprattutto grazie alle tue piccole attenzioni quotidiane se sono riuscito ad arrivare fin qui.

Grazie **mamma**, per avermi insegnato a puntare sempre più in alto, con la stessa determinazione con cui hai sempre sostenuto la nostra famiglia.

Grazie **Giò**, sei stata più di una sorella, sei la mia migliore amica da sempre. Spero di essere stato per te anche solo la metà di quello che sei stata per me.

Grazie **Lucas**, per esserci stato nel modo più naturale e sincero possibile.

Grazie **Biciu**, per avermi aiutato a modo tuo, tra morsi, *mauu* ed agguati. Sei riuscita a tranquillizzarmi come solo tu sai fare. Fai parte del capitolo più speciale della mia vita.

Un grazie speciale a *Nicola, Gorga, Giggi e Toto*. Grazie **Nicola**, per avermi sempre seguito e aiutato, anche quando la mia sospettabile inaffidabilità avrebbe giustificato l'opposto. Grazie **Fra** per essere stato sempre sincero con me, a volte anche troppo, ma con il tempo ho imparato ad apprezzare anche quello. Grazie **Gi**, ho sempre apprezzato la tua capacità di parlare liberamente di tutto ciò che ti passava per la testa. Vederti curioso delle cose noiose che studi mi ha fatto venire voglia di esserlo anch'io. Grazie **Toto**, da te ho imparato a non dare troppo peso alle apparenze. Il tuo modo di fare mi ha fatto capire che fidarsi di sé, a volte, è già abbastanza.

Grazie a **Anuar, Valeria e Vittorio**. In ognuno di voi ho riconosciuto una parte di me, siete stati d'ispirazione.

Troppissimi grazie ad **Arianna**, la mia amica di Analisi I, per la tua genuina semplicità, fondamentale quando c'erano troppi dubbi, angosce e perplessità. A **Debbi** per avermi aiutato a prenotare tutti gli appelli in tempo, la tua mancanza sarà tra le più preoccupanti. A **Gioia**, per avermi sorpresa superando ogni aspettativa, e per essere riuscita, allo stesso tempo, a mantenere sempre alto il nostro morale. A **Luigi** per aver tenuto sempre unito il gruppo, nonostante le tue difficoltà pre-esame. A **Maurizio** per avermi fatto ridere nei

giorni e le notti più stressanti dell'ultimo anno. A **Cristian** per essere riuscito ad alleggerire molte giornate. A **Luca** per avermi fatto capire quanto sia importante circondarsi di buoni amici in qualsiasi occasione. A **Mohamad** per essere proprio uno di quei buoni amici, spero di esserlo stato anche io per te. A **Francesco** per avermi intrattenuto con qualsiasi argomento lo appassionasse, ho imparato molto da te. A **Omar** per avermi mostrato come riuscire a gestire anche le situazioni più disperate con leggerezza.

Ringrazio **Angelo, Anja, Alessandra, Alessandro, Irene, Federico, Giorgia, Giovanni, Jacopo, Luigi, Lukasz, Samuel, Simone, Ubaldo** ed *Eric Smith*, per avermi aiutato a capire chi sono.

Ringrazio **Salvatore** innanzitutto per la pazienza che hai avuto. Grazie a te ho imparato a dare valore a ciò che normalmente si tende a ignorare, anche nelle persone.

La persona più responsabile di aver reso possibile tutto questo è **Elettra**. Gli ultimi tre anni sono stati i più belli della mia vita: abbiamo viaggiato, riso, dormito tanto, pianto e, per fortuna, anche studiato. Ti ringrazio per ogni parola di conforto, per ogni abbraccio, per tutte le volte in cui sei stata il mio unico punto di riferimento quando mi sentivo perso, e per avermi dato il coraggio e la serenità di vivere senza rimpianti. Non vorrei mai dimenticare nulla di tutto questo. Il mio desiderio più grande è che i ricordi che abbiamo vissuto, e quelli che ancora ci aspettano, riescano a tenerci uniti anche quando le città in cui vivremo ci sembreranno troppo lontane.

Tralasciando il problema iniziale che hai avuto con i *notebook*, so bene quante difficoltà hai dovuto affrontare lungo il percorso. Proprio per questo sono profondamente fiero ed orgoglioso di te, non sottovalutarti mai.

Sei il mio più grande *Unfair Advantage*, faccio fatica solo a pensare come avrei fatto tutto questo senza di te. Grazie per completare la mia vita.¹

Ringrazio di cuore la *famiglia e gli amici di Elli*. Avervi conosciuti ha reso tutto più significativo ed importante. Un grazie speciale a **Rosanna**, per aver sempre creduto in me e per avermi fatto sentire parte della vostra famiglia. Mi hai aiutato nei momenti in cui ne avevo più bisogno, spesso senza nemmeno accorgertene. Grazie anche a **Nello**, che, pur non avendo avuto un allievo perfetto, ha saputo essere un ottimo mentore.

Grazie ai miei **nonni, zii e cugini**, per aver contribuito a tirar fuori il meglio di me.

¹Continueremo ad ascoltare *Scream & Shout* prima di ogni esame?

- Ecco una versione migliorata dei ringraziamenti per la tua tesi. Fammi sapere se vuoi un tono più formale o ancora più scherzoso!

- E ora? Abbiamo finito quindi?

- Quasi. Forse dovresti lasciare spazio anche a chi oggi è qui con te.

- Hai ragione.

Questa pagina è vostra.

Scriveteci un ricordo, un pensiero,

una dedica o anche solo una firma.

Sarà il modo più bello per non dimenticarvi.

