UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Sviluppo di una Faster R-CNN per il riconoscimento di oggetti in condizioni meteo avverse a supporto del sistema di percezione delle auto a guida autonoma

RELATORE

Prof. Fabio Palomba

Dott. Giammaria Giordano

Dott.ssa Giusy Annunziata

Università degli Studi di Salerno

CANDIDATO

Mario Cicalese

Matricola: 0512110051

Anno Accademico 2022-2023

Questa tesi è stata realizzata nel

"How big would you dream if you knew you couldn't fail?"

Abstract

Realizzato per la prima volta nel 1995 dalla CMU (Carnegie Mellon University), il primo veicolo a guida autonoma fu in grado di completare un viaggio di 5,000 km da Pittsburgh a San Diego. Il veicolo era in grado di sterzare autonomamente tramite l'utilizzo di reti neurali, sensori e telecamere, mentre, l'acceleratore e il freno, erano controllati da un conducente umano. Dopo quasi 30 anni, i grandi progressi informatici e tecnologici, fatti durante questi decenni, stanno spingendo le aziende automobilistiche alla produzione, su scala globale, di veicoli a guida autonoma. Nell'ambito di questo lavoro di tesi viene introdotto e approfondito il concetto di veicolo a guida autonoma. Inizialmente vengono definiti concetti di base, come: la definizione di un veicolo a guida autonoma e i 6 livelli SAE, con esempi e spiegazioni dei sistemi ADAS e ADS. Viene poi presentata una dettagliata spiegazione della tipica architettura di un veicolo a guida autonoma, composta da vari tipi di sensori e dal sistema di percezione e di decisione. Infine, vengono introdotte le architetture informatiche basate su tecniche di Intelligenza Artificiale; queste si trovano alla base del sistema di percezione e di decisione e sono utilizzate per percepire l'ambiente intorno al veicolo, modificando il suo comportamento di conseguenza. In particolare, vengono approfondite architetture appartenenti alla branca del Deep Learning, come reti neurali e reti convulazionali. L'obiettivo del lavoro di tesi è: implementare, nonchè allenare, una Faster R-CNN su dataset di immagini in condizioni meteo avverse. Il compito della Faster R-CNN è quello di identificare (Object detection) correttamente gli oggetti statici e dinamici presenti nell'ambiente nonostante le condizioni avverse, al fine di migliorare la sicurezza su strada. Per l'implementazione e l'allenamento è stato utilizzato il framework PyTorch che mette a disposizione modelli pre-addestrati e metodi per allenare e valutare i modelli. Le performance del modello vengono poi misurate tramite l'utilizzo di metriche di valutazione: Avergae IoU, Average Recall, Mean Average Precision (mAP) e matrice di confusione. I risultati ottenuti in fase di validazione sono stati: 0.60 di Average IoU, 0.53 di Average Recall, 0.45 di mAP, 1779 veri positivi, 1103 falsi positivi e 1666 falsi negativi

Indice

Elenco delle figure

Elenco delle tabelle

CAPITOLO 1

Introduzione

1.1 Contesto Applicativo

Un veicolo a guida autonoma è un veicolo che è in grado di viaggiare senza un coinvolgimento umano grazie all'utilizzo di un vasto numero di sensori e tecnologie hardware e software sempre più all'avanguardia. Grazie all'introduzione di questa tipologia di veicoli stiamo vivendo una rivoluzione nell'ambito automobilisco con automobili che non solo sono in grado di navigare autonomamente, ma possono anche interagire con altri utenti della strada, prendere decisioni in tempo reale, e soprattutto, garantire una guida sicura e efficiente. L'obiettivo delle aziende automobilistiche è quello di produrre sul libero mercato tale veicolo entro il 2030.
Indipendentemente dall'azienda automobilistica produttrice, l'architettura delle auto a guida autonoma è sempre la medesima, divisa in: sistema di percezione e sistema decisionale. Il sistema di percezione ha lo scopo di creare una rappresentazione in tempo reale dell'ambiente intorno al veicolo tramite i dati e immagini catturate dai sensori installati sul veicolo. Il sistema decisionale, in base ai dati forniti dal sistema di percezione, sarà responsabile del comportamento del veicolo. Entrambi i sistemi sono composti da vari sottosistemi, ognuno dei quali ha il proprio compito e la propria utilità al fine di un corretto comportamento del veicolo. Considernado il

sistema di percezione, esso è composto da vari sottositemi tra cui: il sottosistema di tracciamento degli oggetti in movimento (MOT) e il sottosistema di rilevazione della segnaletica stradale (TSD). Entrambi i sottosistemi utilizzando algoritmi di Deep Learning che sono in grado di riconoscere gli oggetti dinamici e statici presenti all'interno delle immagini catturate dai sensori presenti sul veicolo. Il rilevamento degli oggetti è fondamentale in tale ambito, poichè, il sistema decisionale, in base agli oggetti rilevati dal sistema di percezione, modificherà le traiettorie del veicolo al fine di evitare collissioni ed aumentare la sicurezza. I modelli di Machine Learning, per loro natura, hanno bisogno di dati su cui allenarsi per imparare ad effettuare previsioni. Tuttavia, non esiste una grande disponibilità di dati in determinate condizioni meteo.

1.2 Motivazione e Obiettivi

Yuxiao Zhang et al. [1] hanno dimostrato che le condizioni meteorologiche avverse rappresentano un ostacolo all'implementazione dei sistemi ADS e dei sistemi di Deep Learning basati sull'object detection. Il motivo è legato alla limitata visibilità che si ha dell'ambiente in condizioni meteo avverse come: pioggia, neve e nebbia. Una limitata visibilità si ha anche di notte, di conseguenza, è stata considerata come quarta condizione avversa. Inoltre, gli stessi autori, dimostrano che la maggior parte dei dataset di immagini disponibili in rete contengono nella gran maggioranza dei casi condizioni meteo ottimali, con una presenza limitata o assente di condizioni meteo avverse. Questa presenza minima di dati rappresenta un enorme problema per i modelli di Machine Learning, poichè, un numero limitato di dati potrebbe portare il modello a non imparare correttamente, andando a effettuare predizioni non corrette e non rilevando oggetti fondamentali, come: pedoni, ciclisti, motociclisti e altri veicoli, con conseguenze gravi alla propria sicurezza e quella altrui.
L'obiettivo di questo lavoro di tesi sarà inanzitutto quello di trovare e utilizzare un dataset di immagini che si basi su condizioni meteo avverse. Successivamente l'obiettivo sarà quello di implementare una Faster R-CNN, cioè, un modello di Deep Learning basato sull'object detection, al fine di rilevare tutti gli oggetti, dinamici e statici, presenti all'interno delle immagini del dataset (pedoni, veicoli, segnali

stradali, semafori e altri utenti della strada). Tale modello verrà implementato con lo scopo di migliorare il rilevamento degli oggetti in condizioni meteo avverse al fine di migliorare la sicurezza in strada.

1.3 Risultati Ottenuti

I risultati ottenuti possono essere ritenuti soddisfacenti, considerando la poca mole di dati presenti in rete e gli errori fatti sulle annotazioni di alcune classi dai creatori del dataset utilizzato. Tuttavia, il modello ha ancora un ampio margine di miglioramento. Inizialmente, i risultati ottenuti dalle metriche di valutazioni erano preoccupanti, con un alto numero di falsi positivi predetti dal modello e con un valore basso dell'Average IoU. Questo accadeva perchè l'output del modello non era stato filtrato, considerando anche predizioni con un score di confidenza basso. Successivamente, l'output è stato filtrato considerando solo predizioni dove il modello fosse sicuro delle proprie scelte, con un netto miglioramento dei risultati.

1.4 Struttura della tesi

Il seguente lavoro di tesi è divisa in 5 capitoli ognuno con lo scopo di presentare e descrivere in modo approfondito diversi aspetti della ricerca effettuata e della tecnica utilizzata. Di seguito è riportata una breve descrizione del contenuto dei capitoli:

- Nel capitolo 1 viene fatta un introduzione e una panoramica sull'intero lavoro svolto specificando le motivazioni per il quale si è decisi di intraprendere questo lavoro e gli obiettivi che si vogliono raggiungere il termine.
- Nel capitolo 2 verrà analizzato lo stato dell'arte delle auto a guida autonoma, partendo dalla loro definizione. Successivamente verranno introdotti i 6 livelli SAE e l'architettura software e hardware dei veicoli a guida autonoma. Al termine del capitolo si farà una parentesi sulle architetture informatiche che si utilizzano per realizzare tali veicoli, facendo riferimento in particolare a modelli di Deep Learning come reti neurali e reti convulazionali. Il capitolo si conclude parlando del reserch gap, cioè, dei problemi che sono presenti in letteratura.

- Nel capitolo 3 il focus è sull'implementazione della Faster R-CNN. Inizialmente si concentrerà sui dati raccolti e sulle operazioni preliminari fatti su di essi (Data Preparation e Data visualization). Successivamente si passerà al cuore dell'implementazione, dove si discuterà di tutte le scelte implementative fatte con relative motivazioni. Il capitolo termina facendo riferitmento agli algoritmi di ottimizzazione utilizzati e di come è stato addestrato il modello.

- Nel capitolo 4 sono presenti i risultati ottenuti dal modello sul set di validazione. Si andranno quindi a giudicare le performance del modello analizzando metriche di valutazione utilizzate per calcolare le sue prestazioni. Al termine del capitolo verranno analizzat i risultati, andando a determinare i punti di forza e di debolezza del modello e le motivazioni per il quale non si comporta in maniera ottimale.

- Nel capitolo 5, invece, sono presenti le conclusioni e gli sviluppi futuri. Inizialmente è presente un piccolo recap del lavoro svolto e dei risultati ottenuti, per poi discutere dei possibili miglioramenti da intraprendere in futuro per migliorare le performance del modello implementato e in generale, dei modelli di Deep Learning basati sull'object detection.

CAPITOLO 2

Stato dell'arte

2.1 Definizione di auto a guida autonoma

Un'auto a guida autonoma è un veicolo a motore capace di viaggiare da una destinazione A a una destinazione B in maniera autonoma, con un coinvolgimento del conducente limitato o assente. Un auto, per essere qualificata come pientamente autonoma, dev'essere in grado di viaggiare senza il coinvolgimento del conducente, dovendo quindi, essere capace di gestire tutte le possibili situazioni in cui il veicolo può ritrovarsi [2].
Le auto a guida autonoma si affidano a sistemi avanzati di Intelligenza Artificiale e di Machine Learning per "capire" l'ambiente circostante e gli scenari in cui essi si trovano, reagendo in maniera razionale ad ogni situazione, con l'obiettivo di minimizzare il più possibile le condizioni di rischio. Una combinazione di sensori, telecamere, radar e lidar vengono utilizzati per: creare una mappa dell'ambiente circostante costantemente aggiornata, rilevare la presenza di veicoli,pedoni e ostacoli vicini, misurare distanze, rilevare segnali stradali, semafori, dossi, marciapiedi, etc... [3]
I veicoli autonomi possono anche essere interconnessi con altri dispositivi esterni e infrastrutture intelligenti delle Smartcity come: il proprio smartphone, semafori

e strade intelligenti, che attualmente, sono più un concept piuttosto che la realtà. Infatti, proprio la mancanza di smart city o di strade tecnologicamente adatte a veicoli di questo genere, rappresentano il motivo principale per il quale le auto non sono completamente a guida autonoma, avendo il costante bisogno della supervisione e del coinvolgimento da parte del conducente per riprendere il controllo del veicolo.

2.2 Classificazione delle auto a guida autonoma

Esistono differenti livelli di automazione di un veicolo; essi dipendono dalle azioni che l'auto è in grado di svolgere autonomamente e sul livello di coinvolgimento umano richiesto per il controllo e la gestione delle sue funzionalità. L'automazione di un auto dipende dal tipo e dalla quantità di sistemi di assistenza alla guida che sono installati su di essa.
A classificare i livelli di automazione delle auto è stata la SAE International, un'associazione globale di oltre 128000 ingegneri ed esperti tecnici nei settori aerospaziale, automobilistico e dei veicoli commerciali i quali standard sono utilizzati per far progredire l'ingegneria della mobilità in tutto il mondo. La SAE nel 2014 pubblica il "SAE Levels of Driving Automation (SAE J3016)", un documento nel quale, com'è possibile osservare dalla Figura 2.1, suddivide i veicoli a guida autonoma in 6 differenti livelli. SAE J3016 definisce tali livelli dal livello 0 (nessuna automazione) al livello 5 (veicolo completamente autonomo) nel contesto dei veicoli a motori e del loro funzionamento sulle strade. Tale documento è globalmente riconosciuto ed è in continua fase di aggiornamento poichè la ricerca in tale ambito è ancora in fase di sviluppo [4] .
Di seguito è riportata la spiegazione di tutti i livelli con i relativi nomi assegnati dalla SAE e per ognuno di esso, esempi di sistemi più utilizzati e comuni sul mercato.

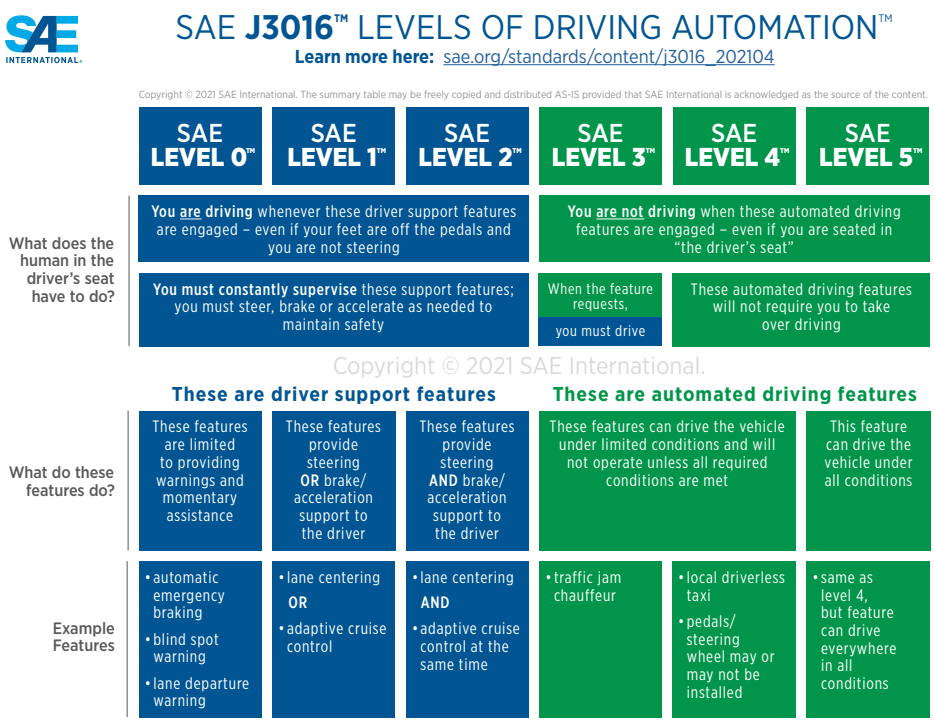

Figura 2.1: Livelli di automazione pubblicati dalla SAE.[1]

2.2.1 Livello 0 (Nessuna Automazione)

l'auto non ha nessun sistema di automazione alla guida, ma ciò non implica che non ci siano sistemi di supporto alla guida. Infatti l'auto potrebbe essere equipaggiata con sistemi di supporto alla guida che aiutano il conducente avvisandolo dei pericoli e fornendo assistenza. Essi, però, non sono in grado di interagire sul comportamento del veicolo, quindi non hanno nessuna automazione. L'esperienza di guida, quindi, è al 100% gestita dal conducente, il quale sarà l'unico ad avere influenza sul comportamento del veicolo [5] [6]. Esempi di sistemi classificati nel livello 0 sono : Avviso di deviazione della corsia, monitoraggio dell'angolo cieco, avviso di collisione frontale, freno di emergenza automatica e tanti altri.

- **Avviso di deviazione della corsia (Lane Departure Warning)**: L'avviso di deviazione della corsia è stato progettato con lo scopo di evitare incidenti dovuti

[1] `https://www.sae.org/blog/sae-j3016-update`

al drift dell'auto o al cambio corsia non intenzionale dovuto dalla distrazione del conducente. Tale sistema, tramite l'utilizzo di una telecamera posizionata vicino allo specchio retrovisore interno, è in grado di rilevare le linee di corsia che si trovano alla sinistra e alla destra del veicolo. Così come mostra la Figura 2.2, nel momento in cui una delle ruote tocca tale linea, il sistema avvisa il conducente tramite una spia gialla nel cruscotto (come in Figura 2.3) e/o un suono di beep. In alcuni sistemi, il volante o il sedile del conducente emettono una breve vibrazione. Generalmente, tale sistema non si attiva quando gli indicatori di direzione sono attivi, perchè in quel caso, il cambio di corsia è intenzionale [7] [8].

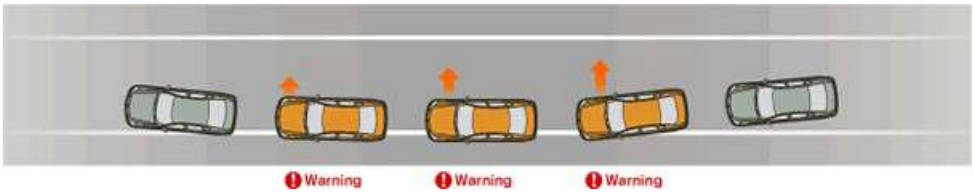

Figura 2.2: Scenario in cui il sistema si attiva[2]

Figura 2.3: Spia gialla visualizzata sul cruscotto [3]

- **Monitoraggio dell'angolo cieco (Blind Spot Warning)**: Un angolo cieco è una parte di strada la cui visibilità è ostruita da un ostacolo o non è riflessa da uno degli specchietti retrovisori. Infatti, gli specchietti retrovisori, non sono in grado di fornire una visuale completa di tutto ciò che si trova dietro o di fianco al veicolo. La porzione di ambiente che non rientra nello specchio retrovisore è definita anche come angolo morto e sono molto pericolosi in fase di: sorpasso, di parcheggio o di cambio corsia.

[2] https://www.nissan-global.com/EN/INNOVATION/TECHNOLOGY/ARCHIVE/LDW/
[3] https://rb.gy/ksws

Per questo motivo è stato inventato il Monitoraggio dell'angolo cieco che, tramite dei sensori posizionati nei lati posteriori del veicolo, è in grado di rilevare veicoli o ostacoli, che come in Figura 2.4, sono presenti nei punti ciechi dell'auto. Il sistema avvisa il conducente tramite una spia luminosa, visibile in Figura 2.5, sullo specchio retrovisore del lato in cui è stato rilevato l'ostacolo. Se gli indicatori di direzioni sono attivi, il sistema avviserà il conducente tramite un suono continuo, o nei sistemi più avanzati, tramite la vibrazione del volante, per evitare l'imminente collisione [9].

Figura 2.4: Funzionamento del Blind Spot Warning[4]

Figura 2.5: Spia luminosa presente sullo specchietto retrovisore[5]

- **Avviso di collisione frontale (Forward collision warning)**: tale sistema utilizza un Radar posizionato anteriormente al veicolo per rilevare la distanza con il veicolo che lo precede e la sua velocità, in modo da avvisare il conducente se il veicolo si avvicina troppo, tramite un suon di beep e una spia visiva, con lo scopo di evitare collisioni [10]. La figura 2.6 mostra il momento in cui il sistema entra in uno stato di allerta a causa della poca distanza con il veicolo avanti.

[4] https://mycardoeswhat.org/safety-features/blind-spot-warning/
[5] https://www.kbb.com/car-advice/blind-spot-monitors/

Figura 2.6: Funzionamento del Forward Collision Warning[6]

2.2.2 Livello 1 (Guida assistita)

rappresenta il livello più basso di automazione alla guida, dove l'auto è equipaggiata da un singolo sistema di automazione per assistere il conducente durante la guida. Rispetto al livello 0, il conducente non è l'unico a poter influenzare il comportamento dell'auto, infatti, il sistema di automazione è in grado di controllare singolarmente l'accelerazione, il freno o il volante del veicolo se viene rilevata una situazione di pericolo. La guida rimane comunque compito del conducente [5] [6]. Esempi di sistemi classificati nel livello 1 sono : Cruise Control adattivo, assistente del mantenimento corsia e assistente del cambio di corsia.

- **Cruise Control Adattivo (Adaptive Cruise Control)** : è un sistema di sicurezza attivo che automaticamente controlla l'accelerazione e la frenata del veicolo mantenendo una distanza di sicurezza dai veicoli che precedono e rispettando i limiti di velocità. Viene attivato dal conducente da un apposito pulsante sul volante [11]. Il conducente, una volta attivato, dovrà definire una velocità di crociera costante e la distanza di sicurezza minima con il veicolo di fronte in modo che il sistema sia in grado di controllare autonomamente l'automobile in termini di accelerazione e frenata (il volante viene controllato completamente dal conducente). Il sistema, tramite l'utilizzo di sensori, laser e radar, è in grado di misurare la distanza e la velocità del veicolo che precede, in modo da poter adattare la velocità di crociera [12]. Infatti, com'è possibile osservare dalla

[6] https://www.silkohonda.com/what-is-forward-collision-warning/

Figura 2.7, se la distanza di sicurezza con il veicolo di fronte viene rispettata, allora il sistema viaggia esattamente alla velocità definita dal conducente. Ma nel momento in cui, s'incontra un veicolo più lento, e quindi, la distanza di sicurezza minima non viene più rispettata, il sistema inizierà a frenare gradualmente l'auto in modo da ricreare la giusta distanza di sicurezza tra i due veicoli. Nel momento in cui il veicolo di fronte aumenterà la propria velocità, o semplicemente, cambierà corsia, la distanza tra i due veicoli aumenterà, e di conseguenza, il sistema aumenterà gradualmente la velocità dell'auto in modo da tornare alla velocità fissata dal conducente.

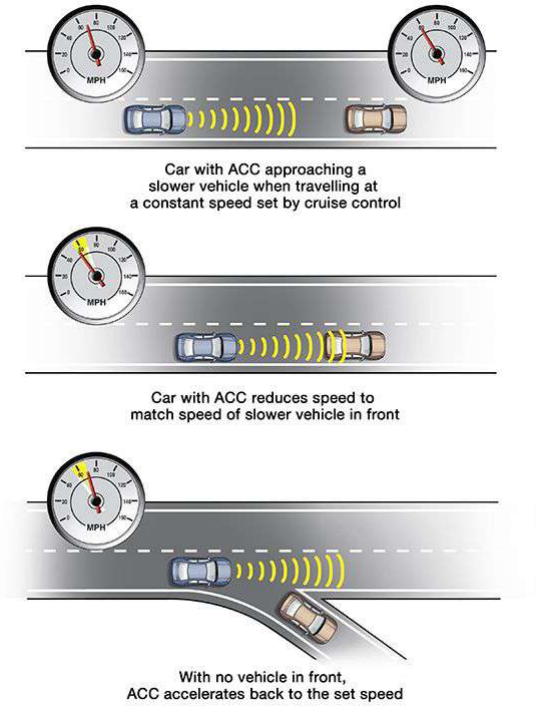

Figura 2.7: Funzionamento dell' Adaptive Cruise Control [7]

- **Assistente del mantenimento corsia (Lane Keep Assist)** : rappresenta l'evoluzione dell'Avviso di deviazione della corsia (Lane Departure Warning) visto nel livello precedente. Il funzionamento è lo stesso, con la differenza che, il sistema visto precedentemente si limitava ad avvisare il conducente, questo invece, è in grado di controllare autonomamente il volante del veicolo in modo da posizionarlo correttamente all'interno delle linee di corsia. Questo sistema

[7] https://mycardoeswhat.org/deeper-learning/adaptive-cruise-control/

oppone una leggera resistenza anche nel caso in cui sia il guidatore a voler cambiare corsia senza utilizzare l'indicatore di direzione, mentre l'attivazione della freccia rende il Lane Keep Assist semplice spettatore della manovra [13]. La Figura 2.8 mostra l'esatto momento in cui il sistema si attiva, modificando la posizione del volante, in modo da riposizionare il veicolo correttamente fra le linee di corsia

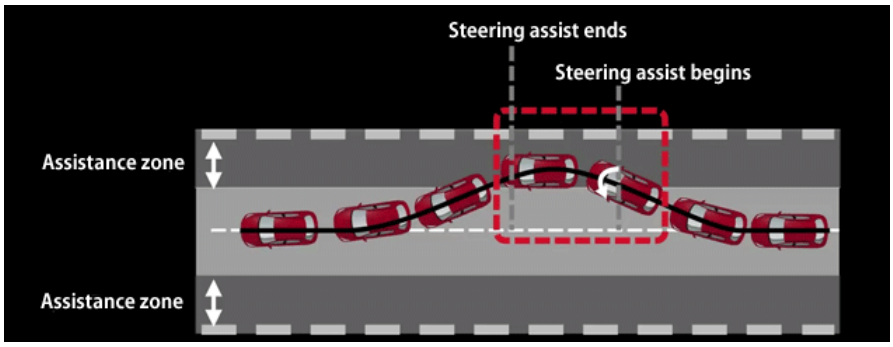

Figura 2.8: Funzionamento del Lane Keep Assist [8]

- **Assistente del cambio corsia (Lane Change Assist)** : è un sistema che aiuta il conducente a cambiare corsia in sicurezza. Il suo funzionamento è molto simile al Monitoraggio dell'angolo cieco (Blind Spot Warning), infatti, tramite dei sensori, è in grado di rilevare veicoli o ostacoli nei punti ciechi del veicolo. Il Lane Change Assist, nel momento in cui il conducente sta avviando una manovra di cambio corsia, come in Figura 2.9, è in grado di calcolare se i veicoli vicini all'auto potrebbero rappresentare un pericolo in base alla loro posizione e alla velocità [14]. In tal caso il sistema avviserà il conducente tramite un suono di beep continuo e interverrà automaticamente sul volante del veicolo, effettuando una correzione della traiettoria, per evitare una collisione con il veicolo che sta per sopraggiungere.

[8] https://www.mazda.com/en/archives/safety2/active_safety/las/

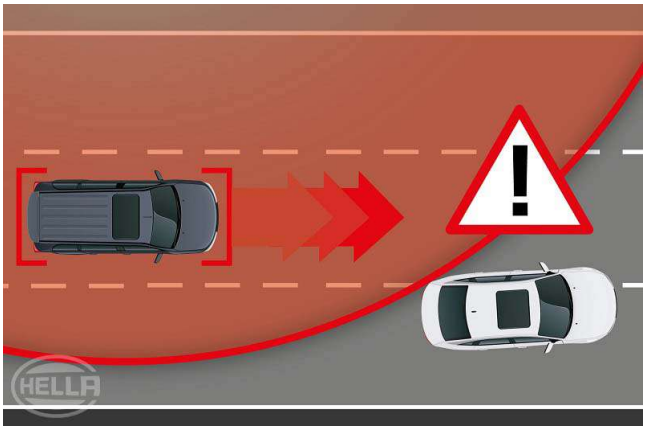

Figura 2.9: Funzionamento del Lane Change Assist [9]

2.2.3 Livello 2 (Automazione parziale)

i veicoli che rientrano in questa categoria sono dotati di un sistema avanzato di assistenza alla guida (ADAS) che è in grado di controllare costantemente e contemporaneamente sia il volante, che l'accelerazione e decelerazione del veicolo; esso, quindi, è in grado di eseguire sia movimenti laterali che longitudinali. Il conducente può riprendere il controllo del veicolo immediatamente su richiesta, disattivando il sistema.
Esempi di sistemi classificati nel livello 2 sono : cruise control adattivo integrato con il mantenimento della corsia, Highway Assist e Traffic Jam Assist.

- **Highway Assist** : è un sistema di guida assistita fino a 150 km/h attivabile su Autostrada o strade simili. Funzionante dall'entrata all'uscita, su tutte le corsie, inclusa quella di sorpasso. Esso, come mostrato in Figura 2.10 assiste il guidatore mantenendo una distanza di sicurezza con il veicolo che precede e mantenendo una velocità costante definita (Adaptive Cruise Control), mantenendo il veicolo al centro della propria corsia (Lane Keep Assist) e aiutandolo nel cambio corsia quando gli indicatori di direzione sono attivi (Lane Change Assist). Il conducente deve attivare manualmente il sistema e monitorarlo costantemente con le mani sul volante. Il conducente può, in qualsiasi momento, bypassare il sistema o disattivarlo [15].

[9] `https://t.ly/lHmLj`

Il sistema fa uso di un insieme di sistemi visti nel livello precedente, che però, essendo utilizzati contemporaneamente, sono in grado di controllare il veicolo sia longitudinalmente che lateralmente,di conseguenza, viene classificato come livello 2. Highway Assist non è l'unico sistema ad utilizzare una combo di sistemi di livello più basso, infatti, d'ora in poi, tutte le tecnologie saranno costruite utilizzando più tecnologie di livello inferiore.

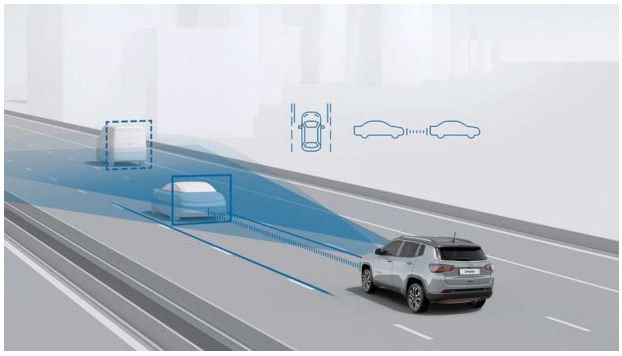

Figura 2.10: Funzionamento del Highway Assist [10]

- **Traffic Jam Assist** : è un sistema di assistenza alla guida attivabile in condizioni di traffico (come moastra la Figura 2.11)intenso a una velocità massima di 60 km/h. Il sistema, quando attivo, tramite l'utilizzo di sensori, camere e radar e tecnologie come Lane assist e Adaptive Cruise Control, sarà in grado di "seguire" il veicolo che lo precede, mantenendo una distanza di sicurezza e azionando automaticamente i freni, fino allo stop totale, quando l'auto avanti sta rallentando [16]. Nel momento in cui, l'auto che precede, partirà e aumenterà progressivamente la propria velocità, il sistema farà lo stesso, finchè la velocità di 60 km/h non sarà raggiunta. In quel caso il sistema si disattiverà automaticamente poichè non ci sarà più la condizione di traffico intenso.

  Il sistema sostanzialmente, aiuta il conducente quando è bloccato nel traffico, accelerando e frenando automaticamente, evitandogli le ripetitive false partenze.

[10] `https://www.jeep-official.it/jeep-compass/motore-termico/sicurezza-protezione`

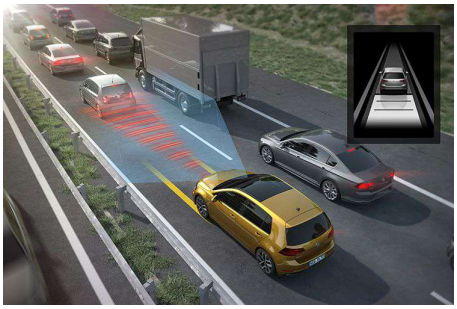

Figura 2.11: Funzionamento del Traffic Jam Assist [11]

2.2.4 Livello 3 (Automazione condizionale)

il salto dal livello 2 al livello 3 è sostanziale da un punto di vista tecnologico, ma sottile da un punto di vista umano [6]. I veicoli classificati in questo livello, hanno una capacità di rilevamento ambientale e possono prendere decisioni in base alle condizioni dell'ambiente circostante, come per esempio : traffico, ostacoli o altri aspetti [5]. Quando sono attivi sono in grado di guidare l'auto, ma richiedono ancora una supervisione umana, infatti, il conducente può immediatamente riprendere il controllo del veicolo se vuole o se gli viene richiesto (in caso di malfunzionamento o incapacità di prendere una decisione sicura da parte del sistema).
Esempi di sistemi classificati nel livello 3 sono : Highway Chauffeur e Traffic Jam Chauffeur.

- **Highway Chauffeur** : è un sistema di guida autonomo funzionante in autostrada o strade simili solo in determinate condizioni ed entro i 130 km/h di velocità. In funzione dall'entrata all'uscita, su tutte le corsie, inclusa quella di sorpasso. Esso rappresenta una versione più avanzata dell'Highway Assist del livello precedente, infatti, oltre che a fornire le stesse funzioni di assistenza alla guida, questo sistema sarà in grado di effettuare cambi di corsia e sorpassi autonomamente. La Figura 2.12 mostra tutte le operazioni che è in grado di svolgere.

  Il conducente deve attivare il sistema, ma non deve monitorarlo, avendo la possibilità di rilasciare il volante. Il conducente può in ogni momento bypassare

[11] `https://www.torque.com.sg/features/how-does-traffic-jam-assist-work/`

il sistema o disattivarlo. In determinate condizioni come: scarsa visibilità o strada scivolosa, il sistema può chiedere al conducente di riprendere il controllo dell'auto e quest'ultimo avrà tempo marginale sufficiente per orientarsi e riprendere la guida. Nel caso in cui il conducente non rispondesse alla richiesta, il sistema andrà a ridurre le condizioni di rischio fermando il veicolo in sicurezza [15]. Se possibile, in base al traffico e alla capacità del sistema, la riduzione delle condizioni di rischio possono includere operazioni come il cambiamento di corsia per fermare il veicolo sulla corsia di emergenza o al lato della strada.

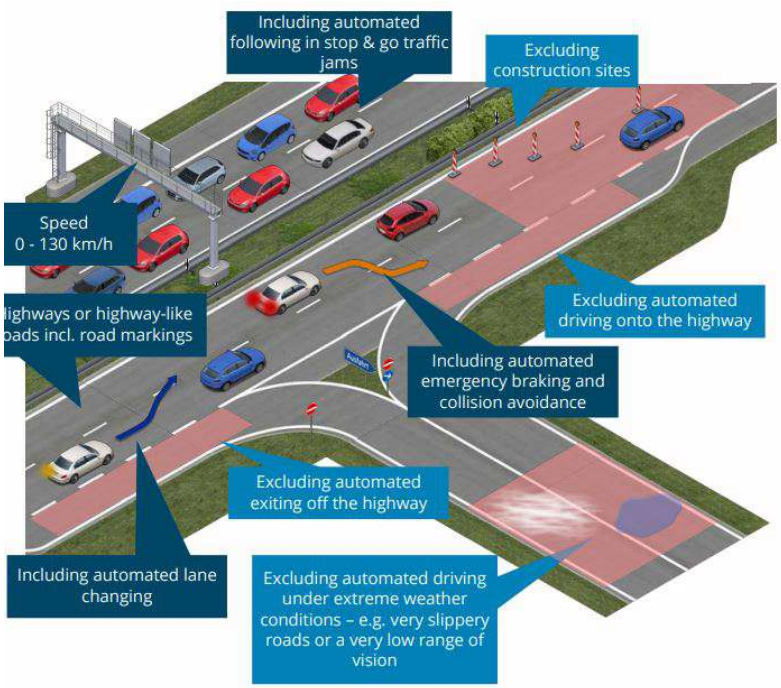

Figura 2.12: Funzionamento Dell'Highway Chauffeur [12]

- **Traffic Jam Chauffeur** : è un sistema di guida autonoma attivabile in condizioni di traffico intenso a una velocità massima di 60 km/h in autostrada o strade simili. Esso rileva un veicolo lento che ci precede e automaticamente lo seguirà, controllando i movimenti laterali e longitudinali del veicolo. Le ultime versioni di questo sistema potrebbero includere la funzionalità di cambiamento automatico della corsia. Il conducente deve attivare il sistema, ma non deve monitorarlo costantemente e può bypassarlo o disattivarlo in qualsiasi momen-

[12] https://www.pegasusprojekt.de/files/tmpl/Pegasus-Abschlussveranstaltung/04_The_Highway_Chauffeur.pdf

to [15]. Come nell'Highway Chauffer il sistema può richiedere al conducente di riprendere il controllo della guida, e se tale richiesta venisse ignorata, il sistema fermerà il veicolo in un posto sicuro per ridurre le condizioni di rischio.

2.2.5 Livello 4 (Alta automazione)

Rispetto al livello 3,i veicoli presenti nel livello 4 sono equipaggiati con un sistema che è in grado di intervenire in caso di malfunzionamento, senza coinvolgere il conducente. L'automazione è alta, infatti, il sistema è in grado di guidare il veicolo in totale autonomia (guida autonoma), ma solo in determinate condizioni legislative e geografiche, come aree urbane in cui i limiti di velocità sono più bassi. Il conducente è sempre in grado di riprendere il controllo del veicolo, disattivando il sistema di guida autonoma [6].
Esempi di sistemi classificati nel livello 4 sono : Highway Autopilot, Urban e Suburban Pilot e Valet Parking.

- **Highway Autopilot** : è un sistema guida altamente automatizzata in autostrada o strade simili ad una velocità massima di 130 km/h. Il sistema è in grado di guidare su queste strade in maniera altamente autonoma. Infatti, viene realizzato includendo tutte le funzionalità offerte dall'Highway Chauffeur e dal Traffic Jam Chauffeur del livello precedente. Il sistema deve sempre essere attivato dal conducente, non dev'essere monitorato e può essere disattivato in qualsiasi momento. Rispetto all'Highway Chauffeur, il sistema non chiederà al conducente di prendere il controllo del veicolo quando esso si troverà nella propria area di funzionamento (esempio : autostrada). In caso di non risposta da parte del conducente, il sistema è in grado di lasciare l'autostrada e di parcheggiare in un posto sicuro.

- **Urban e Suburban Pilot** : è un sistema guida altamente automatizzata in strade urbane o suburbane in funzione fino ai 50 km/h. Tale sistema può essere attivato dal conducente in tutte le condizioni di traffico e sarà in grado di guidare in maniera altamente autonoma. Il conducente può bypassare o disattivare il sistema in ogni momento.

- **Automated Valet Parking** : è una funzione di parcheggio automatica con una velocità operativa di circa 5/10 km/h. Il conducente può uscire dal veicolo e attivare la funzione di parcheggio automatico. Il veicolo inizierà a muoversi autonomamente verso il parcheggio di destinazione e sarà in grado di gestire la maggior parte delle condizioni di errore. Il conducente può comunque gestire o disattivare il sistema da remoto tramite per esempio un APP installata sul proprio Smartphone. Inoltre il veicolo informerà il conducente se non è in grado di completare le manovre di parcheggio a causa di specifiche ragioni. Ad oggi, i sistemi commerciali relativi ai parcheggi, vengono classificati di livello 2. Tale sistema non è ancora disponibile sul libero mercato, ma è stato mostrato al momento solo per scopi pubblicitari da varie case automobilistiche per promuovere i propri veicoli [15]. Il 6 dicembre 2022, La Germania ha approvato tale sistema, realizzato dalla Bosch in collaborazione con Mercedes, per uso commerciale. Al momento, come mostrato in Figura 2.13, è possibile utilizzarlo nell'aeroporto di Stoccarda scaricando un'app sul proprio smartphone [17].

Figura 2.13: Automated Valet Parking nell'aeroporto di Stoccarda [13]

[13] https://www.bosch-press.it/pressportal/it/it/press-release-65792.html

2.2.6 Livello 5 (Automazione completa)

I veicoli di livello 5, secondo la classificazione SAE, raggiungono il più alto livello di automazione grazie alle tecnologie avanzate impiegate. Tali veicoli non richiedono un coinvolgimento umano, infatti, non sono equipaggiate con il volante e con i pedali di accelerazione e decelerazione e sono in grado di andare ovunque e fare tutto ciò che un conducente umano può fare, gestendo autonomamente qualsiasi situazione di errore [15]. Questa tipologia di auto sono ancora in fase di esperimento, e non sono disponibili sul libero mercato. L'obiettivo, così come definito dall'ERTRAC Working Group, nella Roadmap da loro pubblicata (Figura 2.14), è quello di avere disponibili tali veicoli su strada per il 2030[18].

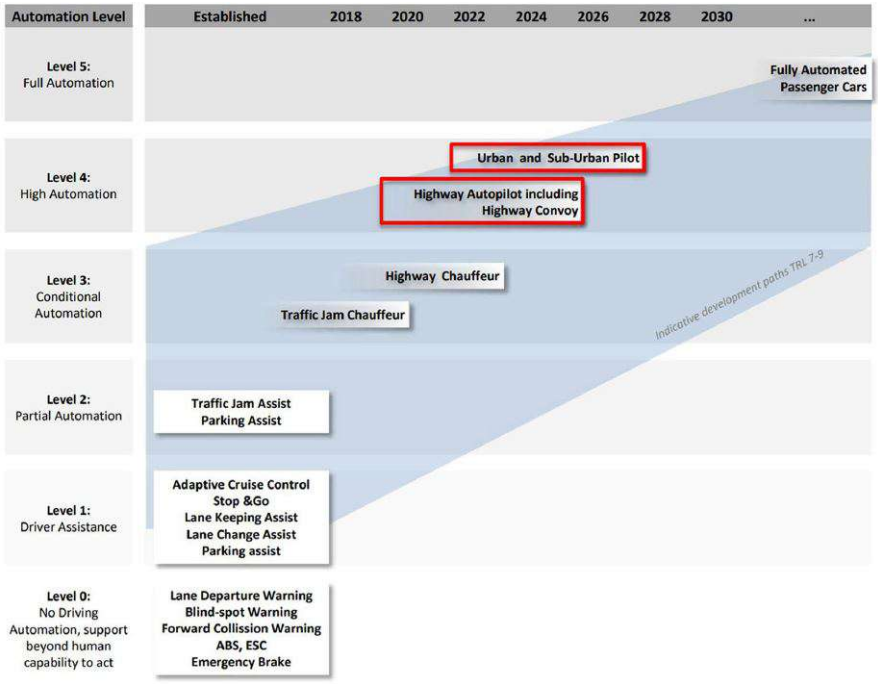

Figura 2.14: ERTRAC Roadmap 2019 [14]

[14] https://www.ertrac.org/wp-content/uploads/2022/07/ERTRAC-CAD-Roadmap-2019.pdf

2.3 Sistemi avanzati di assistenza e automazione alla guida

Dalla Figura 2.1 è possibile notare che le funzionalità sono divise in due gruppi : funzionalità di supporto alla guida (driver support feature, in blu) e funzionalità di guida automatizzata (automated driving features, in verde).

2.3.1 Sistemi di supporto alla guida (ADAS)

I sistemi che forniscono le funzionalità di supporto alla guida, classificati fino al livello SAE 2, sono definiti come Sistemi avanzati di assistenza alla guida (Advanced Driver Assistance Systems, o semplicemente ADAS). Questi sistemi includono funzionalità di supporto per la sicurezza come l'assistenza al mantenimento corsia (line keep assist) o di comodità come l'adaptive cruise control. Durante la guida, queste funzionalità contribuiscono all'aumento di sicurezza fornendo informazioni sotto forma di avvisi acustici e/o visivi. Il conducente ha pieno controllo del veicolo durante l'intero processo dell'operazione, infatti, nel caso ci fosse un malfunzionamento o una condizione in cui le funzionalità di supporto non si comportino come previsto, il conducente è ancora a controllo dell'auto e può ignorarli o disabilitarli. Esistono due tipologie di sistemi ADAS : Sistema ADAS passivo e Sistema ADAS attivo. I sistemi ADAS passivi sono sistemi che semplicemente avvisano il conducente in caso di condizioni pericolose. Non sono in grado di interagire sui movimenti e i comportamenti del veicolo. Il conducente è l'unico a poter gestire la situazione ed evitare incidenti. Metodi tipici di avviso includono suoni e spie luminose, e qualche volta anche feedback fisici, per esempio, la vibrazione del volante per avvisare il conducente che la corsia che sta per occupare è occupata da un altro veicolo (sistema di monitoraggio dell'angolo cieco). Comuni sistemi ADAS passivi sono : Retrocamera, ABS, sensori di parcheggio e tutti quelli visti in precedenza nel livello 0 SAE. I sistemi ADAS attivi, invece, sono in grado di interagire direttamente sul comportamento del veicolo, entrando in azione in situazioni di pericolo, evitando possibili incidenti, anticipando il conducente, che può comunque disattivare il sistema o by-passarlo. Comuni sistemi ADAS attivi sono tutti quelli visti in precedenza nei livelli 1 e 2 SAE.

2.3.2 Sistemi di guida automatizzata (ADS)

I sistemi che forniscono le funzionalità di guida autonoma, classificati dal livello 3 SAE, sono definiti come Sistemi di guida automatizzata (ADS). Questi sistemi sono in grado di prendere il controllo totale del veicolo quando sono soddisfatte determinate condizioni per le quali sono stati progettati. Durante questo tempo, il veicolo può muoversi senza la supervisione del conducente. Queste funzionalità possono prendere il controllo del veicolo e guidare in autonomia senza il coinvolgimento fisico del conducente. Nel momento in cui c'è uno stato dove le condizioni cambiano o il veicolo non è in grado di guidare in autonomia a causa di un errore o di uno stato confusionale, il sistema può richiedere al conducente di riprendere il controllo del veicolo, che avrà tempo marginale sufficiente per orientarsi e riprendere la guida. In realtà la maggior parte di questi sistemi vengono progettati in modo che quando essi sono attivi, e quindi hanno il controllo dell'auto, richiedano, dopo un tempo preciso, solitamente tra gli 8 o i 12 secondi, un feedback da parte del conducente (esempio : toccare il volante) per assicurarsi che esso non si distragga e che quindi sia fisicamente e mentalmente presente. Nel momento in cui tale feedback non venisse fornito, l'autovettura va in uno stato di emergenza e può assumere diversi comportamenti, come la disattivazione dei sistemi ADS con avviso sia acustico che visivo, o, in veicoli più avanzati, come Tesla, il sistema ADS fa accostare l'auto in sicurezza attivando le luci di emergenza. Il comportamento dell'autopilot di Tesla è riportato sul manuale d'uso presente sul loro sito ufficiale [19]. Il sistema è stato progettato affinchè, quando attivo, nel caso in cui non rilevi le mani del conducente sul volante, dopo all'incirca 13 secondi, richieda un feedback fisico per accertarsi della sua attenzione. Quindi, se il conducente ha almeno una mano sul volante, il sistema non richiede alcun feedback. In caso contrario, il sistema si comporterà seguendo tale procedura : dopo all'incirca 13 secondi, appare il primo avviso mostrato in Figura 2.15, infatti, lo schermo principale inizierà a lampeggiare. Basta un tocco dello schermo o una leggera deviazione del volante per farlo scomparire per poi apparire dopo altri 13 secondi (nel caso il conducente rilascia nuovamente il volante). Nel caso in cui tale avviso venisse ignorato, dopo all'incirca 24 secondi, le casse dell'auto inizieranno a riprodurre un suono di beep. Dopo 40 secondi dall'ultimo

feedback fornito, il suono di beep diventerà sempre più continuo e l'auto entrerà in uno stato di emergenza. Infatti, i sistemi ADS inizieranno a frenare gradualmente l'auto accendendo gli indicato di emergenza. Dopo 53 secondi all'incirca, l'auto frena più bruscamente per poi accostare sui margini della strada.

Figura 2.15: Autopilot Tesla che chiede al conducente di collocare le mani sul volante [15]

[15] https://t.ly/dbzwS

2.4 Architettura tipica di una self-driving car

In questa sezione verrà presentata una dettagliata spiegazione della tipica architettura di una self-driving car. L'architettura del sistema autonomo di una self-driving car è tipicamente composto dal sistema di percezione e dal sistema decisionale; ognuno di esso è composto da tanti sottosistemi che comunicano e collaborano gli uni con gli altri così come mostra la Figura 2.16. Il sistema di percezione è diviso in vari sottosistemi che sono responsabili di operazioni di percezione dell'ambiente come: localizzazione del veicolo, riconoscimento e mappatura dei segnali stradali e degli ostacoli statici e dinamici, etc..

Il sistema decisionale è anch'esso diviso in tanti sottosistemi responsabili di operazioni come: pianificazione generica (route planning) e dettagliata (path planning) del percorso verso una destinazione e controllo, movimento e comportamento del veicolo.

Il sistema decisionale, avendo la responsabilità di controllare il veicolo durante tutto il viaggio, dipenderà dalle informazioni ambientali e del veicolo che il sistema di percezione gli fornirà; di conseguenza, i due sistemi cooperano per il sicuro e corretto funzionamento del veicolo [20].

Il sistema di percezione è in grado di percepire l'ambiente circostante e avere informazioni sul veicolo (orientamento, posizione, velocità, etc..) tramite i numerosi sensori installati sul veicolo.

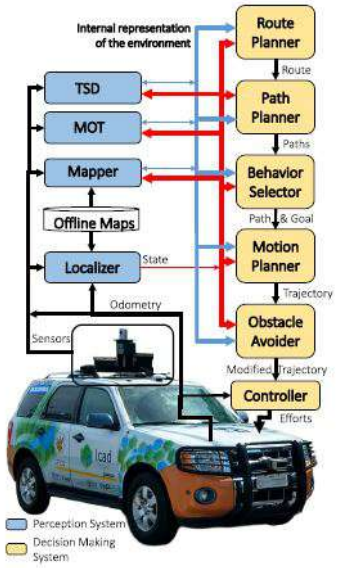

Figura 2.16: Architettura completa di una self-driving car [16]

2.4.1 Tipi di sensori di un'auto a guida autonoma

Come gli umani, anche i veicoli hanno degli "organi di senso" utilizzati per capire l'ambiente in cui si trovano e comportarsi di conseguenza. Questi "organi" sono i sensori. Quando parliamo di sistemi ADAS e ADS, questi sensori giocano un ruolo vitale poichè danno la possibilità di rilevare l'ambiente intorno al veicolo, eseguire determinate azioni e controllare il veicolo [15]. In questa sezione verranno introdotti tutti i tipi di sensori installati sulle auto presenti sul mercato. Tali sensori, singolarmente, non sono solo utilizzati su veicoli completamente autonomi (Livello 5 SAE), ma su tutti i veicoli che presentano almeno un sistema ADAS o ADS, quindi a partire dal livello 1 SAE (a volte anche Livello 0 SAE). E' la combinazione di tutti i sensori (mostrata in Figura 2.17) ad essere utilizzata solo ed esclusivamente per veicoli completamente autonomi.

[16] https://www.sciencedirect.com/science/article/abs/pii/S095741742030628X

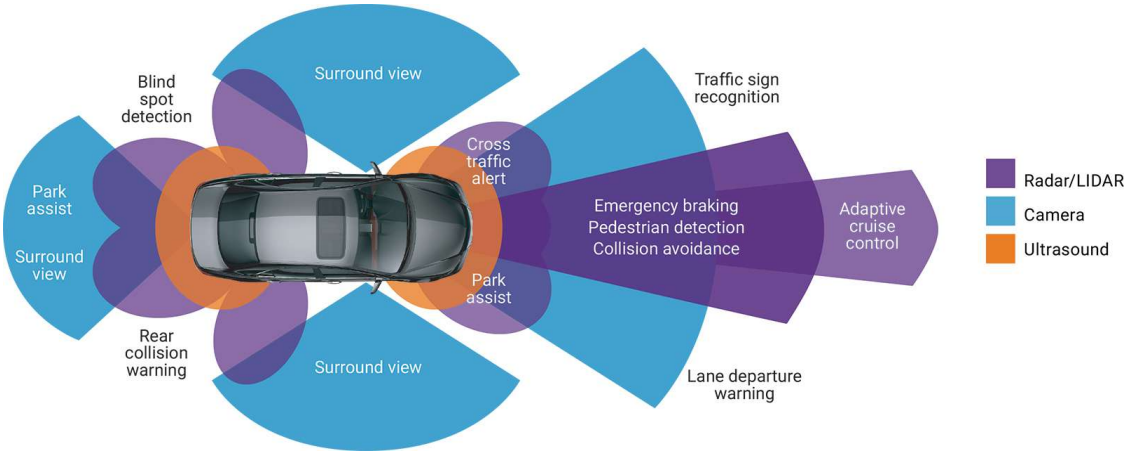

Figura 2.17: Configurazione completa dei sensori su un veicolo [17]

2.4.2 Telecamere

Le telecamere sono uno dei principali sensori presenti in un veicolo; sono utilizzate per realizzare funzionalità di supporto e automazione alla guida e quindi installate su veicoli di livello 1 SAE a salire; esse rilevano l'ambiente circostante grazie alla luce. I veicoli presenti al giorno d'oggi sul mercato arrivano ad avere fino a 13 telecamere in base numero di funzionalità che l'auto offre. Le telecamere aiutano a rilevare oggetti all'interno dell'ambiente, il quale sono classificati utilizzando algoritmi basati su reti neurali. Rilevare gli oggetti è un processo fondamentale poichè aiuta il veicolo a prendere le decisioni necessarie e rispondere in base alle condizioni dell'ambiente. Tali sensori sono utilizzati anche per calcolare le distanze con altri veicoli o con oggetti presenti nell'ambiente. Esistono vari nomi a cui si fa riferimento alle telecamere in base alla posizione dove sono installate e alle funzionalità che offrono, per esempio: retrocamera, camera frontale, camera interna, laterale ed esterna, etc...

Le telecamere di un veicolo possono differire tra di loro in termini di risoluzione, angolo di visione e capacità di rilevamento oggetti, etc...

Esse sono in grado di funzionare in quasi tutte le condizioni ambientali, anche se, basandosi solo e puramente sulla luce ambientale, sarà difficile utilizzarle o fare solo affidamento ad esse in condizioni di scarsa visibilità e/o illuminazione [15].

[17] `https://t.ly/TobEI`

2.4.3 Radar

I Radar (Radio Detection and Ranging) si basano sul principio di riflessione delle onde elettromagnetiche. Sono principalmente utilizzate per rilevare oggetti e calcolare distanze utilizzando il principio dell'effetto Doppler. Un veicolo può avere fino a 8 radar in base alle funzionalità che il veicolo offre. Differenti radar di un veicolo sono classificati in base alla frequenza a cui lavorano. Recentemente all'interno dei veicoli è stato introdotto il radar di imaging (Imaging radars), utilizzato per creare un immagini bidimensionale (2D) dell'ambiente come se fosse una telecamera. Quest ultimo viene anche utilizzato insieme alle telecamere come sensore di ridondanza delle immagini.

I radar sono in grado di rilevare immagini in qualsiasi condizione ambientale; ad esempio, di notte, utilizzano i raggi infrarossi per rilevare oggetti basati sulle onde di calore che trasmettono. In qualsiasi condizione ambientale, i radar emettono e catturano le onde elettromagnetiche riflesse e rilevano la presenza di oggetti. Essi non sono influenzati dalla luce, quindi sono sensori affidabili indipendentemente dalle condizioni di illuminazione. Tuttavia, ci sono alcuni svantaggi dei radar in quanto sono influenzati dal rumore. Questi rumori sono semplici disturbi causati da riflessi indesiderati nell'ambiente [15].

Nei veicoli sono la principale risorsa per il rilevamento, posizionamento e calcolo di distanza per un oggetto presente nelle vicinanze dell'auto. I radar hanno anche nomi differenti in base alla posizione in cui sono installati, range di lavoro e frequenza delle onde che emettono, per esempio : radar a corto (60m), medio(150m) e lungo(250m) raggio, radar frontale, etc...

2.4.4 Lidar

Il Lidar (Light Detection and Ranging) funziona in modo simile al Radar; esso, però, non si basa sul principio di riflessione delle onde elettromagnetiche, ma, emette un impulso laser piuttosto delle onde elettromagnetiche. Tale sensore è molto preciso e affidabile, e sta diventando dominante nel settore automobilistico. Un veicolo che offre funzionalità di guida autonoma o assistita potrebbe non avere un Lidar, o, utilizzarne anche fino a 6 in base al tipo di funzionalità. Il lidar ha una distanza

operativa di 200 metri. Così come i radar, anche il lidar è influenzato dal rumore; questi rumori sono principalmente dovuti dalla dispersione degli impulsi laser. Ogni volta che c'è umidità nell'ambiente o durante la pioggia, l'impulso laser si disperde, e questo potrebbe essere una fonte di rumore. Allo stesso modo, molti oggetti riflettenti presenti nell'ambiente, come ad esempio, la presenza di edifici con facciate vetrate, possono causare un sacco di riflessione, che potrebbe essere un'altra fonte di rumore [15]. La Figura 2.18 mostra la visione del Lidar

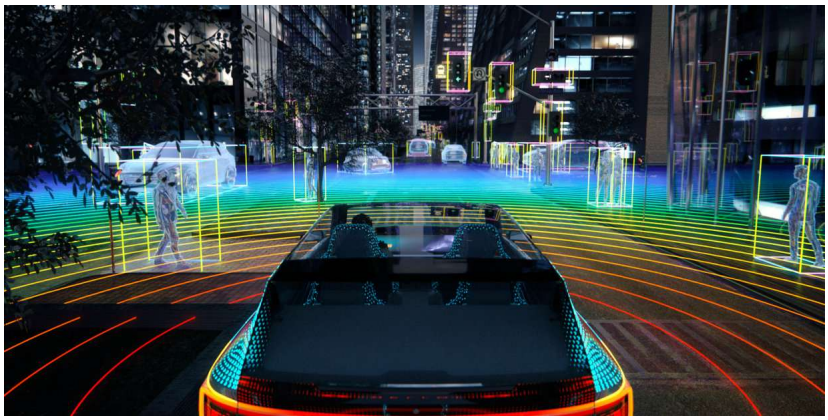

Figura 2.18: Visione del lidar [18]

2.4.5 Sensori ad ultrasuoni

I sensori ad ultrasuoni sono utilizzati da decenni nel mondo automobilistico. Sono uno dei sensori più economici per il rilevamento ambientale. Funzionano basandosi sul principio di riflessione delle onde elettromagnetiche e sono utilizzati, come visibile in Figura 2.19, per rilevare oggetti presenti nelle vicinanze del veicolo [15]. Hanno una distanza operativa di 5 metri e sono utilizzati principalmente per funzionalità di assistenza durante la fase di parcheggio e manovre del veicolo per evitare che il conducente urti oggetti non visibili dalla propria visuale.

[18] https://www.osram.com/os/applications/automotive-applications/sensing_lidar.jsp

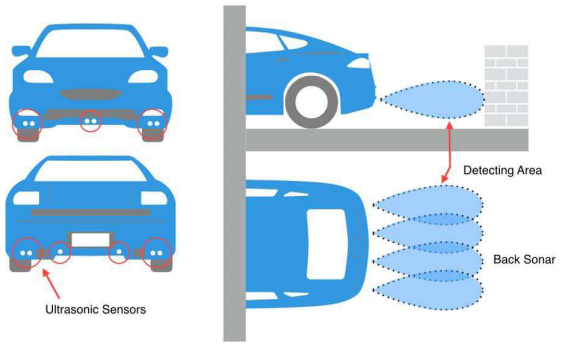

Figura 2.19: Posizione e distanza operativa dei sensori ad ultrasuoni [19]

2.4.6 Sensori per unità di misura inerziale

Un'unità di misura inerziale, conosciuto con la sigla IMU, è uno dei sensori più importanti di un veicolo poichè fornisce lo stato di movimento dell'auto nell'ambiente. Le informazioni principali che offre sono: l'orientamento, il movimento e la posizione angolare del veicolo [15]. In altre parole, l'IMU fornisce dati accurati del veicolo basati su tre assi mostrati nella Figura 2.20: trasversale(Pitch), longitudinale(Roll) e verticale(Yaw). Tali informazioni sono fondamentali poichè influenzano interamente il comportamento e i movimenti autonomi del veicolo (accelerazione, sterzo e freno). per esempio, l'angolo di sterzata sarà differente se il veicolo cambia corsia su una strada curva piuttosto che dritta. Questi sensori offrono anche un sistema di posizionamento globale(GPS) che permette di fornire la posizione geografica dei veicolo.

[19] `https://inrix.com/blog/ultrasonic-sensor-parking-availability-technology/`

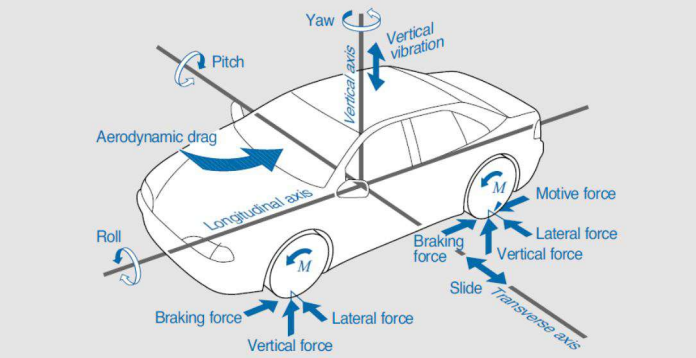

Figura 2.20: Informazioni offerte dal sensore IMU [20]

2.4.7 Sistema di percezione

Il sistema di percezione ha lo scopo di stimare lo stato del veicolo e di creare un'interna (al sistema di auto guida) rappresentazione dell'ambiente, utilizzando dati catturati tramite i vari tipi di sensori installati sul veicolo.

2.4.8 Sottosistema di Localizzazione e Mappe Statiche

Al fine di guidare l'auto all'interno dell'ambiente, il sistema decisionale ha bisogno di conoscere la posizione e lo stato del veicolo. Il sottosistema di localizzazione (Localizer, Figura 2.16) ha il compito di stimare lo stato del veicolo (posizione, velocità lineare, velocità angolare, etc..) in base alla mappa statica dell'ambiente [20]. Le mappe statiche, o anche Mappe Offline (Offline Maps, Figura 2.16), sono calcolate automaticamente prima delle operazione di guida autonoma, utilizzando solitamente i sensori dell'auto, in modo che si conosca la posizione degli oggetti statici all'interno dell'ambiente.

Anche informazioni riguardanti norme e regolamenti (limiti di velocità, linee di demarcazione, senso di marcia, etc.) sono essenziali per il corretto funzionamento del sistema decisionale. Queste informazioni sono solitamente incluse nelle mappe stradali.

Il Localizer riceve in input le Mappe Statiche, i dati dei sensori e L'odometria del veicolo e li utilizza per calcolare lo stato dell'auto (output). E' importante notare che nonostante il GPS aiuti il processo di localizzazione, esso non è sufficiente per la

corretta localizzazione in ambienti urbani a causa di interferenze causate da alberi ad alto fusto, edifici, tunnel, ecc, che rendono il posizionamento GPS inaffidabile [20].

2.4.9 Sottosistema di mappatura (Mapper)

Il sottosistema di mappatura (Mapper, Figura 2.16) riceve in input le mappe statiche (offline) e lo stato del veicolo, e generano come output le Mappe Online. Le Mappe Online sono un'unione di informazioni presenti nelle mappe statiche e una mappa della griglia di occupazione calcolata online utilizzando i dati dei sensori e lo stato della vettura corrente. Le mappe online, però, hanno limitate informazioni sugli oggetti dinamici presenti nell'ambiente. Informazioni come la posizione esatta, la direzione e la velocità di spostamento degli ostacoli dinamici in movimento, non sono presenti nelle mappe Online, nonostante siano fondamentali per il sistema decisionale per evitare collisioni [20].

2.4.10 Sottosistema di tracciamento degli oggetti in movimento (MOT)

Tali informazioni sono fornite dal sottosistema di tracciamento degli oggetti in movimento (MOT). Esso riceve in input le mappe offline e lo stato del veicolo per rilevare e tracciare continuamente la posizione e la velocità degli ostacoli in movimento vicini alla vettura (esempio: pedoni e altri veicoli, come mostrato in Figura 2.21).

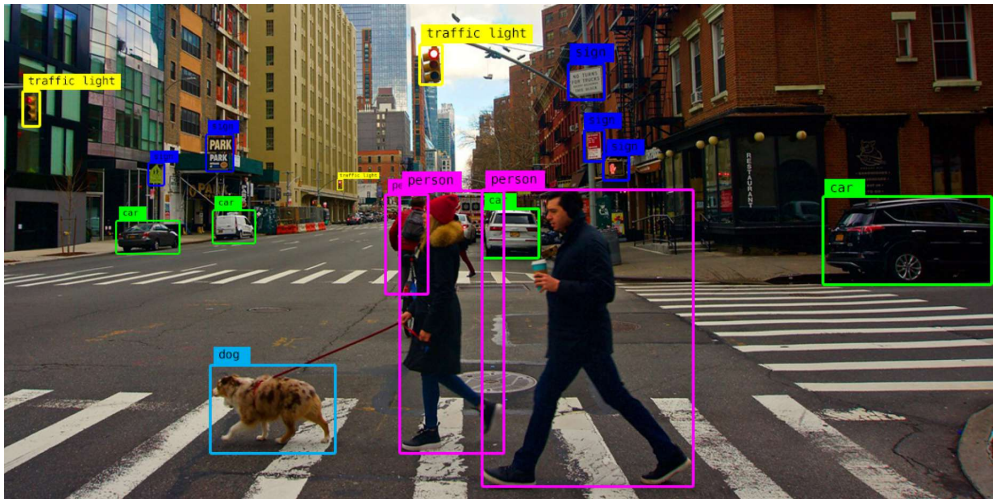

Figura 2.21: funzionamento del sottosistema di tracciamento degli oggetti in movimento[21]

2.4.11 sottosistema di rilevazione della segnaletica stradale (TSD)

La segnaletica stradale orizzontale (linee di demarcazione, precedenza, stop, etc.) e verticale (limiti di velocità, semafori, etc.) sono rilevate dal sottosistema di rilevazione della segnaletica stradale (TSD). Il sottosistema TSD ha il compito di rilevare e riconoscere la segnaletica, come mostra la Figura 2.22. Esso riceve in input i dati dei sensori e lo stato del veicolo e rileva la posizione della segnaletica stradale e riconosce la loro classe di appartenenza e il significato.

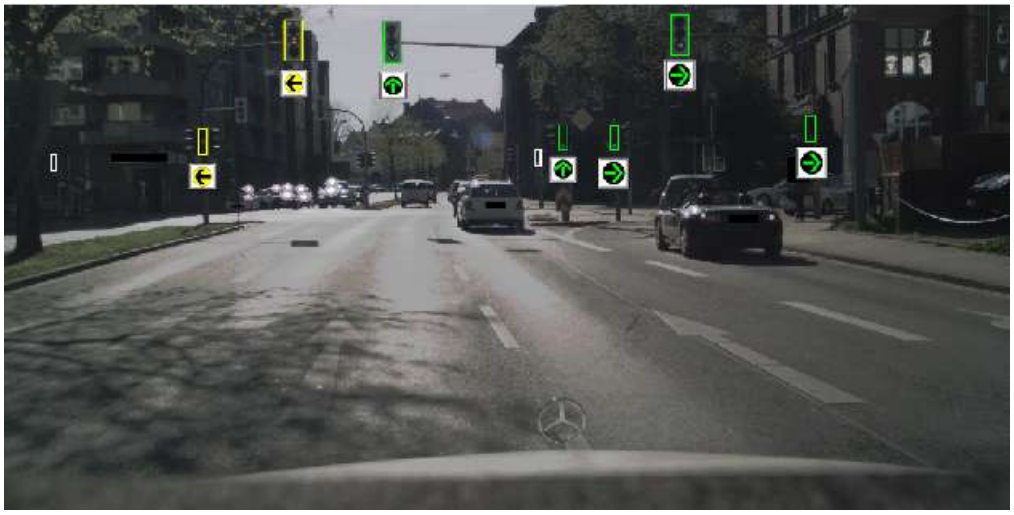

Figura 2.22: funzionamento del sottosistema di rilevazione della segnaletica stradale[22]

[21] https://alwaysai.co/blog/object-detection-for-businesses

2.4.12 Sistema decisionale

Come già accennato precedentemente, il sistema decisionale ha il compito di controllare il veicolo. Le informazioni fondamentali gli vengono fornite in input dal sistema di percezione; esse verranno successivamente processate per determinare il giusto comportamento del veicolo durante il viaggio.

2.4.13 sottosistema di pianificazione dell'itinerario (Route Planner)

Quando il conducente indica una destinazione nelle mappe statiche, il sottosistema di pianificazione dell'itinerario (Route Planner) computa un itinerario W all'interno delle mappe offline, che inizia dalla posizione corrente del veicolo e termina alla destinazione specificata. Un itinerario è composta da sequenza di punti W = {w1,w2,...,w | w | } (punti A,B,C nella Figura 2.23), dove ogni punto Wi, è una coppie di coordinate Wi = (xi,yi) all'interno delle mappe statiche [20].

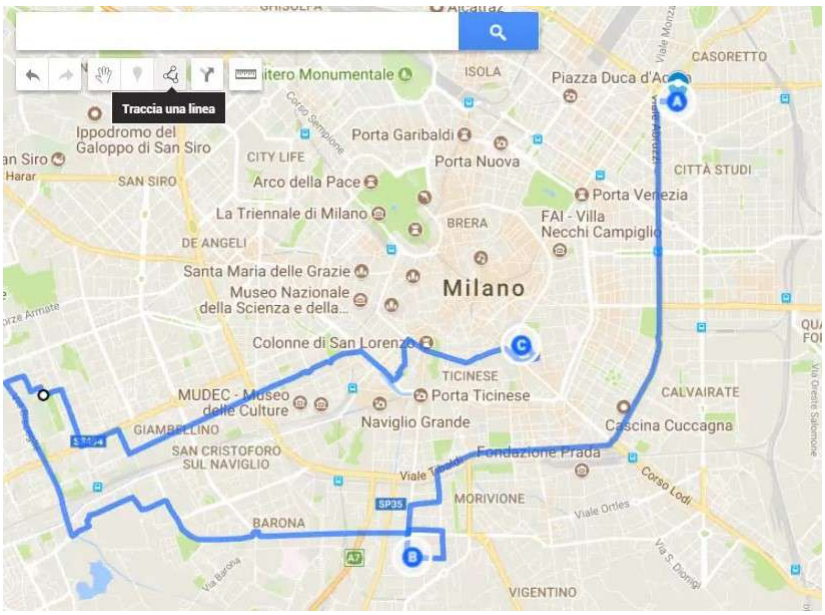

Figura 2.23: esempio di pianificazione di un initerario[23]

[22] https://t.ly/zpcoU

[23] https://www.navigaweb.net/2018/02/creare-percorsi-e-itinerari-sulle-mappe.html

2.4.14 sottosistema di pianificazione del percorso (Path Planner)

Definito un un itinerario, il sottosistema di pianificazione del percorso (Path Planner), calcola, considerando lo stato corrente del veicolo e la rappresentazione interna dell'ambiente (considerando sensi di marcia e norme), un insieme di Percorsi P = {p1, p2,. . . , p | P | } [20]. Un percorso (linea verde nella figura 2.24) è una sequenza di posizioni pi, dove pi è una coppia di coordinate all'interno delle mappe statiche e il senso di marcia da percorrere per raggiungere questa coppia di coordinate.

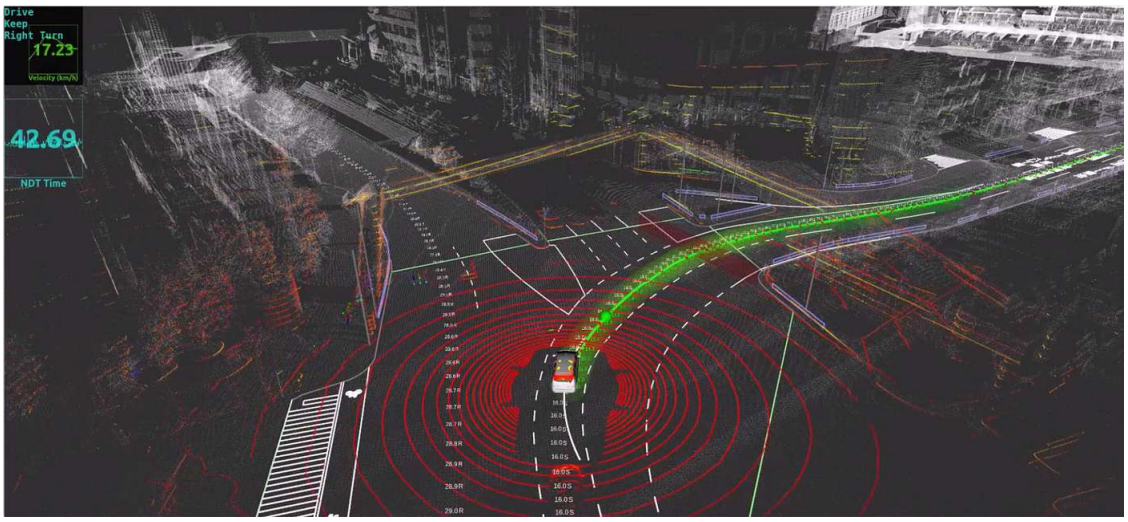

Figura 2.24: Esempio di percorso calcolato dal Path Planner[24]

2.4.15 Sottosistema di selezione del comportamento (Behavior Selector)

Determinato il percorso da percorrere per raggiungere la destinazione, il sottosistema di selezione del comportamento (Behavior Selector) dovrà gestire e definire il comportamento del veicolo durante tutto il viaggio. Esso, infatti, dovrà gestire situazioni comuni e non in cui un veicolo può trovarsi durante un viaggio, come per esempio: mantenimento e cambiamento della corsia, la gestione degli incroci, gestione dei semafori, etc.

[24] https://pub.towardsai.net/towards-autonomous-vehicles-8224bb0bf18d?gi=5c774b5019d8

Tale sistema considera il percorso P e associa ad ogni posizione *pj* una velocità massima di percorrenza *vj* creando un nuovo punto chiamato Goalj = (pj,vj). Il sistema deciderà, in base alla posizione del veicolo, il prossimo Goalj da raggiungere evitando gli ostacoli statici dell'ambiente circostante [20].

2.4.16 Sottosistema di movimento (Motion Planner)

Il sottosistema di movimento (Motion Planner) ha il compito di definire una traiettoria T (linee verdi nella Figura 2.26), considerando la posizione attuale del veicolo e del prossimo Goal, seguendo il percorso definito dal Behavior Selector e soddisfacendo i vincoli dinamici e cinematici del veicolo, offrendo comfort ai passeggeri. Una traiettorie T = {c1, c2,. . . , c | T | } viene definita come una sequenza di tuple *ck* (movimenti), dove ogni tupla è composta da tali informazioni : velocità, angolo di sterzata e la durata del movimento (durata del ck) [20]. Esistono anche altri modi per definire delle traiettorie, tuttavia il loro scopo è sempre quello di guidare il veicolo dalla sua posizione corrente al prossimo goal in maniera sicura e confortevole.

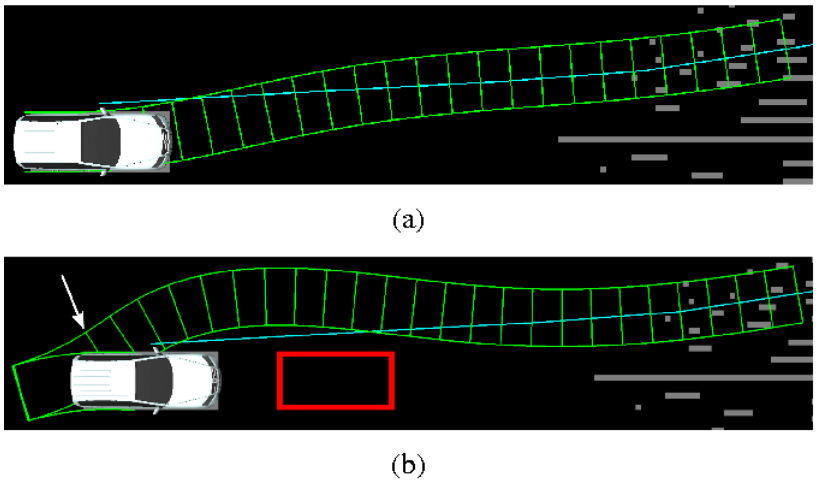

Figura 2.25: Esempi di traiettore definite dal Motion Planner in base alla situazione dell'ambiente[25]

[25] https://rb.gy/it01a

2.4.17 Sottosistema di prevenzione degli ostacoli (Obstacle Avoider)

La traiettoria può essere modificata dal sottosistema che si occupa di evitare gli ostacoli (Obstacle avoider). Esso riceve in input una traiettoria T calcolata dal sottosistema precedente e la modifica (solitamente diminuendo la velocità o cambiando l'angolo di sterzata), se necessario, per evitare collisioni [20]. La Figura 2.26 mostra un esempio di come viene modificata una traiettoria nel momento in cui viene rilevato un ostacolo nella traiettoria originale.

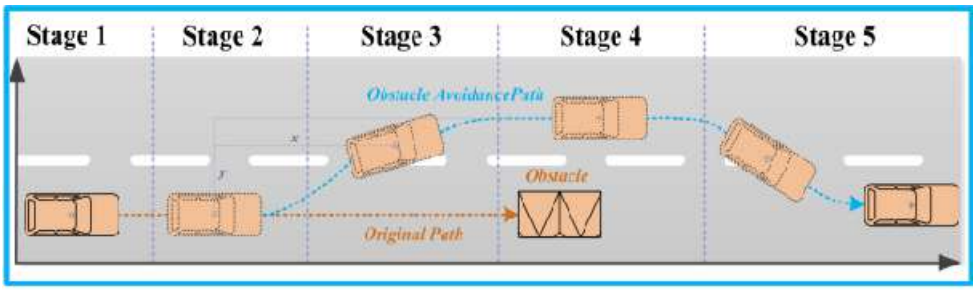

Figura 2.26: Esempi di come L'obstacle avoider rileva gli ostacoli e modifica la traiettoria originale di conseguenza[26]

2.4.18 Sottosistema di controllo (Controller)

Infine esiste il sottosistema di controllo che riceve in input la traiettoria T, eventualmente modificata dal sottosistema Obstacle avoider, calcolando e inviando dei comandi al volante, acceleratore e freno in modo da far percorrere all'auto la traiettoria modificata nel miglior modo possibile (rispettando leggi della fisica) [20].

[26] `https://shorturl.at/lqDUZ`

2.5 Architetture informatiche

L'Intelligenza Artificiale (AI) è la tecnologia che ha reso le auto a guida autonoma una realtà. Infatti, per realizzare questa tipologia di veicoli si fa uso di un sotto campo dell'Intelligenza Artificiale e del Machine Learning (capacità di apprendere dai dati): il Deep Learning. Il Deep Learning è una tecnica di apprendimento che fa uso di reti neurali artificiali.

L'AI viene utilizzata nei 4 pilastri fondamentali delle self-driving car: percezione, decisione, localizzazione e predizione.

Nel sistema di percezione, i dati che vengono catturati dai vari sensori presenti sul veicolo vengono processati e analizzati da algoritmi di AI il quale creeranno una mappa dettagliata dell'ambiente e degli oggetti presenti al suo interno, come: pedoni, veicoli, semafori e altri segnali stradali.

Nel sistema di decisione, l'AI viene utilizzata per prendere decisioni in tempo reale basate sui dati forniti dai sensori. Per esempio: se il veicolo rileva un pedone che attraversa la strada, il sistema di decisione utilizzerà algoritmi di AI per determinare la miglior azione da compiere, come: fermarsi, rallentare o sterzare [21].

L'AI inoltre, può essere utilizzata anche per prevedere il comportamento degli altri veicoli o persone. Questo permette al veicolo di anticipare ed evitare possibili collisioni con gli altri utenti della strada.

2.5.1 Deep Learning

Il Deep Learning è una branca del Machine Learning e fa riferimento a reti neurale composte da tre o più strati. Le reti neurali, tentano di simulare il comportamento del cervello umano permettendo di "imparare" da grandi quantità di dati tramite una combinazione di dati in input, pesi e bias. Questi elementi lavorano insieme per riconoscere, classificare e descrivere accuratamente gli oggetti all'interno dei dati.

Le reti neurali utilizzate consistono in strati multipli di nodi interconnessi tra loro, ciascuno costruito in modo da essere collegato al livello precedente al fine di perfezionare e ottimizzare la previsione o la classificazione. Questa progressione di computazioni attraverso la rete si chiama propagazione in avanti (forward propagation). Gli strati di input e di output di una rete neurale sono chiamati strati visibili.

Lo strato di input è dove il modello di Deep Learning invia i dati con lo scopo di processarli, mentre, quello di output è dove avviene la predizione o la classificazione. Tra questi due strati è possibile inserire multipli strati nascosti (non visibili), utilizzati per ottimizzare la precisione di predizione [22].

Un altro processo, chiamato propagazione all'indietro (backpropagation), utilizza algoritmi, come la discesa in pendenza (gradient descent), per calcolare l'errore della predizione e modificare il peso e il bias della funzione spostandosi all'indietro attraverso gli strati per addestrare il modello.

La forward propagation e backpropagation permettono alla rete neurale di fare previsioni più precise, correggendo eventuali errori fatti in precedenza. Così facendo, nel tempo, la rete migliorerà e diventerà sempre più accurata.

Una delle differenze più importanti tra Machine Learning e Deep Learning è che il Deep Learning elimina alcune operazioni pre-processing fatta sui dati, visibili in Figura 2.27 come: l'estrazione manuele delle feature (Feature Extraction). Infatti, questa tipologia di algoritmi possono lavorare su dati non strutturati, come: testi e foto, automatizzando l'estrazione delle feature ed eliminando eventuali dipendenze legate ai dati. Per esempio: supponiamo di avere un insieme di foto di vari animali e vogliamo classificarli come "cane", "gatto", "criceto", eccetera [22]. Un algoritmo di Deep Learning è capace di determinare quale caratteristiche (per esempio: le orecchie) sono più importanti per distinguere gli animali tra di loro. Nel Machine Learning, questa gerarchia di feature, viene stabilita manualmente dal programmatore.

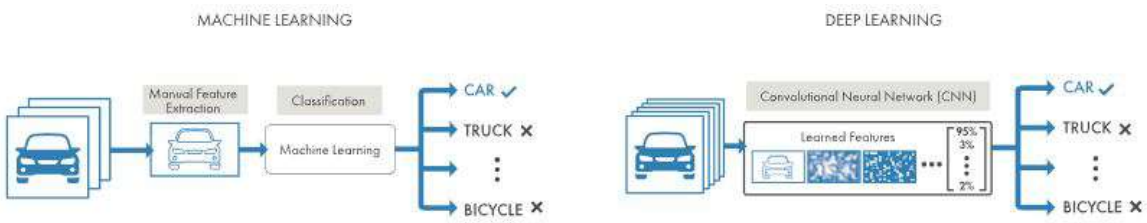

Figura 2.27: Differenza di operazioni tra Machine Learning e Deep Learning[27]

[27] https://it.mathworks.com/discovery/deep-learning.html

2.5.2 Reti neurali artificiali

Le reti neurali artificiali (ANN) sono composte da livelli (layer) di nodi (neuroni artificiali, Figura 2.28). Il primo livello è quello di input, dove i nodi neuroni ricevono i dati (solitamente appartenenti a un dataset) da elaborare. A seguire ci sono uno o più livelli nascosti utilizzati per l'elaborazione e un livello di output che restituisce il risultato finale [23].

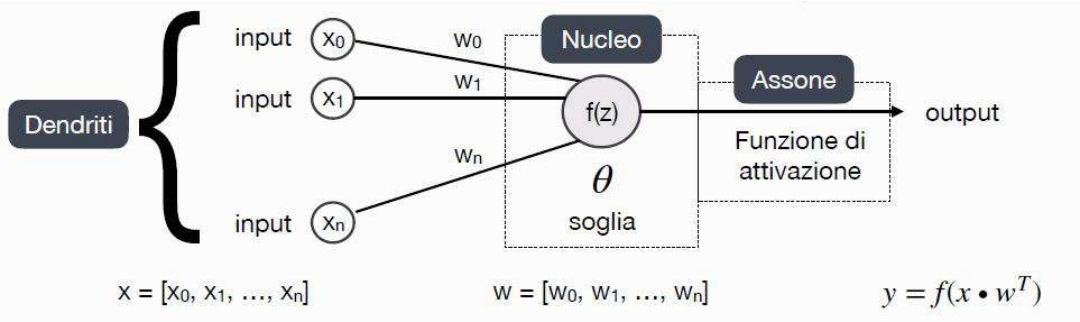

Figura 2.28: Architettura di un neurone artificiale[28]

Una volta determinato il livello di input, vengono assegnati dei pesi ad ogni collegamento (assone). Questi pesi aiutano a determinare l'importanza di quella variabile, cioè, quanto influisce sull'output rispetto agli altri input. Tutti gli input di un neurone, vengono prima moltiplicati per i rispettivi pesi e poi sommati tra di loro. Successivamente, questo risultato, indicato come z nella figura 2.28, viene dato in input a una funzione di attivazione (rappresenta l'elaborazione fatta dal neurone, quindi un neurone può essere visto come una funzione), che determina l'output del neurone. Se l'output supera una certa soglia, esso attiverà il nodo, passando i dati al livello successivo nella rete. Questo processo, visibile in Figura 2.29, di passaggio dati da un livello a quello successivo fino al risultato finale, viene chiamato propagazione in avanti (forward propagation).

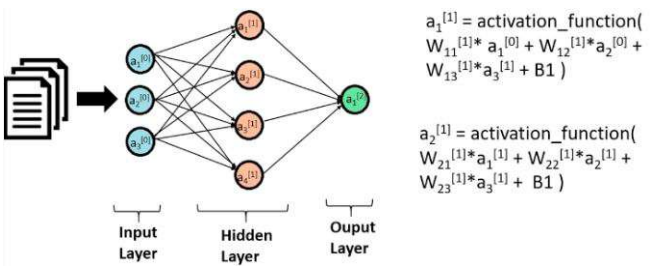

Figura 2.29: Processo di propagazione in avanti[29]

Inizialmente, però, la rete neurale non ha alcuna conoscenza e di conseguenza dev'essere allenata per funzionare correttamente. Per allenare la rete neurale si utilizzano i dati presenti nel dataset, di cui già si conosce l'output. Inizialmente, i valori dei pesi saranno casuali, poichè, la rete non avrà alcun tipo di conoscenza. Calcolato l'output, ottenuto considerando i valori dei pesi casualmente, il modello dovrà capire di quanto ha sbagliato, in modo da migliorare la propria conoscenza. Il modello quindi, andrà a calcolare il proprio errore (differenza tra l'output prodotto e l'effettivo output), tramite una funzione di loss. Il valore della funzione di loss, rappresenta l'errore commesso dal modello, di conseguenza, l'obiettivo sarà quello di minimizzarlo. Questo procedimento dev'essere fatto su ogni istanza del dataset in modo da ricavare l'errore medio e capire di quanto mediamente il modello sbaglia nell'elaborazione.

L'unico modo che il modello ha per ridurre l'errore, e quindi avvicinarsi sempre di più al valore di output corretto, è modificare i parametri modificabili, cioè i pesi e ricalcolare il valore di output. Questo procedimento viene chiamato backpropagation, e viene eseguito finchè la rete non andrà ad azzerare l'errore. Il valore dei pesi varierà in base a un valore decimale, chiamato learning rate, utilizzato per modificare gradualmente i pesi. Ogni iterazione in cui i pesi vengono modificati, è chiamata epoca. La figura 2.30 mostra tutti i passaggi effettuati durante la fase di backpropagation.

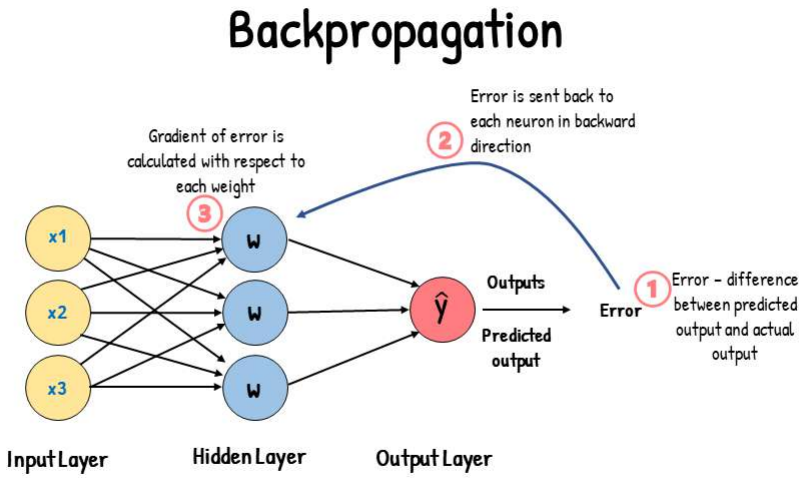

Figura 2.30: Processo di propagazione all'indietro[30]

[29] https://rb.gy/v621d

2.5.3 Reti Neurali Convulazionali

Una rete neurale convoluzionale, anche conosciuta come CNN o ConvNet, è una classe di reti neurali specializzata nell'elaborazione di dati con una topologia a griglia, ad esempio un'immagine. Un'immagine digitale è formata da pixel disposti in modo simile a una grigia nel quale sono riportati dei valori che vanno ad indicare la luminosità e il colore del pixel. I neuroni di una CNN, data un'immagine, sono in grado di rilevare inizialmente pattern semplici (linee,curve, angoli, etc..), e successivamente anche pattern più complessi (volti,oggetti, etc..), fornendo una "vista" al computer [24]. Le CNN trasformano le immagini date in input, in una forma che è più facile da elaborare, senza perdere caratteristiche fondamentali per avere una corretta classificazione.

2.5.4 Architettura di una CNN

Le CNN sono composte da 3 tipi di livelli : convulazione + relu, pooling e completamento connesso. Il livello convulazione è il primo livello di una CNN. I livelli di convulazione possono essere seguiti da ulteriori livelli di convulazione o di pooling, mentre, il livello completamente connesso è il livello finale. L'aggiunta di ogni livello comporta l'aumento della complessità della CNN, tuttavia, vengono identificate porzioni maggiori dell'immagine. I primi livelli hanno lo scopo di rilevare caratteristiche semplici, come colori, angoli, curve, linee, eccetera [25]. Man mano che i dati dell'immagine progrediscono attraverso i livelli, verranno riconosciuti caratteristiche più complesse, elementi e porzioni più grandi dell'immagine che permetteranno la corretta classificazione. La Figura 2.31 mostra tutte le componenti architetturali di una CNN.

[30] `https://rb.gy/3tba1`

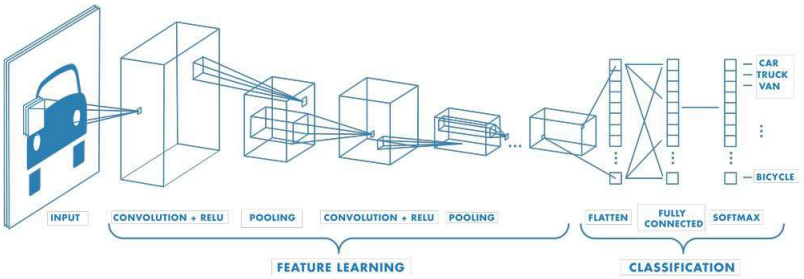

Figura 2.31: Architettura di una CNN[31]

2.5.5 Livello di convulazione

Il livello di convulazione è il livello principale di una CNN. Esso richiede una serie di componenti per funzionare correttamente, come: un immagine in input e il filtro. Il filtro, o anche conosciuto come Kernel, viene definito come il rilevatore delle caratteristiche ed è una matrice (solitamente 3x3, visibile in Figura 2.32) che viene applicata all'immagine in input e crea un immagine di output (nel quale vengono messe in evidenza le sue caratteristiche); questo processo è chiamato convulazione ed è visualizzabile nella Figura 2.32. L'immagine di output, verrà poi inviata ai successivi livelli della CNN. Più nel dettaglio, un pixel dell'immagine di output viene ottenuto moltiplicando un blocco 3x3 dell'immagine di input per la matrice 3x3 che rappresenta il filtro [25]. Per ottenere il prossimo pixel dell'immagine di output, basterà traslare di una posizione verso destra il blocco 3x3 dell'immagine di input. Questa operazione verrà effettuata finchè il filtro non sarà applicato (moltiplicato) ad ogni blocco 3x3 dell'intera immagine di input. L'immagine di output ottenuta da questo processo (convulazione) è conosciuta come mappa delle caratteristiche o mappa di attivazione.

[31] `https://it.mathworks.com/discovery/convolutional-neural-network-matlab.html/`

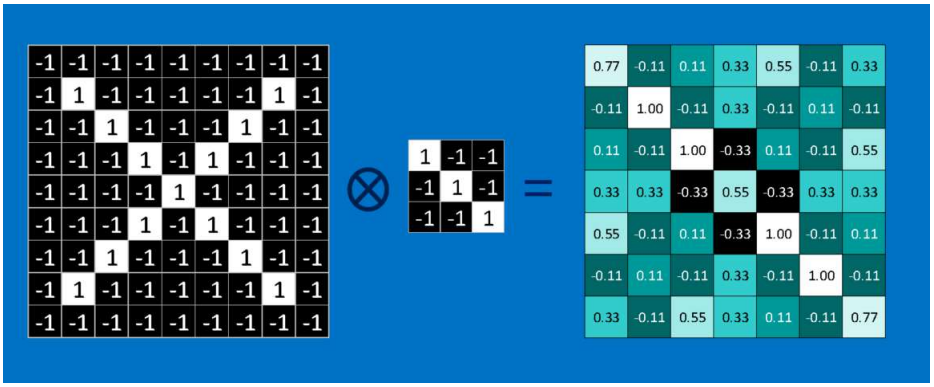

Figura 2.32: Esempio di convulazione su un'immagine[32]

Al termine di ogni operazione di convulazione, la CNN effettua un operazione di rettificazione (visibile in Figura 2.33)tramite il livello ReLU (unità lineare rettificata) il quale andrà a mappare i valori negativi con lo zero e manterrà i valori positivi invariati. Questa operazione è talvolta definita attivazione, dal momento che solo le feature attivate vengono trasmesse al livello successivo.

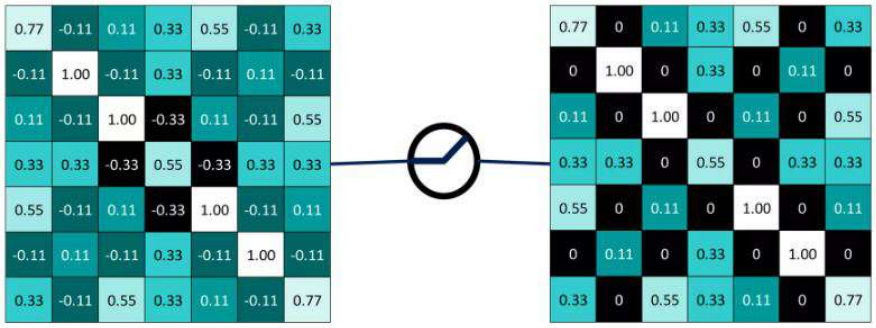

Figura 2.33: Esempio di rettificazione sulla medesima immagine[33]

2.5.6 Livello di Pooling

Il livello di Pooling, anche conosciuto come: livello di sottocampionamento (downsampling), prende in input l'immagine risultante dal livello convulazionale; ha lo scopo di ridurre le dimensioni dell'immagine in input tramite l'esecuzione di un sottocampionamento non lineare. Quest'operazione riduce il numero di caratteristiche e

[32] https://www.slideshare.net/ashraybhandare/deep-learning-cnn-and-rnn
[33] https://www.slideshare.net/ashraybhandare/deep-learning-cnn-and-rnn

parametri da considerare, ma allo stesso tempo, riduce anche la potenza computazionale richiesta per elaborare tali dati. Anche nel processo di pooling esiste un filtro (solitamente di dimensione 2x2), che applicato all'input, genererà un'immagine di output. I due principali tipi di pooling sono [25]:

- Max pooling: il filtro, applicato a un blocco dell'immagine di input, andrà a definire il valore del pixel risultante scegliendo il pixel con valore massimo nel blocco. Un esempio di max pooling viene mostrato in Figura 2.34

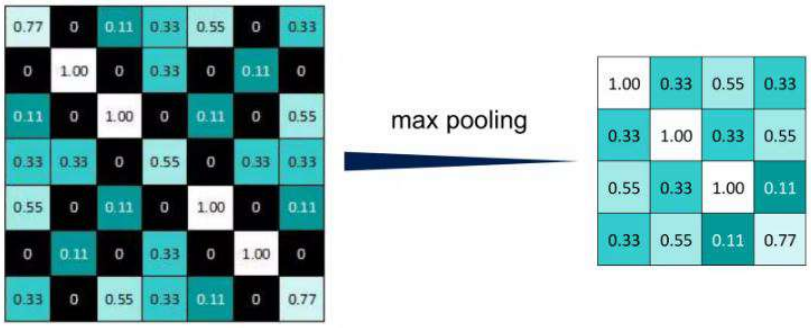

Figura 2.34: Esempio di max pooling[34]

- Average pooling: il filtro, applicato a un blocco dell'immagine di input, andrà a definire il valore del pixel risultante calcolando la media dei valori dei pixel del blocco. Un esempio di average pooling viene mostrato in Figura 2.35

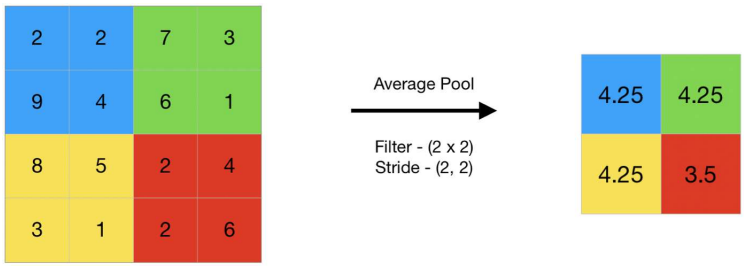

Figura 2.35: Esempio di Average pooling[35]

queste operazioni continueranno finchè il filtro non verrà applicato a tutti i blocchi dell'immagine di input. Durante il processo di pooling molte informazioni vengono perse, ma allo stesso tempo, si riduce la complessità di computazione, si migliora l'efficienza e vengono limitati i rischi di overfitting.

[34] https://www.slideshare.net/ashraybhandare/deep-learning-cnn-and-rnn
[35] https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

2.5.7 Appiattimento (Flattening)

Il processo di flattening, mostrato in Figura 2.36, consiste nell'andare a trasformare la mappa delle caratteristiche data in output dal processo di pooling (Pooled feature map) in un vettore unidimensionale. La ragione per cui viene effettuato questo appiattimento è che adesso tale vettore può essere dato in input a una rete neurale artificiale presente all'interno della nostra CNN [26].

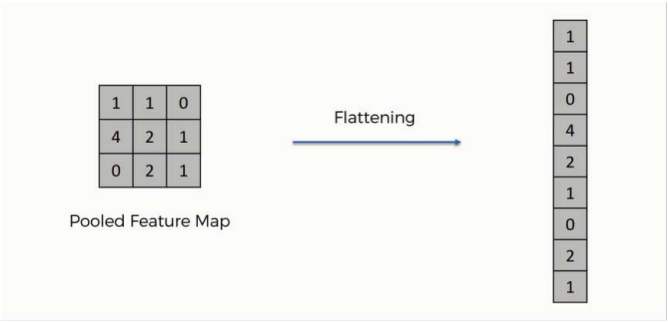

Figura 2.36: Esempio di appiattimento[36]

2.5.8 Livello completamente connesso (Fully Connected Layer)

Il livello completamente connesso prende in input il vettore unidimensionale restituito dal livello precedente con lo scopo di individuare precise caratteristiche di un'immagine. Più precisamente, ogni neurone del livello completamente connesso corrisponde a una determinata caratteristiche che potrebbe essere presente nell'immagine. Il valore che questo neurone poi passerà al prossimo livello rappresenterà la probabilità con cui questa caratteristica è presente nell'immagine [26]. Viene chiamato in questo modo poichè il livello nascosto della rete neurale artificiale viene rimpiazzato da uno specifico tipo di livello nascosto chiamato appunto livello completamente connesso. Cosi come il nome implica, i neuroni di un livello completamente connesso sono connessi a tutti i neuroni del livello successivo (visibili in Figura 2.37). La fine della rete neurale artificiale coincide con la fine della CNN e viene utilizzata per prevedere cos'è contenuto nell'immagine che la CNN sta cercando di riconoscere.

[36] `https://shorturl.at/JNORX`

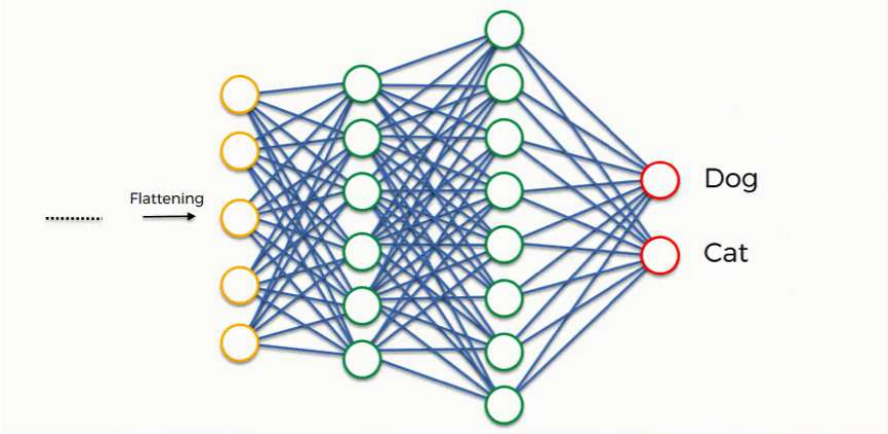

Figura 2.37: Esempio di livello completamente connesso[37]

2.5.9 Limiti delle CNN

Le reti neurali convulazionali classiche (CNN), come descritto in precedenza, sono in grado di, data un immagine, determinare se è presente o meno un determinato oggetto che stiamo cercando. Tuttavia, le CNN, data un'immagine, non sono in grado di riconoscere multipli oggetti appartenenti a classi uguali e/o differenti. Di conseguenza, le CNN non possono essere utilizzate nell'ambito dei veicoli a guida autonoma, poichè, l'ambiente intorno al veicolo presenta multipli di oggetti di diversa categoria, come per esempio: pedoni, segnali stradali, altri veicoli, etc... Si potrebbe risolvere questo problema tramite una "finestra" scorrevole utilizzata per selezionare una regione dell'immagine e applicare la CNN per identificare se è presente o meno l'oggetto che stiamo cercando. Tuttavia, applicare la CNN su ogni regione sarebbe un operazione lenta e costosa, non adatta al contesto dei veicoli a guida autonoma dove il riconoscimento degli oggetti avviene in tempo reale.

2.5.10 R-CNN (Region-based CNN)

Per bypassare il problema dell'enorme numero di regioni da dover selezionare, Ross Girshick et al. [27] nel 2013 propose un metodo dove si utilizzava una ricerca selettiva per estrarre da un'immagine 2000 regioni chiamate "regioni proposte" o "regioni candidate". Così facendo, piuttosto che classificare un enorme numero di

[37] `https://shorturl.at/hjH07`

regioni, possiamo lavorare sulle 2000 generate dalla ricerca selettiva. Successivamente, queste regioni candidate vengono ridimensionate e date in input a una CNN per estrarre le caratteristiche di quella regione. Infine, Le caratteristiche estratte vengono date in input a una SVM (Support Vector Machine), utilizzata per determinare se l'oggetto è presente nella regione candidata considerata. Il problema principale della R-CNN è che dovendo lavorare con 2000 regioni, impiega troppo tempo per riconoscere gli oggetti presenti all'interno dell'immagine. In Figura 2.38 è mostrato il funzionamento e l'architettura di una R-CNN.

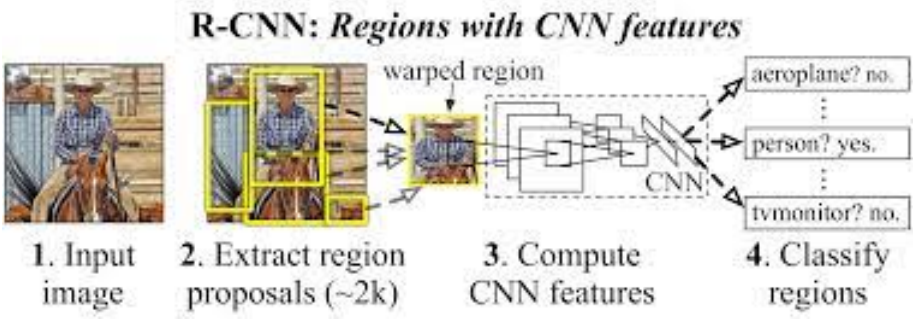

Figura 2.38: Architettura e funzionamento di una R-CNN[38]

2.5.11 R-CNN veloce (fast R-CNN)

Due anni dopo, Ross Girshick et al. [28], risolse i problemi di velocità legati alla R-CNN costruendo un algoritmo di rilevamento oggetti più veloce, chiamato "Fast R-CNN". L'approccio è molto simile all'algoritmo R-CNN, infatti, la CNN, piuttosto che ricevere in input le 2000 regioni proposte, riceverà l'intera immagine. La CNN genererà una mappa delle caratteristiche dell'immagine, dal quale verranno poi identificate le regioni proposte. Le regioni proposte saranno poi date in input a un livello di pooling che avrà lo scopo di ridimensionare queste regioni affinchè abbiano una dimensione fissa. Infine, ogni regione verrà data in input al livello completamente connesso e al livello softmax per classificare gli oggetti presente nella regione proposta. La ragione per il quale la Fast R-CNN è più veloce di una normale R-CNN è che l'operazione di convulazione non viene effettuata 2000 volte

[38] https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.pdf

per immagine, ma un'unica volta per generare una mappa delle caratteristiche. La Figura 2.39 mostra l'architettura di una fast R-CNN

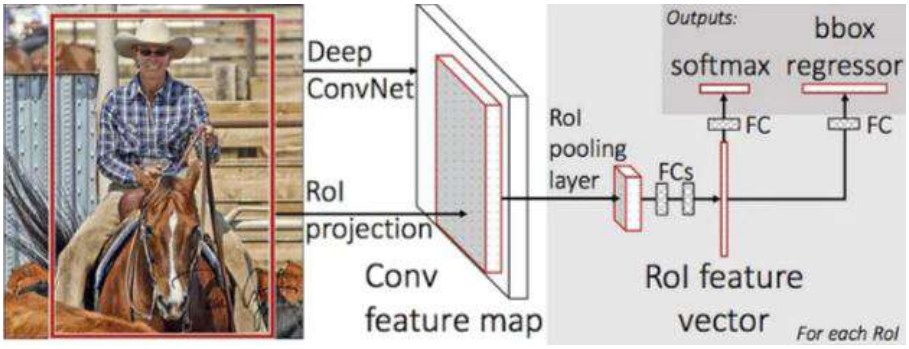

Figura 2.39: Architettura di una Fast R-CNN[39]

2.5.12 R-CNN più veloce (Faster R-CNN)

I due precedenti algoritmi (R-CNN e Fast R-CNN) utilizzano entrambi la ricerca selettiva per trovare le regioni proposte. Il problema principale è che il processo di ricerca è lento e lungo ed influisce di conseguenza sulle prestazioni della rete. Tuttavia, sempre nel 2015, Shaoqing Ren et al. eliminò l'algoritmo di ricerca dal processo di rilevamento oggetti, consentendo alla rete di apprendere le regioni proposte. L'approccio è identico all'algoritmo Fast R-CNN, infatti, l'intera immagine viene data in input a una CNN che genera una mappa delle caratteristiche. Ma, piuttosto che utilizzare una ricerca selettiva su questa mappa per individuare le regioni proposte, viene introdotta una rete separata che prevede le regioni proposte. Successivamente, come nella fast R-CNN, tali regioni vengono poi date in input ai livelli di pooling, completamento connesso e softmax per classificare gli oggetti presenti nelle regioni proposte. Questo piccolo cambiamento, rende il processo di rilevamento oggetti ancora più veloce rispetto ai precedenti algoritmi (Figura 2.40), tanto che la Faster R-CNN può essere utilizzata in contesti in cui il rilevamento dev'essere fatto in tempo reale.

[39] https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf

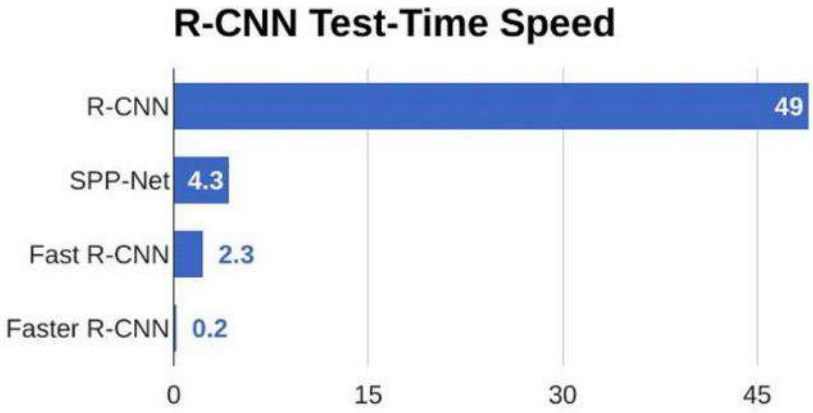

Figura 2.40: Confronto dei tempi di esecuzione[40]

2.6 Research gap

Nonostante la crescita esponenzionale delle tecnologie software e hardware nell'ambito dei veicoli a guida autonoma, sono ancora tante le lacune e i problemi presenti in letteratura che ostacolano la standardizzazione di tali veicoli. Uno dei più grandi problemi presenti in letteratura fa riferimento al comportamento dei veicolo in condizioni meteo avverse e/o con luce scarsa (di notte). Infatti, condizioni meteo avverse, come: neve, nebbia, sabbia etc.., influenzano direttamente l'ambiente circostante il veicolo, compromettendo la visibilità e la capacità di percezione dei sensori, con gravi conseguenze sul riconoscimento degli oggetti. Infatti, Yuxiao Zhang et al. [1], dimostrano che le condizioni meteorologiche avverse sono un ostacolo all'implementazione dei sistemi ADS. Tale gap sulle condizioni meteo non può essere risolto senza dati, infatti, gli algoritmi utilizzati per il riconoscimento degli oggetti all'interno del sistema di percezione hanno bisogno di essere allenati, testati e validati tramite l'utilizzo di dataset. Quindi, per migliorare il comportamento e la percezione del veicolo durante condizioni meteo non ottimali, è essenziale avere abbastanza dati che coprono tutte le tipologie di condizioni meteo in cui il veicolo può trovarsi. Ma, come dimostrano gli stessi Yuxiao Zhang et al. [1] nelle pagine successive dello stesso paper, la maggior parte dei dataset non variano nelle condizioni meteo. Questo perchè la maggior parte dei ricercatori raccolgono questi dati in aree piccole (singole

[40] https://rb.gy/8v5vv

città) e in momenti di condizioni meteo ottimali (di giorno con tempo soleggiato). Allo stesso tempo, raccogliere dati su tutte le possibili condizioni meteo richiederebbe troppo tempo, poichè, particolari condizioni avverse sono rare o addirittura improbabili in base al luogo in cui ci si trova. Per esempio: è impossibile che in un'area tropicana nevichi o che in posti particolarmente freddi ci siano tempeste di sabbia. Il problema però, non rigurda soltanto il sistema di percezione e i dataset, ma anche al testing di tali veicoli, infatti, testare l'efficienza e il funzionamento di un veicolo in condizioni meteo avverse, rappresenterebbe un enorme rischio per la sicurezza in caso di malfunzionamenti. Il problema della mancanza dei dati e del testing possono essere risolti parzialmente tramite l'utilizzo di simulatori. Infatti, simulatori come: CARLA SIMULATOR, permettono ai ricercatori di creare infiniti scenari in cui un veicolo può trovarsi, modificando parametri come: condizioni meteo, tipologia di strada, quantità di traffico, quantità di pedoni etc... Il vantaggio è enorme, infatti, si possono costruire ampi dataset con qualsiasi condizioni meteo e di luce, senza dover raccogliere manualmente i dati, inoltre, i sistemi di percezione, possono essere testati nel medesimo ambiente. Il problema principale è che i dati raccolti dai simulatori sono dati "giocattolo", non reali, di conseguenza i modelli sono molto efficienti nel momento in cui vengono testati in ambiti simulativi, ma inefficienti nell'ambito reale. Tuttavia, i simulatori sono comunque ottimi per il testing dei modelli, poichè non creano pericolo in caso di malfunzionamento.

CAPITOLO 3

Implementazione

Come già accennato nella Sezione 2.6, condizioni meteo avverse e di scarsa luce rappresentano un ostacolo all'implementazione dei sistemi ADS e alla rilevazione di oggetti, sia statici che dinamici, all'interno dell'ambiente da parte dei modelli di Intelligenza Artificiale. Questo accade perchè tali condizioni influenzano l'ambiente andando a peggiorare la visibilità, offuscando o coprendo completamente gli oggetti (esempio: nebbia). In questo capitolo affronteremo questo problema, andando a implementare una Faster R-CNN (modello di Deep Learning accennato nella Sezione 3.10) per il rilevamento di oggetti in condizioni meteo avverse e di scarsa luce utilizzando un dataset contenente immagini riprese da un veicolo in tali condizioni ambientali. Lo scopo sarà quello di definire un modello che sia in grado di rilevare oggetti rilevanti all'interno dell'ambiente come: persone, veicoli, semafori, segnali stradali, etc... in modo da aumentare la sicurezza, propria e altrui, nell'ambito dei veicoli a guida autonoma. La Faster R-CNN implementata prenderà in input le immagini del training set del Dataset considerato, si allenerà su di esse e infine sarà in grado di, date nuove immagini date in input, rilevare gli oggetti presenti in figura disegnando delle bouding box (rettangolo o quadrati attorno all'oggetto) e definendo la classe di appartenenza per ognuna di essa (persona, auto, semaforo, etc...). In Figura 3.1 è possibile osservare l'architettura del modello implementato.

Figura 3.1: Architettura del modello implementato.[1]

3.1 Ambiente di sviluppo e librerie utilizzate

L'ambiente di sviluppo utilizzato per la realizzazione del modello è Google Colab; essa è una piattaforma di Google che permette di eseguire codice in Cloud, utilizzando la CPU e la GPU messe a disposizione. Colab utilizza i Jupyter Notebooks; essi sono celle in cui è possibile scrivere codice. Ogni cella può essere eseguita individualmente permettendo di organizzare e divididere il codice in passi. Inoltre colab può sincronizzarsi con Google Drive, dando la possibilità di avere all'interno del proprio workspace file che si trovano su cloud. Utilizzare una piattaforma in cloud come Google Colab ha molti vantaggi rispetto che implementare il modello in locale; infatti, Colab offre una potente CPU e GPU che velocizzano il training del modello, che, soprattutto nell'ambito di immagini e video, può durare ore e giorni. Ma, il vantaggio non è solo nella velocità di calcolo, ma, anche nello storage, infatti, sfruttando la sincronizzazione fra Google Colab e Google Drive è possibile caricare il Dataset sul Drive e averlo a disposizione all'interno del workspace di Colab; così facendo non siamo obbligati ad avere il Dataset salvato in locale, risparmiando spazio di archivazione del nostro dispositivo. Inoltre, anche tutte le librerie importate e necessarie per l'implementazione del modello vengono memorizzate in cloud, non dovendole installare manualmente in locale. Le principali librerie utilizzate per la realizzazione del modello sono:

- **Pandas e Numpy**: librerie utilizzate per manipolare i dati strutturati in formato csv e per la creazione di DataFrame. Utili anche per il parsing da dati strutturati a non strutturati sintetici.
- **PyTorch**: è un Framework basato sul Machine Learning che offre sia modelli pre-addestrati che metodi per allenare e valutare dei modelli personalizzati e

non.

- **Albumentations**: è una libreria utilizzata per operazioni di image augmentation, cioè, creazione di dati sintetici per creare diversità ed aumentare i dati all'interno del dataset.

3.2 Raccolta Dati

Il dataset utilizzato per allenare il modello è ACDC Dataset (Adverse Conditions Dataset) [29]. Il dataset è formato da 4006 immagini distribuite uniformemente tra condizioni di: nebbia, notte, pioggia e neve. Le immagini sono state registrate in Svizzera, sia su strade urbane che extra-urbane, utilizzando una fotocamera GoPro Hero 5 installata sul parabrezza interno del veicolo. Le immagini, estratte dai video registrati, sono in 1920x1080 e gli oggetti presenti al loro interno sono individuati tramite delle coordinate presenti in un file JSON a parte utilizzato per operazioni di object detection (rilevamento e riconoscimento di oggetti).

3.2.1 Split del Dataset

Come detto in precedenza, il dataset è composto da 4006 immagini divise in 4 condizioni differenti: 1006 di notte, 1000 con nebbia, 1000 con neve e 1000 con pioggia. A loro volta, ogni condizione avversa è divisa in: 400 immagini per il training, 100 per la validazione (106 per la notte) e 500 per il testing, per un totale di 1600 immagini di training, 406 di validazione e 2000 di testing (non annotate per l'object detection). Nonostante 4006 immagini totali possano sembra poche, ACDC Dataset è il Dataset che contiene più immagini in condizioni avverse rispetto tutti gli altri Dataset basati su immagini per auto di guida autonoma. Questo perchè, così come già detto in precedenza nella Sezione 2.6, la maggior parte dei Dataset sono creati durante il giorno e in condizioni meteo ottimali (soleggiate). La Figura 3.3 mostra una tabella in cui si vanno a confrontare vari Dataset in base al numero di immagini in condizioni avverse. Dalla tabella è possibile osservare che nonostante ci siano grandi Dataset come: BDD100K (composto da 100.000 video) [30] e WildDash (circa 5000 imamagini),

ACDC è quello con un numero maggiore di immagini in ambito condizioni meteo avverse.

Dataset	Adverse annot.	Fog	Night	Rain	Snow	Classes
Foggy Driving [38]	101	101	0	0	0	19
Foggy Zurich [9]	40	40	0	0	0	19
Nighttime Driving [10]	50	0	50	0	0	19
Dark Zurich [40]	201	0	201	0	0	19
Raincouver [45]	326	0	95	326	0	3
WildDash [58]	226	10	13	13	26	19
BDD100K [55]	1346	23	345	213	765	19
ACDC	**4006**	**1000**	**1006**	**1000**	**1000**	19

Figura 3.2: tabella comparativa tra il Dataset ACDC ed altri dataset[2]

3.2.2 Struttura del Dataset

Originariamente il Dataset era diviso gerarchicamente per condizioni (fog,night,rain e snow), successivemente in base al set (train, test e val) e infine per frame (GOPRO355,GOPRO356, etc...). Per comodità la struttura del dataset è stata diviso solo in base ai 3 set di: train, test e val, senza considerare le condizioni meteo e i frame. In una cartella a parte è possibile accedere ai file JSON che contengono le coordinate e la classe di appartenenza degli oggetti presenti in ogni immagine. In Figura 3.3 è possibile vedere com'è strutturato il file json:


```
{
    "annotations": [
        {
            "file_name": "fog/train/GOPR0475/GOPR0475_frame_000041_gt_panoptic.png",
            "image_id": "GOPR0475_frame_000041",
            "segments_info": [
                {
                    "area": 331782,
                    "bbox": [
                        0,
                        795,
                        1920,
                        285
                    ],
                    "category_id": 7,
                    "id": 7,
                    "iscrowd": 0
                },
```


Figura 3.3: Parte del file JSON relativo all'object detection delle immagini[3]

[2] https://arxiv.org/pdf/2104.13395.pdf

- **Annotations**: contiene tutte le annotazioni delle immagini.
- **file_name**: è una stringa che rappresenta il nome del file che rappresenta l'immagine
- **image_id**: è una stringa che identifica univocamente ogni immagine
- **segments_info**: contiene tutte le informazioni riguardante gli oggetti presenti in ogni immagine (coordinate, id classe, area, etc...).
- **area**: valore utilizzato per operazioni di object segmentation (non utilizzato)
- **bbox**: contiene le 4 coordinate utili per definire la posizione del singolo oggetto, utilizzate per disegnare le bouding box. Si trovano nel formato COCO [x, y, width, height] e ogni bbox identifica un unico oggetto.
- **category_id e id**: identificativo intero utilizzato per identificare la classe dell'oggetto identificato nel campo bbox.
- **iscrowd**: è un valore binario che indica se l'oggetto fa parte o meno di una "folla" di oggetti. Cioè se l'oggetto è isolato rispetto ad altri oggetti dell'immagine o se si sovrappone con altre figure.

Il file che contiene le annotazioni è fondamentale nell'ambito dell'object detection poichè per ogni immagine contiene le coordinate e l'id della classe degli oggetti presenti nella singola immagine; tali coordinate e id verranno utilizzate successivamente dal modello che addestreremo per imparare a rilevare e classificare correttamente gli oggetti. Infine, in Figura 3.4 è possibile conoscere il nome delle classi associate ad ogni "id_category" del file JSON.

```
[ ] classes= {7: 'road', 8: 'sidewalk', 9: 'parking', 10: 'rail track', 11: 'building', 12: 'wall',
              13:'fence', 14: 'guard rail', 15: 'bridge', 16: 'tunnel', 17: 'pole', 18: 'polegroup',
               19:'traffic light', 20:'traffic sign',24:'person',25:'rider',26:'car',27:'truck',
              28:'bus',31:'train',32:'motorcycle',33:'bicycle'}
```

Figura 3.4: Insieme di classi da cui è composto il Dataset ACDC[4]

3.3 Preparazione dei dati (Data preparation)

La preparazione dei dati è un processo in cui i dati vengono modificati al fine di essere dati in input ai modelli di Machine Learning. Tale processo è fondamentale poichè la maggior parte degli algoritmi di Machine Learning richiedono che i dati siano formattati in un formato preciso, di conseguenza, i dataset devono essere adattati prima di poter essere utilizzati. Solitamente infatti, i dataset hanno dati mancanti o informazioni incomplete, che possono provocare meno accuratezza o predizioni sbagliate da parte dei modelli. Inoltre, una buona pulizia del dataset riduce la complessità e aumenta la capacità di predizione/decisione, con un conseguente aumento delle performance. Le principali operazioni effettuate nel processo di preparazione dei dati sono: Normalizzazione, ridimensionamento delle immagini, codifica e/o modifica delle classi, trasformazione del dataset, data augmentation, data cleaning, etc.. Tutte le operazioni di preparazione dati effettuate sul Dataset ACDC saranno riportate nelle prossime sottosezioni.

3.3.1 Creazione del dataframe

La prima operazione fatta sul dataset è stata trasformarlo in un formato tabellare (Dataframe) in modo che sia più semplice da leggere e modificare. Infatti, nel formato originale, le immagini sono separate dalle proprie annotazioni,e recuperare ogni volta le annotazioni di un'immagine richiederebbe una ricerca e una lettura del file JSON. Di conseguenza, è stata definita una funzione che legge le annotazioni dal file JSON e le associa all'immagine di appartenenza all'interno di una tabella (Dataframe). Ogni riga rappresenta un unico oggetto, quindi, un'immagine che ha più oggetti al suo interno sarà presente in più righe. La Figura 3.5 mostra il nuovo formato tabellare del dataset.

	image_id	filename	labels	bbox	area	category_name
0	GOPR0475_...	fog/train...	19	[715, 758...	149	traffic l...
1	GOPR0475_...	fog/train...	20	[886, 682...	1073	traffic sign
2	GOPR0475_...	fog/train...	26	[771, 777...	507	car
3	GOPR0475_...	fog/train...	24	[892, 771...	267	person
4	GOPR0475_...	fog/train...	26	[832, 773...	679	car
...	...	...	...	...	...	...
16240	GP030176_...	snow/val/...	24	[1007, 74...	470	person
16241	GP030176_...	snow/val/...	24	[0, 624, ...	30322	person
16242	GP030176_...	snow/val/...	26	[928, 753...	1071	car
16243	GP030176_...	snow/val/...	26	[847, 762...	5846	car
16244	GP030176_...	snow/val/...	31	[0, 332, ...	137052	train

[16245 rows x 6 columns]

Figura 3.5: Nuova struttura del dataset[5]

3.3.2 Normalizzazione e modifica delle classi

Originariamente il dataset era composto dalle classi mostrate nella Figura 3.4. Ad ogni classe era associato un id che partiva dal valore 7. Tuttavia, trattandosi di un modello il cui scopo è quello di rilevare oggetti rilevanti in condizioni avverse al fine di aumentare la sicurezza alcune classi sono state eliminate. Questa decisione è stata intrapesa sia perchè avere un numero troppo elevate di classi (21 in questo caso) porta ad un'elevata complessità del modello, sia perchè lo scopo è quello di rilevare gli oggetti più importanti, come: pedoni, segnali, ciclisti e altri veicoli che un veicolo a guida autonoma può urtare in caso di condizioni di scarsa visibilità. Le classi mantenute quindi sono: semaforo, segnale stradale, persona, ciclista, macchina, camion, autobus, treno, moto e bici. Inoltre, le classi sono state normalizzate affinchè il proprio id partisse da 0 e non da 7, questo perchè il modello pre addestrato di Pytorch richiedeva che le classi partissero da 0. La Figura 3.6 mostra la nuova distribuzione delle classi con i relativi id.

```
[7]  name_classes= {0:'traffic light',1:'traffic sign',2:'person',3:'rider',4:'car',5:'truck',
                    6:'bus',7:'train',8:'motorcycle',9:'bicycle'}
```

Figura 3.6: Nuova distribuzione delle classi[6]

3.3.3 Ridimensionamento delle immagini

Le immagini originariamente avevano una risoluzione 1920x1080 pixel. Una qualità così alta porta il vantaggio di mostrare più dettagli che possono essere cruciali per la precisione della rilevazione degli oggetti, specilamente se gli oggetti nelle immagini sono piccoli o richiedono dettagli precisi per una corretta identificazione. Tuttavia, avere delle immagini così dettagliate richiedono molta meoria e richiedono un enorme costo computazionale, con tempi di addestramento molto lunghi. Di conseguenza, essendo che colab fornisce una limitata potenza di calcolo, tutte le immagini sono state convertite in 800x600 pixel.

3.3.4 Ridimensionamento e modifica delle bounding box

Le bounding box sono proporzionate rispetto le dimensioni delle immagini poichè sono disegnate precisamente attorno all'oggetto che rilevano. Di conseguenza, se la dimensioni delle immagini cambiano, anche le bouding box devono essere modificate in modo da riadattarle alle nuove immagini. Tuttavia, non è stata l'unica modifica fatta sulle bounding box, infatti, esse originariamente erano nel formato COCO [x, y, width, height], ma il modello richiedeva il formato [xmin, ymin, xmax, ymax], di conseguenza ogni box è stata adattata a questo formato. Infine, come visibile in Figura 3.5, nel dataframe le coordinate delle bouding box erano racchiuse in un'unica colonna. Ciò comportava a dover iterare su di esse ogni qualvolta che si voleva accedere a una specifica coordinata. Di conseguenza il dataframe è stato modificato in modo che la singola coordinata fosse rappresentato dalla singola colonna. In Figura 3.7 è possibile osservare questi cambiamenti.

xmin	ymin	xmax	ymax
297.916667	421.111111	337.916667	428.888889
369.166667	378.888889	457.083333	422.222222
321.250000	431.666667	334.583333	442.777778
371.666667	428.333333	377.916667	446.111111
346.666667	429.444444	360.416667	445.555556
...	...	...	...
419.583333	411.111111	426.250000	441.666667
0.000000	346.666667	86.250000	556.666667
386.666667	418.333333	404.166667	438.333333
352.916667	423.333333	394.583333	464.444444
0.000000	184.444444	318.333333	485.555556

Figura 3.7: Nuova distribuzione delle coordinate delle bounding box.[7]

3.3.5 Data augmentation

Il data augmentation è una tecnica utilizzata per aumentare la diversità dei dati di addestramento (e non solo) attraverso la generazione di varianti artificiali del dataset originale. Questo processo implica la modifica controllata delle immagini o dei dati esistenti per creare nuovi esempi che mantengano le stesse etichette di classe. Le immagini generate vengono aggiunte al dataset di addestramento esistente. Durante l'addestramento, il modello vedrà varie versioni delle immagini grazie al data augmentation, aiutandolo a imparare feature più generali e a migliorare la sua capacità predittiva. Nel contesto della visione artificiale, il data augmentation può coinvolgere varie trasformazioni, come: rotazione, zoom, ridimensionamento, riflessioni, traslazioni etc... All'interno del dasaset la data augmentation è stata applicata sia sui dati di addestramento e le operazioni effettuate sono state:

- **Horizontal Flip**: è un operazione che applica una flip orizzontale all'immagine, riflettendo l'immagine lungo l'asse verticale, simulando uno specchio orizzontale. Quest'operazione viene applicata con una probabilità del 50%.
- **RandomBrightnessContrast**: è un operazione che modifica il contrasto e la luminosità dell'immagine in modo casuale, aiutando a renderla più varia e resistente alle variazioni di illuminazione. Quest'operazione viene applicata con una probabilità del 20%

3.4 Data Visualization

La Data Visualization è il processo di traduzione dei dati in grafici e altri elementi visivi . Questo processo aiuta a guidare le decisioni durante la preparazione dei dati (se effettuato prima della preparazione dei dati), a individuare problematiche e a garantire che i dati siano pronti per l'analisi e la comunicazione. Tale processo può essere effettuato sia prima che dopo la preparazione dei dati (data preparation). La Figura 3.8 mostra un istogramma che rappresenta il numero di oggetti per classe.

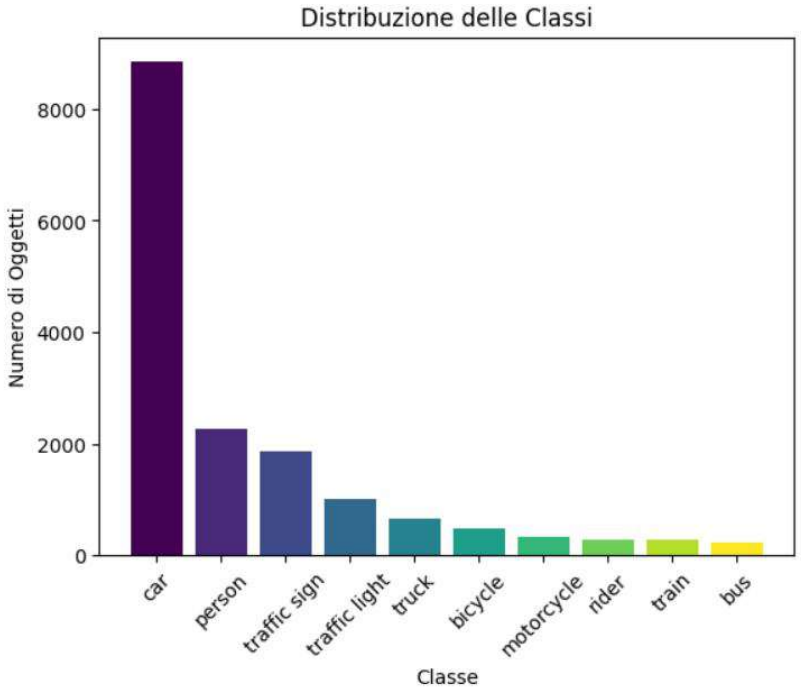

Figura 3.8: Istogramma delle distribuzioni delle classi.[8]

Nel caso di ACDC Dataset, la classe predominate è quella delle macchine, identificate come "car". Questo perchè trattandosi di video registrati da un veicolo, l'elemento più comune da trovare in strada sono proprio le auto. Nella Figura 3.9 invece, è possibile visualizzare un istogramma e un grafico a torta che mostrano la divsione tra set di dati di addestramento e di validazione.

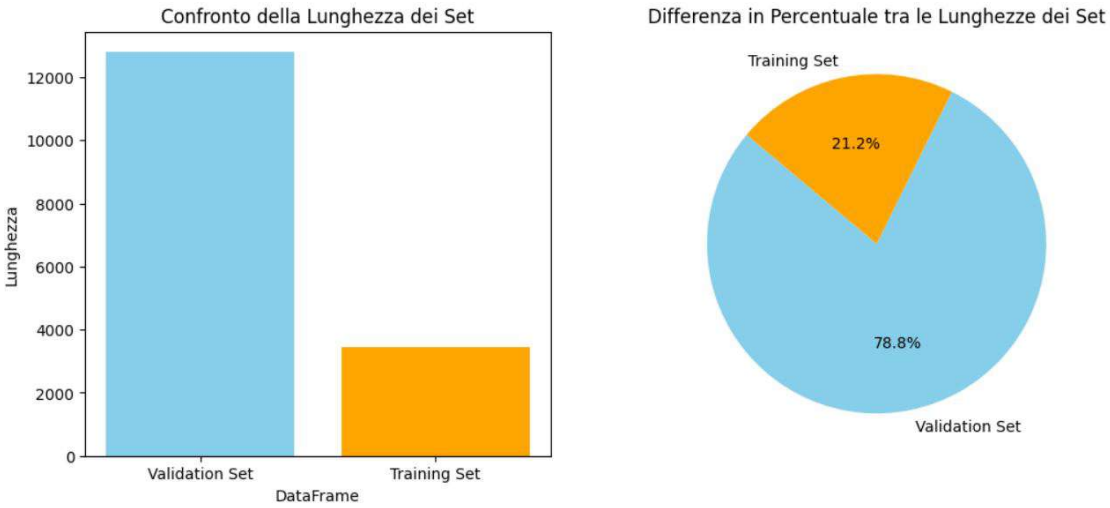

Figura 3.9: Istogramma e grafico a torta rappresentanti la divisione tra set di dati di addestramento e di validazione.[9]

Non esiste una regola sulla divisione del dataset, ne tantomeno una separazione ottimale, tuttavia, la divisione tipica del dataset è: 80% per il training e 20% suddiviso tra validation e testing. Questa soluzione è spesso utilizzata perchè la maggior parte dei dati devono essere utilizzati per addestra il modello, più dati riceve in input, più migliorerà le proprie capacità di apprendimento. Allo stesso tempo, le capacità del modello devono essere valutate, quindi si utilizza un 10-20% del dataset per la validazione. Saremo capace di conoscere la precisione del modello perchè in realtà del set di validazione già si conoscono i risultati, e quindi si andrà a calcolare l'errore effettuato dal modello per poi successivamente, in base ai risultati ottenuti, andare a modificare i parametri di apprendimento di quest'ultimo. Infine, il modello potrà ricevere in input nuovi dati separati completamente dal training e validation test, di cui non si conosce il proprio output, presenti nel test set (solitamente rappresenta il 10-20% del dataset). Questa divisione aiuta a garantire che il modello sia in grado di generalizzare bene su nuovi dati (evitando l'overfitting) e che le valutazioni delle prestazioni siano affidabili e indicative del reale comportamento del modello. Nel caso di ACDC Dataset, il set di training e di validation erano correttamente separati, con una percentuale molto vicina alla regola del 80-20%, il problema ricade sul test set, il quale era composto da un numero di immagini addirittura maggiore rispetto al training set. L'idea era stata quella di espandare il training set utilizzando la maggior parte delle immagini presenti nel test set, purtroppo però, il test set non possedeva il file JSON delle annotazioni, di conseguenza, non si conoscevano gli oggetti presenti. Esistono vari tool online che permettono la classificazione manuale degli oggetti all'interno di immagini, ma trattandosi di oltre 2000 foto, il processo avrebbe richiesto troppo effort. Il training set è stato comunque modificato andando a utilizzare soltanto il 20% della sua grandezza originaria per testare il modello.

3.5 Modello utilizzato

Per implementare il modello ho utilizzato un modello pre-addestrato fornito da PyTorch nell'ambito della libreria torchvision: fasterrcnn_resnet50_fpn. Tale modello si differenzia da una Faster R-CNN standard poichè utilizza un'architettura ResNet-50 per estrarre le feature rilevanti delle immagini date in input. Infatti, una Faster R-CNN standard utilizza come backbone network un'architettura generica, mentre, la Faster R-CNN ResNet-50 utilizza ResNet-50, un'architettura composta da 50 strati che permette di estrarre caratteristiche più avanzate dell'immagine, migliorando la qualità delle predizioni di rilevamneto degli oggetti. La Figura 3.10 mostra l'achitettura di una Faster R-cnn. La Faster R-cnn standard e la Faster R-cnn ResNet50 si differenziano soltanto nella parte inziale di feature extraction in cui si vanno a creare le feature maps per ogni immagine.

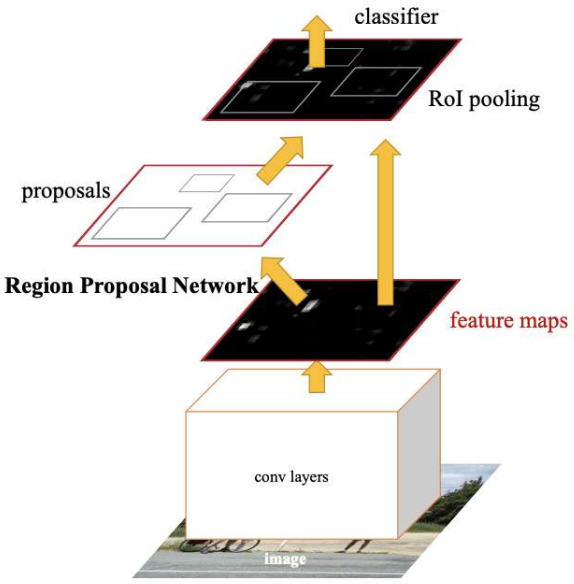

Figura 3.10: Architettura di una Faster R-cnn.[10]

La Faster R-cnn è divisa in due moduli; il primo modulo (Region Proposal Network in Figura 3.10) è una piccola rete neurale completamente connessa che restituisce le regioni proposte; il secondo modulo (classifier in Figura 3.10) è composto dal classificatore della Fast R-cnn che rileva gli oggetti all'interno delle regioni proposte

[10] https://arxiv.org/pdf/1506.01497.pdf

dalla RPN [31]. Quindi Faster R-cnn e Fast R-cnn condividono lo stesso modulo di classificazione. Di seguito vengono riportati i passi effettuati dalla Faster R-cnn ResNet50 pre-addestrata:

1. **CNN Backbone (ResNet50)**: le immagini del training set vengono date in input al CNN Backbone network che rappresenta il livello iniziale dell'architettura del modello. Tale livello è composto da una CNN che si occuperà, utilizzando vari livelli convulazionali e filtri, di estrarre le caratteristiche più rilevanti delle immagini. Le caratteristiche delle immagini vengono raccolte a diversi livelli di astrazione: inizialmente vengono estratte feature più semplici e meno rilevanti come: angoli, linee, curve etc.., mentre i livelli più profondi si occuperanno di estrarre man mano caratteristiche sempre più rilevanti. Così facendo si crea una struttura gerarchica delle caratteristiche (struttura piramidale).

2. **Region Proposal Network (RPN)**: è una rete neurale completamente connessa e ha il compito di generare possibili regioni di interesse (regioni proposte) che potrebbero contenere gli oggetti interessati in base alle feature maps generate dal livello precedente.

3. **Region of Interest (RoI) Pooling**: le regioni proposte vengono sottoposte successivamente al RoI Pooling che ha lo scopo di ridimensionarle in una dimensione comune, in modo da poter essere fornite in input ai livelli successivi.

4. **Feature Extraction e Fully Connected layers**: le regioni proposte ridimensionate vengono poi date in input a una CNN Backbone (la medesima del primo step) per estrarre e creare una mappa delle feature sulle regioni proposte. Successivamente queste mappe vengono date in input a dei livelli completamente connessi che si occuperanno della classificazione degli oggetti.

5. **Classificazione**: per ogni regione proposta il modello andrà ad indicare gli oggetti rilevati e andrà a specificare con che probabilità apparterrà a una specifica classe.

6. **Output**: il modello oltre che a specificare la classe di appartenenza per ogni oggetto e la probabilità di appartenenza (punteggio di confidenza), definisce

anche le bouding box, quindi le coordinate in cui si trova l'oggetto predetto. L'insieme delle bouding box e delle classi di appartenenza degli oggetti al loro interno rappresentano l'output del nostro modello.

3.5.1 Ottimizzazione e allenamento

Prima di allenare il modello è stato definito l'algoritmo di ottimizzazione. In questo caso è stato utilizzato l'algoritmo SGD (Stochastic Gradient Descent). L'obiettivo della funzione di ottimizzazione è quella di, iterazione dopo iterazione, aggiornare i parametri (bias e pesi) in modo che la funzione di perdita (loss function) diminuisca. SGD è un algoritmo di ottimizzazione iterativo in cui piuttosto che utilizzare l'intero dataset, viene scelto in modo casuale un piccolo sottoinsieme (definito come batch) di esso. Tale scelta randomica è il motivo per il quale c'è la parola "Stochastic" all'interno del nome SGD. Successivamente alla scelta del batch, l'algoritmo calcola il gradiente e aggiorna i parametri del modello. Tale metodo è molto efficace a livello computazione poichè ad ogni iterazione non bisognerà considerare l'intero dataset, ma solo una piccola parte di esso.
Inoltre, per ottimizzare il modello è stato utilizzata una strategia di regolazione del tasso di apprendimento durante l'allenamento fornita da PyTorch. Tale tecninca consiste nell'andare a regolare il tasso di apprendimento durante l'addestranemento per migliorare la convergenza del modello. É stata adottata tale tecnica poichè la diminuzione graduale del tasso di apprendimento aiuta a evitare oscillazioni o salti e può aiutare il modello a convergere più lentamente verso un minimo lo cale della funzione di perdita.
Infine, per allenare il modello è stato utilizzato il metodo train_one_epoch del Framework PyTorch nel quale è possibile specificare il numero di epoche da utilizzare per l'addestramento e i parametri di ottimizzazione definiti in precedenza.

CAPITOLO 4

Valutazione

Una volta allenato il modello, non siamo in grado di determinare gli effetti che l'allenamento ha avuto su quest'ultimo in termini di prestazioni. Di conseguenza, per misurare l'accuratezza e la capacità predittiva del modello si utilizza una fase, definita come fase di valutazione del modello. La valutazione di un modello è una fase/processo attraverso il quale vengono testate e misurate le sue prestazioni utilizzando un set di dati (validation set) che non è stato impiegato durante la fase di addestramento (per evitare overfitting). Per valutare un modello si fa uso di metriche di valutazione che variano in base al problema che si sta affrontando (classificazione, object detection, regressione, etc...) e che aiutano a determinare i punti di forza e di debolezza del nostro modello. Nel mio caso, trattandosi di un problema di object detection, sono state utilizzate metriche di valutazione specifiche per questo tipo di problema, il quale permettono di valutare correttamente e chiaramente le performance del nostro modello, sia in termini di localizzazione degli oggetti, sia in termini di classificazione di quest'ultimi.

4.1 Metriche

Come già definito nella Sezione 3.5, il modello implementato restituisce in output sia le bouding box (4 coordinate che rappresentano la posizione dell'oggetto) e sia la classe di appartenenza per ogni oggetto identificato (con anche associato il punteggio di confidenza). Quindi, su ogni oggetto fa una doppia predizione: inizialmente rileva l'oggetto e successivamente definisce la classe di appartenenza di quest ultimo. Di conseguenza, vanno valutate entrambe le decisioni, quindi, verranno utilizzate sia delle metriche di valutazione per valutare le bouding box (posizione) degli oggetti identificati e sia delle metriche per valutare le classi associate agli oggetti identificati. Così facendo si ha una completa conoscenza delle performance del modello appena allenato. Le metriche di valutazione utilizzate sono:

- **Average Intersection over Union (Average IoU)**: l'IoU è una metrica di valutazione utilizzata nei contesti di Object Detection e Segmentation. Tale metrica calcola un valore, definito come IoU, che rappresenta il grado di sovrapposizione tra due aree: le bouding box predette dal modello (Predicted bounding boxes) e le reali bouding box (Ground-truth bounding boxes). L'IoU viene calcolato per valutare la capacità del modello nell'individuazione degli oggetti, senza interessarsi della loro classe; più il valore dell'IoU è alto, più la bouding box predetta coincide con quella reale. Di conseguenza, il nostro obiettivo sarà quello di massimizzare il valore di IoU. L'intervallo dei possibili valori è tra 0 a 1, dove 0 indica che non esiste una sovrapposizione tra le due box (il modello sbaglia completamente a rilevare l'oggetto, Box P7 nella Figura 4.1), e dove 1 indica una sovrapposizione perfetta (il modello rileva perfettamente l'area dell'oggetto). La formula utilizzata per calcolare tale valore è la seguente:

$$IoU = \frac{\text{Area di intersezione}}{\text{Area di unione}}$$

  dove: l'Area di intersezione rappresenta l'area in cui le bouding box si sovrappongono, mentre, l'area di unione rappresenta l'area totale ricoperta dalle boxes. In Figura 4.1 è possibile osservare un esempio della metrica IoU applicata su un immagine. Nel nostro caso considereremo l'average IoU, che rappresenta la

media artimetica della somma di tutti i valori IoU rilevati sulle immagini, in modo da considerare un unico valore.

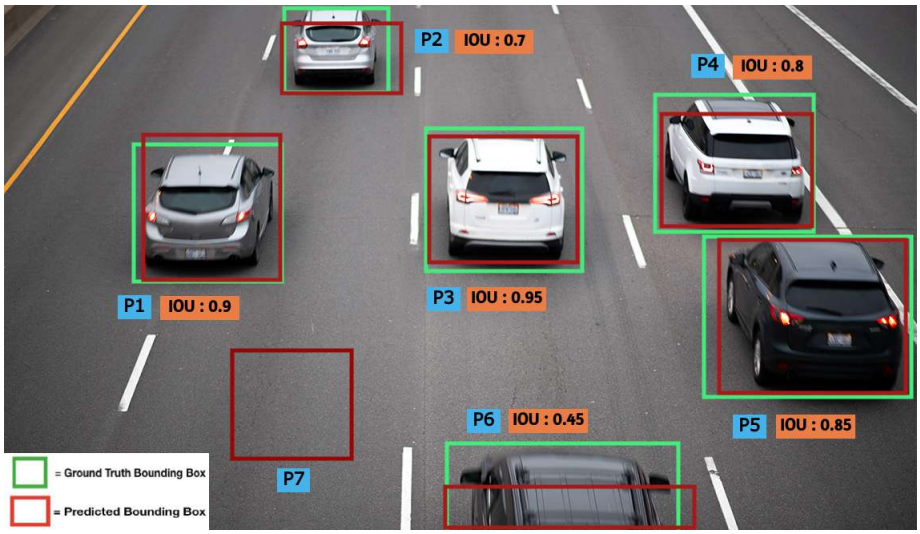

Figura 4.1: Esempio di applicazione della metrica IoU.[1]

- **Average Recall (AR)**: è una metrica utilizzata nella valutazione dei modelli di object detection per misurare quanto spesso il modello riesce a identificare correttamente gli oggetti presenti nelle immagini, senza considerare le classificazioni errate. La recall è il rapporto tra le previsioni corrette (oggetti individuati correttamente) e la somma delle previsioni corrette più le previsioni mancate. La recall si calcola tramite la formula:

$$Recall = \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

dove:

  - **True Positive (TP):** è il numero dei veri positivi. Si ha un vero positivo quando una bounding box predetta si sovrappone a una bounding box reale ha un valore superiore di IoU definito in precedenza (solitamente 0.5). Se consideriamo un IoU = 0.5, nella Figura 4.1, i veri positivi sono: P1, P2, P3, P4, P5.

[1] https://www.v7labs.com/blog/mean-average-precision

  - **False Negative (FN):** è il numero dei falsi negativi. Si ha un falso negativo quando ci sono bouding box vere che il modello non ha rilevato o quando la sovrapposizione tra box predetta e reale non supera la soglia IoU definita. Se consideriamo un IoU = 0.5, nella Figura 4.1, i Falsi negativi sono: P6, P7.

  L'average recall fa la media tra tutti i valori di recall calcolati e il numero totale di oggetti. Così facendo si ha un valore medio di recall che ci indica quanto è preciso il modello nel rilevare gli oggetti.

- **Mean Average Precision (mAP)**: è una metrica più completa e complessa rispetto le precedenti. Infatti, la metrica mAP valuta sia la localizzazione degli oggetti (come AR e IoU) che la loro classificazione. Anche in questa metrica, l'intervallo dei possibili valori è tra 0 e 1, dove un valore maggiore rappresenta una maggior accuratezza del modello, di conseguenza, il nostro obiettivo sarà massimizzare il valore di mAP. Per calcolare la mAP bisogna utilizzare altre metriche come: Precision, Recall e Average Precision. La formula completa è la seguente:

$$mAP = \frac{1}{N} \sum_{n=1}^{N} AP_i$$

  dove:

  - **N:** rappresenta il numero di classi totali. Nel caso del modello implementato N=10
  - **APi:** rappresenta l'average precision per la classe i-esima. Avendo una sommatoria che va da 1 ad N, l'average precision verrà calcolata per ogni classe. Il calcolo dell'average precision viene effettuato nel seguente modo: il modello quando effettua delle predizioni, per ogni immagine restituisce delle bouding box, le classi di appartenenza e un punteggio di confidenza che indica con che probabilità l'oggetto identificato appartiene a quella classe. Successivamente, si calcola la precision e la recall considerando per ogni immagine il punteggio di confidenza maggiore. Calcolando precision e recall, si ottiene una curva Precision-Recall. L'Average Precision

è l'area sotto questa curva, che fornisce una singola cifra che riassume la performance del modello per quella classe specifica. Un AP perfetto avrà un valore di 1. La mean average Precision (mAP) è uguale alla media artimetica delle average precision calcolate su N classi.

4.2 Configurazione del modello

In questa sezione si discuterà della configurazione del modello, cioè, dei parametri (definiti anche come iperparametri) e dei valori a essi associati, utilizzati per costruire il nostro modello e sul quale si baseranno i risultati della Sezione 4.3. La configurazione dei parametri di un modello è un processo fondamentale, poichè, i risultati, e quindi le performance del nostro modello, dipendono direttamente dai valori assegnati a questi parametri. Sul medesimo modello, una scelta errata dei valori può portare ad avere performance pessime, e viceversa, una buona scelta dei valori porta al modello ad avere migliori performance. Di seguito è riportata la lista dei parametri con i valori a essi assegnati sul quale poi successivamente è stato allenato il modello.

- **Batch Size:** è un valore intero che indica il numero di immagini che vengono date in input al modello ad ogni iterazione. Cioè, piuttosto che far allenare il modello su un'immagine per volta, si raggruppano più immagini in un insieme (batch) in modo da poter allenare il modello su più immagini per volta. Uno dei motivi per il quale si considerano le immagini in batch e non singolarmente è perchè le GPU moderne, soprattutto quelle fornite in cloud, sono molto potenti, e utilizzando solo un'immagine non si andrebbe a sfruttare la loro reale potenza, con tempi di training molto lunghi. Solitamente i valori ottimali per la taglia del batch sono 32-64, tuttavia, la GPU e la RAM fornita da Google Colab non avevano abbastanza memoria per supportare un numero così elevato di immagini. Di conseguenza, è stata scelta una **batch size = 8 per il training set** e **un batch size = 4 per il validation set**. Avere dei batch così piccoli non influisce sulle performance del modello, anzi, diminuisce il rischio di overfitting. Lo svantaggio sta nei tempi di training, che sono più lunghi rispetto a batch size di

32-64, ma avendo un dataset non troppo grande, i tempi di training non sono eccessivamente lunghi.

- **Epoche:** nel contesto del Machine Learning, si ha un'epoca quando il modello si è addestrato su tutti i batch del training set, quindi, quando ha ricevuto in input tutte le immagini appartenenti del set di addestramento. In un'epoca sono inclusi anche tutti i processi di forward e back propagation e quindi l'aggiornamento dei pesi e del bias del modello. Avere un numero di epoche troppo grande richiede più tempo di addestramento e si rischia l'overfitting, d'altro canto, potrebbe convergere meglio e avere migliori performance. scegliere invece, un numero di epoche troppo piccolo, potrebbe causare underfitting, cioè, il modello non ha avuto un numero di iterazione sufficiente per imparare dai dati, e quindi, non è in grado di effettuare predizioni correttamente. Solitamente il numero ottimale di epoche sono tra le 25 e i 50. Nel nostro modello sono state utilizzate **25 epoche per allenare il modello.**

- **Learning rate:** il learning rate, anche conosciuto come tasso di apprendimento, è un iperparametro che controlla e modifica i pesi del nostro modello in base alla funzione di perdita. In pratica, controlla quanto velocemente il modello apprende. Con un learning rate troppo basso, il modello potrebbe rimanere bloccato in minimi locali, non trovando soluzioni ottime. Tuttavia, se il learning rate è troppo alto, il modello potrebbe saltare zone della funzione di perdita che rappresentano buone soluzioni. Inizialmente, il **learning rate** è stato impostato a un **valore di 0.005**. Come già specificato nella Sezione 3.5.1, è stato utilizzato un algoritmo di ottimizzazione per regolarizzare il learning rate durante la fase di addestramento. **L'algoritmo di ottimizzazione** è stato configurato in modo tale da andare a **modificare il learning rate ogni 5 epoche** andandolo a moltiplicare con un valore gamma (fattore di riduzione) da noi definito **(gamma = 0.1)** e per un termine di regolarizzazione **(L2 regularization con valore 0.0005)**. L'idea dietro la riduzione del learning rate è quella di prendere passi più grandi all'inizio quando il modello è lontano dalla soluzione ottimale e passi più piccoli quando si avvicina alla convergenza, permettendo una maggiore precisione e stabilità durante l'addestramento.

4.3 Risultati

Dopo aver definito gli iperparametri il modello è pronto per essere allenato. Come già specificato nella Sezione 3.5.1, l'allenamento è stato effettuato utilizzando il metodo train_one_epoch() del framework PyTorch. Dopo l'addestramento, il modello è stato memorizzato su Google Drive in modo che possa essere riutilizzato, senza doverlo allenare ogni volta. Successivamente il modello è stato validato utilizzando una funzione personalizzata nel quale vengono calcolate le metriche: Average Intersection Over Unione (Average IoU), Average Recall (AR) e Mean Average Precision (mAP). La funzione di validazione ha restituito i seguenti valori in output:

Modello	Average IoU	Average Recall	mAP	Tempo di Training
Faster R-CNN ResNet-50	0.60	0.53	0.45	3h 35m

Tabella 4.1: Tabella rappresentante i risultati del modello.

Come già specificato precedentemente nel Capitolo 3, l'addestramento è stato effettuato utilizzando 2000 immagini, mentre, la validazione utilizzando 406 immagini. Le immagini sono nel formato 800x600 pixel in entrambi i set. Com'è possibile osservare dalla Tabella 4.1, il modello ha impiegato circa 3 ore e 30 minuti per completare il training, utilizzando la GPU fornita da Google Colab. Trattandosi di un dataset non troppo numeroso, le immagini sono state ridimensionate con un formato maggiore rispetto ai soliti formati utilizzati in modelli di object detection. Questa scelta è stata presa considerando il tempo di training del modello, che in questo caso, è ancora accettabile. Nel caso in cui avessimo avuto un dataset con un numero maggiore di immagini (10 mila in su), il tempo di training sarebbe diventato troppo lungo, e di conseguenza, avrei considerato formati più piccoli per le immagini in modo da abbassare i tempi di addestramento.

4.4 Matrice di confusione

La matrice di confusione è una metrica di valutazione in forma tabellare che riassume le predizioni fatte dal modello sul validation set. É chiamata in questo modo perchè mostra le istanze vere e false fatte dal modello, quindi dove esso si è "confuso" nel classificare gli oggetti. Solitamente, questa tabella viene utilizzata in problemi di classificazione, in cui ogni immagine contiene un unico oggetto. Nei problemi di object detection, la situazione è leggermente più complessa. Ogni immagine può contenere zero, uno o più oggetti di diverse classi, e la task non è solo classificare un oggetto ma anche localizzarlo all'interno dell'immagine. Pertanto, le metriche di valutazione tradizionali per l'object detection sono basate su concetti come Intersection over Union (IoU) e mean Average Precision (mAP). Tuttavia, se si desidera focalizzarsi esclusivamente sulla capacità del modello di classificare correttamente gli oggetti(ignorando la localizzazione), è possibile derivare una sorta di matrice di confusione, con lo scopo di capire dove precisamente il modello sbaglia. Ad esempio, per ogni bounding box predetta, si potrebbe confrontare la classe predetta con la classe reale dell'oggetto e riempire la matrice di confusione di conseguenza. Nel caso dell'object detection, il modello sbaglia quando si ha un falso positivo o un falso negativo. Ricordiamo che esistono 3 tipi di predizione nell'object detection (nella classificazione sono 4):

- **Vero positivo (TP):** oggetti che il modello ha correttamente identificato e la cui bounding box ha una buona IoU con il ground truth. (Predizione corretta)
- **Falso Positivo (FP):** questi sono gli oggetti che il modello ha predetto, ma che non corrispondono a nessun oggetto reale nell'immagine (o non raggiungono una certa soglia di IoU con alcun ground truth). (Predizione errata)
- **Falso Negativo (FN):** oggetti reali presenti nell'immagine che il modello non è riuscito a rilevare. (Predizione mancante)

In realtà, nei problemi di classificazione esiste anche il vero negativo, ma non viene utilizzato nell'object detection poichè risulta ambiguo. Questo perchè:

- **Veri Negativi (TN):** teoricamente, rappresenterebbe ogni possibile posizione/area nell'immagine dove non c'è un oggetto e il modello correttamente non

fa una predizione. Ma, in un'immagine ci potrebbero essere migliaia o addirittura milioni di potenziali veri negativi, rendendo questo numero non particolarmente "informativo". Per questo motivo, nei contesti di object detection, i veri negativi non sono tipicamente discussi o utilizzati.

La Tabella 4.2 mostra la matrice di confusione ottenuta, con il numero di Veri positivi, falsi positivi e negativi.

Modello	Veri Positivi (TP)	Falsi Positivi (FP)	Falsi Negativi (FN)
Faster R-CNN ResNet-50	1779	1103	1666

Tabella 4.2: Matrice di confusione.

Ricordo che questi valori non rappresentano un'immagine, ma il singolo oggetto. Il validation set è composto da 406 immagini contenente 3445 oggetti totali. Il nostro modello, basandosi sulla matrice di confusione, ha quindi:

- Riconosciuto correttamente 1779 oggetti su 3445 (~ 52% degli oggetti totali).
- Ha predetto 1103 oggetti non realmente esistenti all'interno delle immagini, sbagliando.
- Non ha classificato 1666/3445 oggetti presenti all'interno delle immagini (~ 48% degli oggetti totali).

4.5 Analisi dei risultati

Inizialmente, i risultati erano stati calcolati considerando l'intero output del nostro modello, senza filtrarlo in base allo score di confidenza. Questa scelta però, andava a penalizzare alcune metriche di valutazione come: L'average IoU e Il numero dei falsi positivi. Infatti, il numero dei falsi positivi era spoporzionato rispetto le altre metriche, con un valore di oltre 20.000, che per il numero di oggetti che abbiamo, è molto alto. Questo valore, ha successivamente causato anche problemi all'Average IoU, che aveva un valore di 0.28. La causa di un valore così basso deriva dal fatto che

le bouding box predette erano molte di più di quelle reali, quindi, la media è stata condizionata da molteplici singoli valori 0 di IoU, causando una diminuzione della media totale. Di conseguenza, l'output è stato filtrato mantenendo solo le predizioni che avevano un punteggio di cofidenza maggiore o uguale al 0.5, quindi, tutte quelle predizioni il quale modello era sicuro almeno al 50% della correttezza della propria predizione. Dopo quest'operazione, i valori dell'Average IoU e dei falsi positivi sono stati uguali ai valori presenti nella Tabella 4.1 e 4.2. Tuttavia, il numero dei veri positivi si è leggermente abbassato, ciò significa che il modello aveva correttamente classificato gli oggetti, ma nonostante questo, non era sicuro più del 50% della propria scelta, e quindi le predizioni sono state scartate durante la fase di filtraggio.
Discutiamo adesso, dei risultati ottenuti:

- **Average IoU = 0.60:** un valore di 0.6 è al di sopra la media e mostra una decente capacità del modello di localizzare correttamente gli oggetti.
- **Average Recall = 0.53:** un valore di 0.53 indica che il modello è riuscito a identificare poco più della metà dei veri positivi all'interno del validation set. Ciò suggerisce che, mentre le bouding box predette sono in genere accurate (come mostrato dall'IoU), il modello potrebbe non rilevare tutti gli oggetti che dovrebbe.
- **Mean Average Precision = 0.45:** un valore di 0.45 indica che il modello ha una precisione moderata nell'identificare le classi di appartenenza degli oggetti rilevati, ma ha ancora molto margine di miglioramento.
- **1779 Veri Positivi:** il modello ha dimostrato di rilevare correttamente un buon numero di oggetti, più della metà.
- **1103 Falsi Positivi:** tuttavia, il modello presenta anche molte predizioni errate, suggerendo che potrebbe essere troppo propenso a predire alcune classi, anche quando non sono presenti.
- **1666 Falsi Negativi:** suggeriscono che ci sono molte occasioni in cui il modello non rileva un oggetto che è effettivamente presente nell'immagine.

Quindi, basandoci sui risultati, il modello è in grado di rilevare correttamente gli oggetti all'interno delle immagini, ma, allo stesso tempo, non è in grado di rilevarli tutti, creando dei falsi negativi. I motivi per il quale il modello non è in grado di rilevare gli oggetti all'inteno delle immagini potrebbero essere: scarsa visibilità, oggetti troppo piccoli, tagliati o poco visibili per via del contesto in cui si trovano. Ma le cause potrebbero essere non legate direttamente alle immagini, ma all'addestramento. Infatti, il modello, su alcune classi poco numerose o non classificate correttamente, potrebbe essere andato in underfitting, non riuscendo ad individuare correttamente gli oggetti di alcune classi. Un'esempio potrebbe essere fatto sui cartelli stradali e i semafori. Infatti, all'interno del dataset, mentre tutte le altre classi sono correttamente identificate, con delle bouding box precise su ogni singolo oggetto, per i semafori e i cartelli stradali non è così. Dalla Figura 4.2 è possibile osservare come i semafori e i cartelli stradali non sono classificati singolarmente, ma, tramite un'unica bouding box, includendo molti dettagli ambientali, i quali non sono utili per l'identificazione di tali oggetti, ma anzi, confondono il modello in fase di predizione.

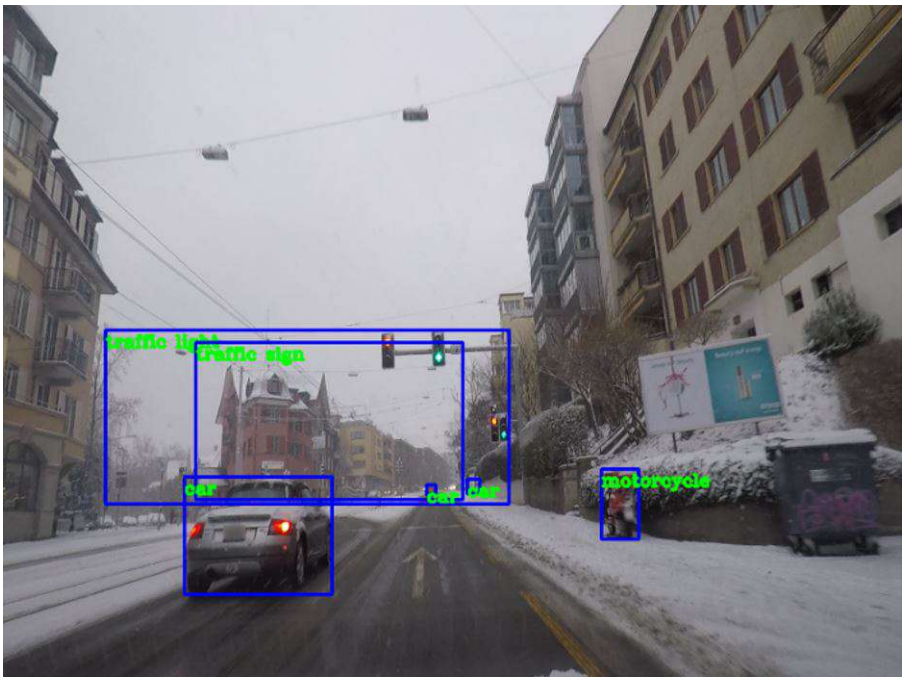

Figura 4.2: Esempio di annotazione per semafori e segnali stradali nel dataset.[2]

Questo è un problema, poichè, all'interno della bouding box vengono introdotti molti più dettagli che fanno confondere il modello in fase di predizione. Inoltre,

questi dettagli legati all'ambiente cambiano in ogni immagine, quindi, il modello non è in grado di "imparare" a riconoscere correttamente tali oggetti. Infatti, nella Figura 4.3 è possibile osservare che il segnale stradale in blu, non viene rilevato dal modello, creando un oggetto falso negativo.

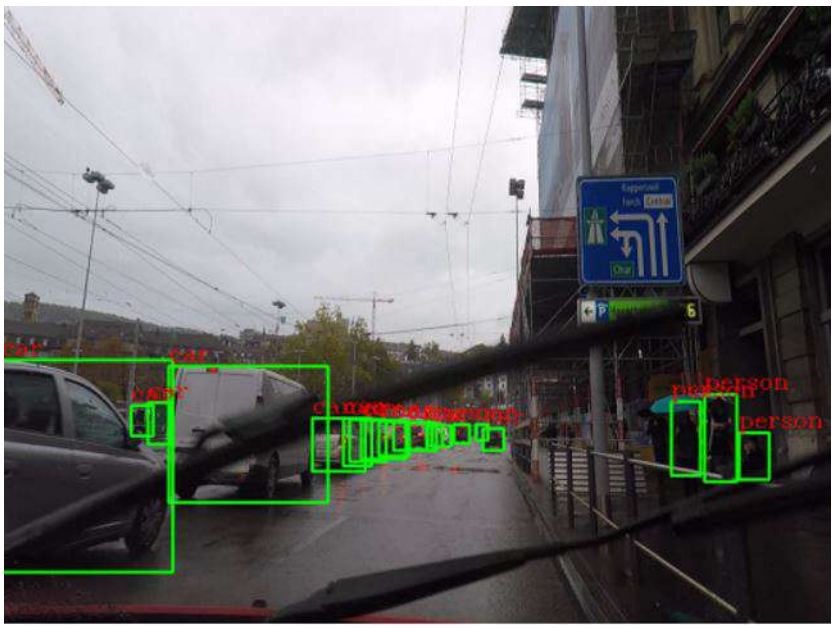

Figura 4.3: Esempio di segnale stradale non riconosciuto dal modello.[3]

Altri motivi per il quale il modello non è in grado di funzionare in maniera ottimale sono:

- **Sbilanciamento delle classi:** il training set potrebbe non avere abbastanza esempi di una determinata classe, non permettendo al modello di generalizzare correttamente.
- **Dimensione degli oggetti:** in alcuni casi gli oggetti da riconoscere sono davvero piccoli e difficili da vedere, causando una non rilevazione da parte del modello
- **Variabilità delle condizioni meteo e di illuminazione:** in alcune immagini è davvero difficile riuscire ad individure gli oggetti per via della scarsa luminazione o delle condizioni meteo.
- **Dati rumorosi:** come nel caso dei semafori e dei segnali stradali, se esistono errori nelle annotazioni delle classi il modello può confondersi. Esistono varie soluzioni: la prima è quella di eliminare tali classi, ma nel caso delle self-driving

car, i semafori e i segnali stradali sono fondamentali e non possono essere rimossi; la seconda soluzione potrebbe essere quella di eliminare le annotazioni esistenti su queste classi e riannotarle manualmente, ma questo implicherebbe l'andare a tracciare manualmente le bouding box su oltre 4000 immagini.

CAPITOLO 5

Conclusioni

Durante lo svolgimento di questo lavoro ci si è concentrati principalmente sul sistema di percezione delle auto a guida autonoma, considerando particolarmente i sottosistemi di tracciamento degli oggetti in movimento (MOT, Sezione 2.21) e di rilevazione della segnaletica stradale (TSD, Sezione 2.22). Questi due sottosistemi utilizzano modelli di Deep Learning per rilevare e tracciare oggetti dinamici e statici (esempio di elementi statici: segnali stradali e semafori) all'interno dell'ambiente. Tali modelli però, in determinate condizioni meteo, non sono efficienti e accurati come lo sono in condizioni meteo ottimali. Di conseguenza, il lavoro svolto nel Capitolo 3, si focalizza proprio su questa problematica, andando ad implementare una Faster R-CNN, cioè, un modello di Deep Learning basato su un problema di object detection, che è in grado di rilevare oggetti all'interno dell'ambiente, considerando un dataset (ACDC Dataset) che contiene immagini scattate durante condizioni meteo avverse (neve, pioggia, nebbia e di notte). Prima di implementare il modello sono state effettuate operazioni di Data Preparation e Data Visualization per adattare il dataset in modo da poter essere utilizzato per addestrare, validare e testare il modello. Successivamente è stato implementato il modello utilizzando il framework PyTorch, che fornisce modelli già definiti e pre-addestrati che possono essere allenati su dataset personalizzati come in questo caso. L'allenamento è stato effettuato utilizzando

nuovamente il framework PyTorch, tramite il metodo train_one_epoch(). Al termine dell'implementazione si sono misurate le prestazioni del modello, utilizzando 3 metriche di valutazione principali: Average IoU, Average Recall e Mean Average Precision e 3 metriche di supporto che rappresentano la matrice di confusione: True Positive, False Positive e False Negative.
I risultati dimostrano che il modello ha una buona capacità di riconoscimento degli oggetti nonostante le condizioni meteo avverse, ma non ottimale. Infatti, metriche come False Positive e False Negative segnalano che il modello ha delle lacune, poichè non identifica tutti gli oggetti presenti nelle immagini (soprattutto semafori e segnali stradali) e identifica oggetti che in realtà non sono presenti. Tuttavia, il problema legato ai Falsi Negativi, quindi agli oggetti non identificati, non dipende direttamente dall'implementazione del modello, ma, da come le classi "Segnale stradale" e "Semaforo" sono state annotate. Infatti, i creatori del dataset, hanno annotato queste classi includendo all'interno delle bounding boxes molti dettagli ambientali che hanno portato al modello a non imparare correttamente come riconoscere tali oggetti, facendolo andare in underfitting su queste classi.
Di conseguenza, in futuro, come sviluppi futuri, si potrebbe pensare di utilizzare un altro dataset, anche se, al momento, il dataset considerato è quello con un numero maggiore di immagini (4006) in condizioni avverse con annotazioni per l'object detection. Inoltre, su Kaggle non esistono dataset di questo tipo, e ciò dimostra quanto questo problema sia nuovo e attuale. Un'idea potrebbe essere quella di unire tutti i dataset trovati in un unico grande dataset, ma, non essendoci un formato standard, ogni dataset ha una propria struttura, e unirli, significherebbe doverli modificare interamente uno ad uno per adattarli in un unico formato comune. Un'idea più azzardata, potrebbe essere quella di creare un proprio dataset utilizzando immagini rilevate da un simulatore, come Carla Simulator. Infatti, ad oggi, esistono molti simulatori che permettono di creari infiniti scenari in cui un veicolo può trovarsi, modificando l'ambiente in termini di: condizioni meteo, quantità e tipologia di oggetti dinamici e statici, tipologia di strada, etc... Così facendo è possibile creare dataset contenenti milioni di immagini comprendendo tutti gli scenari possibili in cui un veicolo può torvarsi. Tale dataset successivamente potrebbe essere utilizzato per allenare il modello e per verificare le capacità di quest'ultimo si potrebbe utilizzare,

nella fase di validazione e testing, un dataset contenente immagini reali per verificare se il modello è capace di rilevare gli oggetti nonostante sia stato allenato con dei dati non realistici. Se il modello dovesse rispondere bene all'addestramento, riuscendo ad avere buone performance, non si avrebbe più il problema legato ai dataset, poichè si possono creare infiniti dataset personalizzati. Tuttavia, il modello potrebbe avere risultati pessimi, poichè i dati utilizzati per l'addestramento, rispetto quelli utilizzati per la validazione e il testing, sono diversi nella propria natura; cioè, le immagini del simulatore sono immagini non reali, "giocattolo", e quindi, il modello allenandosi su quest'ultimi, potrebbe avere buone performance su questa tipologia di dati, ma pessime su dati reali, andando in overfitting.

Bibliografia

[1] "Perception and sensing for autonomous vehicles under adverse weather conditions: A survey," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 196, pp. 146–177, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271622003367 (Citato alle pagine 2 e 48)

[2] techtarget, "self-driving car," https://www.techtarget.com/searchenterpriseai/definition/driverless-car. (Citato a pagina 5)

[3] techopedia, "Autonomous vehicle," https://www.techopedia.com/definition/30056/autonomous-vehicle. (Citato a pagina 5)

[4] S. International, "Sae levels of driving automation," https://www.sae.org/blog/sae-j3016-update. (Citato a pagina 6)

[5] faistgroup, "The 6 levels of autonomy as defined by the sae," https://www.faistgroup.com/news/autonomous-vehicles-levels/. (Citato alle pagine 7, 10 e 15)

[6] synopsys, "Sae levels of driving automation," https://www.synopsys.com/automotive/autonomous-driving-levels.html. (Citato alle pagine 7, 10, 15 e 17)

[7] mycardoeswhat, "Lane departure warning," https://mycardoeswhat.org/deeper-learning/lane-departure-warning/. (Citato a pagina 8)

[8] nissan global, "Lane departure warning," https://www.nissan-global.com/EN/INNOVATION/TECHNOLOGY/ARCHIVE/LDW/. (Citato a pagina 8)

[9] mycardoeswhat, "Blind spot warning," https://mycardoeswhat.org/safety-features/blind-spot-warning/. (Citato a pagina 9)

[10] ——, "Forward collision warning," https://mycardoeswhat.org/safety-features/forward-collision-warning/. (Citato a pagina 9)

[11] tomtom, "What is adaptive cruise control and how does it work?" https://www.tomtom.com/newsroom/explainers-and-insights/what-is-adaptive-cruise-control/. (Citato a pagina 10)

[12] mycardoeswhat, "Adaptive cruise control," https://mycardoeswhat.org/deeper-learning/adaptive-cruise-control/. (Citato a pagina 10)

[13] mazda, "Laslane keep assist system," https://www.mazda.com/en/archives/safety2/active_safety/las/. (Citato a pagina 12)

[14] volkswagen, "Lane change system side assist with rear traffic alert," https://www.volkswagen-newsroom.com/en/lane-change-system-side-assist-3678. (Citato a pagina 12)

[15] P. Pathrose, "Adas and automated driving, a practical approach to verification and validation," pp. 1–10, 2022. (Citato alle pagine 13, 16, 17, 18, 19, 24, 25, 26, 27 e 28)

[16] ŠKODA, "Škoda – traffic jam assist," https://shorturl.at/CW149. (Citato a pagina 14)

[17] Bosch, "Anteprima mondiale: sistema di parcheggio a guida autonoma," https://www.bosch-press.it/pressportal/it/it/press-release-65792.html. (Citato a pagina 18)

[18] E. W. Group, "Connected automated driving roadmap," 2019. [Online]. Available: https://www.ertrac.org/wpcontent/uploads/2022/07/ERTRAC-CAD-Roadmap-2019.pdf (Citato a pagina 19)

[19] Tesla, "Sistema di autosterzatura," https://www.tesla.com/ownersmanual/model3/it_ch/GUID-69AEB326-9831-424E-96AD-4021EABCB699.html. (Citato a pagina 21)

[20] "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742030628X (Citato alle pagine 23, 29, 30, 32, 33, 34 e 35)

[21] K. Gulen, "A match made in transportation heaven: Ai and self-driving cars," https://dataconomy.com/2022/12/28/artificial-intelligence-and-self-driving/. (Citato a pagina 36)

[22] IBM, "What is deep learning?" https://www.ibm.com/topics/deep-learning. (Citato a pagina 37)

[23] ——, "Cos'è una rete neurale?" https://www.ibm.com/it-it/topics/neural-networks. (Citato a pagina 38)

[24] towardsdatascience, "Convolutional neural networks, explained," https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939. (Citato a pagina 40)

[25] IBMCNN, "Convolutional neural networks," https://www.ibm.com/topics/convolutional-neural-networks. (Citato alle pagine 40, 41 e 43)

[26] N. McCullum, "he flattening and full connection steps of convolutional neural networks," https://www.nickmccullum.com/python-deep-learning/flattening-full-connection/. (Citato a pagina 44)

[27] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587. (Citato a pagina 45)

[28] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448. (Citato a pagina 46)

[29] C. Sakaridis, D. Dai, and L. Van Gool, "ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021. [Online]. Available: https://acdc.vision.ee.ethz.ch/ (Citato a pagina 52)

[30] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," 06 2020, pp. 2633–2642. (Citato a pagina 52)

[31] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf (Citato a pagina 62)