Università degli Studi della Calabria

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

---

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea

Riconoscimento e tracciamento di oggetti nelle smart city tramite general-purpose sensing ed intelligenza artificiale

Relatori

Prof. Giancarlo Fortino

Ing. Roberto Minerva

Prof. Noel Crespi

Ing. Claudio Savaglio

Candidato

Stefano Perna

Matr. 235278

Anno Accademico 2022-2023

Alla mia famiglia

Sommario

La realizzazione di Smart City rappresenta una soluzione innovativa per affrontare le sfide dell'urbanizzazione e ottimizzare l'efficienza dei servizi urbani. Il presente lavoro di tesi si inserisce in questo contesto e si focalizza sul concetto di *measurable city*, che riguarda la raccolta e l'analisi dei dati con lo scopo di misurare vari aspetti dell'ambiente urbano, come ad esempio il traffico, nonostante l'assenza di dispositivi e sorgenti dati specifici ed affidabili. Al fine di ovviare a tali criticità, questo lavoro di tesi si concentra nello sviluppo di strumenti innovativi per il monitoraggio dell'ambiente urbano sfruttando dati raccolti da sensori *general-purpose*. Questi sensori possono comprendere telecamere, microfoni, sensori di inquinamento e altri dispositivi in grado di rilevare le grandezze fisiche nell'ambiente circostante.

Attraverso l'implementazione di una pipeline di Machine Learning, il sistema proposto mira all'identificazione e alla classificazione precisa degli oggetti che caratterizzano l'ambiente di interesse, con l'obiettivo di creare una fonte dati affidabile e rappresentativa del contesto urbano. I dati raccolti potranno poi essere utilizzati per il monitoraggio del traffico in diversi contesti all'interno delle Smart City, specie in sinergia con il paradigma innovativo dei Digital Twin e tecnologie di Edge Intelligence: ciò consentirà l'esecuzione di complessi modelli di inferenza in tempo reale e la distribuzione del sistema ai bordi della rete nonostante le limitate risorse hardware dei dispositivi coinvolti, allo scopo di ottenere una rappresentazione digitale olistica degli elementi del traffico cittadino.

Il presente studio ha prodotto un sistema di monitaraggio del traffico, attraverso il rilevamento e l'identificazione degli oggetti in sequenze di immagini. Per valutare l'efficacia del sistema, sono stati condotti test attraverso l'inferenza su video di traffico reale, generando così un dataset ampio e rappresentativo del contesto in analisi. L'implementazione di un framework di valutazione su dataset di riferimento ha confermato l'accuratezza del sistema nel rilevare e tracciare gli elementi del traffico urbano, utilizzando metriche di valutazione specifiche. In particolare, si è ottenuto un valore di mean Average Precision del 44% e di Average Precision per quanto riguarda le sole automobili del 66%. Queste metriche evidenzionano le ottime capacità del sistema nel rilevamento. Per quanto riguardo le abilità di tracciamento, il valore di Multi-Object Tracking Accuracy si attesta sullo 0.35% mentre la Multi-Object Tracking Precision (overlap) è pari a 0.85%. I risultati ottenuti sottolineano l'efficacia del sistema proposto nel contesto del monitoraggio del traffico urbano e forniscono valutazioni quantitative riguardo le sue prestazioni. Inoltre, l'inferenza su dati provenienti da situazioni reali ha contribuito a validare la robustezza e l'applicabilità del sistema in condizioni dinamiche e variabili, fornendo una base solida per ulteriori ricerche e miglioramenti nell'ambito del monitoraggio del traffico urbano.

Indice

Capitolo 1

Introduzione

L'evoluzione delle città verso il paradigma della Smart City rappresenta un passo significativo per affrontare le sfide dell'urbanizzazione moderna e per ottimizzare l'efficienza dei servizi urbani. In tale contesto, questo lavoro di tesi si focalizza sul concetto delle *measurable city*, cioè sulla centralità della raccolta e dell'analisi dei dati per misurare, monitorare, controllare e ottimizzare vari aspetti dell'ambiente urbano, e in particolare il traffico.

Il presente capitolo fornisce un'introduzione al contesto di ricerca e offre un quadro dettagliato del lavoro di tesi. Viene in primis presentata una panoramica sul modello delle Smart City, della quale si discutono le applicazioni e le sfide. Nell'ambito delle measurable city, si discute inoltre un nuovo approccio per la raccolta dati tramite sensori ad uso generale, secondo il paradigma general-purpose sensing. Infine, viene delineata la struttura del lavoro di tesi e i suoi obiettivi.

1.1 Smart city

Le *Smart City* rappresentano un paradigma innovativo nello sviluppo urbano indirizzato a migliorare la qualità della vita dei cittadini ottimizzando l'utilizzo delle risorse e migliorandone l'accessibilità attraverso l'utilizzo di tecnologie di ultima generazione.

Il concetto di Smart City si è evoluto nel corso degli anni, riflettendo gli avanzamenti tecnologici e il cambiamento delle dinamiche del contesto urbano. Nelle fasi iniziali, il focus era posto principalmente nell'uso di tecnologie avanzate e specifiche allo scopo di ottimizzare le operazioni ed i servizi urbani. Più recentemente, invece, l'enfasi si è spostata su un approccio olistico che pone maggiore attenzione alle esigenze dell'individuo (*citizen-centric*) e mira alla crescita sostenibile delle città [1], per sviluppare soluzioni e servizi che tengano conto delle necessità, delle aspettative e delle esperienze quotidiane dei cittadini. Questo approccio mira a creare comunità più consapevoli, partecipative e coinvolte nel processo decisionale e nell'adozione di determinate tecnologie. Le città

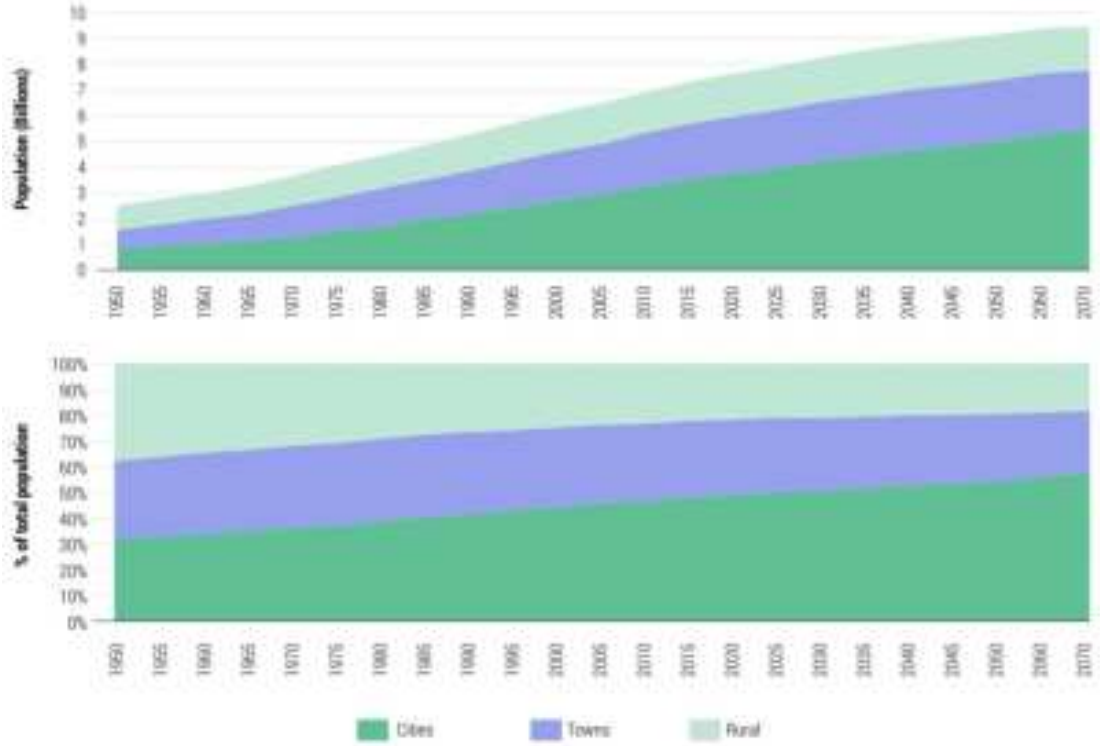


Figura 1.1: *Evoluzione del grado di urbanizzazione della popolazione 1950-2070*

moderne devono fare i conti con una popolazione urbana in costante crescita, stando ad un recente report "Envisioning Future Cities: World Cities Report 2022" del Programma delle Nazioni Unite per gli insediamenti umani (UN-Habitat), infatti, più del 40% della popolazione mondiale vive in città ed ci si aspetta che si raggiunga il 58% entro il 2070. L'aumento demografico porta con sé una serie di sfide urbane complesse, tra cui il proliferare del crimine, la congestione del traffico, la produzione di rifiuti, l'elevata domanda di acqua ed energia, l'inquinamento atmosferico e acustico, l'accesso diseguale ai servizi sanitari, all'edilizia e all'istruzione, oltre a una disparità nella distribuzione della ricchezza. Per affrontare efficacemente le sfide derivanti dall'urbanizzazione, il concetto di smart city si è diffuso ampiamente, proponendosi come soluzione per sostenere la crescita urbana ed economica, e affrontare in modo simultaneo le problematiche ambientali e sociali connesse a tale sviluppo.

Il termine Smart City è oggetto di discussione e non esiste una definizione universalmente accettata. Tra queste, una comunemente citata è stata fornita da Caragliu, Del Bo e Nijkamp (2009)[2], che descrivono come intelligenti città che utilizzano tecnologie dell'informazione e della comunicazione (ICT) per arricchire i servizi urbani e gestire efficacemente le risorse. Ciò implica l'integrazione di infrastrutture avanzate, processi decisionali basati sui dati, e soluzioni innovative per affrontare le sfide urbane.

Le infrastrutture rappresentano un pilastro fondamentale all'interno di una Smart City, abbracciando reti di comunicazione ad alta velocità, sistemi di sensori e dispositivi interconnessi che danno vita all'Internet delle cose (IoT). Queste infrastrutture costituiscono la base tecnologica essenziale per la raccolta e la condivisione in tempo reale di dati che riguardano svariati aspetti della città, dalle dinamiche di utilizzo delle risorse alla mobilità urbana. L'IoT si compone di dispositivi e sensori intelligenti distribuiti su vasta scala all'interno della città, capaci di comunicare tra loro o con dispositivi remoti attraverso la rete internet. Questi strumenti raccolgono dati in tempo reale su parametri diversi, come la qualità dell'aria, il flusso del traffico, il consumo energetico, e sono in grado di eseguire azioni controllate in risposta agli eventi rilevati nell'ambiente circostante. La raccolta di tali dati si traduce in informazioni utili per le decisioni urbane e contribuisce al potenziamento dell'efficienza dei servizi offerti. Tuttavia, la costruzione di un'architettura IoT è un compito estremamente complesso. Essa richiede l'integrazione di una vasta gamma di dispositivi, tecnologie di collegamento e servizi nelle smart cities [3]. Pertanto, la rete dovrebbe essere incorporata con funzionalità di attuazione, networking, elaborazione e rilevamento [4]. Monitorare, raccogliere, archiviare e condividere dati aperti dai sensori IoT è altresì un obiettivo cruciale per agevolare lo sviluppo e l'analisi delle smart cities [5]. La complessa gestione di queste reti di sensori rappresenta un elemento critico che richiede attenta pianificazione, manutenzione, sicurezza e ottimizzazione. Parallelamente, i processi decisionali basati sui dati costituiscono un altro aspetto chiave. Nell'ultimo decennio, l'avvento e l'applicazione di approcci di intelligenza artificiale (IA) hanno dimostrato la loro validità nell'analizzare grandi quantità di dati per ottenere insight significativi. Il termine IA, in questo contesto, include una vasta gamma di tecniche ed algoritmi capaci di apprendere dai dati, come data science, statistical learning, machine learning, e deep learning. Inoltre, si estende a sistemi intelligenti in grado di eseguire svariate attività, tra cui la percezione, il ragionamento e l'inferenza, che comprendono sistemi esperti, modelli grafici probabilistici, e reti bayesiane [6]. Un contributo significativo dell'IA è emerso nell'analisi dei dati raccolti attraverso una moltitudine di sensori IoT, consentendo una gestione e un utilizzo più efficiente delle risorse. Come evidenziato da Greg Stone [7], "Se conosci le domande giuste e comprendi i rischi, i dati possono contribuire a costruire città migliori", e l'IA è fondamentale per estrarre tali intuizioni dai dati. L'integrazione sinergica di IA e IoT apre prospettive sempre più ampie per migliorare la comprensione del contesto urbano, anticipare le esigenze della città e facilitare decisioni informate per una gestione ottimizzata delle risorse.

1.1.1 Applicazioni

Le applicazioni nell'ambito delle Smart City che sfruttano l'integrazione tra IA ed IoT, in particolare attraverso l'uso di sensori, sono molteplici e stanno rivoluzionando la gestione urbana. Diversi studi scientifici evidenziano l'impatto positivo di questa convergenza tecnologica sulla creazione di città più efficienti e orientate al benessere dei cittadini [6].

Un esempio chiave riguarda il settore della mobilità urbana, in cui la combinazione di IA

e IoT rivoluziona il monitoraggio del traffico e la gestione del trasporto pubblico. Sensori intelligenti posizionati in vari punti della città raccolgono dati in tempo reale sulla densità del traffico, la velocità dei veicoli e l'utilizzo dei mezzi pubblici. L'IA analizza questi dati per prevedere modelli di traffico, ottimizzare le rotte e migliorare la sincronizzazione dei trasporti pubblici, riducendo congestionamenti e tempi di percorrenza.

- **Mobilità urbana:** Un esempio chiave riguarda il settore della mobilità urbana, in cui la combinazione di IA e IoT rivoluziona il monitoraggio del traffico e la gestione del trasporto pubblico. Sensori intelligenti posizionati in vari punti della città raccolgono dati in tempo reale sulla densità del traffico, la velocità dei veicoli e l'utilizzo dei mezzi pubblici. Le capacità di riconoscimento e predittive dell'IA possono contribuire alla stima del volume del traffico, fornendo ulteriori dati per l'ottimizzazione delle rotte e la gestione efficace della mobilità urbana [8]. Queste tecniche possono essere impiegate per migliorare la sicurezza stradale, il rilevamento di eventi critici come la distrazione del conducente o incidenti stradali [9] [10] e l'analisi della percorribilità stradale [11] [12], offrendo un approccio completo e avanzato alla gestione del traffico urbano. Tutte queste applicazioni richiedono un monitoraggio costante dell'ambiente urbano di riferimento ed il riconoscimento di diversi eventi. Questo implica spesso la necessità di individuare e riconoscere vari tipi di oggetti come i pedoni, veicoli e oggetti stradali.

- **Sicurezza urbana:** Un'altra applicazione di rilievo riguarda l'ambito della sicurezza urbana. Il riconoscimento, il tracciamento e/o la predizione di crimini all'interno del contesto cittadino ricopre un ruolo cruciale. Tecniche di IA possono supportare il lavoro delle forze dell'ordine in situazioni di emergenza, migliorando la prontezza e l'efficacia del loro intervento.

- **Monitoraggio dell'ambiente:** Altro settore di notevole interesse è quello ambientale, in cui sensori IoT misurano la qualità dell'aria, le emissioni di gas serra e la gestione dei rifiuti. L'elaborazione di questi dati in tempo reale permette di identificare pattern di inquinamento, prevedere situazioni di rischio e ottimizzare le strategie di gestione ambientale.

1.1.2 Measurable city e general-purpose sensing

L'implementazione di una Smart City è un'impresa che si scontra con numerose sfide. In una smart city, le tecnologie non sono isolate ma bensì integrate in modo fluido in vari aspetti della vita urbana. Ciò include infrastrutture, servizi pubblici, trasporti e persino le attività quotidiane dei cittadini, i quali devono essere coinvolti ed educati nell'adozione di nuove tecnologie. Inoltre, una smart city è caratterizzata dalla sua capacità di attuazione. Ciò significa che non si limita a raccogliere dati, bensì li utilizza per rispondere automaticamente mediante l'utilizzo di avanzati sistemi intelligenti e di automazione, anche in tempo reale. Dunque, lo sviluppo di un'infrastruttura ICT così complessa, composta da sensori, attuatori, agenti, sistemi intelligenti nello spazio fisico che comunicano attraverso diversi canali di comunicazione ed interagiscono con l'ambiente urbano ed i

cittadini rappresenta una notevole barriera iniziale per l'avvio di un'iniziativa di questo tipo [13]. La complessità di coordinare e integrare tutti questi elementi all'interno di una città è una sfida di proporzioni considerevoli. Pertanto, prima ancora di diventare *smart* le città moderne devono essere *misurabili*, come evidenziato in [14]. Questo concetto introduce la nozione di Measurable City un approccio che si concentra sulla raccolta e l'analisi dei dati per misurare vari aspetti dell'ambiente urbano, con lo scopo di rappresentare le città attraverso modelli accurati ed eseguibili. Le measurable city non solo forniscono un quadro dettagliato della realtà urbana, ma offrono anche la base necessaria per implementare soluzioni smart in modo più mirato ed efficiente. La raccolta sistematica di dati, unita a modelli di analisi avanzati, consente una comprensione approfondita dei bisogni della città e delle possibili aree di miglioramento. In questo modo, la transizione verso una Smart City diventa più organica e mirata, superando alcune delle sfide intrinseche legate all'adozione di tecnologie all'avanguardia. La measurable city, dunque, emerge come un tassello fondamentale nel complesso puzzle della trasformazione urbana verso un futuro più intelligente e sostenibile. Tuttavia, nonostante si abbia a che fare con un modello più semplice rispetto ad un paradigma più ampio e complesso qual è una Smart City, la realizzazione di una measurable city presenta numerose difficoltà e sfide. Una delle principali sfide è la carenza di fonti di dati affidabili e che siano in grado di rappresentare in modo completo ed esaustivo i diversi aspetti del contesto cittadino. La rappresentazione accurata e dettagliata dello scenario urbano richiede un approccio innovativo per superare questa limitazione. Un'altra sfida riguarda la necessità di integrare, installare, gestire e coordinare una moltitudine di componenti, hardware e software, ed instradare la grande mole di dati prodotta per la sua elaborazione ed analisi.

Le città moderne sono caratterizzate da una grande complessità e varietà di fenomeni da misurare. I dati raccolti da sensori ad uso specifico, come sensori di traffico e inquinamento, sono spesso costosi, difficili da installare e manutenere. Inoltre, i dati raccolti da questi dispositivi sono spesso limitati a un singolo aspetto dell'ambiente urbano. Questo rende difficile ottenere una visione completa della città. Il lavoro di tesi si propone di affrontare questa sfida sviluppando strumenti avanzati per il monitoraggio del contesto urbano attraverso l'impiego di sensori *general-purpose*, come telecamere, microfoni e sensori ambientali, in grado di rilevare una vasta gamma di grandezze fisiche nell'ambiente circostante. Il general-purpose sensing si propone di superare i limiti di una raccolta di dati legati ad uno scopo specifico, aprendo la strada a una piattaforma versatile in grado di rilevare una vasta gamma di informazioni ambientali utilizzando un ridotto insieme di dispositivi. Questo approccio contrasta con l'approccio "single-purpose", in cui un sensore è dedicato a raccogliere dati su una singola variabile [15]. L'approccio general-purpose sensing, invece, si presenta come una soluzione più efficace quando le caratteristiche da monitorare aumentano, e richiederebbero quindi una più ampia infrastruttura di rilevamento. L'adozione di sensori *general-purpose* offre numerosi vantaggi, tra cui una riduzione dei costi di implementazione, manutenzione e impatti sociali ed estetici, spostando la complessità dalla parte hardware a quella software. Questo approccio non solo ottimizza l'efficienza operativa ma affronta anche le questioni

estetiche e sociali legate all'installazione di strumenti di rilevamento. Tuttavia, è necessario considerare come l'applicazione di sensori general-purpose in contrapposizione a quella di sensori specifici, potrebbe dimostrare effetti negativi sull'accuratezza e la precisione dei dati raccolti, rendendo più difficoltosa la fase di elaborazione, di analisi e l'estrapolazione di informazioni rilevanti. Di fatto, l'approccio del general-purpose sensing sposta la complessità dalla fase di distribuzione, gestione dell'hardware e del rilevamento dati, all'elaborazione tramite strumenti software. Pertanto, diviene ancora più decisiva l'integrazione di tecniche di intelligenza artificiale per ampliare le capacità sensoriali.

1.2 Multi-Object Tracking nel traffico

Nell'ambito della mobilità urbana, il tracciamento di oggetti multipli rappresenta una sfida cruciale per migliorare la gestione del traffico e garantire la sicurezza stradale, attraverso il monitoraggio. Il riconoscimento e il tracciamento di veicoli, pedoni, e altri elementi nell'ambiente urbano sono fondamentali per ottimizzare le infrastrutture cittadine e prevenire situazioni critiche.

Il Multi-Object Tracking (MOT) nel traffico si propone di monitorare simultaneamente la posizione e il movimento di più oggetti in movimento all'interno di un'area urbana. Questo compito è complesso a causa della varietà degli oggetti, delle possibili occlusioni e delle mutevoli condizioni ambientali.

L'implementazione di tecniche avanzate di apprendimento automatico, in particolare algoritmi di visione artificiale e deep learning, è fondamentale per affrontare queste sfide. Le potenzialità del MOT nel traffico sono molteplici. Innanzitutto, consente un monitoraggio continuo e in tempo reale del flusso veicolare, permettendo una gestione più efficiente del traffico e l'ottimizzazione dei tempi di percorrenza. Inoltre, il tracciamento degli oggetti contribuisce significativamente alla sicurezza stradale, consentendo la rilevazione tempestiva di comportamenti pericolosi, incidenti o situazioni di emergenza. L'applicazione di queste tecniche di MOT nell'ambito urbano, supportate dalla raccolta sistematica di dati provenienti da sensori general-purpose, come telecamere e sensori ambientali, che forniscono informazioni dettagliate sulla dinamica del traffico urbano, può contribuire in modo determinante allo sviluppo di una measurable city.

1.3 Obiettivi del lavoro di tesi

Il presente lavoro di tesi è il frutto di una ricerca portata avanti nel corso del periodo di mobilità all'interno del progetto Erasmus+ in collaborazione tra il team di ricercatori del laboratorio diretto dal Professore Noel Crespi (DICE Lab, Telecom SudParis, Parigi - Francia), e il laboratorio diretto dal Professore Giancarlo Fortino (SPEME Lab, Unical, Rende - Italia) del Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica.

La ricerca, inserita nel contesto delle measurable city, si propone di affrontare le sfide legate al riconoscimento e tracciamento di oggetti nell'ambiente urbano. L'obiettivo principale è applicare e valutare le più recenti tecniche di IA e di Machine Learning per implementare un efficace sistema di MOT utilizzando dati provenienti da sensori general-purpose. I passi chiave della ricerca includono la selezione e l'implementazione di algoritmi avanzati di visione artificiale e deep learning, la creazione di dataset rappresentativi dell'ambiente urbano, e la valutazione delle prestazioni del sistema di tracking sviluppato. I dati raccolti saranno fondamentali per supportare la trasformazione delle città in entità misurabili, aprendo la strada alla realizzazione di soluzioni smart più mirate ed efficienti. I risultati ottenuti saranno valutati nel contesto del monitoraggio del traffico nelle Smart City. Inoltre, la ricerca terrà conto della crescente importanza dell'Edge Intelligence, esplorando la possibilità di distribuire il sistema ai bordi della rete. Sarà fondamentale considerare le limitate capacità hardware dei dispositivi Edge, soprattutto per esecuzioni di complessi modelli di inferenza in tempo reale.

Il lavoro di tesi è strutturato come segue: nel Capitolo 2 verrà esplorato il panorama attuale del tracciamento di oggetti in ambienti urbani, analizzando le più recenti proposte nel campo presenti in letteratura. Nel Capitolo 3 si andrà a dettagliare lo sviluppo di un framework di Multi-Object Tracking (MOT), con un focus sulla metodologia, le tecnologie coinvolte e i dati utilizzati. Verrà presentata la creazione di un dataset di riferimento attraverso l'utilizzo del sistema proposto. Sarà inoltre introdotto un framework di valutazione del MOT basato sul dataset UA-Detrac. Nel Capitolo 4 si esamineranno i risultati sperimentali, si effettuerà un'analisi del dataset generato, una valutazione dei tempi di inferenza del sistema proposto e verranno presentati i risultati della valutazione del MOT attraverso metriche come mean Average Precision (mAP), Multi-Object Tracking Accuracy (MOTA) e Multi-Object Tracking Precision (MOTP). Conclusioni e sviluppi futuri sono infine riportati nel Capitolo 5.

Capitolo 2

Stato dell'arte

In questo capitolo, viene fornita una panoramica sullo stato dell'arte nel campo del tracciamento di oggetti in ambienti urbani. La crescente urbanizzazione ha portato ad un incremento vertiginoso delle attività all'interno del contesto urbano, con un conseguente incremento del traffico stradale, rendendo fondamentale l'adozione di sistemi intelligenti per ottimizzare la mobilità. In questo contesto, la visione artificiale emerge come uno strumento chiave per sviluppare soluzioni rigorose ed affidabili per l'estrazione di informazioni rilevanti. Questi processi, attraverso l'analisi del contesto di interesse, possono contribuire alla gestione ed alla risoluzione di numerosi problemi che caratterizzano l'ambiente cittadino. La capacità di identificare, localizzare, tracciare e categorizzare oggetti specifici nel contesto di riferimento costituisce la base per la comprensione dello stesso. In particolare, in questa ricerca si porrà particolarmente attenzione sulle categorie di oggetti rilevanti per il traffico, veicoli e pedoni su tutti. La presente disamina si concentrerà sui metodi per il tracciamento degli oggetti nell'ambiente urbano presenti ad oggi in letteratura, esaminando approcci come la object detection attraverso reti neurali convoluzionali (CNN) e algoritmi di tracciamento basati su filtri di Kalman. Verranno presentati anche dataset di riferimento come UA-Detrac e Waymo Open Dataset, essenziali per la valutazione delle performance. Infine, si esploreranno metriche di valutazione come Precision, Recall, mAP, MOTA e MOTP, le quali forniscono una valutazione completa delle capacità di un sistema di tracciamento di oggetti.

2.1 Metodi per il tracciamento degli oggetti nell'ambiente urbano

Quando si parla di Multi-object Tracking (MOT), si fa tipicamente riferimento a due task: il rilevamento degli oggetti (Object Detection) ed il loro tracciamento (tracking) attraverso i differenti frame, entrambi sono strettamente correlati poiché il tracciamento

degli oggetti dipende dalla loro rilevazione. Lo scopo della object detection è determinare se ci sono istanze di oggetti appartenenti a categorie specifiche (come persone, automobili, biciclette, ecc.) in un'immagine e, se presenti, restituire la posizione spaziale e l'estensione di ogni istanza di oggetto (ad esempio, attraverso un bounding box) [16]. Mentre, il tracciamento degli oggetti può essere definito come il processo di monitoraggio di un oggetto lungo diversi frame, mediante l'associazione di un identificatore univoco, rendendo nota la sua posizione e direzione durante l'intera sequenza. Questo processo è fondamentale per comprendere il movimento e l'interazione degli oggetti in un contesto dinamico come l'ambiente urbano. La struttura di base di un algoritmo MOT, rappresentata in fig. 2.1 inizia con l'acquisizione di un video o di una sequenza di frame. Inizialmente, un algoritmo di rilevamento individua gli oggetti nei diversi frame, restituendo un bounding box per ciascun oggetto rilevato. Successivamente, queste rilevazioni sono analizzate dall'algoritmo di tracciamento, che estrae le caratteristiche per cercare un'affinità con gli oggetti precedentemente rilevati in frame precedenti (nel caso di elaborazione online) e/o con oggetti rilevati in frame precedenti o successivi (nel caso di elaborazione offline). Questa ricerca di affinità porta all'associazione di oggetti esistenti e nuovi tra i frame, restituendo la traccia degli oggetti. L'architettura appena descritta però non è l'unica possibile per il task di MOT. Ad esempio è possibile considerare due approcci diversi sulla base del metodo di inizializzazione dell'algoritmo di tracking. In particolare, si parla di metodi Detection-Based (DBT) e Detection-Free (DFT). Nel primo caso si utilizza un modello di rilevamento degli oggetti frame per frame, come nell'architettura precedentemente descritta, nel secondo caso, invece, gli oggetti da tracciare sono selezionati manualmente dal primo frame in analisi. Un'altra classificazione riguarda la modalità di tracciamento, online o offline. L'online tracking è caratterizzato dall'uso del frame corrente o di quelli passati per identificare e/o reidentificare gli oggetti rilevati. Mentre, nell'offline tracking si fa uso di tutti i frame della sequenza, passati presenti e futuri [17].

2.2 Tecniche di object detection

Nelle ultime due decadi i progressi nell'ambito della object detection hanno attraversato due periodi storici 2.2: il primo, contraddistinto dall'utilizzo di metodi cosídetti *tradizionali* (prima del 2014) ed i metodi basati sulle *deep neural network* (DNN) [18]

I metodi tradizionali si basano su caratteristiche create manualmente ed algoritmi complessi per la rilevazione degli oggetti nelle immagini. Per citarne alcuni: Viola Jones [19], utilizzato per il riconoscimento di volti in tempo reale, si basa sul metodo delle *sliding-window* per analizzare le immagini in diverse zone ed a scale diverse, utilizzando determinate maschere per individuare i punti caratteristici del viso. HOG (Histograms Oriented Gradients), introdotto da Navneet Dalal e Bill Triggs nel 2005 [20], si basa sull'analisi delle distribuzioni dei gradienti di intensità nelle immagini. In sostanza, l'im-

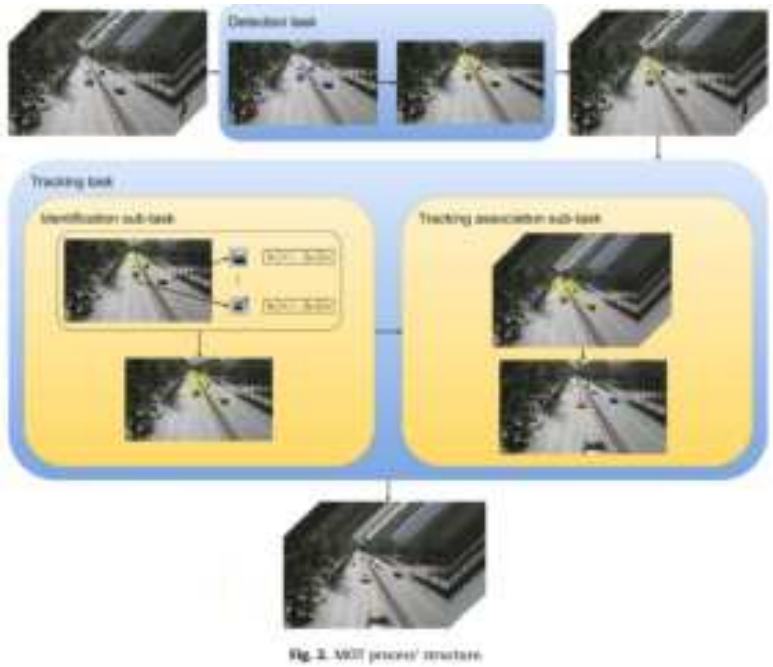


Figura 2.1: *Struttura base del processo di MOT (fonte: [17])*

magine viene suddivisa in piccole regioni chiamate celle, e per ciascuna cella, vengono calcolati gli orientamenti dei gradienti. I risultati di questi calcoli vengono successivamente organizzati in un istogramma, generando così una rappresentazione compatta ma informativa dell'immagine. Le tecniche tradizionali di object detection, come Viola-Jones e HOG, hanno contribuito significativamente allo sviluppo della visione artificiale, ma presentano alcuni limiti che le rendono meno adatte a gestire sfide complesse, quali la rigidità nelle caratteristiche manualmente progettate, la difficoltà nella gestione di diverse scale e orientamenti, limitata capacità di apprendimento e sfide nella cattura di oggetti complessi. Queste limitazioni hanno stimolato lo sviluppo delle Deep Neural Network (DNN), introducendo un approccio più flessibile e adattabile che supera molti dei vincoli delle metodologie tradizionali, permettendo un apprendimento automatico di features complesse e migliorando la performance nella rilevazione di oggetti in contesti visivi variabili. L'avvento delle Deep Neural Network (DNN) ha innescato una trasformazione significativa nel campo della visione artificiale, portando a risultati prima ritenuti difficilmente raggiungibili. Negli ultimi anni, la letteratura scientifica ha visto la proliferazione di approcci basati su DNN, sia nell'ambito urbano che in altri scenari.

2.2.1 Convolutional Neural Network

Attualmente, i metodi più promettenti si fondano sulle Convolutional Neural Network (CNN), dove reti profonde apprendono caratteristiche latenti e complesse durante la fase di addestramento, cercando di interpretare il contenuto delle immagini. Questi

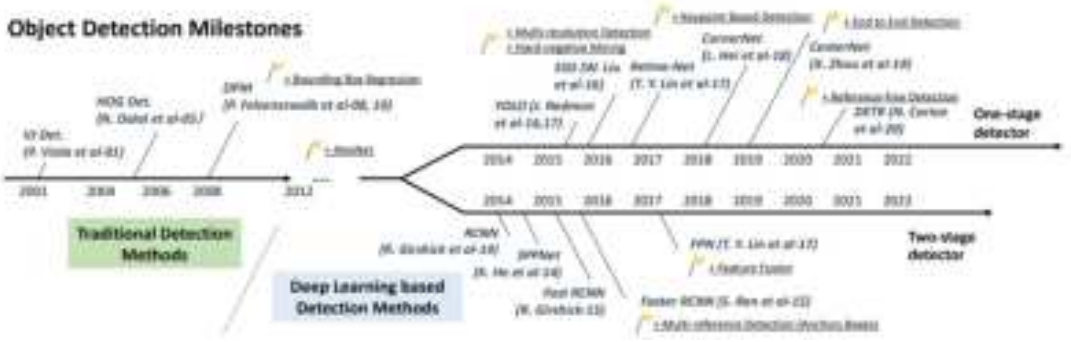


Figura 2.2: *Tappe nello sviluppo di metodi per l'Object Detection (fonte:[18])*

modelli possono essere distinti principalmente in due categorie [21]: algoritmi a due fasi e algoritmi a una fase. Le CNN a due fasi seguono un processo sequenziale, in cui la prima fase è dedicata al rilevamento di regioni di interesse (ROI) attraverso l'uso di algoritmi che generano proposte di regioni potenzialmente contenenti oggetti. Nella seconda fase, queste regioni proposte vengono sottoposte a classificazione e raffinamento per identificare con precisione la presenza degli oggetti. Un esempio paradigmatico di questo approccio è rappresentato da R-CNN (Region-based Convolutional Neural Network). Dall'altro lato, le CNN a una fase integrano il rilevamento e la classificazione in un'unica passaggio, operando direttamente sull'intera immagine senza necessità di una fase preliminare di generazione di proposte. Gli algoritmi a due fasi includono R-CNN (Region-based Convolutional Neural Network) [22], Fast R-CNN [23], Faster R-CNN [24] e Mask R-CNN [25], mentre gli algoritmi a una fase comprendono la serie di algoritmi YOLO (you only look once) [26–30] e gli algoritmi SSD (Single Shot MultiBox Detector) [31], tra gli altri.

Architettura di una CNN

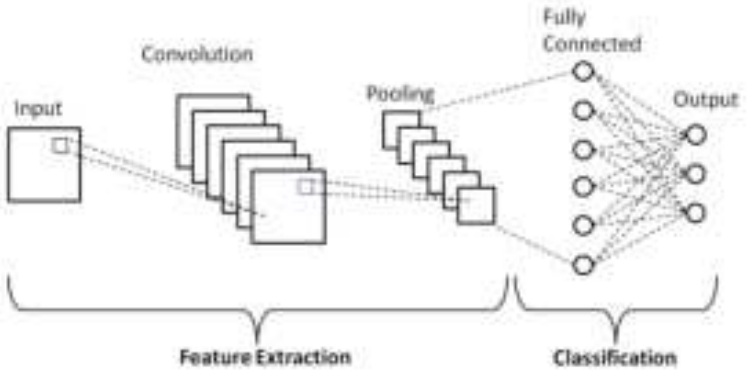


Figura 2.3: *Architettura di una CNN (fonte: Medium)*

Le Convolutional Neural Networks (CNN) rappresentano un tipo di rete neurale progettata specificamente per l'analisi di dati bidimensionali, come le immagini. L'architettura delle CNN è ispirata dalla biologia visiva, cercando di emulare il modo in cui il cervello umano elabora e comprende le informazioni visive.

L'architettura di base di una CNN include tre tipi principali di strati:

- Strati di Convoluzione: Questi strati applicano filtri (kernel) alle regioni sovrapposte dell'input per estrarre caratteristiche come contorni, texture e pattern più complessi. L'uso di convoluzioni consente alle CNN di mantenere la relazione spaziale tra i pixel, catturando così informazioni significative dalle immagini.
- Strati di Pooling: Questi strati riducono la dimensionalità spaziale dell'input, riducendo il numero di parametri e di calcoli nella rete. La tecnica comune di pooling è il max pooling, che seleziona il valore massimo da una finestra mobile. Ciò aiuta a mantenere le caratteristiche rilevanti e a rendere la rappresentazione più gestibile.
- Strati Fully Connected (FC): Questi strati connettono ogni neurone dello strato precedente a ciascun neurone, producendo alla fine l'output della rete. Gli strati fully connected sono tipicamente posizionati alla fine della CNN per eseguire la classificazione o la regressione.

Regional-based CNN Le Regional-based CNN (R-CNN) rappresentano l'applicazione classica delle CNN per il rilevamento di oggetti tramite un approccio a due fasi. L'idea dietro le R-CNN è semplice: si inizia con un primo step denominato *Proposal Generation*. In questa fase vengono generate regioni di interesse (proposals) che potrebbero contenere oggetti. Questo passo aiuta a ridurre il numero di regioni da esaminare, rendendo il processo più efficiente. Quindi, ogni *proposal* è trasformata in un formato standardizzato e viene utilizzata per alimentare una rete CNN pre-addestrata per l'estrazione delle *feature*. Infine, un rete Fully Connected viene usata per la classificazione e la raffinazione della posizione dell'oggetto, producendo l'etichette di classi ed i relativi bounding box. Nel corso degli anni, sulla base di questa architettura, si sono susseguiti diversi proposte volte a migliorarne l'efficienza. Uno dei limiti delle R-CNN, infatti, riguarda la necessità di alimentare una modulo CNN per ogni regione di interesse. Fast R-CNN risolve questo problema, alimentando l'architettura CNN con l'intera immagine e le regioni di interesse in un solo step di propagazione. A questo punto il collo di bottiglia riguarda la fase di generazione delle regioni, basti pensare che da una sola immagine è possibile generare migliaia di *proposal*. Faster R-CNN si focalizza su questo aspetto, posticipando la generazione delle regioni solo dopo aver trasformato l'immagine originale in una rappresentazione compatta tramite CNN, detta *feature map*. La feature map sono quindi utilizzate per generare dei box di interesse [22] [23] [24].

Nonostante la loro capacità di raggiungere un'elevata precisione, questi metodi sono di rado utilizzati nella pratica a causa della loro complessità che si traduce in tempi

di inferenza prolungati. Al contrario, l'approccio opposto, che esegue il rilevamento in un singolo passaggio di inferenza, offre prestazioni notevolmente superiori, adatte anche per applicazioni in tempo reale. Tuttavia, ciò si accompagna a un significativo calo di accuratezza quando si tratta di rilevare oggetti in scene dense o di dimensioni ridotte. SSD (Single Shot MultiBox Detector) e YOLO (You Only Look Once) sono due dei principali metodi che usano un approccio ad una fase [18].

SSD (Single Shot MultiBox Detector) La novità principale portata da SSD è l'introduzione di tecniche di *multi-reference* e *multi-resolution*, grazie alle quali si analizza l'immagina a diverse scale permettendo prestazioni migliori nell'individuazione di oggetti di diversa dimensione all'interno dell'immagine, contribuendo ad una maggiore flessibilità ed accuratezza.

YOLO (You Only Look Once) Il metodo YOLO, acronimo di "*You Only Look Once*", introdotto per la prima volta da Redmon ed altri nel 2016 [26], è caratterizzato dall'utilizzo di una singola rete CNN alimentata direttamente dall'immagine di input, la quale viene suddivisa in sezioni. Su ognuna di queste porzioni produce i bounding box e le probabilità di classe per ogni oggetto rilevato in modo del tutto simultaneo. Una delle novità introdotte da YOLO consiste nel fatto che affronta il compito di rilevamento come un problema di regressione, analizzando in una singola passata l'intera immagine. Grazie a quest'approccio, YOLO riesce ad essere estremamente veloce nella fase di inferenza e risulta essere la soluzione più adatta nell'implementazione di applicazioni in tempo reale.Tuttavia, a causa di questo approccio unificato, YOLO ha difficoltà a rilevare piccoli oggetti e oggetti che sono vicini tra loro. Questo perché la griglia su cui YOLO fa le sue previsioni limita il numero di oggetti che può rilevare in ogni cella.

L'architettura di YOLO, rappresentata in fig. 2.4, è ispirata al modello convoluzionale GoogLeNet [32] per la classificazione di immagini. La rete ha 24 strati convoluzionali seguiti da 4 strati di max-pooling e 2 strati completamente connessi. La rete si distingue per alcune caratteristiche chiave:

- **Preprocessing dell'immagine:** L'immagine di input è adeguatamente adattata e ridimensionata a 448x448 pixel prima di essere elaborata dalla rete convoluzionale.
- **Strati convoluzionali:** Contrariamente a GoogLeNet, YOLO utilizza strati di riduzione 1×1 seguiti da strati convoluzionali 3×3. La funzione di attivazione è ReLU, ad eccezione dello strato finale che utilizza una funzione di attivazione lineare.
- **Divisione dell'immagine in griglia:** L'immagine è suddivisa in una griglia $S \times S$, in cui ogni cella prevede B bounding box e i punteggi di confidenza relativi a ciascuna di esse. Questi punteggi indicano quanto il modello sia sicuro della presenza di un oggetto e quanto sia precisa la classificazione.

- **Bounding box:** Ogni bounding box è descritta da 5 componenti: le coordinate (x, y), che indicano il centro della casella, e le sue dimensioni w e h predette rispetto all'intera immagine. Infine ad ogni bounding box è associato un livello di confidenza, rappresentante l'Intersection over Union tra la bounding box predetta e quella di verità.
- **Anchor Box:** Il processo di predizione delle boundinx box è guidato dall'utilizzo di anchor box. Rettangoli dalla forma predefinita allo scopo di rilevare gli oggetti nell'immagine.
- **Probabilità delle classi:** Ogni cella della griglia è associata a delle probabilità condizionali delle classi, $Pr(Classi|Oggetto)$. Queste probabilità indicano la confidenza del modello riguardo la presenza di specifiche classi in relazione alla presenza di un oggetto nella cella.
- **Regolarizzazione:** Per prevenire l'overfitting, YOLO incorpora tecniche come la batch normalization e il dropout, fornendo stabilità e regolarizzazione al modello.
- **Non Maximum Suppression:** È una fase di post-elaborazione che ha lo scopo di eliminare predizioni duplicate. Dopo che il modello ha previsto molteplici bounding box per uno stesso oggetto, questa tecnica seleziona solo le predizioni più accurate, evitando sovrapposizioni e migliorando la precisione complessiva del modello.

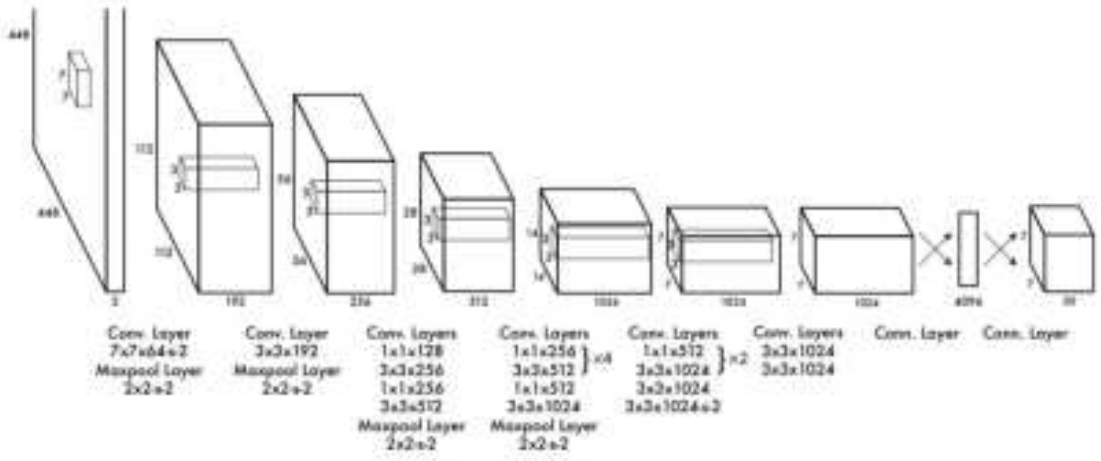


Figura 2.4: *Architettura di YOLO (fonte: [26])*

2.3 Algoritmi di tracciamento

In seguito al rilevamento degli oggetti, il secondo aspetto su cui bisogna porre l'attenzione per il task di Multi-Object Tracking riguarda l'identificazione e, se prevista, la

re-identificazione degli oggetti in movimento, attraverso la sequenza. Idealmente, vorremmo che una volta rilevato e classificato un oggetto nell'immagine gli venga associato un ID univoco e che quando lo stesso oggetto viene riproposto nei frame successivi venga riconosciuto e quindi gli venga associato il medesimo codice identificativo. Per risolvere questo problemi, i *tracker* utilizzano una varietà di metodi per modellare l'aspetto degli oggetti in movimento.

I principali metodi proposti in letteratura ed utilizzati nelle applicazioni reali utilizzano un filtro di Kalman (KF) come modulo per predire la posizione dell'oggetto di interesse nel frame successivo [33]. Tra questi, un esempio significativo è SORT (Simple Online Real-Time Tracking), [34]. SORT rappresenta gli oggetti in movimento attraverso un modello lineare a velocità costante, applica il filtro di Kalman per prevederne i parametri. Successivamente, risolve il problema di associazione tra le bounding box predette e gli oggetti target preesistenti mediante l'utilizzo dell'algoritmo di ottimizzazione noto come "Hungarian Algorithm." Nel corso degli anni, diverse ottimizzazioni sono state apportate a questo metodo. Ad esempio, Deep SORT [35] introduce una rete neurale convoluzionale (CNN) preaddestrata per riconoscere specifici oggetti, mirando a re-identificare gli oggetti target e migliorare le prestazioni, specialmente in presenza di occlusioni o quando alcuni oggetti sono assenti in determinati frame. ByteTrack, invece, propone un approccio diverso. Mentre SORT utilizza solo le detection box aventi score elevato, ByteTrack separa i rilevamenti in quelli ad alto e basso score ed esegue due fasi di associazione utilizzando la IoU come misura di similarità. L'utilizzo delle box con score bassi ha lo scopo di ridurre eventuali rilevamenti mancanti [36].

2.4 Dataset

L'importanza dei dataset di riferimento nel campo della computer vision è cruciale per la valutazione delle prestazioni degli algoritmi MOT, poiché consentono di testare e valutare modelli e algoritmi proposti su una serie di sequenze video opportunamente annotate. Tuttavia, non tutti i dataset sono adeguatamente preparati per affrontare specificamente le sfide del tracciamento di oggetti in ambienti stradali.

Tra i dataset più utilizzati, MOT16 [37] e BU-TIV [38] sono comunemente adottati, anche se presentano solo una preparazione parziale per il tracciamento di veicoli in contesti urbani. Ad esempio, il dataset MOT16 è progettato principalmente per il tracciamento di persone in contesti di sorveglianza, mentre BUT-TV si concentra principalmente sul tracciamento di pedoni e oggetti in video termici a infrarossi. Inoltre, BU-TIV non fornisce un'ampia varietà di scenari di tracciamento urbano, limitando così la sua applicabilità a un insieme più ampio di contesti urbani e scenari di tracciamento. Questo sottolinea l'importanza di sviluppare e utilizzare dataset di riferimento che siano specificamente progettati e ottimizzati per le sfide uniche presentate dal tracciamento di veicoli in ambienti urbani. D'altra parte, la comunità di ricerca ha risposto alle esigenze specifiche del tracciamento veicolare, rilasciando dataset mirati come Waymo Open Dataset

Tabella 2.1: *Statistiche sul dataset UA-Detrac (fonte: [40])*

Tipo	N. Frame	Tracce	N. Bounding box
Training set	84k	5.9k	578k
Test set	56k	2.3k	632k

[39] e UA-Detrac [40]. Questi dataset, progettati specificamente per l'identificazione dei veicoli, offrono sequenze video che rappresentano scenari realistici di traffico stradale.

UA-Detrac Il dataset UA-Detrac rappresenta un prezioso supporto per la sperimentazione nel campo della rilevazione multi-oggetto e del tracciamento multi-oggetto nel contesto del traffico reale. Esso comprende 100 sequenze video catturate in diverse location, con oltre 140.000 frame e dettagliate annotazioni, quali occlusione, condizioni meteorologiche e informazioni temporali. Il dataset contiene 8250 veicoli annotati manualmente, generando un totale di 1,21 milioni di bounding boxes etichettate degli oggetti. Le sequenze video sono registrate a 25 frame al secondo con una risoluzione di 960×540 pixel. Il dataset è suddiviso in un set di addestramento (DETRAC-train) e un set di test (DETRAC-test). Il set di addestramento contiene 83.791 frame con 577.899 bounding boxes, mentre il set di test contiene 56.340 frame con 632.270 bounding boxes, come mostrato in tabella 2.4. Il dataset UA-Detrac fornisce strumenti di valutazione e un nuovo protocollo di valutazione per la rilevazione di oggetti e il sistema di tracciamento multi-oggetto, considerando la rilevazione di oggetti e il tracciamento di oggetti in tandem.

Waymo Open Dataset Il dataset Waymo Open più complesso e dettagliato rispetto al precedente. È composto da dati proveniente da più sensori come telecamere 3d e LIDAR, raccolti da veicoli autonomi operati dal sistema di guida autonoma Waymo Driver, in una vasta gamma di condizioni. Il dataset è scomposto in due parti: il dataset Perception con dati di sensori ad alta risoluzione ed etichette, e il dataset Motion con traiettorie di oggetti e mappe 3D corrispondenti.

2.5 Metriche di valutazione

La valutazione accurata delle performance degli algoritmi di Multi-Object Tracking (MOT) rappresenta un elemento chiave per comprenderne l'efficacia nell'ambito del tracciamento di oggetti in contesti urbani. Le metriche di valutazione svolgono un ruolo cruciale, offrendo una visione dettagliata delle capacità di un algoritmo MOT e facilitando valutazioni obiettive e confronti significativi tra diverse soluzioni proposte.

Molteplici studi adottano metriche non specificamente adatte al task di MOT, come Precision, Recall e mAP (mean Average Precision). Queste misure sono utili per valutare il task sottostante di object detection all'interno di un sistema MOT, ma non

forniscono una valutazione completa e rappresentativa del sistema di tracciamento. Nel corso degli anni, sono state introdotte diverse categorie di metriche per valutare le performance dei sistemi MOT nella loro completezza, suddivise in gruppi come VACE (Visual Assessment of the Cleanness of Events), CLEAR (Classification of Events, Activities, and Relationships), e altre.

Precision e Recall Precision e Recall sono due metriche fondamentali nell'ambito del riconoscimento di oggetti. La precisione è la percentuale di identificazioni positive che risultano effettivamente corrette, mentre la misura di recall è la percentuale di positivi che sono stati identificati correttamente.

Intersection Over Union IoU, o Intersection Over Union, è una misura utilizzata nel rilevamento di oggetti per quantificare l'overlap tra la bounding box prevista e quella reale, fornendo una valutazione quantitativa della corrispondenza tra le previsioni e la realtà (vedi fig. 2.5). Un IoU di 1 indica una corrispondenza perfetta tra le bounding box, mentre un IoU di 0 indica nessun overlap. All'utilizzo di IoU, si affianca una soglia (threshold), la quale indica il livello minimo di sovrapposizione accettato per considerare la previsione come corretta.

mAP (mean Average Precision) A completare la valutazione dei sistemi di object detection, la metrica mAP (Mean Average Precision) assume un ruolo cruciale nel determinare la precisione media di un sistema di rilevamento oggetti, considerando tutte le classi di oggetti coinvolte. Essa si basa sulla media delle precisioni a vari livelli di Recall, offrendo una valutazione complessiva dell'efficacia del sistema. Tipicamente, l'approccio adottato prevede il calcolo di una curva precision-recall per ciascuna classe di oggetti, variando la soglia di IoU. Successivamente, si calcola la precisione media (average precision) considerando una serie di punti lungo la curva precision-recall e, infine, si determina la media finale di queste valutazioni. Questo metodo offre una misura accurata e completa delle capacità del sistema, permettendo una valutazione approfondita delle prestazioni in diverse situazioni e su diverse categorie di oggetti [41].

Metriche VACE

Le metriche VACE, proposte da Wu e Nevatia [42], si concentrano sulla misurazione di diversi tipi di errori nei sistemi MOT. Alcune riportate di seguito:

- **FP (Falsi Positivi):** Indica il numero totale di falsi positivi nell'intera sequenza. I falsi positivi sono determinati dal numero di bounding box predette che non è possibile associare a nessun oggetto reale.
- **FN (False Negatives):** Rappresenta il numero totale di falsi negativi nell'intera sequenza. I falsi negativi corrispondono alle ground truth che non possono essere collegate a nessuna ipotesi.

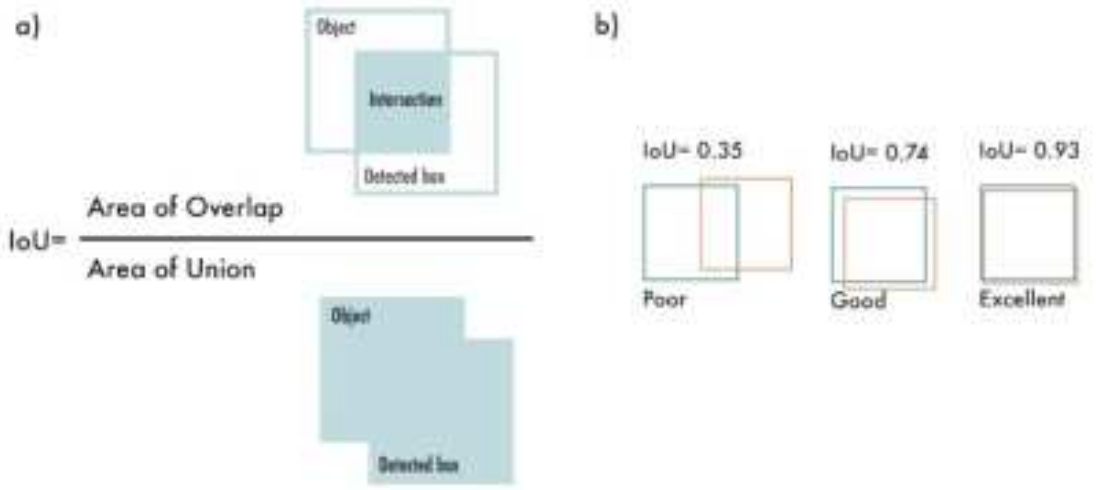


Figura 2.5: *Intersection over Union (fonte: [41])*

- **MT trajectories (Mostly Tracked trajectories):** Misura il numero di traiettorie del ground-truth che sono correttamente tracciate in almeno l'80% dei frame del video.
- **Fragments:** Indica il numero totale di traiettorie predette che coprono al massimo l'80% dei frame della sequenza. Una stessa traiettoria può essere coperta da più di un frammento.
- **ML trajectories (Mostly Lost trajectories):** Rappresenta il numero di traiettorie del ground-truth che sono tracciate correttamente in meno del 20% dei frame del video.
- **False trajectories:** Si riferisce alle traiettorie previste che non coprono la traiettoria di un oggetto reale.
- **ID switches (Identity switches):** Indica il numero di volte in cui un oggetto viene correttamente rilevato, ma l'ID viene riassegnato in modo errato lungo i diversi frame della sequenza.

Metriche CLEAR

Le metriche CLEAR (Classification of Events, Activities and Relationships) [43], utilizzano diverse misure VACE combinandole tra loro ed introducendo il concetto di IoU (Intersection over Union) per valutare la forza dell'associazione tra ground truth[1] e predizione. Le principali metriche CLEAR sono:

[1]Il termine "ground truth" si riferisce ai dati reali e di riferimento utilizzati per valutare le prestazioni degli algoritmi di tracciamento di oggetti.

- **MOTA (Multiple Object Tracking Accuracy):** Misura l'accuratezza complessiva di un algoritmo MOT. Può essere calcolata utilizzando la seguente formula:
$$1 - \frac{\text{FN} + \text{FP} + \text{IDSW}}{\text{GT}}, \tag{2.1}$$
dove GT rappresenta il numero di ground truth boxes.

- **MOTP (Multiple Object Tracking Precision):** Misura la precisione complessiva di un algoritmo MOT. Viene calcolata con l'equazione
$$\frac{\sum_{t,i} d_{t,i}}{\sum_t c_t}, \tag{2.2}$$
dove $d_{t,i}$ misura la distanza tra gli oggetti rilevati e la ground truth corrispondente, mentre c_t è il numero totale di associazioni tra predizioni e ground-truth. Questa metrica si concentra sull'accuratezza nella localizzazione delle detection box e tratta solo l'output dell'object detector, senza considerare l'output del tracker.

Queste metriche offrono una valutazione dettagliata delle prestazioni dei sistemi MOT, consentendo una comprensione approfondita delle loro capacità in scenari di tracciamento di oggetti in ambienti urbani. L'utilizzo accurato di tali metriche è essenziale per garantire una valutazione obiettiva e informativa degli algoritmi MOT proposti.

2.6 Hardware utilizzato per il MOT

L'implementazione di algoritmi di tracciamento di oggetti richiede una potente infrastruttura hardware, soprattutto quando si lavora con grandi volumi di dati e modelli di apprendimento automatico complessi. Sistemi di questo tipo richiedono di:

- raccogliere dati dall'ambiente circostante, utilizzando una moltitudine di sensori elementari;
- processare le informazioni acquisite garantendo bassi tempi di latenza e costi di comunicazione sostenibili;
- elaborare ed archiviare i dati;
- applicare tecniche di IA per l'identificazione ed il rilevamento degli oggetti fisici.

Per soddisfare queste esigenze, i moderni sistemi di monitoraggio, che spesso integrano algoritmi di object detection basati su reti neurali profonde, richiedono l'uso di GPU ad alte prestazioni. Le GPU sono ampiamente utilizzate nell'addestramento e nell'inferenza di reti neurali, poiché sono in grado di gestire parallelamente operazioni complesse tipiche delle reti neurali profonde. Tuttavia, sebbene garantiscano prestazioni eccezionali, tecnologie di questo tipo richiedono un grosso dispendio sia energetico che economico. La principale limitazione è legata al fatto che l'esecuzione degli algoritmi di deep learning

richiede una capacità di calcolo massiccia, per questo motivo vengono principalmente eseguiti nei data center basati su cloud. Nel contesto del monitoraggio del traffico, però, una soluzione *cloud-centric* non sembra essere ottimale ed affronta diverse sfide per soddisfare i requisiti delle smart city, quali [44]:

- Tempi di latenza: La trasmissione dei dati raccolti dai dispositivi terminali al cloud richiede inevitabilmente tempo, mentre le applicazioni critiche richiedono rigorosamente una bassa latenza per reagire istantaneamente a comportamenti anomali.
- Sicurezza e privacy: La trasmissione dei dati sulla rete e lo stoccaggio di questi dati su server remoti possono generare numerosi problemi legati alla sicurezza e alla privacy, specialmente per dati sensibili sui comportamenti degli utenti.

Una soluzione praticabile è rappresentata dalla distribuzione di sistemi intelligenti ai bordi della rete (*edge*) secondo il paradigma *edge computing*, complementare a quello cloud. Spostare le attività di calcolo più vicino all'ambiente in analisi può ridurre considerevolmente la latenza e i rischi per la sicurezza. Un approccio del genere si avvale di dispositivi edge per l'elaborazione dei dati e l'inferenza. Questi dispositivi, come le board Google Coral, Jetson Nano e Raspberry Pi, presentano dimensioni contenute e un consumo energetico relativamente basso. Pur avendo prestazioni inferiori rispetto alle *server farm*, negli ultimi anni hanno raggiunto risultati considerevoli, integrando acceleratori hardware specializzati come le Unità di Elaborazione Tensoriale (TPU).

Le TPU rivestono un ruolo significativo nel contesto degli edge device. La loro architettura ottimizzata consente di affrontare specificamente carichi di lavoro associati alle reti neurali, offrendo prestazioni superiori a soluzioni più generaliste. Nell'ambito dell'elaborazione edge, in cui la riduzione della latenza e il risparmio energetico sono prioritari, le TPU emergono come componenti chiave. Grazie alle loro dimensioni compatte e al basso consumo energetico, le TPU sono ideali per essere integrate in dispositivi edge come telecamere intelligenti, board, sensori IoT e altri sistemi embedded. Le TPU risultano particolarmente vantaggiose per applicazioni di visione artificiale, gestendo con efficienza complessi algoritmi di riconoscimento di oggetti, segmentazione di immagini e altre attività visive, ottimizzando le risorse di calcolo per le specifiche esigenze delle reti neurali coinvolte e migliorando l'efficienza complessiva del sistema di edge computing [45].

2.7 Sfide aperte

Nonostante i significativi progressi raggiunti nel campo del tracciamento di oggetti, alcune sfide rimangono aperte e richiedono ulteriori ricerche e sviluppi, in particolar modo nel complesso contesto urbano.

- La gestione delle occlusioni emerge come uno degli ostacoli più significativi. Immersi nel caos delle strade affollate, con veicoli in movimento, pedoni attraversanti e strutture che ostacolano la visuale, la necessità di sviluppare algoritmi in grado di prevedere il movimento degli oggetti al di là delle ostacolazioni diventa evidente. Superare questa sfida richiede non solo avanzati approcci algoritmici, ma anche una profonda comprensione della dinamica urbana.

- La diversità degli scenari urbani costituisce un ulteriore elemento di complessità. Le strade, variabili nelle dimensioni e soggette a variazioni improvvise di illuminazione, richiedono algoritmi flessibili capaci di adattarsi dinamicamente a differenti condizioni. Incroci, semafori e varie configurazioni stradali contribuiscono a rendere ogni scenario unico, richiedendo soluzioni di tracciamento altamente adattabili.

- L'efficienza computazionale si pone come una sfida ineludibile. La richiesta di tracciamento in tempo reale impone una pressione significativa sulla potenza di calcolo. Riuscire a sviluppare algoritmi efficienti diventa quindi cruciale per consentire l'integrazione su svariate piattaforme, dai veicoli autonomi ai dispositivi embedded, garantendo al contempo prestazioni ottimali.

- La sicurezza, soprattutto nel contesto della guida autonoma, costituisce una priorità assoluta. Gli algoritmi di tracciamento devono essere in grado di rilevare e rispondere prontamente a situazioni di emergenza, ponendo l'accento sulla sicurezza stradale come principio fondamentale.

- La mancanza di dataset affidabili rappresenta un altro nodo cruciale. Senza dati di alta qualità, gli algoritmi rischiano di operare nell'oscurità, compromettendo l'efficacia del tracciamento. Affrontare questa sfida implica la creazione di dataset dettagliati, capaci di catturare la complessità di una gamma completa di scenari urbani.

- Infine, il versante etico e della privacy emerge come una considerazione imprescindibile. Il tracciamento, sebbene promettente in termini di avanzamenti tecnologici, solleva questioni sensibili riguardo alla privacy delle persone coinvolte. È imperativo sviluppare soluzioni etiche e rispettose della privacy, bilanciando l'utilità dei dati di tracciamento con la necessità di tutelare i diritti individuali.

Capitolo 3

Sistema proposto per il riconoscimento e tracciamento degli oggetti nell'ambiente urbano

Il seguente capitolo delinea il processo di sviluppo del sistema MOT, evidenziando la metodologia applicata, le tecnologie coinvolte e i dati utilizzati per garantire un'elevata precisione nelle operazioni di riconoscimento e tracciamento degli oggetti. Successivamente, a partire dal sistema MOT creato, si suggerisce la creazione di un dataset di riferimento. Questo dataset è ottenuto mediante l'applicazione di tecniche di online-tracking su sequenze video di traffico reale. Tale approccio non solo consente di valutare in modo più robusto le prestazioni del sistema su situazioni reali e dinamiche, ma contribuisce anche a arricchire il campo dei dataset disponibili, offrendo una base solida per futuri sviluppi e confronti nel campo del riconoscimento e tracciamento degli oggetti in ambienti urbani. Il capitolo prosegue presentando un framework di valutazione del Multi-Object Tracking (MOT) che si basa sul dataset di benchmark UA-Detrac. La descrizione dettagliata del framework include metodi e metriche chiave utilizzate per valutare l'efficacia del tracciamento degli oggetti.

3.1 Metodologia

Nella prima fase sperimentale del lavoro di tesi l'attenzione è stata incentrata sulla creazione di un sistema di MOT per l'estrapolazione e la raccolta di dati affidabili riguardo il traffico urbano, in uno specifico ambiente selezionato. Attraverso l'elaborazione e l'analisi di sequenze video rappresentanti scene di traffico urbano, utilizzando tecniche all'avanguardia di object detection e di tracciamento, l'obiettivo è fornire un dataset affidabile per il monitoraggio del traffico urbano. In particolare, il dataset creato è stato adeguatamente adattato ed utilizzato con successo come base di partenza per sviluppare avanzati modelli di Machine Learning per il task di predizione delle traiettorie dei veicoli

nel contesto di riferimento, ricerca svolta da parte del collega Fabrizio Mangione nel suo lavoro di tesi. Nella seconda fase, il focus si è spostato sulla valutazione del sistema utilizzato e delle sue performance sul dataset di benchmark UA-Detrac [40] attraverso lo sviluppo di un framework di valutazione.

3.2 Tecnologie utilizzate

3.2.1 Python

Python, linguaggio di programmazione di alto livello e ampiamente adottato, ha guadagnato un ruolo di rilievo nell'ambito del Machine Learning (ML) grazie alla sua sintassi chiara, flessibilità e vasta comunità di sviluppatori. La sua versatilità si manifesta attraverso la facilità di apprendimento e l'adattabilità a diversi paradigmi di programmazione, come *scripting* e programmazione orientata agli oggetti. Nel contesto del ML, Python è diventato linguaggio principale soprattutto per l'ampio ecosistema di librerie specializzate, tra cui spiccano TensorFlow e PyTorch per il deep learning e scikit-learn per algoritmi tradizionali. La comunità attiva contribuisce a una crescita continua, offrendo supporto tempestivo e aggiornamenti frequenti. La facilità di integrazione con altre tecnologie, insieme a librerie di visualizzazione come Matplotlib e Seaborn, rende Python ideale per implementare modelli ML in applicazioni reali e per la visualizzazione e l'analisi dei dati. La sua scalabilità e adozione industriale consolidano la sua posizione come strumento fondamentale per lo sviluppo di applicazioni ML su larga scala, sottolineando il ruolo cruciale che Python svolge nell'innovazione nell'Intelligenza Artificiale.

3.2.2 Pytorch

PyTorch [46], una delle librerie più influenti nel campo del Machine Learning, si è affermata come un framework di deep learning flessibile e potente. PyTorch è ampiamente utilizzato per lo sviluppo e l'addestramento di reti neurali, comprese le reti neurali profonde (DNN) utilizzate in applicazioni di visione artificiale, elaborazione del linguaggio naturale e altro. La novità dell'architettura di PyTorch è che utilizza un approccio dinamico per la definizione e l'esecuzione dei modelli di apprendimento profondo, invece di costruire un grafo statico di calcolo. Questo design offre i seguenti vantaggi:

- Facilità d'uso: PyTorch si integra perfettamente con l'ecosistema Python e consente di scrivere modelli, ottimizzatori e caricatori di dati come normali programmi Python, sfruttando tutte le funzionalità del linguaggio, come il debug, la stampa e la visualizzazione.
- Flessibilità: PyTorch permette di implementare qualsiasi nuova architettura di rete neurale con PyTorch, senza dover adattarsi a interfacce rigide o limitazioni di espressività. PyTorch supporta anche la differenziazione automatica di funzioni personalizzate e la gestione della memoria di tensori in maniera dinamica.

- Interoperabilità: PyTorch facilita lo scambio di dati con altre librerie Python, come NumPy, SciPy e Pandas. PyTorch fornisce anche un'API C++ per l'esecuzione di modelli al di fuori dell'interprete Python.

Un altro punto di forza di PyTorch è l'ecosistema crescente di estensioni e librerie, come TorchVision per visione artificiale. L'adozione di PyTorch da parte della comunità accademica e industriale ha portato a un continuo sviluppo e miglioramento, contribuendo alla sua reputazione di strumento affidabile per le sfide avanzate del deep learning.

3.2.3 YOLOv8

YOLO, nella sua versione 8, rappresenta lo stato dell'arte per quanto riguarda la serie di modelli di object detection *You Only Look Once*. Permette di eseguire l'identificazione e la segmentazione degli oggetti nell'immagine garantendo prestazioni elevate, sia in termini di mAP (*mean Average Precision*) che per quanto riguarda i tempi di inferenza, entrambi riassunti in tabella 3.2. Inoltre, grazie alla possibilità di accedere a modelli di grandi dimensioni preallenati sul dataset COCO [47] permette di disporre un punto di partenza valido per il task di rilevamento anche nel contesto urbano, senza la necessità di allenare il modello in locale con i conseguenti limiti dettati dall'hardware a disposizione. In aggiunta, i modelli *pre-trained* sono disponibili in diverse dimensioni (*nano*, *small*, *medium*, *large*, *extra-large*). Questo garantisce grande flessibilità, soprattutto in ottica di una distribuzione del sistema su dispositivi con hardware limitato, come dispositivi edge.

Architettura di YOLOv8

Alcune delle novità e delle caratteristiche di YOLOv8 sono [41]:

- Nuova backbone network:YOLOv8 utilizza una backbone network chiamata NAS (Neural architecture search), una tecnica di ricerca per trovare l'architettura di rete neurale ottimale per il rilevamento di oggetti. Questo processo di ottimizzazione massimizza le prestazioni rispetto a un compromesso tra accuratezza e velocità. Grazie all'uso della NAS, YOLOv8 minimizza i tempi di inferenza, rendendolo uno dei modelli di rilevamento di oggetti più veloci disponibili ed adatto a contesti di rilevamento in tempo reale, anche su dispositivi edge.

- Anchor-free: YOLOv8 adotta un approccio anchor-free, che elimina la necessità di usare delle ancore predefinite per predire le bounding box degli oggetti. Le ancore, applicate nei metodi precedenti, consistono in box con una forma predefinita, utilizzate per individuare gli oggetti. Un approccio anchor-free rende il modello estremamente flessibile, più semplice e robusto, e migliora la precisione soprattutto per gli oggetti di piccole dimensioni.

person	fire hydrant	elephant	skis	wine glass	broccoli	dining table	toaster
bicycle	stop sign	bear	snowboard	cup	carrot	toilet	sink
car	parking meter	zebra	sports ball	fork	hot dog	tv	refrigerator
motorcycle	bench	giraffe	kite	knife	pizza	laptop	book
airplane	bird	backpack	baseball bat	spoon	donut	mouse	clock
bus	cat	umbrella	baseball glove	bowl	cake	remote	vase
train	dog	handbag	skateboard	banana	chair	keyboard	scissors
truck	horse	tie	surfboard	apple	couch	cell phone	teddy bear
boat	sheep	suitcase	tennis racket	sandwich	potted plant	microwave	hair drier
traffic light	cow	frisbee	bottle	orange	bed	oven	toothbrush

Figura 3.1: *Classi del dataset COCO (fonte: SuperAnnotate)*

Dataset COCO

Il dataset COCO (Common Objects in Context) è una risorsa chiave utilizzata per addestrare e valutare modelli di object detection come YOLOv8. Contiene immagini complesse e diverse, con oggetti in contesti realistici. Il dataset include oltre 330.000 immagini, di cui più di 200 mila etichettate con 1,5 milioni di bounding box, appartenenti a 80 categorie (vedi fig. 3.1). La diversità delle immagini e delle categorie nel dataset COCO contribuisce a migliorare la capacità dei modelli di generalizzare e rilevare oggetti in scenari realistici. Oltre ad annotazioni per il rilevamento di oggetti, il COCO dataset fornisce anche maschere di segmentazione pixel-per-pixel per gli oggetti presenti nelle immagini. Questo permette di sviluppare modelli che siano in grado di non solo di identificare gli oggetti nelle immagini ma anche a delineare in modo preciso i contorni degli oggetti attraverso la segmentazione. YOLOv8 sfrutta il dataset COCO durante la fase di addestramento, permettendo al modello di apprendere da un'ampia gamma di situazioni e oggetti.

Modelli pre-addestrati YOLOv8

Model	size (pixels)	mAP^{val} 50-95	Speed CPU (ms)	Speed T4 GPU (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	-	-	3.2	8.7
YOLOv8s	640	44.9	-	-	11.2	28.6
YOLOv8m	640	50.2	-	-	25.9	78.9
YOLOv8l	640	52.9	-	-	43.7	165.2
YOLOv8x	640	53.9	-	-	68.2	257.8

Figura 3.2: *Performance di YOLOv8 su COCO dataset (fonte: [48])*

Ultralytics [48], YOLOv8 offre una varietà di modelli YOLOv8 pre-addestrati su dataset COCO, per soddisfare diverse esigenze di prestazioni e applicazioni. Di seguito un elenco dei modelli pre-allenati disponibili, mentre in fig. 3.2 si mostrano le loro prestazioni dichiarate.

- YOLOv8 n (Nano)
- YOLOv8 s (Small
- YOLOv8 m (Medium)
- YOLOv8 l (Large)
- YOLOv8 x (Extra-large)

Ogni modello è progettato per bilanciare l'accuratezza e la velocità di inferenza, permettendo agli utenti di scegliere il modello più adatto alle loro esigenze specifiche. Questi differiscono principalmente per la loro dimensione e quindi per il numero di parametri utilizzati. Tutti i modelli sono progettati per lavorare con immagini di dimensioni 640x640 pixel. La metrica di valutazione utilizzata per questi modelli è la mAP (Mean Average Precision), un indice comune per la valutazione dei modelli di rilevamento di oggetti. L'intervallo "val 50-95" si riferisce al calcolo della mAP su diverse soglie di Intersection over Union (IoU) , da 0.5 a 0.95 con passo di 0.05. Questo fornisce una misura più robusta delle prestazioni del modello, poiché tiene conto della sua capacità di rilevare oggetti con diverse soglie di precisione. [48]

3.2.4 StrongSORT

Come precedentemente analizzato, il MOT non riguarda solo la fase di rilevamento degli oggetti, ma è necessario individuare un algoritmo accurato ed affidabile per il tracciamento degli oggetti attraverso l'assegnamento consistente di un ID univoco a ciascun oggetto tra i vari frame della sequenza, al fine di comprendere la dinamica del traffico e analizzarne il comportamento nel tempo. Tra i diversi algoritmi di tracciamento si è selezionato StrongSort [49], un approccio robusto ed efficiente che migliora le prestazioni del precedente DeepSort. StrongSORT si basa sul paradigma tracking-by-detection, ovvero rileva gli oggetti in ogni frame e poi li associa in base a delle caratteristiche di apparenza e/o movimento. StrongSORT migliora il precedente DeepSORT da più punti di vista, attraverso l'utilizzo di una serie di tecniche avanzate per migliorare l'accuratezza e l'efficienza del tracciamento multi-oggetto. In particolare:

- Filtro di Kalman adattivo (NSA): questo metodo adatta il calcolo del vanilla Kalman Filter pesando la confidenza del rilevamento, in modo da dare più peso ai rilevamenti più accurati e meno a quelli più incerti e rumorosi.
- Meccanismo di aggiornamento delle caratteristiche basato sulla media mobile esponenziale (EMA): questo metodo aggiorna lo stato rappresentante le caratteristiche

di ogni tracciato con una media ponderata tra le caratteristiche del tracciato fino a quel momento ed il rilevamento corrente. Questo permette di sfruttare le informazioni sui cambiamenti inter-frame e di ridurre il rumore dato del singolo rilevamento.

- Modello di compensazione dei movimenti di camera (ECC): questo metodo stima la rotazione e la traslazione globale tra frame adiacenti. Questo permette di compensare il rumore di movimento causato dal movimento della camera e di estrarre informazioni di movimento più precise.

StrongSORT, combinando questi componenti, fornisce un baseline forte e equo per il task di MOT, e raggiunge risultati all'avanguardia su diversi benchmark pubblici, come MOT17, MOT20, DanceTrack e KITTI.

3.3 Sviluppo del framework per il MOT

Lo sviluppo del framework per il Multi-Object Tracking (MOT) ha ricoperto un ruolo cruciale nel lavoro di ricerca, in quanto ha richiesto la progettazione e l'implementazione di un sistema robusto che integrasse modelli di object detection e algoritmi di tracking avanzati, per il tracciamento di oggetti nel contesto urbano e che permettesse l'estrapolazione e l'archiviazione dei dati inferiti per la creazione di un dataset.

3.3.1 Architettura del Framework

Il punto di partenza per la realizzazione del framework è stata la libreria Python AS-One [50], adeguatamente estesa e personalizzata per soddisfare le specifiche esigenze della ricerca. Questa libreria semplifica il processo di integrazione di algoritmi di object detection all'avanguardia con gli algoritmi di tracking più performanti, offrendo una gamma diversificata di metodi per il rilevamento e il tracciamento degli oggetti. AS-One include tracker di rilievo come ByteTrack, DeepSORT e StrongSORT, oltre alle più recenti versioni della serie di modelli YOLO, in diversi formati e dimensioni.

Detector I detector messi a disposizione sono YOLO-NAS, YOLOv8, V7, V6, V5, R e X. Tutti utilizzabili nei tre formati ONNX, PyTorch e CoreML. Sulla base dei pesi (*weights*) applicati in fase di creazione dell'oggetto detector è possibile caricare il modello prescelto, questo può essere un modello YOLO addestrato da zero su uno specifico dataset selezionato, oppure uno dei modelli disponibili preaddestrati su COCO, nelle diverse dimensioni: nano, small, medium, large ed extra-large.

- ONNX (Open Neural Network Exchange): È un formato open-source utilizzato per rappresentare modelli di apprendimento automatico. È progettato per consentire l'interoperabilità tra diverse piattaforme.

- PyTorch: È una libreria di apprendimento automatico open source basata su Torch, che fornisce un'ampia varietà di algoritmi di apprendimento. PyTorch è noto per

la sua facilità d'uso e la sua flessibilità, permettendo un controllo dettagliato del processo di apprendimento.

- CoreML: È un framework di Apple progettato per integrare facilmente i modelli di apprendimento automatico su sistemi iOS e macOS.

L'algoritmo di MOT, presente nella classe Python 'asone.py' della libreria AS-One, integra il funzionamento di tracker e detector. Data una sequenza di frame, come un file video in archivio, un flusso continuo proveniente da webcam o un url o un insieme di immagini, il metodo itera alimentando il detector prescelto frame per frame. Il tracker viene aggiornato con le nuove predizioni, comprendenti i bounding box e i relativi score di classe e valori di confidenza, per l'identificazione degli oggetti.

3.3.2 Estrapolazione e archiviazione di dati di secondo livello

Tale sistema è stato integrato sviluppando dei metodi per l'estrapolazione di dati di più alto livello, a partire dagli output di detector e tracker, con l'obiettivo di supportare la raccolta di dati completa ed affidabile. Lo pseudocodice 1 descrive le operazioni principali eseguite dall'algoritmo di MOT implementato.

Il sistema proposto permette, contemporaneamente all'esecuzione dell'analisi di sequenze di immagini, consente di arricchire i dati inferiti, combinandoli ed integrandoli con informazioni di contesto. In particolare, durante l'inferenza, il framework archivia i seguenti dati:

- Dataset delle traiettorie: Ogni record nel dataset è caratterizzato da un identificativo di frame e un identificativo univoco dell'oggetto. Sono stati registrati, inoltre, i valori di posizione del bounding box all'interno dell'immagine, il valore di confidenza restituito da YOLO e la classe rilevata.

- Crop dei frame: I ritagli delle immagini rappresentanti gli oggetti identificati, per ogni frame. Questi ritagli corrispondono all'area dell'immagine definita dal bounding box individuato per ciascun oggetto. I crop delle immagini possono essere utilizzati per eseguire un'analisi più attenta e dettagliata sugli oggetti rilevati, come ad esempio il rilevamento di caratteristiche specifiche oppure, nel caso di veicoli, il riconoscimento del modello degli stessi.

- Video di output: Video per la visualizzazione degli oggetti identificati. La sequenza di input viene arricchita ed esportata in formato video aggiungendo le bounding box rilevate, gli ID univoci, la categoria ed i valori di confidenza associati ad ogni oggetto rilevato. Questo flusso video può essere utile per eseguire un'analisi qualitativa sull'accuratezza del rilevamento e del tracciamento degli oggetti, nonché per scopi di presentazione e comunicazione dei risultati.

Di seguito una descrizione dettagliata di una entry del dataset delle traiettorie:

- **frame_id**: Codice identificativo del frame nella sequenza.
- **id**: Identificativo dell'oggetto assegnato dal tracker nel processo di identificazione.
- **bbox_left**: La distanza della bounding box rilevata dal bordo sinistro dell'immagine, in pixel.
- **bbox_top**: La distanza della bounding box rilevata dal bordo in alto dell'immagine, in pixel.
- **bbox_w**: La larghezza del bounding box rilevato, in pixel.
- **bbox_h**: La larghezza del bounding box rilevato, in pixel.
- **conf**: Score di confidenza del modello di detection nel classificare l'oggetto rilevato nella bounding box ad una specifica classe.
- **class_id**: La classe dell'oggetto rilevato dal modello di detection.

Le coordinate sono ottenute rispetto agli assi cartesiani disposti come evidenziato nella figura 3.3. In questa rappresentazione, l'asse delle ordinate (y) si estende verticalmente verso il basso dalla parte superiore dell'immagine, mentre l'asse delle ascisse (x) si estende orizzontalmente dalla sinistra verso destra dell'immagine stessa.

In figura 3.4, uno snapshot esemplificativo del video di output prodotto.

(a) *(b)*

Figura 3.3: *Coordinate di riferimento*

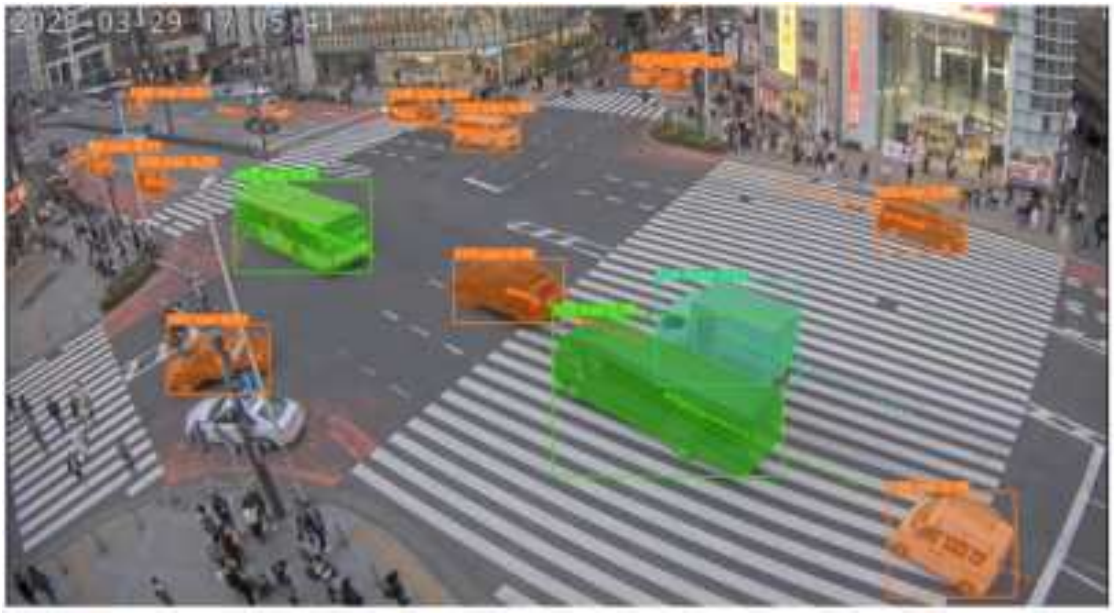


Figura 3.4: *Applicazione di YOLOv8 nano*

Algorithm 1 Algoritmo di tracciamento

```
function TRACKING(stream, target_fps, video_writer)
    frame_id ← 1
    video_fps ← stream.get_fps()
    frame_2_skip ← round(video_fps/target_fps) - 1
    while stream not empty do
        frame ← stream.read()
        frame_id ← frame_id + 1
        bboxs, ids, scores, class_ids ← detect_and_track(frame)
        for i ← 1 to len(bboxs) do
            save_crops(frame, bboxs[i])
            update_trajectories_dataset(bboxs[i], frame_id, ids[i], scores[i], class_ids[i])
        frame2 ← decorate_frame(bboxs, scores, class_ids)
        video_writer.add_frame(frame2)
        for _ ← 1 to frame_2_skip do
            frame_id ← frame_id + 1
            stream.read()                                          ▷ salta frame
```

3.4 Creazione di un dataset di riferimento tramite online-tracking su video di traffico reale

3.4.1 Contesto di riferimento

Figura 3.5: *Tokyo Cam (fonte: [51])*

Figura 3.6: *Stazioni AQI nei pressi dell'incrocio di riferimento*

Allo scopo di raccogliere dati significativi che potessero supportare la creazione di modelli precisi ed efficaci per il monitoraggio il traffico urbano ed in particolare per il task di predizione delle traiettorie dei veicoli, la fase di iniziale di ricerca ha riguardato la selezione del contesto di riferimento da monitorare e la metodologia di acquisizione

dei dati grezzi da elaborare. La scelta del contesto di riferimento è una scelta cruciale per garantire un'adeguata rappresentatività del dataset, e può variare in base al task per cui è pensato. Considerato il complesso task di predizione delle traiettorie dei veicoli, il contesto di riferimento deve essere caratterizzato ovviamente da un numero relativamente elevato di veicoli ed inoltre si deve garantire che le traiettorie dei veicoli presenti siano abbastanza eterogenee. Ad esempio, se il caso di studio facesse riferimento al traffico in autostrada, le traiettorie sarebbero eccessivamente regolari e l'applicabilità del dataset in contesti più complessi sarebbe limitata. Per questo motivo, la ricerca si è diretta verso l'identificazione di un ambiente reale, in particolare, un incrocio stradale. Per lavorare su tale contesto, si sono prese in considerazione due soluzioni. La prima soluzione ipotizza la presenza di risorse locali, come sensori di inquinamento, telecamere, microfoni da utilizzare in un ambiente reale. La seconda soluzione prevede, invece, di recuperare informazioni accessibili online per creare un ambiente virtuale di lavoro. Poiché nella fasi iniziali della ricerca non è stato possibile reperire risorse interne, si è optato per il secondo approccio.

Identificazione di un caso reale: Un incrocio stradale a Tokyo

In rete sono reperibili diverse sequenze video di traffico. Tra le diverse alternative la scelta è ricaduta su una specifica telecamera di traffico in tempo reale situata a Tokyo [51], in Giappone, la quale offre un flusso video continuo di 24 ore accessibile da YouTube e un archivio contenente i dati storici. Questo ambiente in particolare rappresenta un contesto estramemente complesso ed eterogeneo, come si può vedere in figura 3.5. La scelta dell'incrocio stradale a Tokyo come contesto principale per l'analisi del traffico è stata guidata dalla sua eccezionale complessità e rappresentatività delle dinamiche urbane. Questo scenario offre una varietà di elementi che contribuiscono alla sua unicità e rilevanza per la presente ricerca sul monitoraggio del traffico ed il task di predizione delle traiettorie dei veicoli:

- Elevato numero di corsie: L'incrocio in questione presenta una configurazione stradale notevolmente articolata, con otto corsie sulla strada principale e tre sulla strada secondaria. Questa molteplicità di corsie introduce una complessità aggiuntiva dovuta alle numerose interazioni tra veicoli che si spostano in direzioni diverse. L'analisi di un ambiente così articolato fornisce un quadro completo delle dinamiche di traffico in un contesto urbano.

- Prossimità ad una stazione dei treni: La presenza di una stazione dei treni a pochi metri dall'incrocio aggiunge una dimensione significativa al contesto. Questo elemento comporta flussi aggiuntivi di pedoni provenienti e dirigendosi verso la stazione, veicoli legati ai trasporti pubblici e possibili congestioni durante gli orari di punta dei treni. La coesistenza di flussi veicolari e pedonali complessi rende questo scenario particolarmente sfidante per il monitoraggio, richiedendo modelli di tracciamento robusti.

- Diversità di veicoli: La varietà di veicoli che attraversano l'incrocio aggiunge un ulteriore strato di complessità. Dai veicoli privati, ai taxi e ai bus, ogni categoria presenta caratteristiche differenti. Valutare le tecnologie selezionate in questo contesto richiede non solo un'elevata precisione nel rilevamento degli oggetti ma anche la capacità di discernere tra le diverse categorie in tempo reale.

- Alta densità di pedoni in una metropoli affollata: Tokyo è nota per la sua densità di popolazione e l'incrocio in esame riflette questa realtà. La presenza di un gran numero di pedoni attraversanti aggiunge ulteriori sfide al tracciamento delle traiettorie.

- Possibile integrazione di dati ambientali: Infine, nell'ottica di estendere il dataset con informazioni di contesto aggiuntive, nelle vicinanze della telecamera sono presenti due stazioni AQI (Air Quality Index) che forniscono dati sulla qualità dell'aria in termini di inquinanti e dati ambientali come temperatura, umidità, vento, ecc. Tali dati possono essere facilmente ottenuti tramite API in formato JSON [1, 2].

In questo contesto, le classi selezionate per il rilevamento includono pedoni, auto, autocarri (truck), motocicli, bici e bus.

3.4.2 Acquisizione dati

La fase di acquisizione dati ha riguardato il download di 70h di video di traffico dalla sorgente precedentemente identificata [51], tramite la libreria *youtube-dl* [52]. In particolare, si è scelto di monitorare il traffico durante l'orario di punta della giornata, ovvero tra le 12:00 e le 14:30 (ora locale), recuperando i flussi video dall'archivio Youtube contenente le *live* registrate, a partire dalla fine dell'anno 2022 fino alla metà del 2023, periodo in cui è stata svolta ricerca. Analizzare video relativi allo stesso time-slot ha contribuito a creare un dataset omogeneo e rappresentativo delle dinamiche del traffico durante l'orario di punta, migliorando così la qualità e l'utilità delle informazioni ottenute.

La seguente tabella mostra in dettaglio alcune statistiche sulle sequenze di video acquisite.

Tabella 3.1: *Statistiche sui video di traffico acquisiti*

N. Video	Date di Pubblicazione	Time slot	Durata	Totale Ore
28	Nov 2022 - Giu 2023	12:00 - 14:30	2h30mn	70h

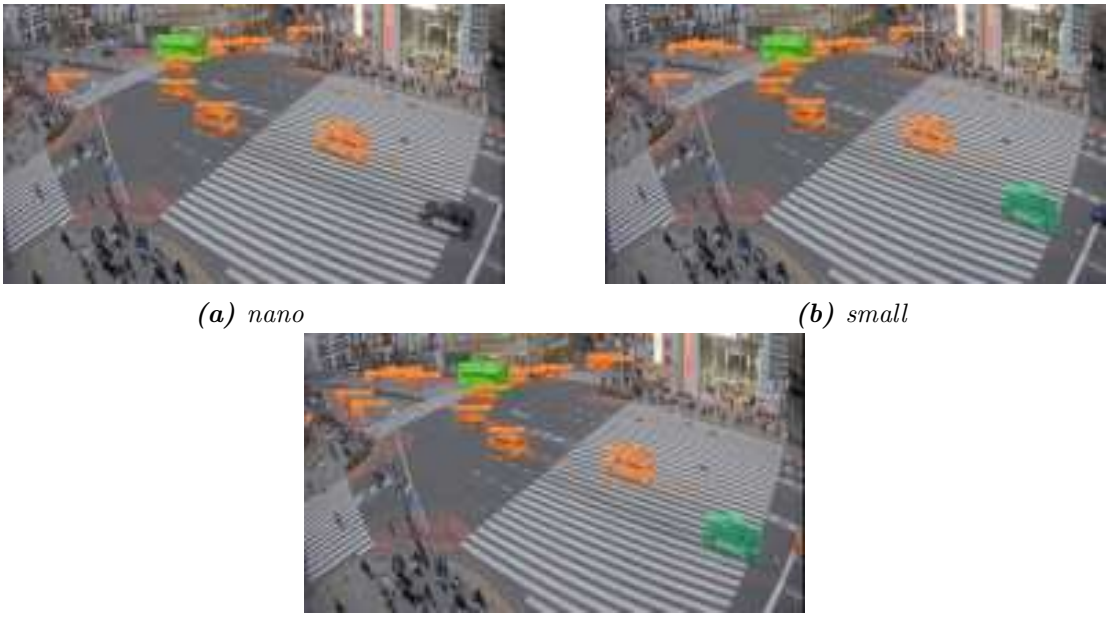
3.4.3 Pre-processing della sequenza video

Per ridurre i tempi di inferenza sui video in input, è stato deciso di diminuire il *frame rate* originale da 25fps a 10fps. Questo è stato ottenuto mediante adeguate modifiche al codice, consentendo di impostare facilmente il numero di frame da saltare nella sequenza,

durante la fase di inferenza. È importante sottolineare che una riduzione eccessiva del frame rate comporterebbe delle difficoltà nel corretto tracciamento degli oggetti lungo la sequenza video. Pertanto, la scelta di diminuire il frame rate a 10fps è stata un compromesso tra la riduzione dei tempi di inferenza e la necessità di mantenere una frequenza sufficiente per un tracciamento efficace degli oggetti nel video.

3.4.4 Scelta del modello di detection

Per quanto riguarda la scelta del modello, sono stati presi in considerazione i modelli preaddestrati di YOLOv8 su dataset COCO. Nelle fasi preliminari del lavoro, si è svolta un'analisi qualitativa per volta a determinare la dimensione dei modelli più adatta, tenendo in considerazione i limiti hardware e la necessità di produrre risultati significativi ed accurati. È stata eseguita una fase di detection su un video di test, utilizzando tre diverse taglie di YOLOv8: nano, small e medium. Come si può vedere in figura 3.7, analizzando i video prodotti in output applicando rispettivamente la versione nano 3.7a, small 3.7b, e medium 3.7c, non si è riscontrato un calo rilevante delle prestazioni al variare delle dimensioni dei modelli, anzi, i risultati sono del tutto paragonabili. Dunque, tenendo conto dei risultati ottenuti, avvalorati anche dalle prestazioni dichiarate da Ultralytics (tabella 3.2), e delle limitate capacità computazionali disponibili durante lo studio, si è deciso di utilizzare la versione *nano* di YOLOv8.



(a) *nano* ***(b)*** *small*

(c) *medium*

Figura 3.7: *Confronto YOLOv8 nano - small - medium*

3.4.5 Scelta dei parametri

La fase successiva ha coinvolto la scelta dei parametri. In particolare, la *confidence threshold* e la soglia di *Intersection over Union*. La prima specifica la soglia minima di confidenza necessaria per classificare l'oggetto rilevato ad una determinata classe, questa è stata posta a 0.3. La seconda invece, è utilizzata in YOLOv8 nella fase di Non-Maximum Suppression (NMS) per eliminare iterativamente le bounding box sovrapposte con punteggio più basso. In pratica, una bounding box viene rimossa se la sua IoU con un'altra bounding box di punteggio più alto è maggiore della soglia specificata. Questa tecnica, riportata sotto forma di pseudocodice in fig. [2], aiuta a mantenere solo le rilevazioni più accurate ed evita sovrapposizioni eccessive tra bounding box. La *IoU threshold* è stata posta pari a 0.45. L'attenta selezione di questi parametri permette di bilanciare la precisione dei rilevamenti, evitando falsi positivi e contemporaneamente ottenere una buona copertura di oggetti rilevati in relazione a quelli effettivamente presenti nella sequenza.

Algorithm 2 Non Maximum Suppression

```
1: procedure NMS(B, c, threshold)
2:     B_nms ← {}                                   ▷ Initialize empty set
3:     for b_i ∈ B do                               ▷ Iterate over all the boxes
4:         discard ← False
5:         for b_j ∈ B do
6:             if IoU(b_i, b_j) > threshold then
7:                 if score(c, b_j) > score(c, b_i) then
8:                     discard ← True
9:             if discard = False then
10:                B_nms ← B_nms ∪ {b_j}
11:    return B_nms
```

3.5 Sviluppo del framework di valutazione

3.5.1 UA-Detrac dataset

Come precedentemente illustrato nel capitolo dello stato dell'arte (vedi Par. 2.4), il dataset UA-Detrac si rivela estremamente utile nella valutazione del framework proposto per il Multi-Object Tracking (MOT). Concepito come una risorsa per la rilevazione multi-oggetto e il tracciamento multi-oggetto nel contesto del traffico reale, UA-Detrac fornisce dati dettagliati e situazioni complesse che lo rendono un'eccellente risorsa di benchmarking.

Per l'integrazione di UA-Detrac nel framework di valutazione, si è focalizzata l'attenzione sul sottoinsieme di (DETRAC-train). Il dataset è composto da 60 sequenze video catturate in diverse location, offrendo oltre 84 mila frame e dettagliate annotazioni. Le

annotazioni comprendono un totale di 5.9 mila veicoli manualmente etichettati con 578 mila bounding boxes. Le categorie dei veicoli sono suddivise in 'car', 'bus', 'van' e 'others'. Ogni sequenza è costituita da un insieme di immagini sequenziali, accompagnate dal relativo dataset di annotazioni, nel formato XML, Detrac-Train-Annotation. Ogni sequenza video è annotata come segue:

- **ignored region**: Il campo 'ignored_region' specifica una serie di bounding box, i quali identificano zone dell'immagine ignorate durante la fase di annotazione manuale degli oggetti presenti nell'immagine (vedi fig. [3.8]).
- **frame**: La lista di frame presenti nell'immagine. Per ogni frame sono presenti gli oggetti identificati 'target'.
- **target**: Ogni frame contiene una 'target_list', vale a dire una lista di oggetti target rilevati. Ogni target è rappresentato dalle informazioni posizionali della corrispondente bounding box nell'immagine, il tipo del veicolo ed altre informazioni utili riguardo la traiettoria.

Figura 3.8: *Zone ignorate in fase nell'annotazione UA-Detrac*

Per garantire una perfetta integrazione con il framework MOT, si è proceduto a trasformare le annotazioni iniziali in formato CSV (vedere Tabella 3.2), seguendo la struttura: *frame_id, id, bbox_left, bbox_top, bbox_w, bbox_h, class_id*. Questa preparazione del dataset, come illustrato in precedenza, assicura la compatibilità con il sistema precedentemente proposto e ne facilita l'integrazione.

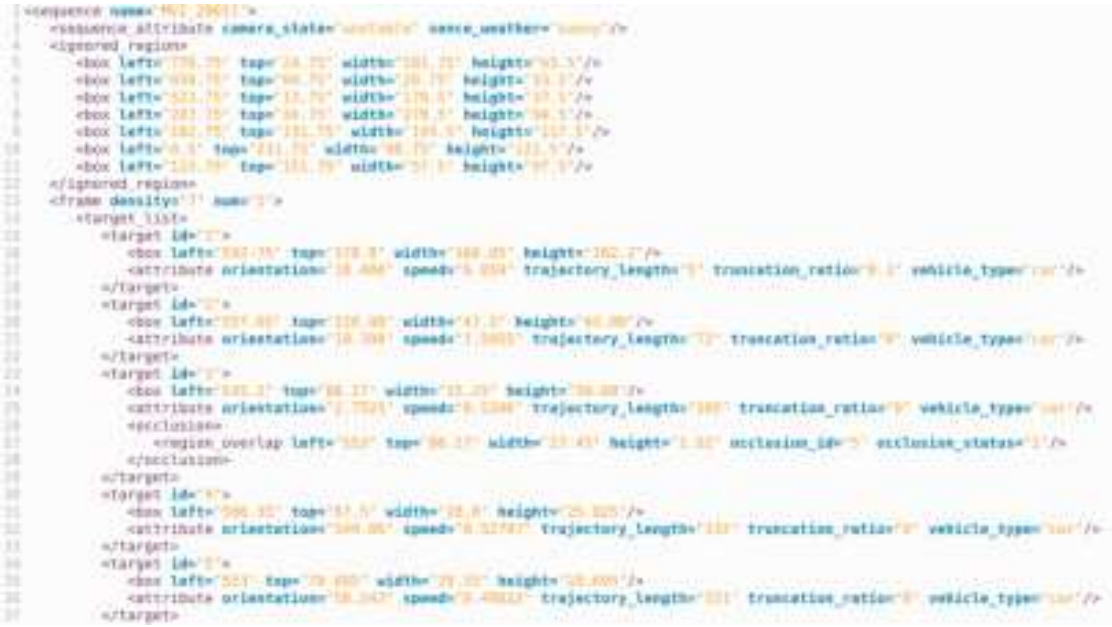


Figura 3.9: *UA-Detrac dataset: Esempio di annotazioni in formato XML*

Tabella 3.2: *Annotazioni nel formato MOT prestabilito*

frame id	id	bbox left	bbox top	bbox w	bbox h	class id
1	1	592.75	378.8	160.05	162.2	car
1	2	557.65	120.98	47.2	43.06	car
1	3	545.2	88.27	35.25	30.08	car
1	4	508.35	67.5	28.0	25.925	car
1	5	553	70.095	29.55	19.695	car
1	6	731.1	114.23	52.4	39.95	car
1	7	902.15	250.12	58.85	107.99	car
2	1	594.2	391.81	161.5	149.19	car
2	2	557.7	121.87	48.1	44.12	car
..	..	..	..	..	..	..

3.5.2 Valutazione della fase di detection

Per eseguire una valutazione accurata del modello di detection utilizzato nella precedente fase di raccolta dati ed inferenza si è scelto di calcolare le metriche nella sezione 2.5. Precision, Recall e mAP (mean Average Precision). Innanzitutto è stato necessario eseguire YOLOv8 Nano, utilizzando i medesimi parametri scelti in precedenza, sul dataset di benchmark UA-Detrac per ottenere le predizioni, frame per frame. Ottenuti i dati di output, si è proceduto all'implementazione delle suddette metriche. Dato che il dataset di benchmark include categorie leggermente differenti rispetto a quelle del dataset COCO, su cui è stato allenato il modello YOLO utilizzato è nella fase di valutazione è stato necessario realizzare un mapping tra le classi, cercando di ridurre al minimo la possibilità di errore e rendere compatibili i dati di output con le annotazioni di UA-Detrac. In particolare, oltre all'associazione ovvia 'car-car' e 'bus-bus, è stato associata la categoria 'truck' di YOLO alla classe 'van' di UA-Detrac, mentre le categorie 'bycycle' e 'motorcycle' sono state associate alla classe 'others'.

Precision e Recall

Precision e recall rappresentano rispettivamente la fedeltà del modello nell'evitare di rilevare oggetti quando non sono presenti e la copertura del modello, o meglio, l'abilità del modello di rilevare quanti più oggetti nell'immagine. Queste misure, richiedono il calcolo preliminare del numero di rilevamenti veri positivi e falsi positivi, che rispettivamente rispecchiano quanti oggetti rilevati dal modello di detection trovano o meno riscontro nella ground truth. Dati un insieme di bounding box predette e di bounding box rappresentano la ground truth, il compito del sistema di valutazione sviluppato si è focalizzato sul trovare una corrispondenza, frame per frame, tra questi due insiemi. È possibile associare due bounding box sulla base della loro sovrapposizione nell'immagine, quindi, attraverso il calcolo dell'IoU ed impostando una soglia minima di sovrapposizione accettata per considerare la previsione come corretta. L'algoritmo presentato (vedi fig. [3]), fa esattamente questo. Iterando sulle bounding box ottenute da YOLOv8, per ciascun frame, cerca una corrispondenza nell'insieme delle ground truth, tale che la sovrapposizione superi la soglia prestabilita ed a parità di classe assegnata all'oggetto. Una volta ottenuti il numero di falsi positivi (FP) e veri positivi (TP), si calcola la precisione come la percentuale di veri positivi su tutte le predizioni e la recall come la percentuale di veri positivi rispetto a tutte le ground truth.
Tuttavia, l'algoritmo [3], non tiene conto delle '*ignored region*' (IR) presenti nel dataset UA-Detrac. È stato quindi opportunamente modificato, in modo tale che durante la valutazione vengano scartate le ipotesi posizionate all'interno delle IR, che altrimenti non troverebbero nessun riscontro tra le ground truth. Questo è stato fatto utilizzando ancora una volta la IoU tra le bounding box dei veicoli identificati e quelle rappresentanti le IR. Come si può vedere in fig. [3.8], però, le '*ignored region*' consistono in bounding box piuttosto grandi rispetto alle dimensioni di un singolo veicolo, per questo motivo si è deciso di porre una soglia di IoU bassa (0.1). Questa scelta è stata avvalorata da un'analisi qualitativa, valutando degli *snapshot* contenenti sia le IR che le predizioni, e

Algorithm 3 Calcolo della Precision e Recall

procedure EVALUATE_DETECTION($predictions, ground\_truths, threshold$)
Initialize $TP \leftarrow 0$, $FP \leftarrow 0$, $NP \leftarrow 0$

for $prediction \in predictions$ **do**
Try to find the ground truth with same class and largest IoU
if IoU$(prediction, gt) > threshold$ **then** ▷ $(threshold = 0.5)$
if gt has not been assigned yet **then**
assign this ground truth to the prediction
$TP \leftarrow TP + 1$
else
$FP \leftarrow FP + 1$ ▷ corresponding gt has been assigned or IoU < threshold

$Precision \leftarrow \frac{TP}{TP+FP}$
$Recall \leftarrow \frac{TP}{\text{number of ground truths}}$

verificando se quest'ultime fossero correttamente ignorate o meno, nella maggiorparte dei casi. Di seguito è riportato uno snapshot esplicativo dell'analisi effettuata 3.10. In blu sono rappresentate le IR, in nero i veicoli rilevati ma ignorati poichè considerati all'interno della IR, mentre in giallo i veicoli rilevati ma che non hanno trovato una corrispondenza nelle ground truth. In viola le ground truth mancate dal detector. In verde le predizioni che hanno trovato una corrispondenza nelle ground truth in rosso. Analizzando l'immagine si può capire come la scelta della soglia IoU per l'identificazione degli oggetti nella IR possa contribuire significativamente nel conteggio dei falsi positivi (FP). Infatti, gli oggetti rilevati in giallo saranno considerati erroneamente tali, poichè nonostante corrispondano effettivamente a dei veicoli, facendo parte della IR, ma non essendo stati scartati, non troveranno una corrispondenza con la ground truth, introducendo un ulteriore margine di errore.

mean Average Precision

Per il calcolo della mean Average Precision (mAP) in letteratura esistono due metodi principali. Uno proposto ed utilizzato sul dataset VOC ed un altro sul dataset COCO [41].

mAP su VOC

1. Per ogni categoria, si calcola la curva *Precision-Recall*, ottenuta calcolando diverse misure di precision e recall al variare della soglia di confidenza delle previsioni del modello.

2. Calcola la precisione media (AP) per ciascuna categoria utilizzando un campionamento interpolato a 11 punti della curva Precision-Recall.

Figura 3.10: *Snapshot contenente predizioni e ignored region*

3. Calcola la precisione media finale (AP) prendendo la media delle AP su tutte le categorie.

mAP su COCO In questo caso, piuttosto che utilizzare una interpolazione a 11 punti, si utilizza un'interpolazione a 101 punti, ossia si calcola la precision per 101 soglie di recall da 0 a 1 con incrementi di 0.01. Inoltre, la Precisione Media (AP) viene ottenuta mediando su più soglie di IoU anziché uno solo, ottenendo molteplici AP. Ad esempio AP50, rappresenta l'AP per una singola soglia di IoU pari a 0.5. I passaggi per calcolare l'mAP in COCO sono i seguenti:

1. Per ogni categoria, calcola la curva precision-recall variando la soglia di confidenza delle previsioni del modello.
2. Calcola la Precisione Media (AP) per ciascuna categoria utilizzando 101 soglie di recall.
3. Calcola l'AP a diversi valori di IoU, tipicamente da 0.5 a 0.95 con un passo di 0.05.
4. Per ogni soglia di IoU, si considera la media delle AP attraverso tutte le categorie.
5. Infine, calcola l'AP complessiva facendo la media dei valori di AP calcolati per ciascuna soglia di IoU.

Curva Precision-Recall La curva Precision-Recall è quindi ottenuta variando la soglia di confidenza del modello per determinare quali previsioni considerare positive. A

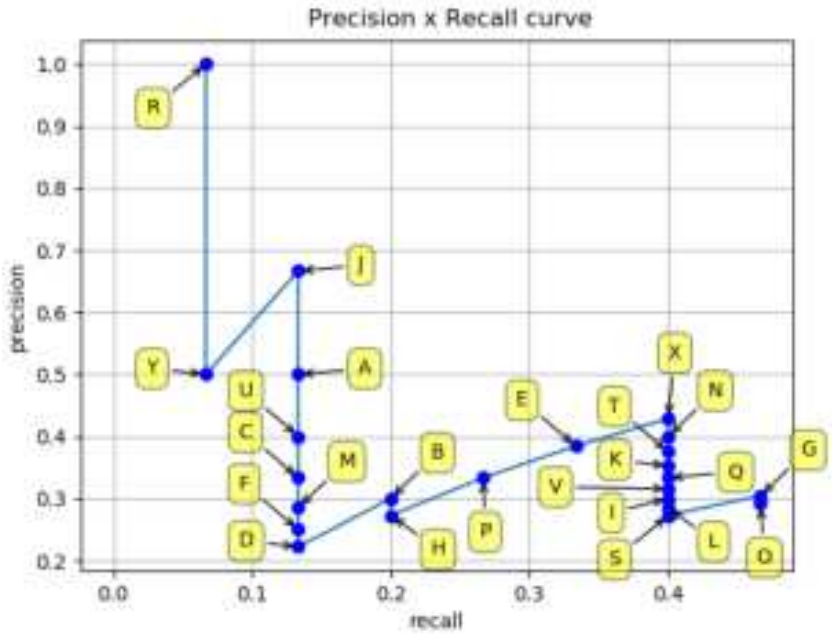


Figura 3.11: *Esempio di curva Precision-Recall*

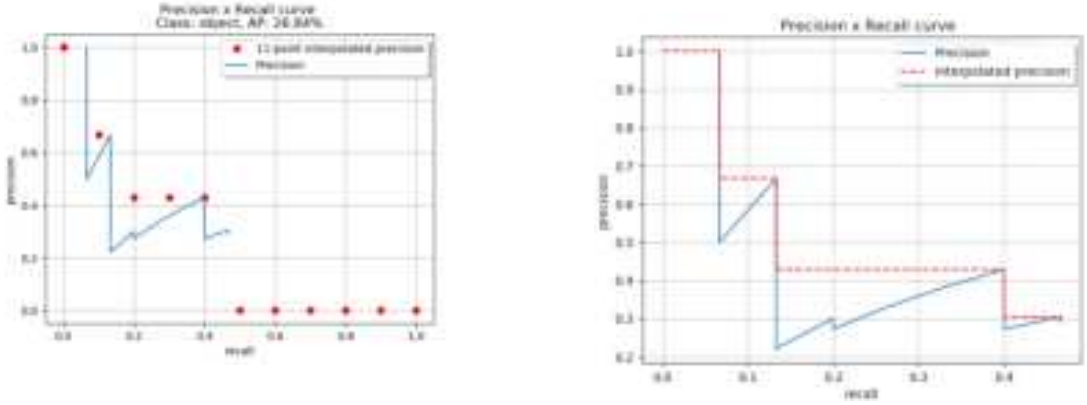
ogni valore di soglia, si calcolano i valori di precisione e recall. Il risultato è una curva che evidenzia come la precisione e il recall cambiano al variare della soglia di confidenza (vedi Fig. 3.12. Idealmente, si vorrebbe avere un modello che mantenga una precisione elevata anche a livelli alti di recall, ma in pratica c'è spesso un trade-off tra queste due metriche.

Campionamento interpolato a diversi valori di recall Nel contesto della valutazione della mean Average Precision (mAP), l'interpolazione a diversi valori di recall è utilizzata per ridurre le fluttuazioni nel grafico Precision-Recall ed ottenere un grafico più *smooth* attraverso una stima della precisione su diverse soglie di recall. Ad ogni valore di recall, si sostituisce la precision con il massimo valore di precision per quel livello di recall, come mostrato in Fig. 3.12.

3.5.3 Valutazione MOT

Precision, recall e mAP offrono una panoramica sulla bontà del metodo di detection utilizzato nel catturare gli oggetti all'interno dei frame in analisi. Tuttavia, il MOT non si limita a questo. Vogliamo stabilire, piuttosto, qual è la qualità dell'intero processo di tracciamento in termini quantitativi, utilizzando misure rappresentative dell'efficacia complessiva del sistema nel tracciamento degli oggetti in ambiente urbano.

Prima di scegliere il metodo di valutazione da applicare è importante illustrare in modo



(a) *(b)*

Figura 3.12: *Esempio di Interpolazione a 11 punti*

dettagliato le caratteristiche che ci si attende da un sistema di MOT. Esso dovrebbe, in ogni istante temporale, identificare correttamente il numero di oggetti presenti e stimare la posizione di ciascun oggetto con la massima precisione possibile (notando che proprietà quali contorno, orientamento o velocità degli oggetti non sono esplicitamente considerate in questa analisi). Inoltre, dovrebbe mantenere un tracciamento coerente di ciascun oggetto nel tempo, assegnando a ognuno un ID di traccia univoco che rimanga costante per l'intera sequenza, persino dopo eventuali occlusioni temporanee.

Sulla base di tali criteri, i metodi utilizzati per questo task, si rifanno alle misure *CLEAR* precedentemente introdotte (2.5) e presentate da Bernardin e Stiefelhagen nello studio 'Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics' [53]. Il metodo proposto è il seguente: presumendo che per ogni istante di tempo t un modulo di MOT produca un insieme di ipotesi $\{h_1, \ldots, h_m\}$ per un insieme di oggetti visibili $\{o_1, \ldots, o_n\}$, la procedura di valutazione comprende i seguenti passaggi.

Per ogni istante di tempo t:

- Stabilire la migliore corrispondenza possibile tra le ipotesi h_j e gli oggetti o_i,
- Per ciascuna corrispondenza trovata, calcolare l'errore nella stima della posizione dell'oggetto,
- Accumulare tutti gli errori di corrispondenza:
  - Contare tutti gli oggetti per i quali non è stata prodotta nessuna ipotesi come mancate (MISSES),
  - Contare tutte le ipotesi del tracker per le quali non esiste alcun oggetto reale come falsi positivi (FP),

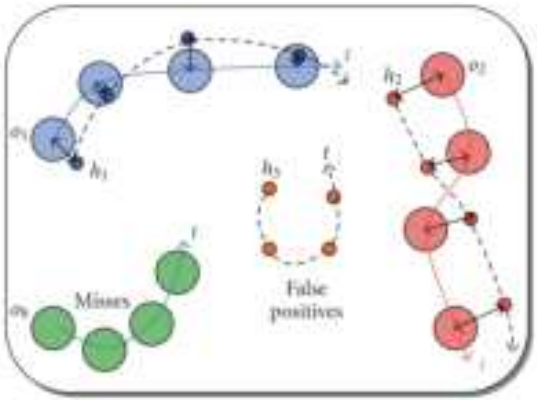


Figura 3.13: *Associazioni tra ipotesi del tracker e oggetti della ground truth*

– Contare tutte le occorrenze in cui l'ipotesi di tracciamento per un oggetto è cambiata rispetto ai frame precedenti come errori di associazione (*mismatch*). Ciò potrebbe accadere, ad esempio, quando due o più oggetti vengono scambiati mentre passano vicino l'uno all'altro, o quando una traccia di oggetti viene reinizializzata con un diverso ID di traccia, dopo essere stata precedentemente persa a causa di un'occlusione.

La figura 3.13 riassume quanto appena descritto. Quindi, le prestazioni di tracciamento possono essere espresse in modo intuitivo con due metriche: la "precisione del tracker" (MOTP), che esprime quanto bene sono stimate le posizioni esatte delle persone, e la "accuratezza del tracciamento" (MOTA), che mostra quanti errori ha commesso il tracker in termini di mancate, falsi positivi, errori di corrispondenza, mancate recuperi di tracce, e così via.

Stabilire la corrispondenza tra oggetti e ipotesi del tracker

Come anticipato, il primo passo nella valutazione riguarda il trovare un'associazione tra la sequenza di ipotesi di oggetti $h_1, ..., h_m$ prodotta dal tracker in ogni frame e gli oggetti reali $o_1, ..., o_n$.

Corrispondenze valide Per determinare corrispondenze valide, come visto precedentemente per il conteggio di falsi positivi e veri positivi nel calcolo di precision e recall, ci affidiamo ad una misura di distanza ($dist_{i,j}$) tra un oggetto o_i ed un'ipotesi o_j e la scelta di una threshold T che ammetta un minimo margine ddi errore nella stima della posizione. Questo è raffigurato in Figura 3.14(a). In particolare, se si considera l'IoU come misura di similarità. La distanza può essere vista come $dist_{i,j} = 1 - IoU(o_i, h_j)$ e consideriamo le corrispondenze come valide se $dist_{i,j} < T$.

Tracciamento coerente nel tempo In secondo luogo, per misurare la capacità del tracker di etichettare gli oggetti in modo coerente, è necessario rilevare quando sono

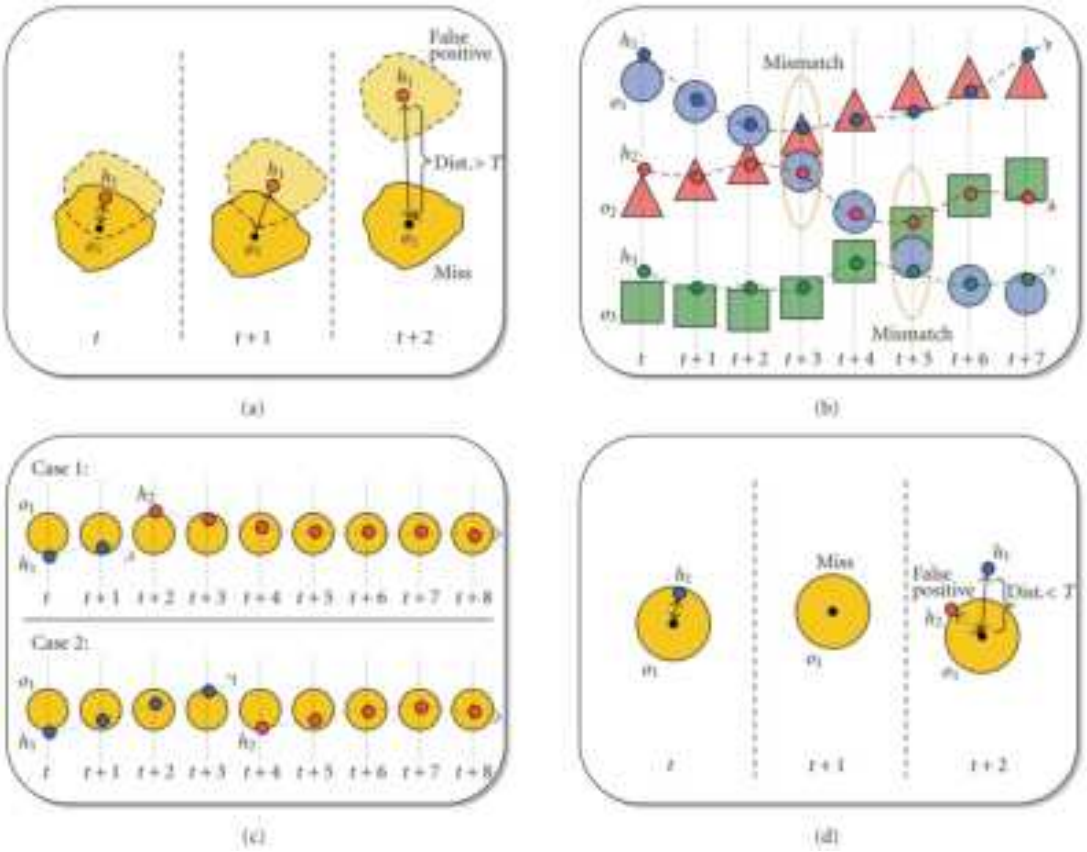


Figura 3.14: *Corrispondenze ottimali e misure di errore*

state effettuate corrispondenze in conflitto per un oggetto nel tempo. La Figura 3.14(b) illustra il problema. Qui, una traccia è stata erroneamente assegnata a 3 oggetti diversi nel corso del tempo. Un mismatch può verificarsi quando gli oggetti si avvicinano l'uno all'altro e il tracciatore scambia erroneamente le loro identità. Può anche verificarsi quando una traccia è stata persa e reinizializzata con un'identità diversa. Un modo per misurare tali errori potrebbe essere decidere su una mappatura "migliore" (o_i, h_j) per ogni oggetto o_i e ipotesi h_j, ad esempio, sulla base della corrispondenza iniziale fatta per o_i, o la corrispondenza (o_i, h_j) più frequentemente fatta nell'intera sequenza. Si dovrebbero quindi contare tutti gli abbinamenti in cui questa mappatura viene violata come errori. In alcuni casi, questo tipo di misura può tuttavia diventare non intuitiva. Come mostrato nella Figura 3.14(c), se, ad esempio, l'identità dell'oggetto o_i viene scambiata solo una volta nel corso della sequenza di tracciamento, l'intervallo temporale in cui avviene lo scambio influenza drasticamente il valore prodotto da tale misura di errore.

Per questo motivo, nelle metriche CLEAR si utilizza un approccio differente: si contano i *mismatch* solo nei frame in cui viene rilevato un cambiamento (identity switch o IDS) nella mappatura tra oggetto ed ipotesi, mentre si considerano le corrispondenze rilevate nei segmenti intermedi come corrette. Per rilevare quando si verifica un *IDS* viene costruita una lista di coppie oggetto-ipotesi $M_t = (o_i, h_j)$ di associazioni eseguite fino al tempo t. Quindi, se viene trovata una corrispondenza al tempo $t + 1$ tra o_i ed h_k che contraddice una mappatura (o_i, h_j) in M_t, si conta come errore di *mismatch* e (o_i, h_j) viene sostituito da (o_i, h_k) in M_{t+1}. La lista di mappatura così costruita M_t può ora aiutare a stabilire corrispondenze ottimali tra oggetti e ipotesi al tempo $t + 1$, quando esistono scelte multiple valide. La Figura 3.14(d) mostra un caso del genere. Quando non è chiaro quale ipotesi abbinare a un oggetto o_i, viene data priorità a h_o con $(oi, ho) \in Mt$, sebbene possa essere più distante rispetto ad un'altra ipotesi. Questo perché essendo stata associata ad o_i nel frame precedente è molto probabilmente la traccia corretta. Altre ipotesi sono considerate falsi positivi e potrebbero essere verificate perché il tracker emette diverse ipotesi per lo stesso oggetto o_i, o perché un'ipotesi che in precedenza ha tracciato un altro oggetto è accidentalmente passata a o_i.

In dettaglio, la lista delle mappature è così costruita. Si considera inizialmente $M_0 =$ e per ogni frame t:

1. Per ogni mappatura (o_i, h_j) in M_{t-1}, verifica se è ancora valida. Se l'oggetto o_i è ancora visibile e l'ipotesi del tracker h_j esiste ancora al tempo t, e se la loro distanza non supera la soglia T, effettua la corrispondenza tra o_i e h_j per il fotogramma t.

2. Per tutti gli oggetti per i quali non è stata ancora effettuata alcuna corrispondenza, cerca di trovare un'ipotesi corrispondente. Consenti solo abbinamenti uno a uno e coppie per le quali la distanza non supera T. L'abbinamento dovrebbe essere effettuato in modo da minimizzare l'errore totale di distanza oggetto-ipotesi per gli oggetti interessati. Se viene fatta una corrispondenza (o_i, h_k) che contraddice

una mappatura (o_i, h_j) in M_{t-1}, sostituisci (o_i, h_j) con (o_i, h_k) in M_t. Conta questo come un errore di *mismatch*.

3. Dopo i primi due passaggi, si ha a disposizione un insieme completo di coppie corrispondenti per il frame corrente, t. Sia c_t il numero di corrispondenze trovate per il tempo t. Per ciascuna di queste corrispondenze, calcola la distanza d_{it} tra l'oggetto o_i e la sua ipotesi corrispondente.

4. Tutte le ipotesi rimanenti sono considerate falsi positivi. Allo stesso modo, tutti gli oggetti rimanenti sono considerati mancanti. Siano fp_t e m_t il numero di falsi positivi e mancanti, rispettivamente, per il fotogramma t. Sia anche g_t il numero di oggetti presenti al tempo t.

5. Ripeti la procedura dal passaggio 1 per il frame successivo.

Sulla base della strategia appena presentata si definiscono due metriche:

Multiple Object Tracking Accuracy (MOTA) Il MOTA tiene conto di tutti gli errori nel rilevamento ed nell'identificazione degli oggetti fatti dal tracker, compresi falsi positivi, mancanti, mismatch, su tutti i frame. Sommando i diversi errori, rapportati al numero di oggetti totali, otteniamo il tasso di errore totale E_{tot}, e $1 - E_{tot}$ che rappresenta l'accuratezza complessiva del tracciamento (vedi 3.1).

$$MOTA = 1 - \frac{\sum (m_t + fp_t + mme_t)}{\sum g_t}, \tag{3.1}$$

dove m_t, fp_t e mme_t rappresentano rispettivamente il numero di oggetti mancati, falsi positivi e mismatch al tempo t.

Multiple Object Tracking Precision (MOTP) MOTP si ottiene, come si può vedere in 3.2, confrontando la posizione predetta e quella reale per le coppie oggetto-ipotesi su tutti i fotogrammi, mediato per il numero totale di corrispondenze effettuate. Mostra la capacità del tracker di stimare posizioni precise degli oggetti, indipendentemente dalla sua abilità nell'identificazione degli oggetti, nel mantenere traiettorie consistenti, e così via.

A seconda se venga usata una misura di similarità o dissimilarità al numeratore, MOTP rappresenta rispettivamente la precisione o l'errore totale del sistema nello stimare la posizione degli oggetti rilevati. Ad esempio, considerato un oggetto o_i ed un'ipotesi h_j, $d_{i,j}$ potrebbe essere interpretata come misura di dissimilarità tra i due componenti e posto pari a $d_{i,j} = 1 - IoU(o_i, h_j)$. Al contrario, piuttosto che la distanza si potrebbe misurare la sovrapposizione tra o_i ed h_j ponendo $d_{i,j} = IoU(o_i, h_j)$. In quest'ultimo caso, valori elevati di MOTP (vicini a 1) saranno da considerarsi positivi mentre valori bassi (vicini allo zero) indicheranno una precisione del sistema bassa. Nella prima soluzione, l'interpretazione va ovviamente invertita.

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{3.2}$$

Capitolo 4

Analisi dei Risultati

In questo capitolo saranno discussi ed illustrati i risultati ottenuti durante la fase sperimentale della ricerca. Questi riguardano una presentazione ed un'analisi qualitativa del dataset generato eseguendo il framework MOT sviluppato sui dati di traffico video di Toyko. Una discussione dei risultati ottenuti sul task di predizione delle traiettorie tramite modelli di apprendimento automatico allenati sul dataset proposto, ricerca riguardante il lavoro di tesi del collega Mangione. Inoltre, si aprirà una parentesi sui tempi di inferenza ottenuti durante l'esecuzione del sistema proposto. Nella seconda parte si presenteranno i risultati ottenuti dalla fase di valutazione del sistema MOT tramite le metriche precedentemente descritte, mAP, MOTA e MOTP.

4.1 Valutazione dei dati raccolti tramite online-tracking su video di traffico RT

I risultati presentati di seguito sono stati ottenuti eseguendo il modello in locale, utilizzando un personal computer comune. La configurazione utilizzata è descritta in modo dettagliato nella seguente tabella 4.1:

Tabella 4.1: *Configurazione iniziale del sistema*

Modello	YOLOv8
Dimensione	Nano
Tracker	StrongSORT
Formato del modello	PyTorch
FPS	10
Confidence Threshold	0.3
IoU	0.45
Hardware - CPU	Intel® Core™ i5-8250U CPU @ 1.60GHz × 8
Hardware - GPU	GeForce MX150 2GB (GPU mobile integrata)
Hardware - RAM	8GB

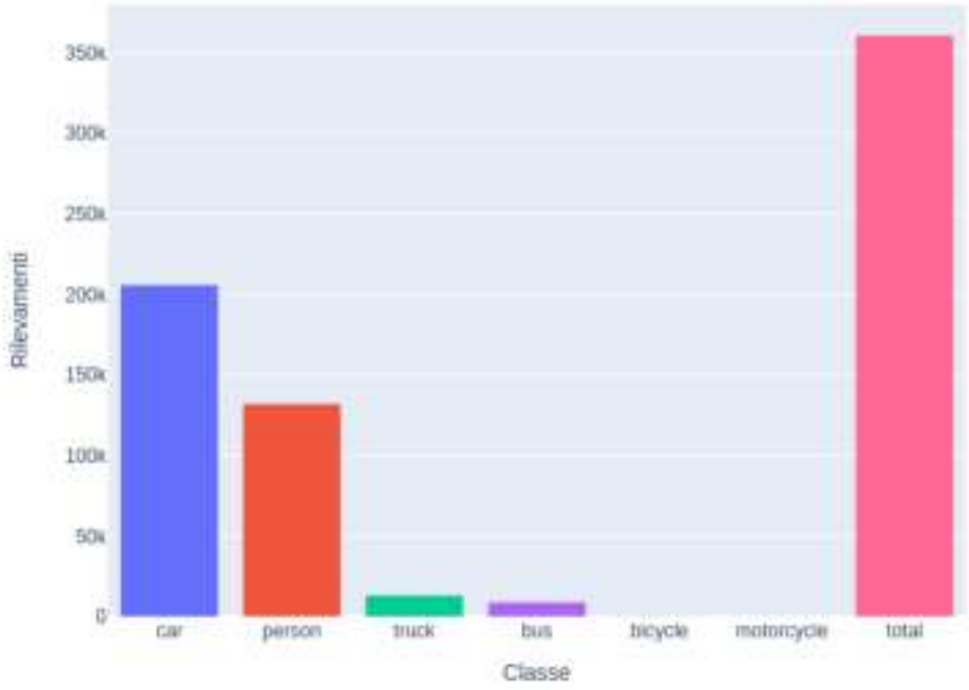


Figura 4.1: *Numero di oggetti unici identificati*

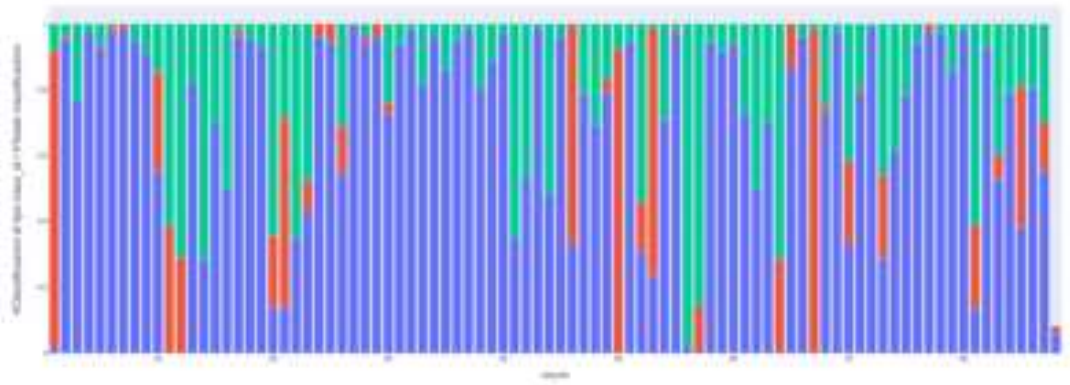


Figura 4.2: *Inconsistenze nella classificazione dei veicoli rilevati*

Analizzando oltre 70 ore di video, il sistema ha identificato oltre 360 mila oggetti unici, generando un dataset con oltre 1.7 milioni di record contenenti le traiettorie degli oggetti identificati, appartenenti alle categorie pedoni, auto, autocarri, motocicli, bici e bus.

Tabella 4.2: *Statistiche sul dataset prodotto*

Ore di traffico video analizzate	70h
Oggetti identificati	360k
Dimensione del dataset	1.7mln
Auto	206k
Pedoni	130k
Autocarri	13247
Bus	9107
Bici	213
Motocili	123

4.1.1 Analisi qualitativa dei risultati ottenuti

L'esame dei risultati, come illustrato nella Figura 4.1 e in tabella 4.2, rivela che la categoria preponderante è rappresentata dalle automobili, con un sorprendente rilevamento di oltre 200 mila auto diverse. La seconda posizione è invece occupata dalla categoria dei pedoni, superando i 130 mila. Seguono 'truck' e 'bus', mentre le moto e le biciclette si attestano ad un livello prossimo allo zero.

Approfondimento dei risultati riguardo i pedoni Tuttavia, una valutazione basata esclusivamente sui dati presentati nella Figura 4.1 potrebbe erroneamente suggerire che il modello sia competente nel rilevamento e tracciamento dei pedoni. Un'analisi più approfondita, che tiene in considerazione anche i video di output generati durante la fase

di inferenza, evidenzia una situazione più complessa. Come si può vedere in figura 4.3, è evidente che, in presenza di un numero cospicuo numero di persone, queste vengono in larga misura ignorate dal detector.

È cruciale, quindi, considerare che la quantità di pedoni rilevati non può costituire da sola un indicatore affidabile delle prestazioni del modello nell'identificazione di questa categoria. Questa discrepanza può essere attribuita principalmente a due fattori di rilevanza.

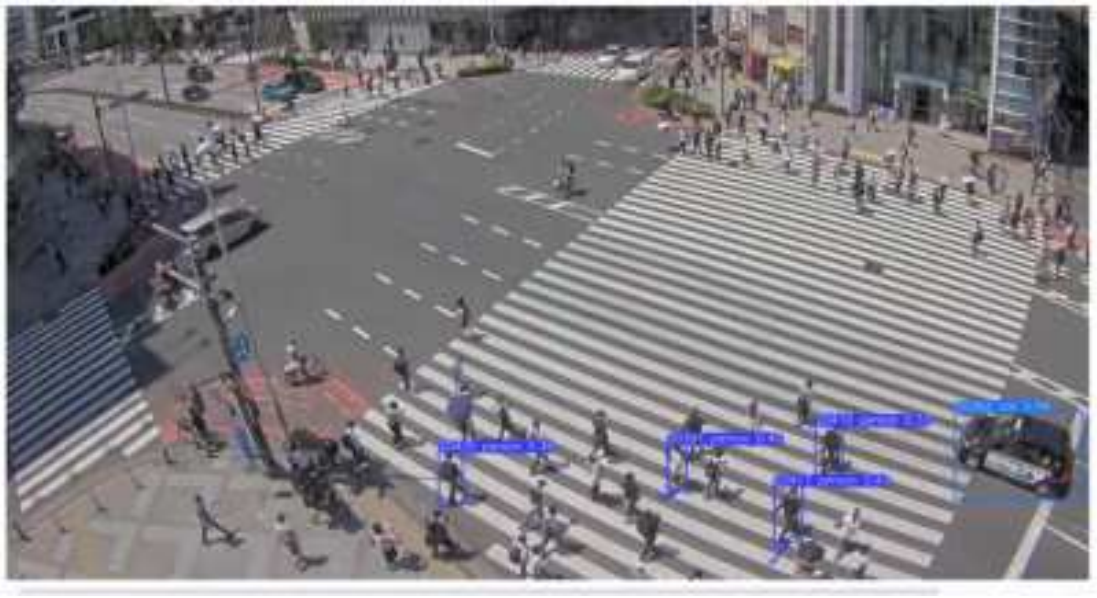


Figura 4.3: *Rilevamento dei pedoni*

- *Incapacità del Detector nel rilevare pedoni:* L'analisi qualitativa evidenzia un numero significativo di casi in cui il detector non è in grado di individuare i pedoni (*misses*). I pedoni, specialmente nel contesto preso in considerazione a causa dell'altezza della telecamera, costituiscono oggetti di difficile individuazione.
- *Ruolo del Tracker:* Un secondo aspetto cruciale riguarda il ruolo del tracker. È plausibile che il notevole numero di pedoni rilevati sia dovuto agli errori del tracker nell'assegnare allo stesso oggetto due o più identificativi differenti nel corso dei frame analizzati.

Pertanto, una valutazione esaustiva delle prestazioni del modello richiede una considerazione attenta di entrambi questi fattori, al fine di ottenere una comprensione più accurata delle capacità di rilevamento dei pedoni.

4.1.2 Inconsistenze nella classificazione degli oggetti

Una prima analisi in merito ai dati prodotti riguarda le inconsistenze nel tracciamento degli oggetti identificati. Nel contesto del MOT, idealmente vorremmo che per ogni oggetto o_i presente in una determinata sequenza di immagini, questo venga consistentemente rilevato, classificato ed identificato allo stesso modo tra i vari frame. Ciò implica che, per ogni frame in cui o_i è presente, innanzitutto deve essere rilevato, ed inoltre gli deve associata la stessa classe e stesso identificativo.

In primo luogo, si considerano gli oggetti identificati consistentemente dal tracker, ma classificati in modo differente nel corso del video. In pratica, si considerano tutti i record presenti nel dataset, avente medesimo codice identificativo dell'oggetto rilevato. Analizzando i dati prodotti, è possibile notare come il dataset contenga diverse inconsistenze, in quanto gli oggetti sono classificati differentemente tra i vari frame. In particolare la figura 4.2 mostra le classi associate, in percentuale, ad un insieme di 200 veicoli identificati. Mentre per alcuni veicoli le classificazioni risultano robuste, avendo la medesima classificazione per l'80%, o più, dei frame. In altri casi, invece, la classificazione è più incerta ed in questo caso diventa difficile determinare con una buona accuratezza la categoria del veicolo, non avendo una classe dominante sulle altre.

4.1.3 Predizione delle traiettorie del veicolo tramite il dataset prodotto

Nel lavoro di ricerca tesi del collega Fabrizio Mangione, il dataset prodotto è stato applicato per addestrare due modelli di apprendimento automatico per il task di predizione delle traiettorie dei veicoli. Sono stati sviluppati due modelli LSTM distinti, rispettivamente **HIST** e **NBRS**, che differiscono l'un l'altro per l'input utilizzato. Esso è indicato come:

$$X = \{x_{t-5}, x_{t-4}, \ldots, x_t\}, \text{ dove } |X| = 6.$$

restituiscono in output un vettore $Y = (x_{t+1}, y_{t+1}), (x_{t+2}, y_{t+2}), (x_{t+3}, y_{t+3})$ contenente le coordinate del veicolo target nei prossimi tre istanti temporali. Attraverso l'unione dei punti, cioè le coordinate predette, viene generato un segmento rappresentante la predizione della traiettoria del veicolo.

Prima di alimentare le reti, durante la fase di training, i dati precedentemente raccolti sono stati processati. In particolare, sono state effettuate operazioni di filtraggio delle voci del dataset. Nello specifico, avendo a disposizione i dati posizionali riguardanti le bounding box dei veicoli identificati, per ogni veicolo, sono state ricavate le coordinate corrispondenti al centro del veicolo. In riferimento all'attributo class id, ovvero la classe di identificazione del veicolo, sono state mantenute solo le voci corrispondenti alle categorie pertinenti, come auto, moto, bus e autocarri eliminando quelle non riconducibili a categorie di veicoli.

Per valutare la precisione delle previsioni relative alla traiettoria di un veicolo, è stato implementato un meccanismo che confronta l'andamento delle curve ottenute unendo i punti nel piano corrispondenti alle coordinate reali, con l'andamento delle curve ottenute unendo i punti nel piano corrispondenti alle coordinate predette dal modello, come si può vedere in fig. 4.4. Questo meccanismo aiuta a determinare quanto le previsioni del modello siano coerenti con i dati reali e se riescano a seguire in modo adeguato l'andamento della traiettoria e può essere utilizzato come misura di accuratezza relativa alle previsioni.

Di seguito è riportato il codice Python che esegue questa verifica, incrementando un contatore ogni volta che viene rilevato un caso in cui l'andamento della curva prevista corrisponde a quello della curva reale:

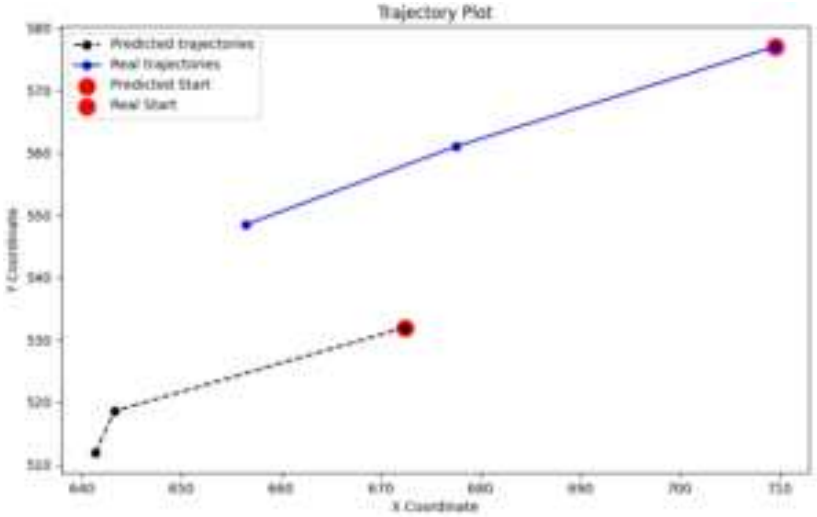
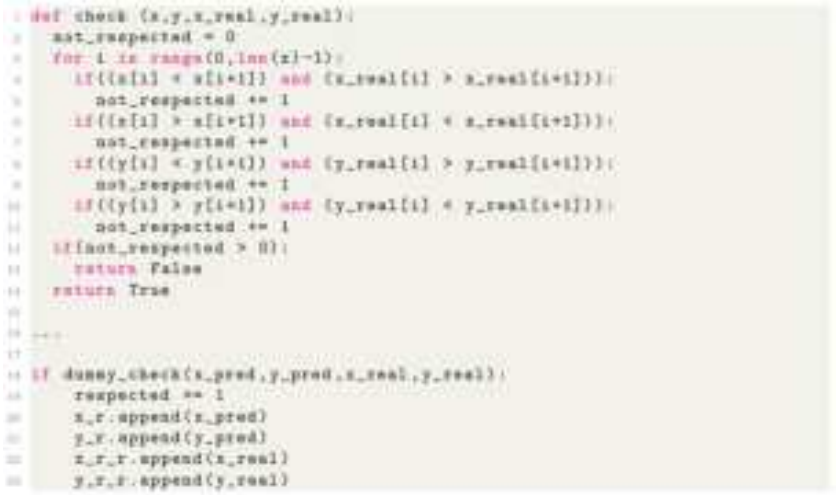


Figura 4.4: *Confronto tra traiettoria reale e predizione*



```
def check (x,y,x_real,y_real):
  not_respected = 0
  for i in range(0,len(x)-1):
    if((x[i] < x[i+1]) and (x_real[i] > x_real[i+1])):
      not_respected += 1
    if((x[i] > x[i+1]) and (x_real[i] < x_real[i+1])):
      not_respected += 1
    if((y[i] < y[i+1]) and (y_real[i] > y_real[i+1])):
      not_respected += 1
    if((y[i] > y[i+1]) and (y_real[i] < y_real[i+1])):
      not_respected += 1
  if(not_respected > 0):
    return False
  return True

...

if dummy_check(x_pred,y_pred,x_real,y_real):
    respected += 1
    x_r.append(x_pred)
    y_r.append(y_pred)
    x_r_r.append(x_real)
    y_r_r.append(y_real)
```

Figura 4.5: *Valutazione modelli di predizione delle traiettorie*

In questo codice vengono confrontate le coordinate predette (x, y), con le coordinate reali (x_{real}, y_{real}), punto per punto. Se in uno qualsiasi dei casi l'andamento previsto non fosse coerente a quello reale si considererebbe la traiettoria predetta come errata.

Attraverso questo approccio, i risultati ottenuti mostrano un tasso di accuracy relativamente basso, sebbene i risultati relativi alla predizione delle coordinate dei veicoli siano promettenti. La causa di tali risultati, potrebbe risiedere in un meccanismo di verifica troppo stringente. A tal punto nello studio portato avanti da Mangione, si è deciso di ripetere la valutazione, considerando come andamenti coerenti, quelli caratterizzati da al più una dissimilarità. In termini percentuali, l'accuratezza aumenta significativamente, passando dal 13% al 41%. Tali valori sono ottenuti prendendo come modello di riferimento HIST, caratterizzato dal maggior numero di traiettorie correttamente predette.

4.2 Valutazione dei tempi di inferenza

Nonostante le limitate risorse computazionali a disposizione (vedi tabella 4.1), durante la fase di inferenza sulla sequenza video estratta da telecamera situata a Tokyo, il sistema ha garantito una velocità di elaborazione pari a 3 frame al secondo, per l'intero il processo di inferenza.

Confrontando tali risultati con quelli riportati da Ultralytics riguardo i benchmark di YOLO (vedi fig. 3.2), si è riscontrata una riduzione delle prestazioni pari a tre volte. Tuttavia, bisogna sottolineare che le prestazioni presentate dal sistema MOT includono, non solo la fase di rilevamento ma anche quella di tracciamento, attraverso il metodo StrongSORT.

Sebbene i risultati siano più che soddisfacenti e mostrino come nonostante le limitate risorse si possa eseguire il framework ottenendo prestazioni di tutto rispetto, qualora si considerasse di applicare il sistema nel contesto di un applicativo *real time*, con un campionamento del flusso video di 10fps, non si riuscirebbe a processare tutti i dati in tempo. Per affrontare questa sfida, si possono considerare diverse tecniche di ottimizzazione, in particolar modo per quanto riguarda il modello di detection YOLO. Ad esempio, si può pensare di applicare tecniche di *model reduction*, o esportare tali modelli in formati ottimizzati per l'esecuzione su dispositivi con hardware limitato, come *TensoRT*, *TF Lite* e *TF Edge TPU* pensato per l'elaborazione su *edge device*. Quando si parla di *model reduction* si fa riferimento a metodi di riduzione delle reti neurali, questi si dividono tipicamente in due macro-categorie: riduzione con perdita e senza perdita. La riduzione senza perdita mira a ridurre la complessità della rete senza compromettere la dimensione del modello. In altre parole, si cerca di comprimere la rete senza perdere informazioni rilevanti per le prestazioni del modello. I metodi comuni includono:

- Pruning: Rimozione di connessioni o neuroni meno importanti senza compromettere l'accuratezza del modello.
- Quantizzazione: Riduzione della precisione numerica dei pesi del modello senza introdurre perdite di informazioni significative.

La riduzione con perdita, invece, comporta una compressione più aggressiva della rete, spesso a costo della perdita di alcune informazioni. L'obiettivo è ridurre la complessità strutturale della rete, ad esempio, diminuendo il numero di strati o di unità neurali in ciascuno strato, accettando una certa quantità di errore o degradazione delle prestazioni. Tali ottimizzazioni potrebbero contribuire significativamente a migliorare le prestazioni del sistema in scenari real-time, garantendo una gestione più efficiente dei flussi video.

4.3 Valutazione del framework MOT utilizzato su UA-Detrac dataset

Di seguito vengono presentati i risultati ottenuti durante la valutazione del framework MOT proposto, utilizzando il dataset UA-Detrac come riferimento.

L'obiettivo della valutazione di un'architettura di questo tipo riguarda il quantificare, secondo metriche accurate ed adatte al contesto in analisi, in modo preciso le prestazioni del sistema. Nel nostro caso, quindi, è necessario fornire dei risultati sia per quanto riguarda le performance nel rilevamento degli oggetti in ambito urbano che l'identificazione ed il tracciamento degli stessi degli oggetti. Un'attenta analisi ci permette di capire quali sono i limiti del sistema proposta, i casi di inconsistenza o errore in cui cade, e l'applicabilità dello stesso in un ambiente reale. Le metriche utilizzate includono la *mean Average Precision* (mAP), *Multi-Object Tracking Precision* (MOTP) e *Multi-Object Tracking Accuracy* (MOTA). Le prime due riguardano principalmente il task di detection, esprimendo dei valori di precisione relativamente al rilevamento degli oggetti

nella scena, mentre MOTA esprime un valore di accuratezza nell'identificazione degli oggetti e quindi nella definizione del tracciato. Verrano inoltre discussi i valori riguardanti il *False Positive Rate* (FPR), il numero relativo di *Identity Switch* (IDS) e di oggetti mancati (MISS).

Curve Precision-Recall e mean Average Precision

La curva Precision-Recall rappresenta uno strumento fondamentale per valutare le performance di un sistema di rilevamento, fornendo una visione dettagliata della capacità del modello di gestire precisione e recall al variare di una soglia di confidenza. La valutazione è stata condotta categoria per categoria (bus, van, car) con l'obiettivo di analizzare le performance specifiche per ciascuna classe. Va notato che la quarta categoria prevista dal dataset UA-Detrac, "others", è stata ignorata poiché, in seguito alla fase di inferenza, non sono state riscontrate rilevazioni per questa specifica classe. Dopo l'inferenza delle sequenze video fornite da UA-Detrac, sono stati calcolati i valori di precision e recall, fissando una categoria per volta e variando i valori di confidenza del modello di detection da 0.3 a 0.9. I dati divisi per video sono stati aggregati per ottenere una singola coppia precision-recall per ogni valore di confidenza e categoria.

Il grafico 4.7 mostra tre curve Precision-Recall per ciascuna categoria, ottenute tramite l'interpolazione di 11 punti di recall. Questo grafico evidenzia la risposta del sistema a diverse soglie di confidenza del modello, riflettendo il compromesso tra precisione e recall per un dato valore di confidenza.

Il grafico a dispersione (scatter plot) tra precision e recall, rappresentato in Fig. 4.6, fornisce un'ulteriore prospettiva sulle performance del sistema MOT e sul bilanciamento tra precisione e recall. Ogni punto è caratterizzato da un valore di precisione sull'asse delle ascisse e un valore di recall sull'asse delle ordinate. La dimensione dei punti è proporzionale ai valori di confidenza del modello per quella specifica coppia precision-recall. Punti posizionati in alto a sinistra indicano una configurazione del sistema con elevati valori sia di precisione che di recall, evidenziando un modello capace di identificare correttamente la maggior parte degli oggetti e farlo con alta precisione. Al contrario, punti situati in basso a destra indicano una configurazione con bassi valori sia di precisione che di recall, segnalando una capacità limitata del sistema di riconoscere e tracciare gli oggetti correttamente.

Utilizzando le curve Precision-Recall descritte in precedenza, sono stati calcolati i valori di Average Precision (AP) per categoria e, infine, la mean Average Precision (mAP) attraverso il metodo VOC. Nella Figura 4.8, sono riportati i valori di AP per categoria. Come ci si poteva aspettare, la categoria per cui si ottiene il valore più elevato è quella delle auto con il 66%; risultati simili si hanno con i bus, dove l'AP si attesta al 57%, mentre per i van si ottiene una AP del 5%.

Sulla base dei valori di AP precedentemente elencati, la mAP del framework MOT

proposto si attesta al 43%. Tuttavia, è importante notare che escludendo la categoria 'van' il sistema ottiene una mAP del 62%.

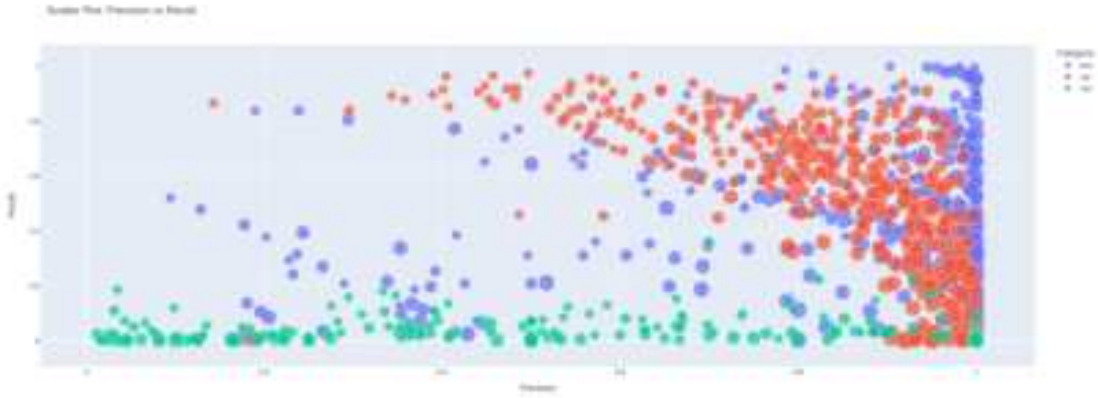


Figura 4.6: *Grafico a dispersione di Precision e Recall*

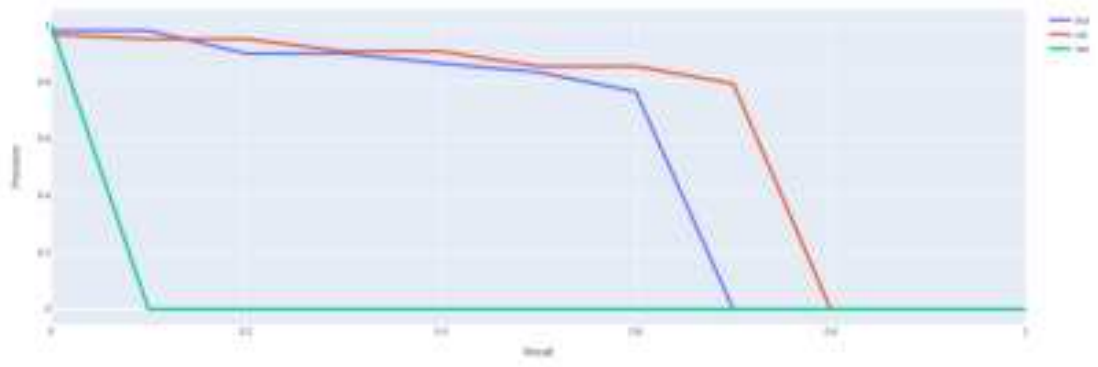


Figura 4.7: *Interpolazione della curva Precision-Recall per categoria*

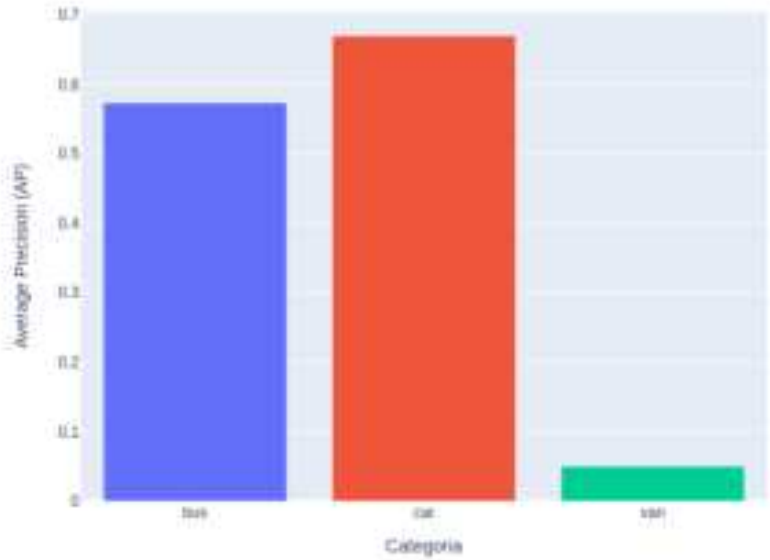


Figura 4.8: *Average Precision per categoria*

Considerazioni sui risultati ottenuti Nel complesso, il sistema di tracciamento oggetti per quanto riguarda la sola fase di rilevamento mostra risultati promettenti. È evidente che per le categorie di auto e bus, il sistema ottiene prestazioni migliori rispetto alla categoria van, evidenziando una maggiore capacità di identificare correttamente gli oggetti desiderati. La categoria van mostra un AP significativamente più basso, indicando sfide specifiche in questa classe. Le sfide riscontrate nella categoria 'van' possono essere attribuite principalmente alla sua confusione con la classe 'car'. I veicoli di tipo van possono essere facilmente scambiati dal modello come semplici automobili, portando a una classificazione errata. Questo fenomeno può essere spiegato dal fatto che la totalità degli ampi dataset di riferimento per l'object detection presentano un forte sbilanciamento sulle classi ed in particolare le automobili sono spesso sovrarappresentate rispetto alle altre categorie di veicoli, creando un dislivello nelle prestazioni tra le diverse classi. Il forte sbilanciamento delle classi nei dataset di addestramento può influenzare la capacità del modello di discriminare tra categorie simili, come nel caso dei veicoli di tipo 'van' e 'car'. Lo stesso vale per il dataset COCO, utilizzato per allenare il modello YOLOv8 utilizzato nel sistema MOT proposto. L'esclusione della categoria 'van' dall'analisi complessiva della mean Average Precision (mAP) migliora notevolmente il rendimento globale del sistema. Questo suggerisce che, nonostante le sfide specifiche nella categoria 'van', le prestazioni complessive del sistema sono positive. Affrontare le disuguaglianze nelle distribuzioni delle classi potrebbe costituire una strategia cruciale per ulteriori miglioramenti, specialmente nelle categorie meno rappresentate, garanten-

do così una valutazione più equa e accurata delle capacità complessive del sistema di tracciamento oggetti.

Metriche CLEAR per il tracciamento

Multiple Object Tracking Accuracy (MOTA)

$$MOTA = 1 - \frac{\sum (m_t + fp_t + mme_t)}{\sum g_t}$$

Dove m_t, fp_t e $mme_t \in (-\inf, 1]$ rappresentano rispettivamente il numero di oggetti mancati, falsi positivi e di *identity switch* al tempo t.

Il MOTA tiene conto di tutti gli errori nel rilevamento ed nell'identificazione degli oggetti fatti dal tracker, compresi falsi positivi, oggetti mancanti e mismatch, su tutti i frame. Il suo range di valori è compreso da un massimo di 1 a $-\inf$. Se MOTA è 1, allora l'accuratezza del sistema è ottima. Se MOTA è intorno a zero o inferiore a zero, allora l'accuratezza del sistema è scarsa.

In Figura 4.9 viene illustrata la distribuzione del valore MOTA su ogni video del dataset UA-Detrac.

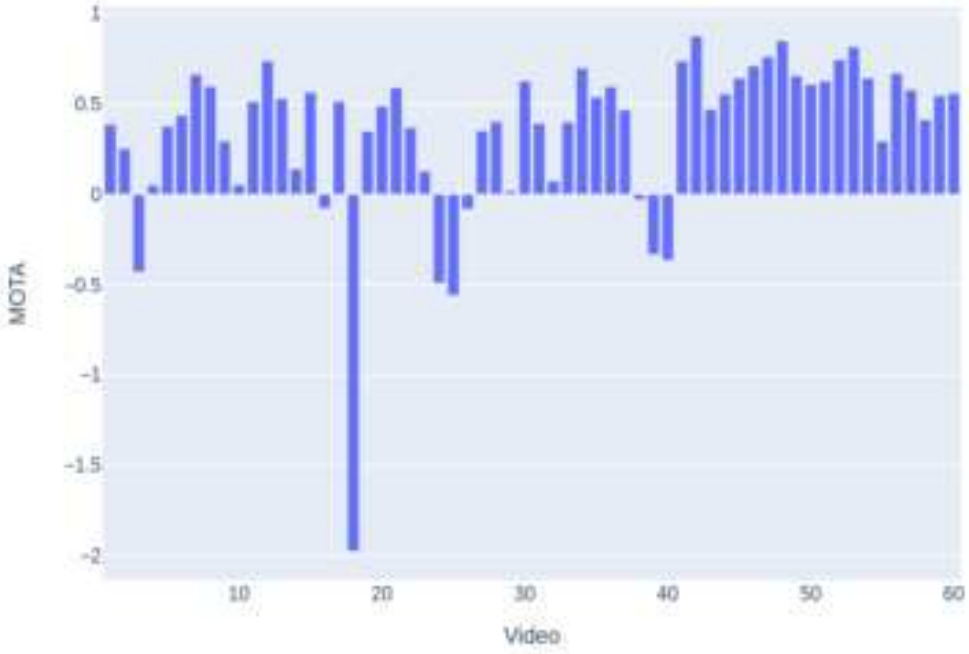


Figura 4.9: *Distribuzione della MOTA per ogni video nel dataset UA-Detrac.*

Nell'esaminare i risultati, emerge un valore di accuratezza notevolmente basso, in netto contrasto con i valori ottenuti per le altre sequenze, per quanto concerne il video 18 (MVI_39811). L'analisi del tasso di falsi positivi (Figura 4.13) rivela che ciò è attribuibile alla rilevazione di oggetti nelle immagini da parte del sistema che non corrispondono a nessuna verità di riferimento. Questo è ulteriormente evidenziato nell'analisi dell'immagine, che comprende sia le ground truth che le predizioni effettuate (Figura 4.10). Le regioni da ignorare (IR) sono rappresentate in blu, i veicoli rilevati ma ignorati poiché considerati all'interno delle IR sono in nero, mentre in giallo sono indicati i falsi positivi. Le ground truth mancanti dal detector sono in viola, mentre in verde sono evidenziate le predizioni che hanno trovato corrispondenza con le verità di riferimento in rosso. Nonostante il numero di falsi positivi (bounding box in giallo) sia notevolmente elevato, è evidente che corrispondano effettivamente a veicoli.

Analizzando le IR del video MVI_39811, come mostrato in Figura 4.11, emerge che il problema risiede nella fase di valutazione, e ci suggerisce che diverse predizioni dovrebbero essere state ignorati nel processo di associazione tra oggetti e ipotesi. Tale risultato è incoraggiante poiché dimostra la capacità del sistema nel rilevare gli oggetti, anche a lunga distanza e in presenza di immagini sfocate, e suggerisce che errori di MOTA bassi non siano necessariamente attribuibili a errori da parte del tracker nella fase

di identificazione.

In generale, i valori di MOTA sono incoraggianti ed il valore medio di *accuracy* del sistema si attesta intorno al 35%. Escludendo il filmato MVI_39811, viziato da alcune imprecisioni nella fase di associazione oggetto-ipotesi a causa delle IR, il sistema raggiunge un valore medio di MOTA, in percentuale, oltre il 39%.

Figura 4.10: *Predizioni nel filmato MVI_39811 di UA-Detrac*

Figura 4.11: *Ignored region nel filmato MVI_39811 di UA-Detrac*

MOTP (Multiple Object Tracking Precision) I risultati ottenuti sulla metrica MOTP evidenziano un livello estremamente positivo in termini della capacità del tracker nel fornire stime precise sulla posizione degli oggetti. Nel calcolo di tale metrica, sulla base della formula 3.2, è stata adottata una misura di similarità, specificamente l'indice di sovrapposizione (IoU), ponendo $d_{i,j} = IoU(o_i, h_j)$. Così facendo, valori elevati di MOTP (prossimi a 1) indicano un'eccellente precisione del sistema nel seguire e stimare la posizione degli oggetti nel tempo, mentre valori vicini a zero riflettono una precisione del sistema più bassa. Il risultato aggregato, con una media di MOTP pari a 0.85 su tutti i video, sottolinea in modo significativo la coerenza delle stime del tracker e la loro prossimità alle posizioni reali degli oggetti nel contesto del dataset UA-Detrac.

La Figura 4.12 fornisce una rappresentazione della distribuzione della metrica MOTP per ciascun video nel dataset, ulteriormente confermando l'alto livello di precisione ottenuto dall'algoritmo di tracciamento su diverse sequenze.

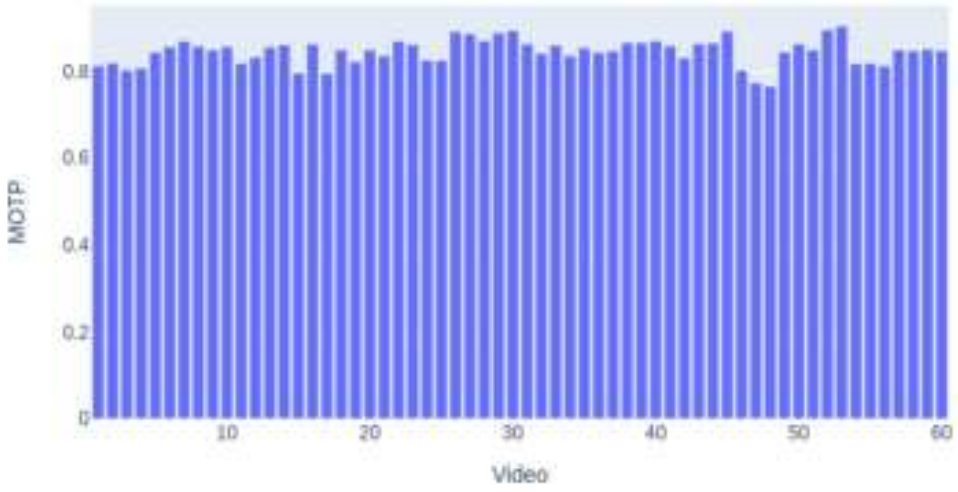


Figura 4.12: *Distribuzione della MOTP per ogni video nel dataset UA-Detrac*

False Positives Rate (FPR) Il tasso medio di falsi positivi (FPR), calcolato come il rapporto tra la somma totale dei falsi positivi e la somma delle ground truth, su tutti i video si attesta al valore di 0.48. Questo valore risulta influenzato dal numero significativamente elevato di falsi positivi nel video MVI_39811, che è approssimativamente triplo rispetto al numero di oggetti annotati nel dataset. Tale discrepanza è evidenziata nella Figura 4.13, la quale illustra la distribuzione dei falsi positivi per ciascun video nel dataset.

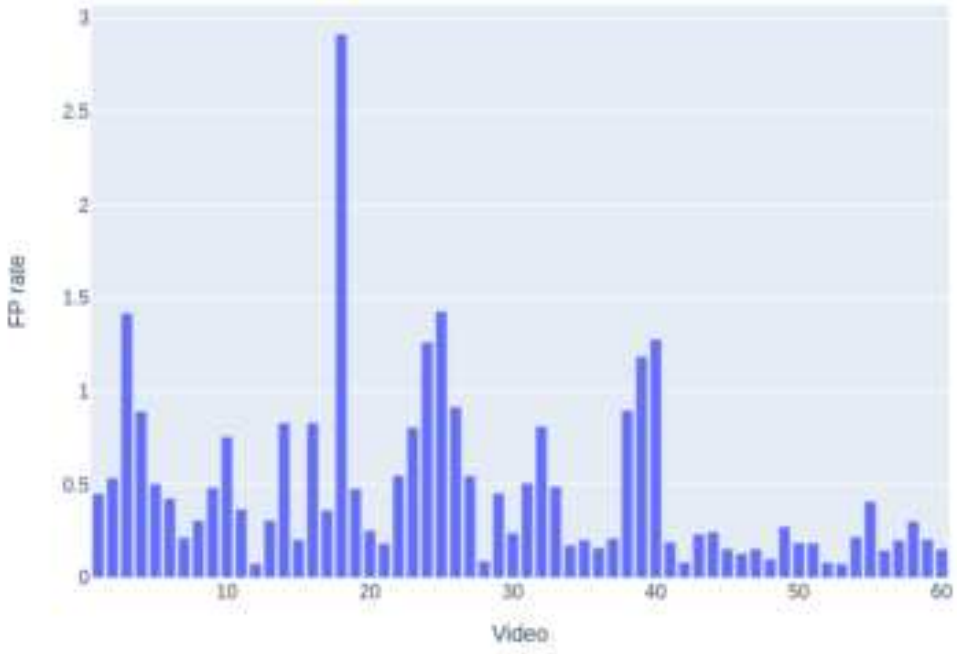


Figura 4.13: *Tasso dei FP per ogni video nel dataset UA-Detrac*

Inoltre, la Figura 4.14 illustra la distribuzione del *misses rate* per ogni video nel dataset. Ovvero, il tasso di oggetti mancati dal sistema. Mentre la Figura 4.15 mostra la distribuzione degli identity switch per ogni video nel dataset, la quale rappresenta il rapporto tra il numero di identity switch, o errori di mismatch, e le ground truth per ciascun filmato. Nel primo caso il valore medio si attesta su 0.16, mentre nel secondo caso su 0.01.

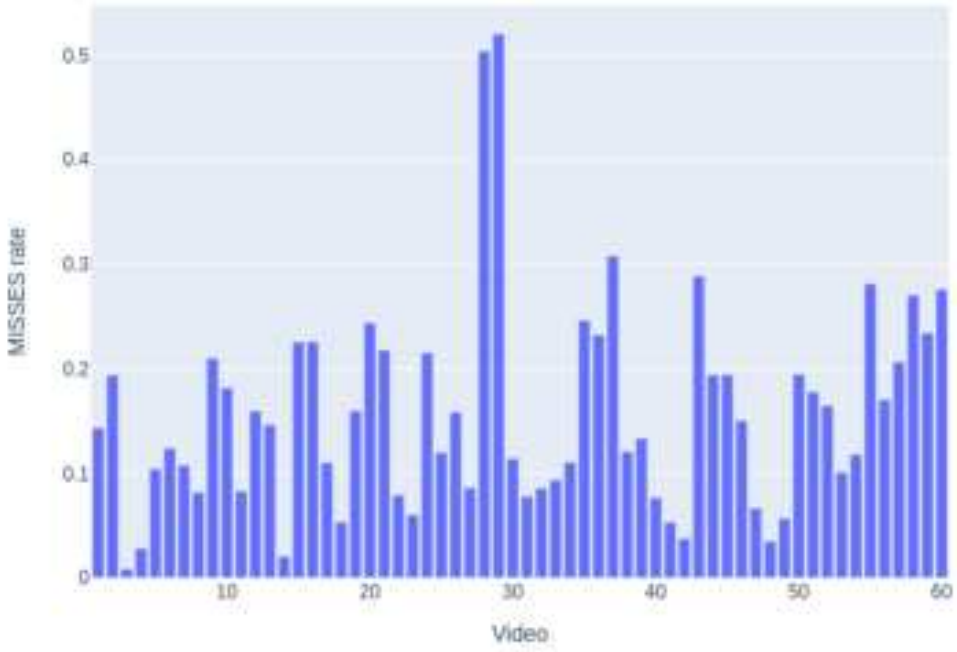


Figura 4.14: *Tasso degli oggetti mancati per ogni video nel dataset UA-Detrac*

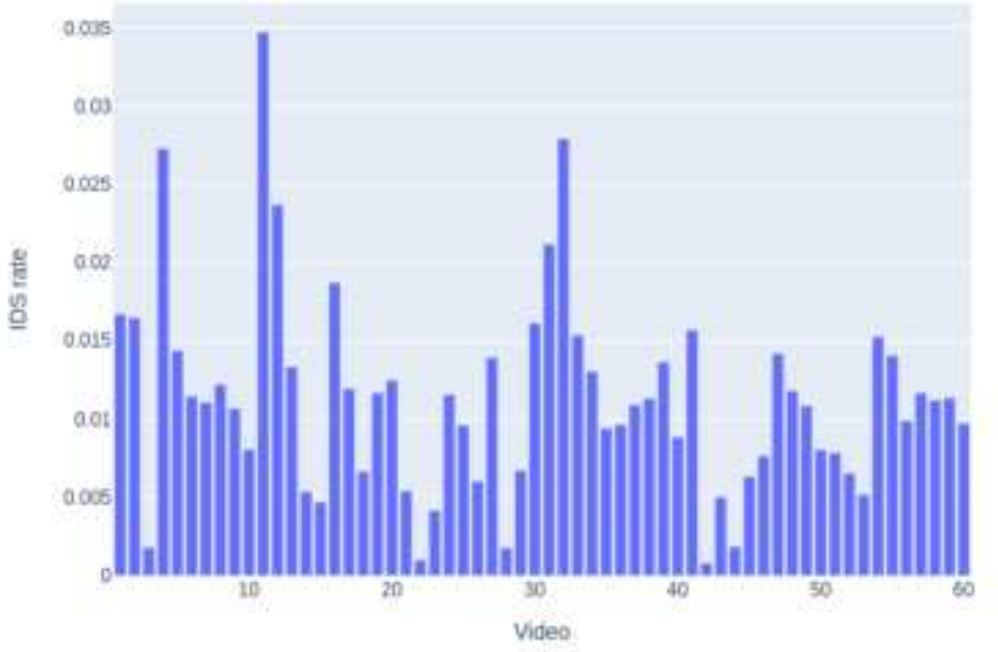


Figura 4.15: *Tasso degli IDS per ogni video nel dataset UA-Detrac*

Nella seguente tabella 4.3, riassunte le prestazioni del sistema sulla base delle metriche implementate:

Tabella 4.3: *Prestazioni del sistema su UA-Detrac*

mAP	44%
AP (car)	66%
MOTA	35%
MOTP	85%

4.3.1 Considerazioni Finali

In conclusione, l'approfondita analisi delle prestazioni del sistema di rilevamento e tracciamento oggetti, valutate attraverso metriche quali precisione, recall, mAP, e indicatori chiave come MOTA, MOTP, FPR, IDS rate e misses rate, offre una panoramica com-

pleta delle capacità del framework rispetto al dataset UA-Detrac. Nonostante alcune sfide riscontrate, evidenziate, ad esempio, nel video MVI_39811, il sistema dimostra una notevole abilità nel rilevare e tracciare oggetti. Tale capacità evidenzia una gestione efficace di scenari diversificati e situazioni complesse, come nel caso dell'analisi di immagini sfocate. In particolare, l'analisi delle curve Precision-Recall mette in luce una differenziazione delle prestazioni tra le categorie di oggetti. Mentre le metriche CLEAR, mettono in luce un'elevata precisione nella stima della posizione degli oggetti ed una buona accuratezza nell'identificazione e re-identificazione degli stessi.

Capitolo 5

Conclusioni e sviluppi futuri

5.1 Conclusioni

Durante il percorso condotto nel presente lavoro di tesi, è stato sviluppato un approccio innovativo per il tracciamento degli oggetti nell'ambiente urbano, focalizzando l'attenzione sulla creazione di una "measurable city". Attraverso l'integrazione di tecnologie avanzate di MOT e l'impiego di sensori general-purpose, sono state poste le basi per lo sviluppo di un sistema per il monitoraggio efficace del traffico in contesti urbani complessi. La ricerca si è concentrata sulle sfide del riconoscimento e tracciamento di oggetti, applicando le più recenti tecniche di apprendimento automatico per implementare un efficace sistema di MOT.

In tale scenario, il lavoro di tesi ha fornito dapprima una valutazione delle tecnologie e dei metodi coinvolti nel task di MOT in contesti urbani. L'analisi dello stato dell'arte ha evidenziato l'importanza della visione artificiale e delle reti neurali convoluzionali. Successivamente, è stato sviluppato un sistema di MOT che si distingue per la sua coerenza nel rilevamento e tracciamento degli oggetti nelle sequenze di immagini, consentendo l'estrazione e l'archiviazione di dati di alto livello, i quali possono essere arricchiti con ulteriori informazioni di contesto tramite l'applicazione di sensori e stazioni di monitoraggio aggiuntive.

A partire dal sistema sviluppato, è stato identificato un contesto di studio e generato, tramite inferenza, un dataset di riferimento utile per altri compiti, contribuendo già all'implementazione di modelli per la previsione di traiettorie dei veicoli nel traffico. Infine, è stato sviluppato un framework di valutazione, basato sul dataset di benchmark UA-Detrac, nel quale sono state individuate e implementate le metriche e le tecniche più adeguate.

Il sistema si è dimostrato consistente nel rilevamento degli oggetti e capace di garantire un'elevata precisione nella determinazione della loro posizione e una discreta accuratezza

nell'identificazione ed il tracciamento degli stessi, nonostante le sfide proprie del MOT, quali oggetti di dimensioni variabili, occlusioni e condizioni delle immagini analizzate non ottimali. Tuttavia, sono stati riscontrati problemi nel rilevamento di specifiche classi di oggetti, come pedoni, van o autocarri, sottolineando l'importanza di dataset bilanciati e rappresentativi di tutte le classi e evidenziando che le sfide legate all'identificazione di oggetti di dimensioni ridotte sono ancora una questione aperta.

In conclusione, questa ricerca ha fornito contributi innovativi quali un approccio originale e multidisciplinare, un dataset di riferimento e uno strumento di valutazione robusto per il tracciamento di oggetti in ambienti urbani. Le sfide affrontate e superate durante questo lavoro di tesi indicano chiaramente le prospettive promettenti di ulteriori sviluppi e applicazioni pratiche in contesti reali, che sono discusse di seguito.

5.2 Sviluppi futuri

Il campo delle Smart City e del monitoraggio del traffico evolve rapidamente, aprendo nuove prospettive e sfide. Di seguito vengono discusse nuove direzioni per ampliare il sistema proposto, abbracciando l'integrazione di ulteriori componenti, l'applicazione innovativa delle sue funzionalità e valutando una sua possibile distribuzione. Le prospettive che saranno brevemente presentate includono l'integrazione multisensoriale, la generazione di signature e Digital Twin e la distribuzione del sistema ai bordi della rete.

Integrazione multisensoriale

L'espansione del framework MOT verso l'integrazione di dati sensoriali diversificati emerge come una prospettiva cruciale per ampliare la comprensione dell'ambiente urbano. L'inclusione di informazioni provenienti da microfoni, sensori ambientali e di inquinamento può notevolmente arricchire i dati raccolti dal tracciamento degli oggetti. La rilevazione di suoni ambientali, la misurazione della qualità dell'aria e la rilevazione di sostanze inquinanti forniscono un quadro più completo, consentendo una gestione più accurata del contesto urbano.

Signature e Digital Twin

L'implementazione di signature, basate su approcci avanzati come quelli proposti nel paper di Laput su Synthetic Sensing [15], promette di elevare ulteriormente le capacità di un'infrastruttura di monitoraggio. La creazione di signature basate su caratteristiche uniche e l'identificazione più precisa di oggetti simili rappresentano un terreno fertile per futuri sviluppi. Inoltre, l'adozione del concetto di Digital Twin si configura come un'innovazione chiave, aprendo la strada a simulazioni dettagliate del flusso del traffico, previsioni di congestioni stradali e analisi approfondite del comportamento dei pedoni.

Distribuzione del sistema ai bordi della rete

La distribuzione del sistema ai bordi della rete, mediante l'adozione di modelli di apprendimento automatico ottimizzati per dispositivi edge come YOLO, costituisce un passo significativo verso l'efficienza del tracciamento degli oggetti in tempo reale. L'elaborazione diretta sui dispositivi edge e la riduzione della dipendenza da risorse centralizzate offrono vantaggi tangibili. La progettazione di un'infrastruttura modulare, con sensori general-purpose, dispositivi edge e l'uso del cloud per l'archiviazione e l'analisi a lungo termine, crea un ambiente flessibile e adattabile a scenari urbani dinamici.

Questi sviluppi proiettano il sistema verso un futuro in cui la gestione ed il monitoraggio dell'ambiente urbano possono migliorare notevolmente gli attuali livelli di precisione e adattabilità.

Bibliografia

[1] Narmeen Zakaria Bawany e Jawwad A. Shamsi. «Smart City Architecture: Vision and Challenges». In: *International Journal of Advanced Computer Science and Applications* 6.11 (2015). DOI: 10.14569/IJACSA.2015.061132. URL: http://dx.doi.org/10.14569/IJACSA.2015.061132.

[2] Andrea Caragliu, Chiara Del Bo e Peter Nijkamp. «Smart Cities in Europe». In: *VU University Amsterdam, Faculty of Economics, Business Administration and Econometrics, Serie Research Memoranda* 18 (gen. 2009). DOI: 10.1080/10630732.2011.601117.

[3] Andrea Zanella et al. «Internet of Things for Smart Cities». In: *IEEE Internet of Things Journal* 1.1 (2014), pp. 22–32. DOI: 10.1109/JIOT.2014.2306328.

[4] Wenjia Li, Houbing Song e Feng Zeng. «Policy-Based Secure and Trustworthy Sensing for Internet of Things in Smart Cities». In: *IEEE Internet of Things Journal* 5.2 (2018), pp. 716–723. DOI: 10.1109/JIOT.2017.2720635.

[5] T. Pflanzner, K. Zs. Leszko e A. Kertesz. «SUMMON: Gathering smart city data to support IoT-Fog-Cloud simulations». In: *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*. 2018, pp. 71–78. DOI: 10.1109/FMEC.2018.8364047.

[6] Kashif Ahmad et al. *Developing Future Human-Centered Smart Cities: Critical Analysis of Smart City Security, Interpretability, and Ethical Challenges*. 2021. arXiv: 2012.09110 [cs.CY].

[7] Arup. *If you know the right questions and understand the risks, data can help build better cities*. 2023. URL: https://www.arup.com/perspectives/if-you-know-the-right-questions-and-understand-the-risks-data-can-help-build-better-cities.

[8] Sangmin Lee et al. «Intelligent traffic control for autonomous vehicle systems based on machine learning». In: *Expert Systems with Applications* 144 (2020), p. 113074. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2019.113074. URL: https://www.sciencedirect.com/science/article/pii/S0957417419307912.

[9] Guofa Li et al. ≪A Temporal–Spatial Deep Learning Approach for Driver Distraction Detection Based on EEG Signals≫. In: *IEEE Transactions on Automation Science and Engineering* 19.4 (2022), pp. 2665–2677. DOI: 10.1109/TASE.2021.3088897.

[10] Shuai Bai et al. ≪Traffic Anomaly Detection via Perspective Map based on Spatial-temporal Information Matrix≫. In: *CVPR Workshops*. 2019. URL: https://api.semanticscholar.org/CorpusID:198167881.

[11] Kashif Ahmad e Nicola Conci. ≪How Deep Features Have Improved Event Recognition in Multimedia: A Survey≫. In: 15.2 (2019). ISSN: 1551-6857. DOI: 10.1145/3306240. URL: https://doi.org/10.1145/3306240.

[12] Kashif Ahmad et al. ≪Automatic detection of passable roads after floods in remote sensed and social media data≫. In: *Signal Processing: Image Communication* 74 (2019), pp. 110–118. ISSN: 0923-5965. DOI: https://doi.org/10.1016/j.image.2019.02.002. URL: https://www.sciencedirect.com/science/article/pii/S0923596518311536.

[13] Zakaria e A. ≪Smart City Architecture: Vision and Challenges≫. In: 11 (2015). DOI: 10.14569/ijacsa.2015.061132.

[14] Claudio Savaglio et al. ≪Opportunistic Digital Twin: An Edge Intelligence Enabler for Smart City≫. In: (2023). ISSN: 1550-4859. DOI: 10.1145/3616014. URL: https://doi.org/10.1145/3616014.

[15] Gierad Laput, Yang Zhang e Chris Harrison. ≪Synthetic Sensors: Towards General-Purpose Sensing≫. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2017). URL: https://api.semanticscholar.org/CorpusID:21299622.

[16] Li Liu et al. *Deep Learning for Generic Object Detection: A Survey*. 2019. arXiv: 1809.02165 [cs.CV].

[17] Diego M Jim'enez-Bravo et al. ≪Multi-object tracking in traffic environments: A systematic literature review≫. In: *Neurocomputing* 494 (2022), pp. 43–55.

[18] Zhengxia Zou et al. *Object Detection in 20 Years: A Survey*. 2023. arXiv: 1905.05055 [cs.CV].

[19] Yi-Qing Wang. ≪An analysis of the Viola-Jones face detection algorithm≫. In: *Image Processing On Line* 4 (2014), pp. 128–148.

[20] Navneet Dalal e Bill Triggs. ≪Histograms of oriented gradients for human detection≫. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee. 2005, pp. 886–893.

[21] Haitong Lou et al. ≪DC-YOLOv8: Small-Size Object Detection Algorithm Based on Camera Sensor≫. In: *Electronics* 12.10 (2023). ISSN: 2079-9292. DOI: 10.3390/electronics12102323. URL: https://www.mdpi.com/2079-9292/12/10/2323.

[22] Ross Girshick et al. ≪Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation≫. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 580–587.

[23] Ross Girshick. ≪Fast R-CNN≫. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448.

[24] Shaoqing Ren et al. ≪Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks≫. In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 28. 2015.

[25] Kaiming He et al. ≪Mask R-CNN≫. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 386–397.

[26] Joseph Redmon et al. ≪You Only Look Once: Unified, Real-Time Object Detection≫. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.

[27] Joseph Redmon e Ali Farhadi. ≪YOLO9000: Better, Faster, Stronger≫. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6517–6525.

[28] Alexey Bochkovskiy, Chien-Yao Wang e Hong-Yuan Mark Liao. ≪YOLOv4: Optimal Speed and Accuracy of Object Detection≫. In: *arXiv preprint arXiv:2004.10934*. 2020.

[29] Chenchen Li et al. ≪YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications≫. In: *arXiv preprint arXiv:2209.02976*. 2022.

[30] Chien-Yao Wang, Alexey Bochkovskiy e Hong-Yuan Mark Liao. ≪YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors≫. In: *arXiv preprint arXiv:2207.02696*. 2022.

[31] Wei Liu et al. ≪SSD: Single Shot MultiBox Detector≫. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Vol. 9905. 2016, pp. 21–37.

[32] Christian Szegedy et al. ≪Going deeper with convolutions≫. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[33] Ricardo Pereira et al. ≪Sort and Deep-SORT Based Multi-Object Tracking for Mobile Robotics: Evaluation with New Data Association Metrics≫. In: *Applied Sciences* 12.3 (2022). ISSN: 2076-3417. URL: https://www.mdpi.com/2076-3417/12/3/1319.

[34] Alex Bewley et al. ≪Simple Online and Realtime Tracking≫. In: *CoRR* abs/1602.00763 (2016). arXiv: 1602.00763. URL: http://arxiv.org/abs/1602.00763.

[35] Nicolai Wojke, Alex Bewley e Dietrich Paulus. ≪Simple Online and Realtime Tracking with a Deep Association Metric≫. In: *CoRR* abs/1703.07402 (2017). arXiv: 1703.07402. URL: http://arxiv.org/abs/1703.07402.

[36] Yifu Zhang et al. *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*. 2022. arXiv: 2110.06864 [cs.CV].

[37] Anton Milan et al. «MOT16: A Benchmark for Multi-Object Tracking». In: *CoRR* abs/1603.00831 (2016). arXiv: 1603.00831. URL: http://arxiv.org/abs/1603.00831.

[38] Zheng Wu et al. «A Thermal Infrared Video Benchmark for Visual Analysis». In: *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014, pp. 201–208. DOI: 10.1109/CVPRW.2014.39.

[39] Pei Sun et al. «Scalability in Perception for Autonomous Driving: Waymo Open Dataset». In: *CoRR* abs/1912.04838 (2019). arXiv: 1912.04838. URL: http://arxiv.org/abs/1912.04838.

[40] Longyin Wen et al. «DETRAC: A New Benchmark and Protocol for Multi-Object Tracking». In: *CoRR* abs/1511.04136 (2015). arXiv: 1511.04136. URL: http://arxiv.org/abs/1511.04136.

[41] Juan Terven e Diana Cordova-Esparza. *A Comprehensive Review of YOLO: From YOLOv1 and Beyond*. 2023. arXiv: 2304.00501 [cs.CV].

[42] Bo Wu e R. Nevatia. «Tracking of Multiple, Partially Occluded Humans based on Static Body Part Detection». In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 1. 2006, pp. 951–958. DOI: 10.1109/CVPR.2006.312.

[43] Rainer Stiefelhagen et al. «The CLEAR 2006 evaluation». In: vol. 4122. Apr. 2006, pp. 1–44. ISBN: 978-3-540-69567-7. DOI: 10.1007/978-3-540-69568-4_1.

[44] Jinjun Tang et al. «Towards AI-Based Traffic Counting System with Edge Computing». In: *Journal of Advanced Transportation* 2021 (2021), p. 5551976. ISSN: 0197-6729. DOI: 10.1155/2021/5551976. URL: https://doi.org/10.1155/2021/5551976.

[45] Ahmad Ammar Asyraaf Jainuddin et al. «Performance Analysis of Deep Neural Networks for Object Classification with Edge TPU». In: *2020 8th International Conference on Information Technology and Multimedia (ICIMU)* (2020), pp. 323–328. URL: https://api.semanticscholar.org/CorpusID:226854445.

[46] Adam Paszke et al. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». In: *Advances in Neural Information Processing Systems*. A cura di H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[47] Tsung-Yi Lin et al. «Microsoft COCO: Common objects in context». In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[48] *Ultralytics YOLOv8*. https://github.com/ultralytics/ultralytics?ref=blog.roboflow.com. Accessed: June 24, 2023. 2023.

[49] Yunhao Du et al. *StrongSORT: Make DeepSORT Great Again*. 2023. arXiv: 2202.13514 [cs.CV].

[50] 'Augmented Startups'. *AS-One*. https://github.com/augmentedstartups/AS-One. 2023.

[51] Tokyo Shinjuku Live Ch. *Tokyo Shinjuku Live Cam*. YouTube video. 2023. URL: https://www.youtube.com/watch?v=RQA5RcIZlAM.

[52] *Youtube-DL*. https://github.com/ytdl-org/youtube-dl.

[53] Keni Bernardin e Rainer Stiefelhagen. «Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics». In: 2008 (2008). ISSN: 1687-5176. DOI: 10.1155/2008/246309. URL: https://doi.org/10.1155/2008/246309.