



UNIVERSITÀ DEGLI STUDI DI FIRENZE  
SCUOLA DI INGEGNERIA - DIPARTIMENTO DI INGEGNERIA  
DELL'INFORMAZIONE

---

Tesi di Laurea Triennale in Ingegneria Informatica

**RICONOSCIMENTO DI OGGETTI IN IMMAGINI  
TERMICHE USANDO TECNICHE DI  
AUGMENTATION**

*Candidato*  
Borsi Leonardo

*Relatore*  
Prof. Bertini Marco

---

Anno Accademico 2023/2024

*A Marga e Augusto,  
per aver supportato ogni mia scelta,  
seguendomi in ogni strada volessi imboccare,  
per tutto ciò che mi avete trasmesso  
e che continuate ad insegnarmi giorno per giorno*

*A Martina e Niccolò,  
per avermi sempre spronato ad essere la versione migliore di me stesso,  
il fratello maggiore che vi meritate,  
e per tutti i sorrisi strappati anche nei momenti più bui*

*Ai due Luca, Benedetta e Marta,  
per essere sempre stati il mio rifugio sicuro,  
la famiglia che ti scegli e ti porti dietro per tutta la vita*

*A Lorenzo, Filippo, Luca, Niccolò e Edoardo,  
i migliori compagni d'avventura che potessi trovare,  
per tutta la spensieratezza e la serenità che mi portate ogni giorno*

*A Stefano,  
per esser stato il mio mentore  
ed aver creduto in me sin dal primo momento*

*A Laura, Fausto, Concetta e Angelo,  
per avermi reso l'uomo che sono oggi,  
con tutto l'amore che continuate a darmi,  
perfino da lassù*

# Indice

<b>Introduzione</b>	<b>i</b>
<b>1 Object Detection</b>	<b>1</b>
1.1 Il Problema dell'Object Detection . . . . .	1
1.1.1 Computer Vision . . . . .	1
1.1.2 Object Detection . . . . .	3
Feature Extraction . . . . .	4
Bounding Box Prediction e Classification . . . . .	6
Non-Maximum Suppression . . . . .	9
1.2 Modelli per l'Object Detection . . . . .	10
1.2.1 R-CNN e Varianti . . . . .	10
R-CNN . . . . .	10
Fast R-CNN . . . . .	11
Faster R-CNN . . . . .	12
1.2.2 SSD . . . . .	14
1.2.3 YOLO . . . . .	15
1.3 Applicazioni dell'Object Detection . . . . .	17
<b>2 YOLO</b>	<b>18</b>
2.1 You Only Look Once . . . . .	18

2.2	Evoluzione di YOLO . . . . .	20
2.2.1	YOLOv1 . . . . .	20
2.2.2	YOLOv2 . . . . .	22
2.2.3	YOLOv3 . . . . .	23
2.2.4	YOLOv4 . . . . .	24
2.2.5	YOLOv5 . . . . .	26
2.2.6	YOLOv6 . . . . .	27
2.2.7	YOLOv7 . . . . .	28
2.3	Modelli utilizzati . . . . .	29
2.3.1	YOLOv8 . . . . .	29
	Miglioramenti introdotti . . . . .	29
	Architettura . . . . .	31
2.3.2	YOLO-World . . . . .	33
	Open-Vocabulary Object Detection . . . . .	33
	Innovazioni introdotte . . . . .	34
	Architettura . . . . .	34
2.3.3	RT-DETR . . . . .	36
	Caratteristiche principali . . . . .	36
	Architettura . . . . .	36
<b>3</b>	<b>Data Augmentation</b>	<b>38</b>
3.1	Introduzione alla Data Augmentation . . . . .	38
3.1.1	Definizione . . . . .	38
3.1.2	Scopo . . . . .	39
3.2	Tecniche di Data Augmentation . . . . .	40
3.2.1	Tecniche Base di Data Augmentation . . . . .	40
	Trasformazioni Geometriche . . . . .	40
	Modifiche dei Colori . . . . .	41

Distorsioni e Rumore . . . . .	42
3.2.2 Tecniche Avanzate di Data Augmentation . . . . .	44
Augmentation tramite GAN . . . . .	44
Mixup, Cutout e Cutmix . . . . .	45
<b>4 Sperimentazione e Risultati</b>	<b>46</b>
4.1 Obiettivo . . . . .	46
4.2 Preparazione del dataset . . . . .	47
4.2.1 Dataset termico FLIR . . . . .	47
4.2.2 Estrazione e Formattazione dei subset . . . . .	48
4.2.3 Analisi delle classi . . . . .	50
4.2.4 Nuovo dataset filtrato . . . . .	51
4.3 Modelli di partenza . . . . .	52
4.3.1 Ultralytics . . . . .	52
4.3.2 Modelli di Ultralytics utilizzati . . . . .	52
4.3.3 Valutazione dei modelli . . . . .	53
Metriche . . . . .	53
4.3.4 Test dei modelli non addestrati . . . . .	55
4.4 Addestramento su dataset filtrato . . . . .	57
4.4.1 Addestramento dei modelli . . . . .	57
Parametri . . . . .	57
Augmentations . . . . .	58
4.4.2 Test dei modelli addestrati sul dataset filtrato . . . . .	59
4.5 Addestramento su dataset aumentato . . . . .	62
4.5.1 Albumentations . . . . .	62
4.5.2 Prima Data Augmentation . . . . .	63
Primo dataset aumentato . . . . .	63
Test modelli addestrati sul primo dataset aumentato . .	65

4.5.3	Seconda Data Augmentation . . . . .	68
	Secondo dataset aumentato . . . . .	68
	Test modelli addestrati sul secondo dataset aumentato	70
	Ulteriore addestramento sul secondo dataset aumentato	73
4.6	Modello Ottimale . . . . .	76
	<b>Conclusioni</b>	<b>78</b>
	<b>Bibliografia</b>	<b>79</b>

# Introduzione

L'Object Detection è una delle aree più dinamiche ed importanti della Computer Vision, che mira ad individuare e classificare oggetti all'interno di un'immagine. Questa tecnologia trova applicazione in diversi settori, tra cui la sorveglianza, la guida autonoma e la robotica, dove è fondamentale riconoscere oggetti e persone in tempo reale e in condizioni variabili.

In particolare, le immagini termiche offrono un vantaggio significativo in situazioni di scarsa visibilità, come di notte o in condizioni atmosferiche avverse, permettendo di rilevare oggetti sulla base del calore che emettono. Tuttavia, lavorare con immagini termiche presenta delle sfide uniche, tra cui una minore risoluzione spaziale ed un minor contrasto rispetto alle immagini a luce visibile.

Questa tesi si propone di esplorare l'efficacia di diversi modelli di Object Detection applicati a immagini termiche e di migliorare le loro prestazioni attraverso l'uso di tecniche di Data Augmentation. Il lavoro sperimentale alla base di questa ricerca si concentra su tre tipologie di modelli YOLO: YOLOv8, YOLO-World e RT-DETR.

I primi tre capitoli della tesi forniscono una base teorica essenziale per comprendere le tecniche e le metodologie utilizzate, mentre il quarto capitolo è dedicato al lavoro sperimentale svolto con l'obiettivo di identificare il modello più efficiente nel riconoscimento di oggetti in immagini termiche.

# Capitolo 1

## Object Detection

### 1.1 Il Problema dell'Object Detection

#### 1.1.1 Computer Vision

La Computer Vision è una branca dell'intelligenza artificiale che si occupa dello sviluppo di metodi che consentano ai computer di interpretare e comprendere il mondo visivo in modo simile agli esseri umani. Questa capacità è cruciale per sviluppare applicazioni avanzate che richiedono l'analisi e l'elaborazione delle immagini e dei video. La Computer Vision comprende quindi le basi teoriche, gli algoritmi e i modelli matematici necessari per acquisire, elaborare, analizzare e comprendere le immagini digitali, con l'obiettivo di estrarre informazioni significative dall'ambiente visivo.

Tra i problemi centrali di cui si occupa la Computer Vision, troviamo:

- **Image Recognition:** Identificare e classificare oggetti o scene all'interno di un'immagine. Questo include compiti come il riconoscimento facciale, il riconoscimento di oggetti e il riconoscimento di azioni.

- **Object Detection:** Localizzare e identificare oggetti di interesse all'interno di un'immagine, determinando la loro posizione attraverso bounding box. Questo è fondamentale per applicazioni come il tracciamento degli oggetti, la guida autonoma e la sorveglianza.
- **Semantic Segmentation:** Dividere un'immagine in segmenti che corrispondono a oggetti o regioni con caratteristiche simili. Questo aiuta a comprendere la struttura dell'immagine a livello di pixel.
- **Scene Recognition:** Interpretare e descrivere l'intera scena rappresentata in un'immagine, inclusi gli oggetti presenti e le loro relazioni spaziali.
- **Pose Estimation:** Determinare la posizione e l'orientamento di un oggetto nello spazio tridimensionale a partire da immagini bidimensionali.
- **3D Reconstruction:** Ricostruire modelli tridimensionali di oggetti o scene a partire da immagini bidimensionali. Questo è cruciale per applicazioni come la realtà aumentata e la robotica.
- **Object Tracking:** Seguire il movimento di uno o più oggetti attraverso una sequenza di immagini o video. Questo è essenziale per applicazioni come la sorveglianza video e la guida autonoma.
- **Super-Resolution:** Migliorare la risoluzione di un'immagine partendo da versioni a bassa risoluzione, utile per migliorare la qualità visiva e per analisi più dettagliate.

Questi problemi sono alla base di molte applicazioni pratiche della Computer Vision e sono spesso interconnessi.

### 1.1.2 Object Detection

L'Object Detection è quindi uno dei problemi fondamentali della Computer Vision. Esso riguarda la localizzazione e identificazione di uno o più oggetti di interesse all'interno di un'immagine o di un video. A differenza dell'Image Recognition, che si limita a classificare un'intera immagine, l'Object Detection deve determinare sia la presenza degli oggetti sia le loro posizioni precise.

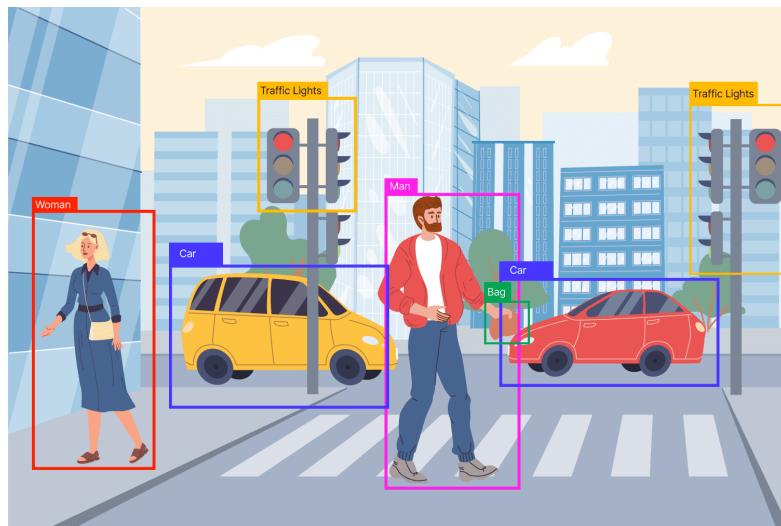


Figura 1.1: Esempio di Object Detection di un'immagine [4]

L'Object Detection ha due obiettivi principali:

- **Localizzazione:** Determinare la posizione degli oggetti all'interno dell'immagine, solitamente rappresentata da bounding box (rettangoli che contornano gli oggetti).
- **Classificazione:** Identificare la classe a cui appartiene ogni oggetto rilevato (ad esempio persona, automobile, animale, ecc...).

Questi obiettivi richiedono algoritmi capaci di elaborare immagini complesse e di generare previsioni accurate e efficienti in termini di calcolo.

L'Object Detection è pertanto un processo complesso che combina diverse fasi, ognuna delle quali gioca un ruolo cruciale nel garantire l'accuratezza e l'efficienza del sistema complessivo.

I principali passaggi coinvolti nell'Object Detection sono:

- **Feature Extraction**
- **Bounding Box Prediction**
- **Classification**
- **Non-Maximum Suppression (NMS)**

### Feature Extraction

La fase di Feature Extraction è cruciale nell'Object Detection, in quanto le caratteristiche estratte rappresentano la base su cui si fondano le successive fasi di localizzazione e classificazione. Questa fase coinvolge l'uso di reti neurali convoluzionali (CNN) per identificare caratteristiche rilevanti delle immagini, come bordi, texture, forme e colori.

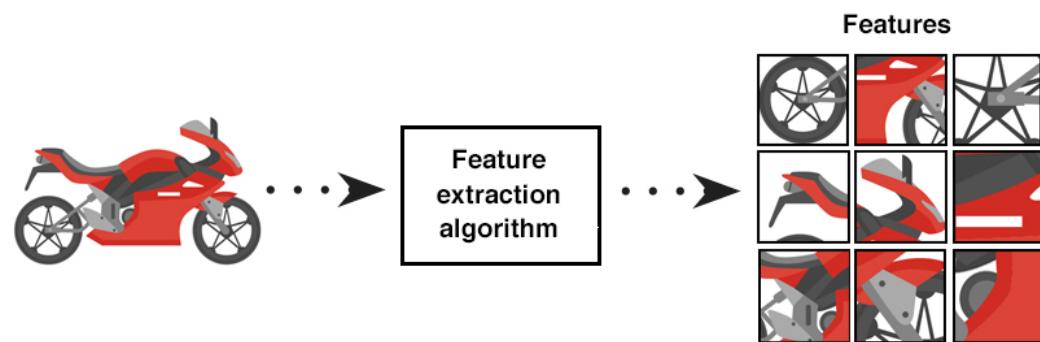


Figura 1.2: Esempio di Feature Extraction di un'immagine [11]

Le CNN sono una classe di reti neurali composte da vari layer di diversi tipi, ciascuno con una funzione specifica:

- **Convolutional Layers:** estraggono caratteristiche locali come bordi, angoli e texture tramite l'applicazione di filtri (kernel) sull'immagine di input. Ogni kernel convoluziona l'immagine generando una feature map che evidenzia la presenza di specifici pattern.
- **Activation Layers:** applicano una funzione di attivazione non lineare (ad esempio, ReLU - Rectified Linear Unit) dopo ogni convolutional layer, la quale introduce non-linearità nel modello, permettendo alla rete di apprendere rappresentazioni più complesse.
- **Pooling Layers:** riducono le dimensioni spaziali delle feature map (tipicamente mediante operazioni di max pooling o average pooling), riducendo così il numero di parametri e computazioni nella rete, e introducendo invarianza rispetto alle traslazioni. Questo aiuta a rendere il modello più robusto e a prevenire l'overfitting.

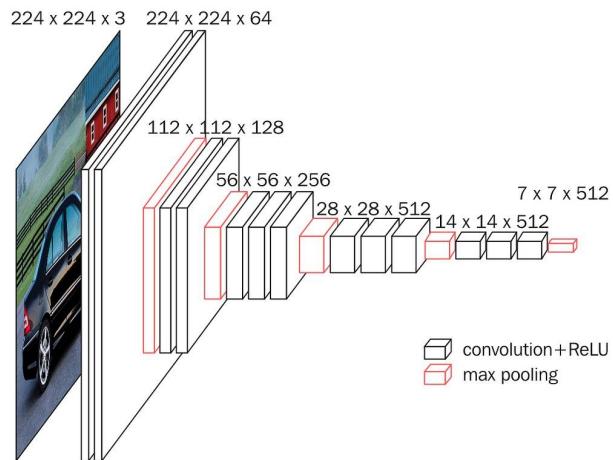


Figura 1.3: Layers della CNN adibiti alla Feature Extraction [18]

Quando un’immagine viene data in input ad una CNN, questa passa attraverso la serie di convolutional e pooling layers. I primi convolutional layers tendono a rilevare caratteristiche di basso livello, come bordi e texture, mentre i successivi catturano caratteristiche di livello più alto, come parti di oggetti e forme complete. L’uso di multipli convolutional layers permette quindi alla rete di costruire una rappresentazione gerarchica delle caratteristiche dell’immagine.

Il risultato finale della fase di Feature Extraction è una serie di feature maps che catturano informazioni spaziali e di contesto sull’immagine. Queste feature maps sono poi utilizzate nei passaggi successivi per la predizione delle bounding box e la classificazione degli oggetti.

### Bounding Box Prediction e Classification

La predizione delle bounding box e la classificazione degli oggetti sono due passaggi cruciali nell’Object Detection. A seconda dell’architettura del modello, questi passaggi possono essere eseguiti separatamente o simultaneamente.

Nei modelli basati su region proposals, come R-CNN e le sue varianti, la predizione delle bounding box e la classificazione avvengono in due passaggi distinti:

1. **Generazione delle Proposal Regions:** in questa fase, il modello utilizza algoritmi come il Selective Search o una Region Proposal Network (RPN) per generare un insieme di region proposals, che sono regioni dell’immagine candidate a contenere oggetti. Il Selective Search utilizza tecniche di segmentazione per trovare regioni simili, mentre l’RPN è una rete neurale che produce direttamente proposte di regioni.

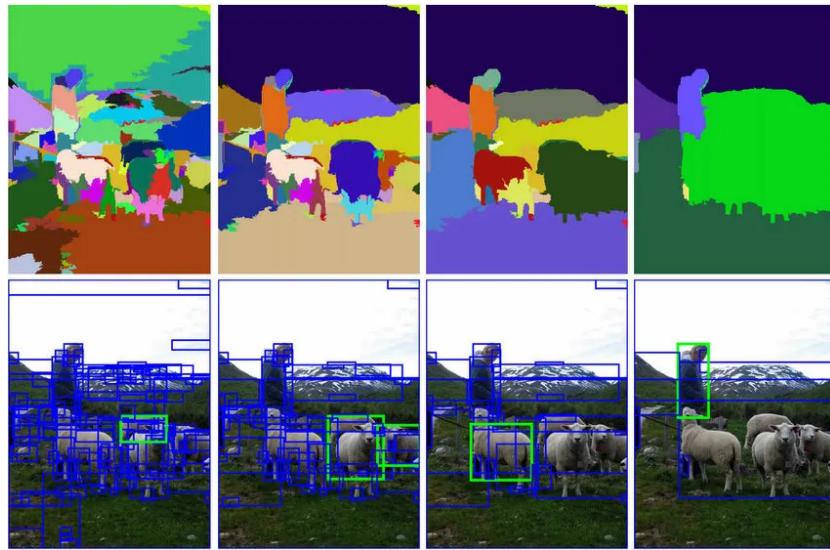


Figura 1.4: Esempio di generazione delle Proposal Regions con l'algoritmo Selective Search [16]

**2. Predizione delle Bounding Box e Classificazione:** ogni region proposal viene estratta dalle feature maps e passata attraverso una CNN che effettua la predizione delle bounding box precise e la classificazione degli oggetti. In R-CNN, questo passaggio viene eseguito separatamente per ciascuna region proposal, mentre in varianti come Fast R-CNN e Faster R-CNN, queste operazioni vengono ottimizzate per migliorare la velocità e l'efficienza.

Nei modelli single-shot, come SSD e YOLO, la predizione delle bounding box e la classificazione degli oggetti avvengono simultaneamente in un unico passaggio.

Questo approccio è caratterizzato dai seguenti passaggi chiave:

**1. Suddivisione in Griglia:** dopo la feature extraction, l'immagine di input viene suddivisa in una griglia di dimensioni predefinite, ad esempio 7x7 o 13x13, a seconda delle specifiche del modello.

**2. Predizione delle Bounding Box:** per ogni cella della griglia, il modello utilizza le feature maps per predirre un insieme di bounding box, ciascuna definita da:

- Coordinate (x, y) del centro della bounding box rispetto alla cella della griglia
- Altezza e larghezza (h, w) della bounding box, normalizzate rispetto alle dimensioni dell'immagine
- Un punteggio di confidenza (objectness score) che riflette la probabilità che la bounding box contenga un oggetto e la precisione con cui la bounding box delimita l'oggetto

**3. Classificazione:** oltre alle coordinate e all'objectness score, per ogni bounding box viene predetta una distribuzione di probabilità sulle classi degli oggetti. Il modello quindi stima la probabilità che l'oggetto contenuto nella bounding box appartenga a ciascuna delle classi target.

**4. Calcolo del Punteggio Finale:** l'objectness score per ogni bounding box viene moltiplicato per le probabilità di classe per ottenere un punteggio finale per ogni combinazione di bounding box e classe. Questo punteggio finale aiuta a determinare quali bounding box e classi saranno considerate nella fase finale di rilevamento.

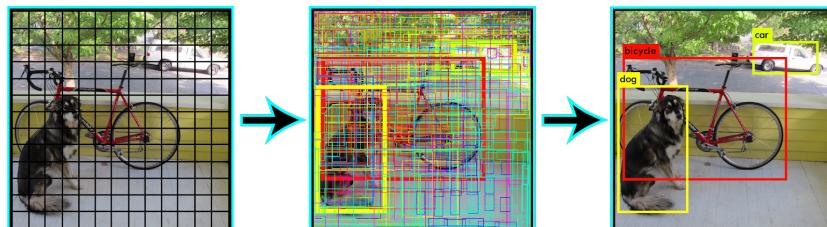


Figura 1.5: Bounding Box Prediction e Classification dei modelli single-shot [14]

## Non-Maximum Suppression

La Non-Maximum Suppression (NMS) è una tecnica essenziale utilizzata nei modelli di Object Detection per ridurre le proposte ridondanti di bounding box o regioni che coprono gli oggetti rilevati. Questa viene applicata dopo la fase di generazione delle proposte (bounding box o regioni), le quali vengono valutate e ordinate in base a un punteggio di confidenza, che rappresenta la probabilità che una proposta contenga un oggetto.

L’NMS identifica e rimuove le proposte che hanno un’elevata sovrapposizione con altre proposte di punteggio più alto, mantenendo solo quelle più accurate e significative.

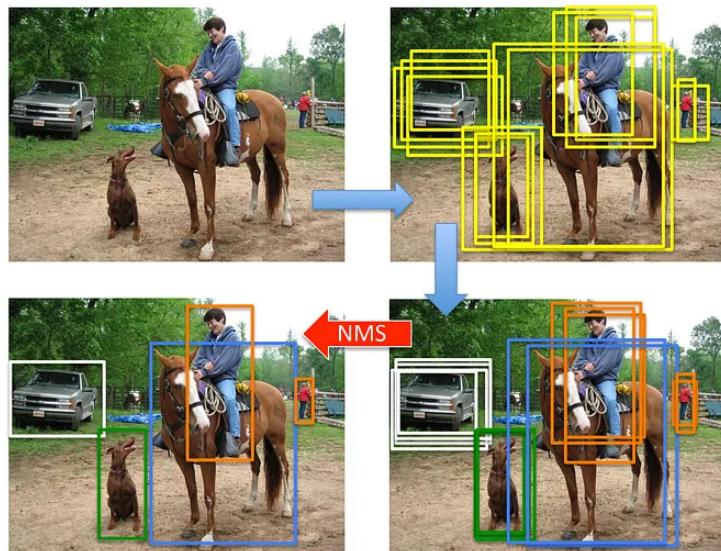


Figura 1.6: Esempio di rimozione delle bounding box ridondanti tramite Non-Maximum Suppression [14]

## 1.2 Modelli per l'Object Detection

Nel campo dell'Object Detection, vari modelli hanno rivoluzionato il rilevamento degli oggetti nelle immagini, contribuendo significativamente alla precisione e all'efficienza di questa tecnologia fondamentale della Computer Vision. Ogni modello presenta un approccio distintivo per affrontare le sfide di localizzazione e classificazione degli oggetti.

### 1.2.1 R-CNN e Varianti

#### R-CNN

R-CNN (Region-based Convolutional Neural Network) è stato uno dei primi modelli ad adottare un approccio basato sulle region proposals per l'Object Detection. Proposto da Ross Girshick, Jeff Donahue, Trevor Darrell e Jitendra Malik nel 2014 [33], ha segnato un punto di svolta nell'utilizzo delle reti neurali convoluzionali per il rilevamento degli oggetti. Prima dell'introduzione di R-CNN, i metodi tradizionali per l'Object Detection erano basati su tecniche di estrazione di features manuali e classificatori separati. Questi approcci erano spesso limitati dalla necessità di progettare manualmente le features e dalla loro mancanza di robustezza e generalizzazione.

R-CNN si distingue per il suo approccio in tre fasi visto precedentemente:

1. **Generazione delle Proposal Regions:** genera un insieme di proposal regions candidate tramite algoritmi come il Selective Search, le quali vengono poi selezionate in base a caratteristiche di basso livello come colore, texture e forma.
2. **Feature Extraction:** ogni proposal region viene trasformata in una dimensione fissa e passata attraverso una rete convoluzionale pre-addestrata,

come AlexNet o VGG-16. Questo processo genera una feature map per ciascuna region proposal, catturando le informazioni rilevanti dell'area proposta.

**3. Classificazione e Ottimizzazione delle Bounding Box:** le feature estratte da ciascuna region proposal vengono fornite a un classificatore SVM per determinare la presenza di oggetti e per la classificazione, mentre un regressore lineare ottimizza le coordinate della bounding box prevista per migliorare la precisione della localizzazione dell'oggetto.

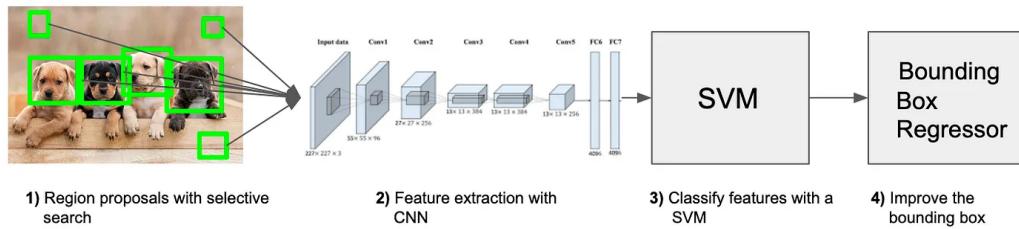


Figura 1.7: Architettura di R-CNN [34]

## Fast R-CNN

Fast R-CNN, proposto da Ross Girshick nel 2015 [24], migliora l'efficienza di R-CNN introducendo le seguenti migliorie:

- **Utilizzo di una sola CNN:** integra l'intero processo di rilevamento degli oggetti in una singola rete, combinando l'estrazione di features e la generazione di region proposals

- **RoI Pooling:** introduce la tecnica di RoI (Region of Interest) pooling per estrarre features dalle feature map generate, gestendo efficacemente regioni di dimensioni variabili.

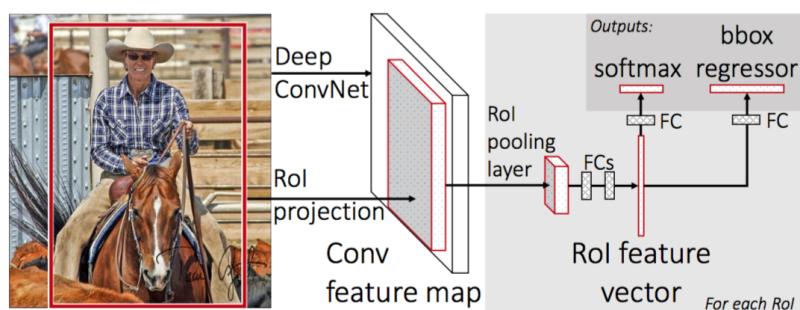


Figura 1.8: Architettura di Fast R-CNN [22]

## Faster R-CNN

Faster R-CNN, proposto da Shaoqing Ren, Kaiming He, Ross Girshick e Jian Sun nel 2016 [36], rappresenta un ulteriore miglioramento rispetto a Fast R-CNN, introducendo un approccio più efficiente per la generazione delle region proposals. Principali migliorie:

- **Region Proposal Network (RPN):** introduce una RPN che opera condividendo la feature map con la rete principale, permettendo la generazione di proposte di regioni in modo efficiente e integrato.
- **Architettura two-stage:** divide il processo di rilevamento degli oggetti in due fasi: prima, il RPN propone regioni d'interesse basate sulla

feature map condivisa; successivamente, queste proposte vengono elaborate da un classificatore e un regressore per la classificazione degli oggetti e l'ottimizzazione delle bounding box.

- **RoI Pooling migliorato:** implementa un RoI pooling migliorato all'interno dell'RPN e nella fase successiva di elaborazione delle proposte. Questo approccio ottimizza la gestione delle regioni di interesse, migliorando la precisione complessiva del rilevamento degli oggetti.

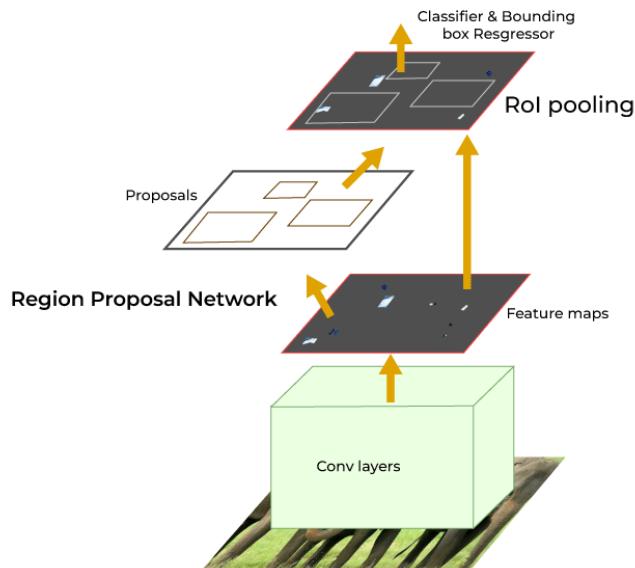


Figura 1.9: Architettura di Faster R-CNN [22]

### 1.2.2 SSD

Il modello SSD (Single Shot MultiBox Detector) è stato un importante avanzamento nell’ambito dell’Object Detection. Proposto da Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu e Alexander C. Berg nel 2016 [42], ha introdotto un approccio single-shot per il rilevamento degli oggetti, combinando predizione delle bounding box e classificazione degli oggetti in un unico passaggio. Prima dell’SSD, i modelli two-stage come R-CNN e le sue varianti dominavano il campo, con una fase separata di generazione delle region proposals seguita dalla classificazione. Questi approcci, sebbene accurati, erano spesso limitati dalla complessità computazionale e dalla lentezza del processo. Eliminando questa fase separata adottata dai modelli two-stage, rende il processo di rilevamento più rapido ed efficiente. Questa caratteristica ha reso SSD ideale per applicazioni che richiedono un’elaborazione veloce delle immagini, come la videosorveglianza e i sistemi di guida autonoma.

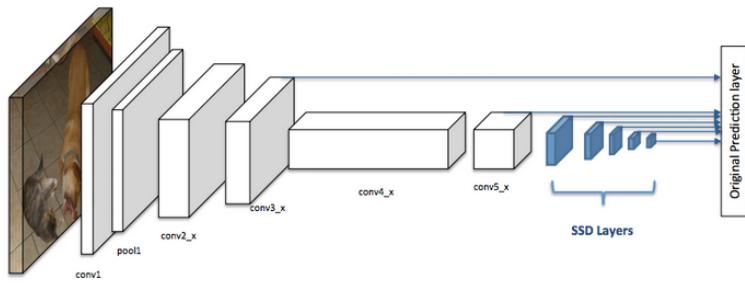


Figura 1.10: Architettura di SSD [6]

Un’altra innovazione chiave di SSD è l’utilizzo delle **anchor box**: bounding box con dimensioni predefinite disposte su una griglia che suddivide l’immagine. Durante l’addestramento, SSD predice l’offset rispetto a ciascuna anchor box per adattare la posizione e le dimensioni della bounding box alle caratteristiche specifiche dell’oggetto da rilevare.

Questo approccio consente a SSD di individuare oggetti di diverse dimensioni e forme con maggiore accuratezza e robustezza.

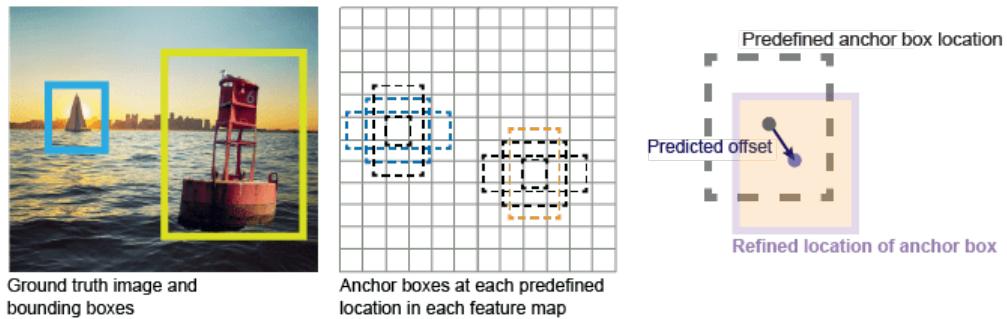


Figura 1.11: Utilizzo delle anchor box nell'SSD [5]

SSD rappresenta quindi un passo significativo verso modelli più efficienti e versatili per l'Object Detection, combinando alta precisione con prestazioni ottimali in tempo reale.

### 1.2.3 YOLO

Il modello YOLO (You Only Look Once) rappresenta un'altra pietra miliare nell'ambito dell'Object Detection, affiancandosi all'SSD come uno dei primi modelli single-shot a rivoluzionare il campo. Proposto originariamente da Joseph Redmon, Santosh Divvala, Ross Girshick e Ali Farhadi nel 2016 [27], YOLO segue l'approccio innovativo che combina la predizione delle bounding box e la classificazione degli oggetti in un singolo passaggio. Come nell'SSD, l'immagine viene suddivisa in una griglia e vengono predette bounding box e probabilità di classe direttamente per ciascuna cella della griglia.

YOLO si distingue per la sua capacità di considerare il contesto globale dell'immagine durante il processo di rilevamento. Ogni cella della griglia non solo predice le bounding box ma valuta anche la confidenza di ciascuna predi-

zione in relazione a tutte le altre predizioni nell'immagine. Questo approccio globale contribuisce a migliorare la coerenza e la precisione complessiva del modello.

Inoltre, YOLO utilizza un'unica CNN per l'intero processo di rilevamento degli oggetti, semplificando ulteriormente l'architettura e migliorando l'efficienza computazionale. Questa caratteristica lo rende particolarmente adatto per dispositivi con risorse limitate e per applicazioni embedded.

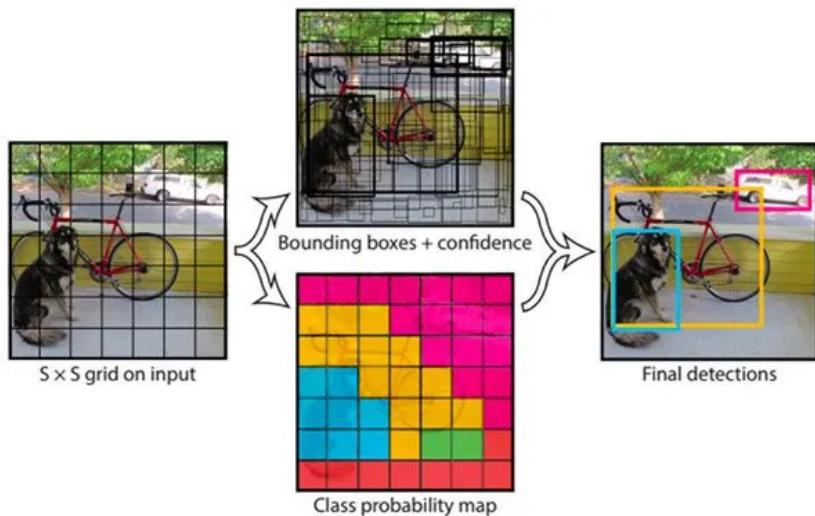


Figura 1.12: Object Detection tramite YOLO [38]

YOLO ha trovato ampio impiego in una vasta gamma di applicazioni, dalla videosorveglianza alla guida autonoma, grazie alla sua velocità, precisione e capacità di gestire oggetti di diverse dimensioni e forme all'interno di un'immagine.

### 1.3 Applicazioni dell'Object Detection

L'object detection è una tecnologia fondamentale con numerose applicazioni pratiche in vari settori, quali:

1. **Sicurezza e Sorveglianza:** utilizzata per monitorare ambienti e individuare attività sospette. Esempi includono il rilevamento di persone, veicoli o oggetti non autorizzati in aree sensibili come aeroporti, banche e stazioni ferroviarie.
2. **Automazione Industriale:** usata per il controllo di qualità, l'ispezione automatizzata dei prodotti e la gestione degli inventari.
3. **Guida Autonoma e Veicoli Intelligenti:** consente ai veicoli di rilevare e reagire in tempo reale a pedoni, veicoli, segnaletica stradale e altri ostacoli.
4. **Sanità e Medicina:** supporta la diagnostica medica automatizzata, il monitoraggio dei pazienti e la ricerca scientifica.
5. **Commercio e Marketing:** impiegata per migliorare l'esperienza del cliente, ottimizzare la gestione del magazzino e analizzare il comportamento dei consumatori, tramite l'analisi delle immagini e dei video permette di monitorare gli scaffali nei negozi e di rilevare automaticamente il movimento dei prodotti.
6. **Sicurezza Stradale e Controllo del Traffico:** fondamentale per il miglioramento della sicurezza stradale e l'ottimizzazione del controllo del traffico urbano, grazie a sistemi avanzati che possono rilevare incidenti, monitorare il flusso veicolare, riconoscere targhe di registrazione e applicare le normative stradali.

# Capitolo 2

## YOLO

### 2.1 You Only Look Once

Il nome "You Only Look Once" deriva dalla filosia di design che sta alla base del modello ideato da Joseph Redmon, Santosh Divvala, Ross Girshick e Ali Farhadi nel 2016 [27]: l'immagine deve essere elaborata una sola volta per effettuare sia la predizione delle bounding box che la classificazione degli oggetti. A differenza dei modelli two-stage, che richiedono un primo passaggio per generare le region proposals e un secondo passaggio per classificarle, YOLO effettua tutto in un unico passo. Questo approccio consente di ridurre drasticamente il tempo di elaborazione e semplifica l'architettura complessiva del modello.

Esistono quindi alcuni principi chiave che hanno influenzato il design e lo sviluppo del modello:

- **Approccio Globale all'Immagine:** A differenza dei modelli precedenti che esaminano porzioni dell'immagine in sequenza, YOLO considera l'intera immagine in un singolo passaggio. Questo consente al mo-

dello di avere una visione globale del contesto, riducendo la probabilità di predizioni incoerenti tra le diverse regioni dell'immagine.

- **Efficienza Computazionale:** YOLO è progettato per essere estremamente veloce. L'approccio single-shot consente di ridurre drasticamente il tempo di elaborazione, rendendolo adatto per applicazioni in tempo reale come la videosorveglianza, la guida autonoma e altre situazioni dove la velocità è critica.
- **Semplicità e Generalizzazione:** La semplicità dell'architettura di YOLO non solo facilita l'implementazione, ma migliora anche la capacità del modello di generalizzare su diverse tipologie di immagini e scenari. Utilizzando un'unica rete neurale convoluzionale per predirne sia le bounding box che le probabilità di classe, YOLO riesce a mantenere un equilibrio tra complessità e prestazioni.

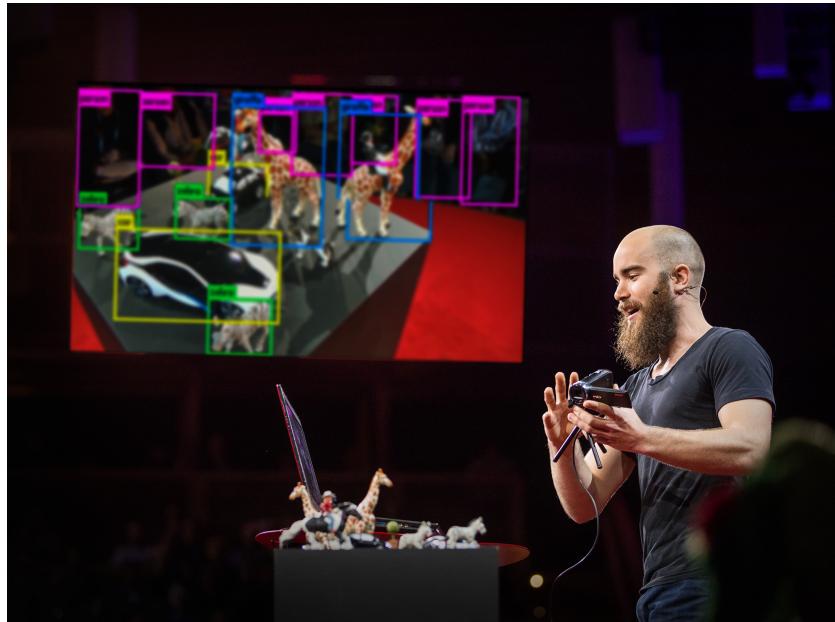


Figura 2.1: Joseph Redmon durante un TED Talk su YOLO nel 2017 [31]

## 2.2 Evoluzione di YOLO

YOLO ha subito un’evoluzione significativa nel corso degli anni, con ogni nuova versione che ha introdotto importanti miglioramenti in termini di precisione, velocità e capacità di generalizzazione.

### 2.2.1 YOLOv1

La prima versione di YOLO, proposta nel 2016 [27], è composta da una singola rete neurale convoluzionale che processa l’intera immagine di input in un singolo passaggio. Nella rete troviamo le seguenti tipologie di layers:

- **Convolutional Layers:** Applicano convoluzioni con filtri di diverse dimensioni per estrarre feature di alto livello dall’immagine di input. Questi strati sono responsabili della rilevazione dei pattern, come bordi, texture e forme complesse.
- **Max-Pooling Layers:** Ridimensionano le feature map riducendo la risoluzione spaziale, mantenendo le feature più importanti e riducendo la quantità di parametri e il costo computazionale. Max-pooling seleziona il valore massimo in ogni finestra del filtro, riducendo così l’informazione spaziale ma conservando le feature dominanti.
- **Fully Connected (FC) Layers:** Connettono ogni unità di un layer a tutte le unità del layer successivo, combinando le feature estratte per produrre le predizioni finali. Questi strati sono utilizzati per la classificazione finale e la regressione delle bounding box.

L’architettura di YOLOv1 si ispira a GoogLeNet (rete nota per i suoi moduli inception che permettono di utilizzare filtri di diverse dimensioni in parallelo) e utilizza 24 convolutional layers per estrarre feature rilevanti a vari

livelli di astrazione, alternati a max-pooling layers per ridurre la dimensione spaziale delle feature maps, mantenendo al contempo le informazioni più importanti. Successivamente, 2 FC layers combinano queste feature estratte per predire le bounding box e le probabilità di classe.

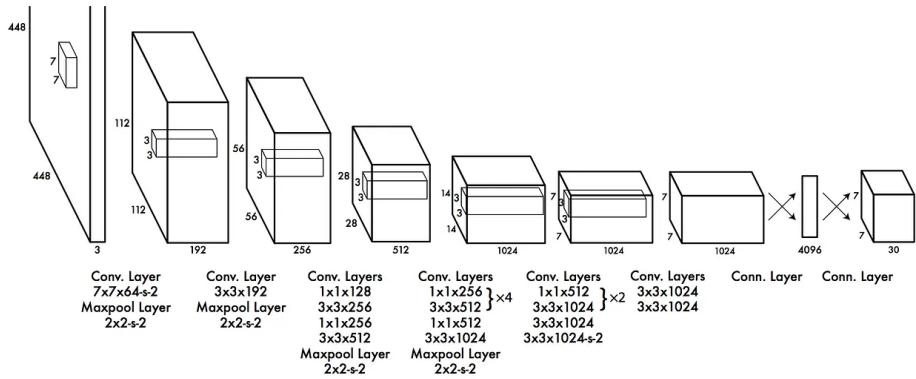


Figura 2.2: Architettura di YOLOv1 [17]

L'output finale della rete è un tensore di dimensione

$$S * S * (B * 5 + C)$$

dove S è la dimensione della griglia, B è il numero di bounding box predette per cella, e C è il numero di classi. Questo tensore contiene tutte le predizioni delle bounding box, i punteggi di confidenza relativi a ciascuna predizione e le probabilità di classe per ogni oggetto rilevato.

### 2.2.2 YOLOv2

YOLOv2, anche conosciuto come YOLO9000, è stato introdotto nel 2016 [25]; questa versione è stata progettata per essere più veloce e precisa, capace di rilevare una gamma più ampia di classi di oggetti.

Le caratteristiche principali includono:

- **Backbone CNN Darknet-19:** YOLOv2 utilizza Darknet-19 come Backbone, una variante dell'architettura VGGNet, con strati di convoluzione progressiva e max-pooling.
- **Anchor Boxes:** introduce le Anchor Boxes, che sono bounding boxes predefinite con diverse proporzioni e scale., le quali permettono al modello di gestire meglio oggetti di varie dimensioni e proporzioni all'interno di un'immagine.
- **Batch Normalization:** l'introduzione della normalizzazione batch standardizza le attivazioni di ogni layer, riducendo il rischio di overfitting e accelerando la convergenza durante l'addestramento. Questo processo aiuta a mantenere l'output di ogni layer stabile e ben bilanciato, facilitando l'apprendimento di feature utili.
- **Strategia di Training Multi-Scale:** viene adottata una strategia di training multi-scala, che consiste nell'addestrare il modello su immagini di diverse scale e risoluzioni. Questa tecnica permette al modello di essere robusto e flessibile, adattandosi meglio a variazioni nelle dimensioni degli oggetti e migliorando la capacità di generalizzazione.
- **Nuova Loss Function:** introdotta una nuova loss function progettata specificamente per l'object detection, la quale è basata sugli errori quadratici tra le bounding box predette e quelle reali, e tiene conto anche

delle probabilità di classe. La nuova loss function aiuta il modello a migliorare la precisione delle predizioni, penalizzando in modo appropriato le deviazioni tra le bounding box previste e quelle effettive, e ottimizzando la classificazione degli oggetti rilevati.

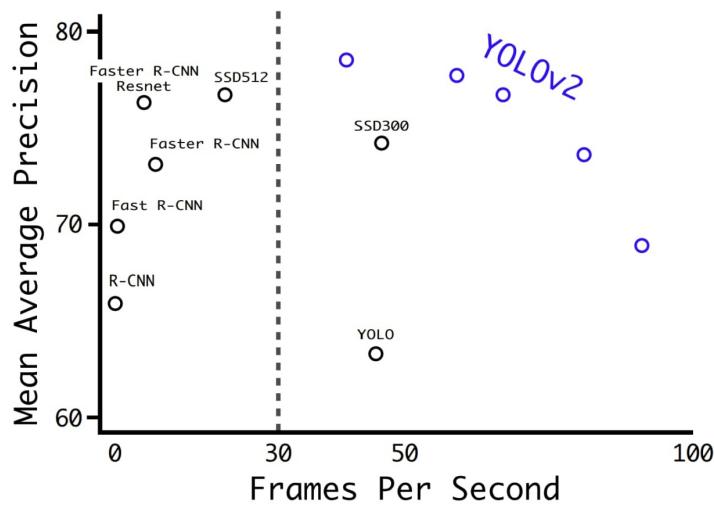


Figura 2.3: Confronto della mAP di YOLOv2 con quelle di altri modelli [28]

### 2.2.3 YOLOv3

YOLOv3 è la terza versione dell'algoritmo YOLO, introdotta nel 2018 [26] con l'obiettivo di aumentare l'accuratezza e la velocità dell'algoritmo. Caratteristiche chiave di YOLOv3 sono:

- **Architettura CNN Darknet-53:** Darknet-53 è una variante dell'architettura ResNet, progettata specificamente per le attività di object detection con 53 convolutional layers.
- **Anchor Boxes migliori:** Implementa anchor boxes scalate e con diversi aspect ratio per adattarsi meglio alle dimensioni e alle forme degli oggetti da rilevare.

- **Feature Pyramid Networks:** Introduce le FPN, una architettura CNN utilizzata per rilevare oggetti a diverse scale, migliorando le prestazioni di detection sugli oggetti di piccole dimensioni.

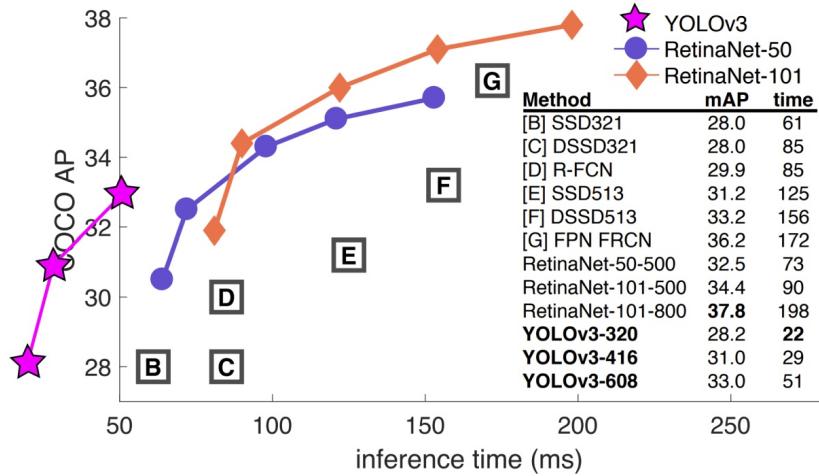


Figura 2.4: mAP ottenuta su dataset COCO con YOLOv3 [28]

## 2.2.4 YOLOv4

YOLOv4 viene rilasciata nel 2020 da Bochkovskiy [15] e presenta le seguenti novità:

- **Architettura CSPNet:** adotta una nuova architettura CNN chiamata CSPNet (Cross Stage Partial Network), una variante della ResNet progettata specificamente per task di object detection.
- **Anchor Box con k-means clustering:** introdotto un nuovo metodo di generazione per le anchor box che sfrutta un algoritmo di clustering per raggruppare le bounding box del ground truth in cluster, utilizzando i centroidi di questi cluster come anchor box; questo permette alle

anchor box di essere più precise rispetto alle dimensioni e alla forma degli oggetti rilevati.

- **GHM Loss:** introdotta una variante della Focal Loss, progettata per migliorare le prestazioni del modello su dataset con distribuzione non uniforme delle classi.
- **Miglioramenti alle FPN:** revisione che migliora ulteriormente la capacità del modello di rilevare oggetti di piccole dimensioni, in modo da operare efficacemente su scale diverse all'interno delle immagini.

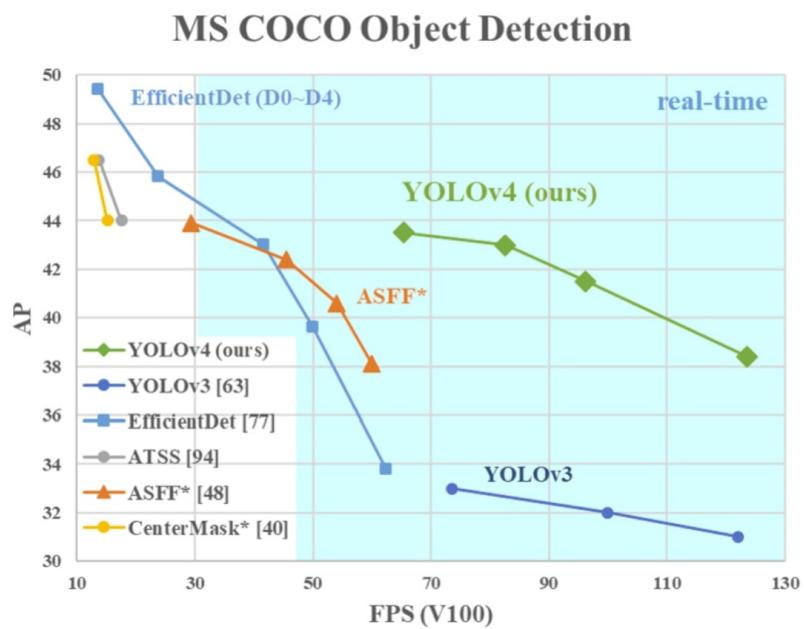


Figura 2.5: mAP di YOLOv4 confrontata con quelle di modelli precedenti [28]

### 2.2.5 YOLOv5

Introdotta nel 2020 dallo stesso team che ha sviluppato l'algoritmo YOLO originale, YOLOv5 è un progetto open-source mantenuto da Ultralytics [2]. Introduce diverse nuove funzionalità e miglioramenti tra cui:

- **Architettura EfficientDet:** utilizza una architettura più complessa chiamata EfficientDet, basata sull'architettura di rete EfficientNet, la quale consente a YOLOv5 di raggiungere una maggiore accuratezza e una migliore generalizzazione su un'ampia gamma di categorie di oggetti rispetto alle versioni precedenti.
- **Addestramento su Dataset Diversificato (D5):** a differenza di YOLO, che è stato addestrato sul dataset PASCAL VOC composto da 20 categorie di oggetti, YOLOv5 è addestrato su un dataset più ampio e diversificato chiamato D5, che include un totale di 600 categorie di oggetti.
- **Spatial Pyramid Pooling (SPP):** introduce un nuovo tipo di layer di pooling che riduce la risoluzione spaziale delle feature maps, il quale è particolarmente efficace nel migliorare le prestazioni di rilevamento sugli oggetti di piccole dimensioni, consentendo al modello di analizzare gli oggetti a diverse scale.
- **Loss Function CIoU:** introduce una nuova loss function chiamata "CIoU loss", variante della loss function IoU (Intersection over Union). La CIoU loss è specificamente progettata per migliorare le prestazioni del modello su dataset sbilanciati, contribuendo a migliorare l'accuracy e la stabilità durante l'addestramento.

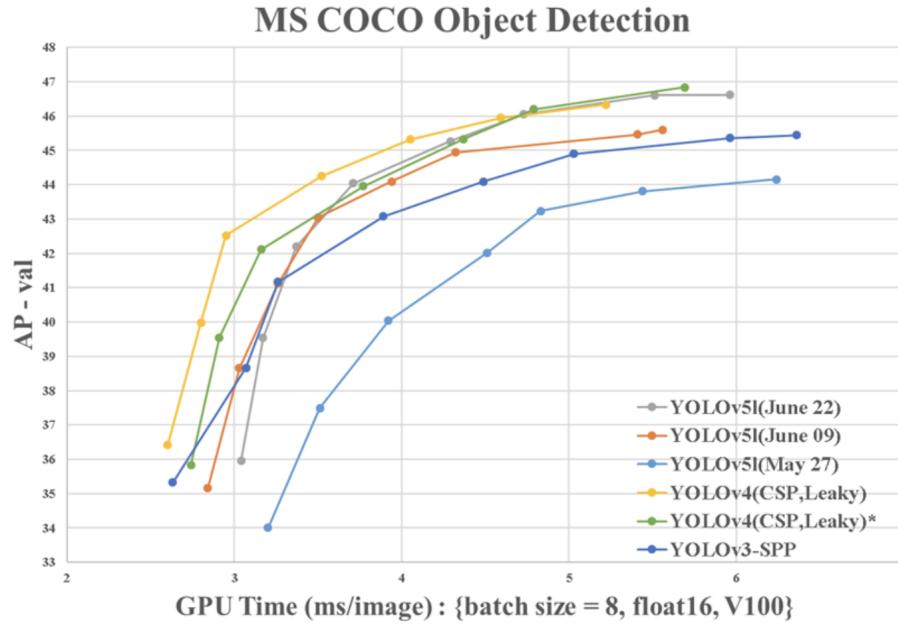


Figura 2.6: mAP di YOLOv5 confrontata con quelle di modelli precedenti [39]

### 2.2.6 YOLOv6

Proposta nel 2020 da Li et al. [20], va a migliorare le precedenti versioni introducendo una nuova architettura CNN chiamata **EfficientNet-L2**, una variante dell’architettura EfficientNet che migliora l’efficienza computazionale del modello riducendo il numero di parametri, pur mantenendo elevate prestazioni su vari benchmark di object detection.

### 2.2.7 YOLOv7

La settima versione di YOLO, rilasciata nel 2022 da Chien-Yao Wang, Alexey Bochkovskiy e Hong-Yuan Mark Liao [19], introduce diverse migliorie significative:

- **Anchor boxes migliorate:** YOLOv7 aumenta a 9 il numero di anchor boxes, il quale permette al modello di rilevare una gamma più ampia di forme e dimensioni degli oggetti rispetto alle versioni precedenti, riducendo così il numero di falsi positivi.
- **Focal Loss:** introdotta una nuova loss function chiamata "focal loss", la quale riduce il peso della loss per esempi ben classificati e si concentrano sugli esempi difficili, ovvero gli oggetti difficili da rilevare.
- **Risoluzione più alta e velocità migliorata:** YOLOv7 processa le immagini ad una risoluzione superiore rispetto alle versioni precedenti, ottenendo una precisione complessiva maggiore, ed ad una velocità maggiore, rendendolo adatto per applicazioni sensibili in tempo reale dove velocità di elaborazione più elevate sono cruciali.

## 2.3 Modelli utilizzati

I tre modelli selezionati per la sperimentazione di questa tesi (YOLOv8, YOLO-World e RT-DETR) rappresentano l'avanguardia nell'ambito dell'Object Detection. Ognuno di essi presenta caratteristiche uniche che li rendono adatti a diversi scenari applicativi, rappresentando un ulteriore sviluppo rispetto alle precedenti iterazioni dei modelli YOLO.

### 2.3.1 YOLOv8

YOLOv8 rappresenta l'ultima evoluzione della famiglia YOLO, sviluppata da Ultralytics e rilasciata nel 2023 [21]. Questa versione introduce una serie di miglioramenti significativi rispetto alle precedenti, rendendolo uno degli algoritmi di rilevamento degli oggetti più avanzati e performanti disponibili oggi.

#### Miglioramenti introdotti

YOLOv8 introduce le seguenti migliorie:

- **Architettura Backbone:** YOLOv8 utilizza un'architettura backbone più avanzata che combina convolutional layers ottimizzati con tecniche di normalizzazione e attivazione migliorate, progettata per massimizzare l'estrazione delle feature mantenendo un alto livello di efficienza computazionale.
- **Residual Blocks e Dense Connections:** l'introduzione di residual blocks (simili a quelli di ResNet) e dense connections (ispirate a DenseNet) contribuisce a una migliore propagazione del gradiente durante l'addestramento. Vengono ridotti problemi come il vanishing gra-

dient e migliorata la capacità della rete di apprendere rappresentazioni complesse, permettendo alla rete di addestrarsi.

- **GIoU Loss:** la loss function è stata migliorata con l'introduzione della Generalized Intersection over Union (GIoU) loss, la quale tiene conto non solo della sovrapposizione tra le bounding box previste e quelle reali, ma anche della distanza tra di esse.
- **Miglioramenti nelle FPN:** le feature pyramid networks sono state potenziate per fornire una migliore rappresentazione multi-scala delle feature dell'immagine, fattore che aiuta il modello a rilevare oggetti di diverse dimensioni, specialmente quelli piccoli.
- **Inferenza Accelerata:** grazie a ottimizzazioni specifiche nell'architettura e nell'implementazione del codice, YOLOv8 è in grado di eseguire l'inferenza a velocità superiori rispetto ai suoi predecessori.
- **Efficienza Computazionale:** la struttura generale è stata ottimizzata per essere eseguita su dispositivi con risorse limitate, come i dispositivi edge e embedded.

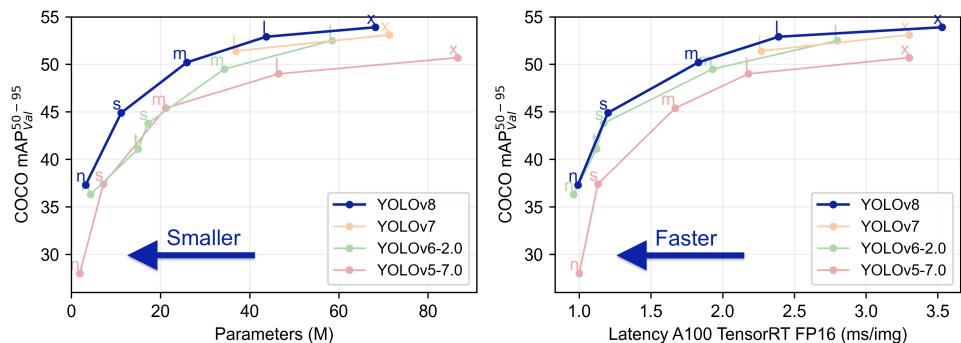


Figura 2.7: mAP ottenuta da YOLOv8 su dataset COCO confrontata con quelle delle precedenti versioni [13]

## Architettura

L'architettura di YOLOv8 si suddivide in 3 blocchi principali:

- **Backbone:** chiamato anche Feature Extractor, è responsabile dell'estrazione delle feature significative dall'input. Nei suoi strati iniziali, rileva schemi semplici come bordi e texture, e man mano che si avanza, il backbone rileva feature a più scale, ottenendo rappresentazioni a diversi livelli di astrazione.
- **Neck:** funge da ponte tra il Backbone e il blocco successivo (Head). Esso costruisce piramidi di feature aggregando le feature maps ottenute dal backbone ed eseguendo la concatenazione o fusione di feature a diverse scale per garantire che la rete possa rilevare oggetti di diverse dimensioni. Inoltre, integra informazioni contestuali per migliorare l'accuratezza del rilevamento e riduce la risoluzione spaziale e la dimensionalità delle risorse per facilitare il calcolo, aumentando la velocità.
- **Head:** è la parte finale dell'architettura e si occupa di generare gli output della rete. Genera le bounding boxes associate ai possibili oggetti nell'immagine, assegna punteggi di confidenza a ciascuna bounding box per indicare la probabilità della presenza di un oggetto, e classifica gli oggetti nelle bounding boxes in base alle loro categorie.

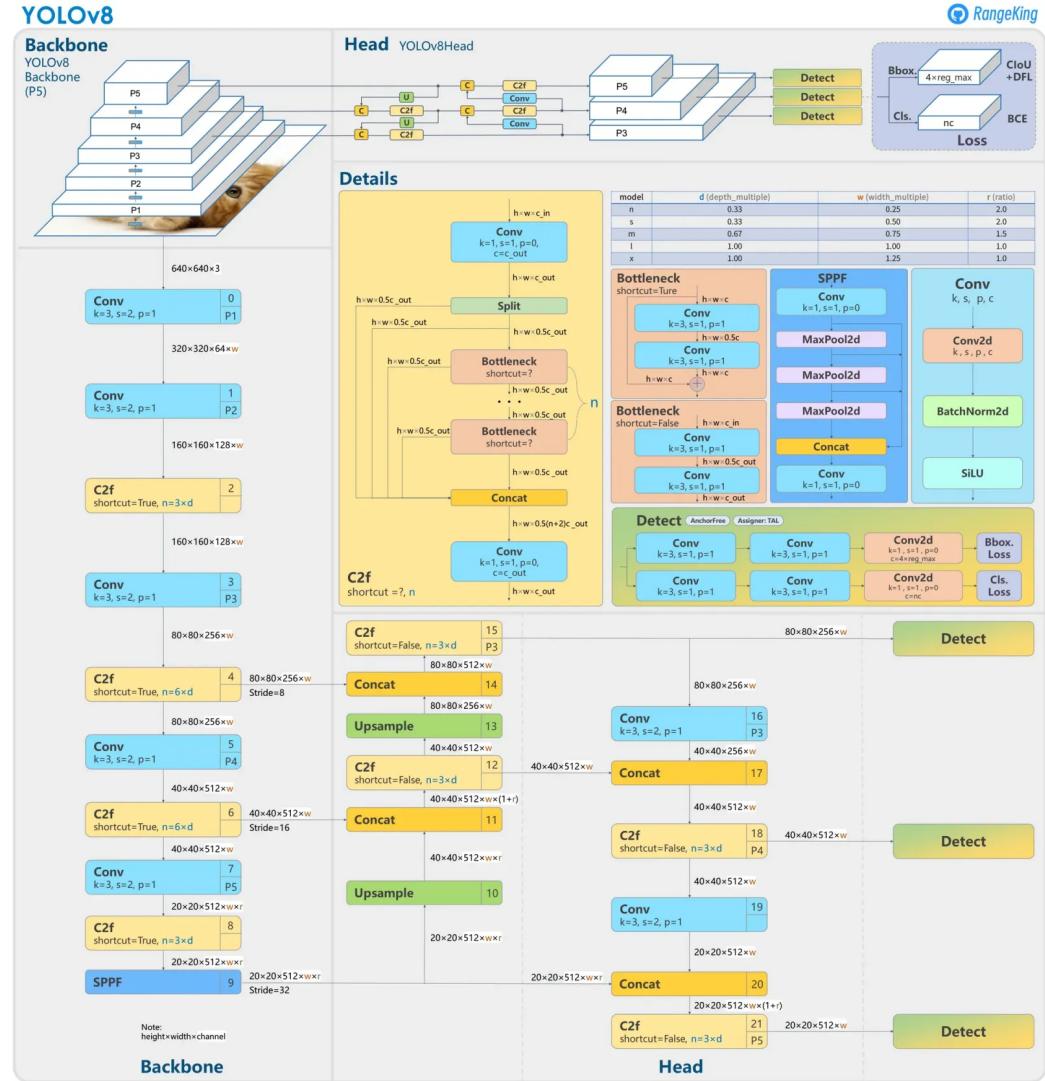


Figura 2.8: Architettura di YOLOv8 [40]

### 2.3.2 YOLO-World

YOLO-World è una recente evoluzione della famiglia di algoritmi YOLO [41], rivoluzionaria nella open-vocabulary object detection, che combina efficacemente l'efficienza di YOLOv8 con la potenza della modellazione vision-language ed un massiccio pre-training.

#### Open-Vocabulary Object Detection

L'object detection open-vocabulary (OVD) si propone di rilevare qualsiasi oggetto descritto tramite linguaggio naturale, anche se non è stato "visto" durante l'addestramento, superando così le limitazioni dei detector tradizionali che si concentrano su categorie di oggetti fissate.

YOLO-World in particolare consente di specificare dinamicamente le classi tramite prompt personalizzati, permettendo così agli utenti di adattare il modello alle proprie esigenze senza doverlo addestrare nuovamente. Questa caratteristica è particolarmente utile per adattare il modello a nuovi domini o compiti specifici che non erano originariamente parte dei dati di addestramento. Impostando prompt personalizzati, gli utenti possono guidare il focus del modello verso gli oggetti di interesse, migliorando l'accuratezza dei risultati di detection.

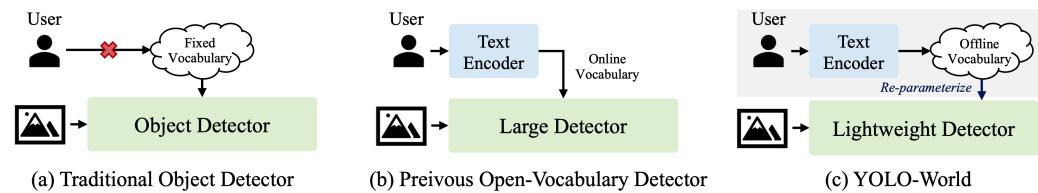


Figura 2.9: Open-Vocabulary Object Detection di YOLO-World [12]

## Innovazioni introdotte

YOLO-World introduce le seguenti innovazioni:

- **Paradigma Prompt-Then-Detect:** precomputando vocabolari off-line basati su prompt definiti dall'utente, YOLO-World evita la necessità di codifica del testo in tempo reale, migliorando la velocità di inferenza e semplificando la personalizzazione.
- **Vision-Language Grounding:** la capacità del modello di comprendere e correlare informazioni visive con descrizioni in linguaggio naturale porta a rilevazioni più precise e consapevoli del contesto.
- **Ridotta Complessità del Modello:** la sua architettura evita le pesanti componenti basati su Transformer presenti nei modelli OVD precedenti, promuovendo l'accessibilità.
- **Etica Open-Source:** rilasciato sotto la GPL (General Public License), favorisce la collaborazione, accelera l'innovazione e garantisce che la tecnologia rimanga accessibile alla comunità più ampia.

## Architettura

Le componenti principali dell'architettura di YOLO-World:

- **YOLOv8 Backbone:** si occupa dell'estrazione delle feature, fornendo dettagliate rappresentazioni multi-scala delle immagini in input.
- **Text Encoder:** un encoder basato su Transformer (che sfrutta i progressi ottenuti da CLIP di OpenAI) utile a trasformare efficientemente i prompt in linguaggio naturale in embedding testuali informativi.

- **RepVL-PAN**: il Re-parameterizable Vision-Language Path Aggregation Network combina in modo efficace le feature visive e gli embedding testuali. Questa sinergia è ottenuta tramite:
  - **Text-guided Cross Stage Partial Layer (T-CSPLayer)**, il quale potenzia le feature dell'immagine tramite le indicazioni degli embedding testuali, dirigendo l'attenzione del modello verso aree rilevanti. Questo avviene tramite il Max Sigmoid Attention Block, che calcola i pesi di attenzione basati sull'interazione tra le indicazioni testuali e le caratteristiche spaziali dell'immagine.
  - **Image-Pooling Attention**, che "raffina" gli embedding testuali incorporando il contesto visivo globale e così migliorando la comprensione complessiva.

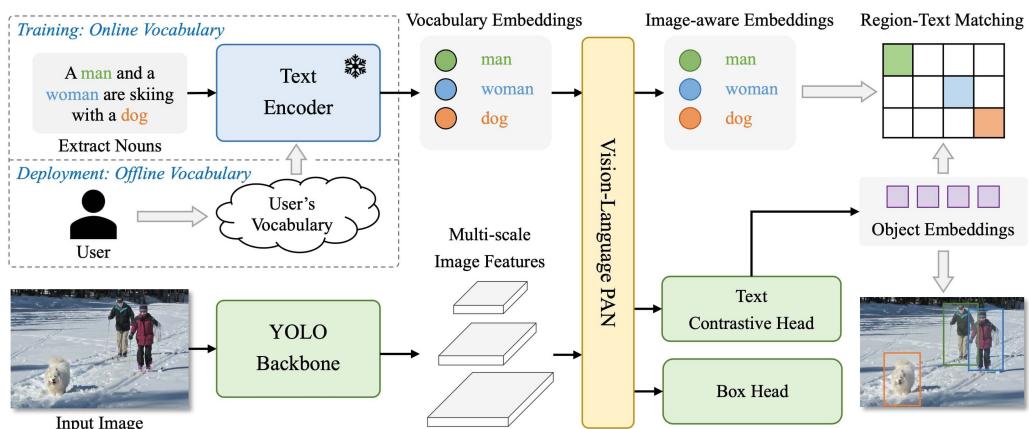


Figura 2.10: Architettura di YOLO-World [12]

### 2.3.3 RT-DETR

RT-DETR, o Real-Time DETR, è un’evoluzione del modello DETR (DEtection TRansformer) sviluppato da Baidu [43], che mira a migliorare le prestazioni di object detection in tempo reale.

#### Caratteristiche principali

Le principali caratteristiche del modello RT-DETR sono:

- **Efficient Hybrid Encoder:** incorpora un encoder che sfrutta i Vision Transformers (ViT) per gestire efficientemente le feature multiscala, permettendo così al RT-DETR di acquisire una comprensione globale delle immagini, migliorando la capacità di rilevamento degli oggetti senza dipendere esclusivamente da filtri locali come nelle CNN.
- **Selezione delle Query consapevole dell’IoU:** migliora l’inizializzazione delle query degli oggetti attraverso una selezione consapevole dell’indice di sovrapposizione (IoU), permettendo al modello di concentrarsi sugli oggetti più pertinenti della scena e migliorando l’accuratezza della detection.
- **Velocità di Inferenza Adattabile:** offre flessibilità nella regolazione della velocità di inferenza utilizzando diversi strati del decoder senza necessità di riaddestramento

#### Architettura

L’architettura di RT-DETR consiste sempre in 3 componenti principali:

- **Backbone:** utilizza un approccio innovativo il quale non si limita alla selezione delle feature map finali del backbone, bensì utilizza feature

provenienti da tre diversi livelli diversi di quest'ultimo. Gli stadi S3, S4 e S5 del backbone vengono impiegati come input per l'encoder, fungendo così da estrattore di caratteristiche multiscala per il modello.

- **Efficient Hybrid Encoder:** trasforma le feature in una sequenza di feature dell'immagine, utilizzando due altri moduli:

- **AIFI (Attention based Intra Scale Feature Interaction)**, il quale si occupa esclusivamente della feature map S5 per estrarre interpretazioni semantiche più elaborate, aumentando l'accuratezza complessiva.
- **CCFM (CNN based Cross-scale Feature-fusion Module)**, che estrapola interpretazioni semantiche sia da S4 che da S5, ed incorpora un Fusion Block che fonde le caratteristiche di due blocchi adiacenti in una nuova feature.

- **IoU Aware Query Selection e Transformer Decoder:** il primo è il modulo adibito alla selezione delle feature in base all'indice di sovrapposizione, le quali serviranno da input per il decoder, responsabile della generazione delle bounding box e dei punteggi di confidenza.

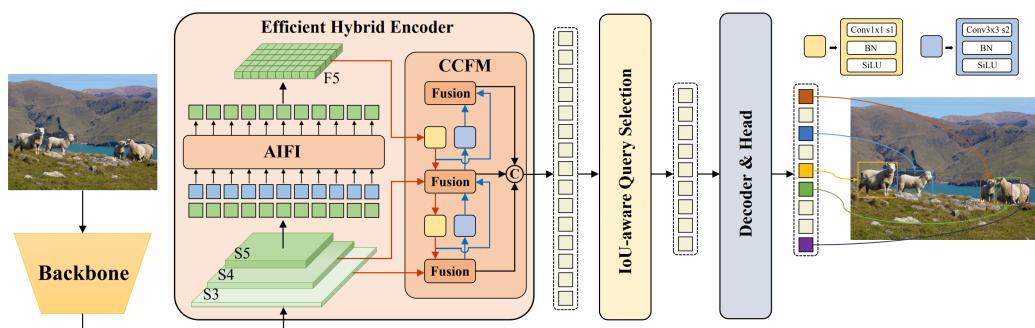


Figura 2.11: Architettura di RT-DETR [9]

# Capitolo 3

## Data Augmentation

### 3.1 Introduzione alla Data Augmentation

#### 3.1.1 Definizione

La data augmentation è una tecnica utilizzata nel campo del machine learning per incrementare la quantità e la varietà dei dati di addestramento a disposizione.

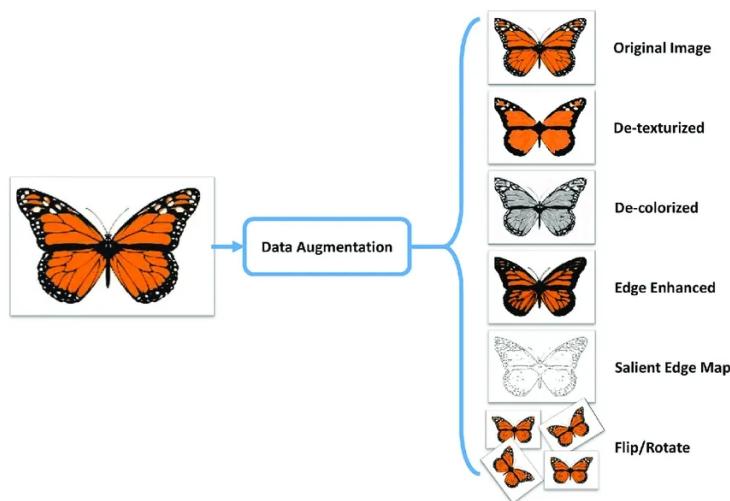


Figura 3.1: Esempi di Augmentation su un'immagine [37]

Questa tecnica consiste nel generare nuovi dati a partire da quelli esistenti, applicando una serie di trasformazioni come rotazioni, traslazioni, modifiche di colore, aggiunta di rumore, e molte altre. Si può quindi definire come un insieme di operazioni che modificano i dati originali in modo da creare nuove versioni, pur mantenendo inalterata l'informazione essenziale contenuta nei dati stessi.

### 3.1.2 Scopo

L'obiettivo principale della data augmentation è quello di migliorare la generalizzazione dei modelli di machine learning, ovvero la capacità del modello di performare bene su dati mai visti prima. Questo viene ottenuto in vari modi:

- **Aumento della Varietà dei Dati:** aggiungendo varianti dei dati esistenti, si espande il dataset senza la necessità di raccogliere nuovi dati. Questo è particolarmente utile quando i dati disponibili sono limitati o costosi da ottenere.
- **Riduzione dell'Overfitting:** creando una maggiore diversità nel dataset, si riduce il rischio che il modello impari a memorizzare dettagli specifici dei dati di addestramento piuttosto che generalizzare dai pattern sottostanti (fenomeno dell'overfitting), e si costringe quindi il modello a imparare caratteristiche più generali e robuste.
- **Bilanciamento dei Dataset:** nei casi in cui alcuni gruppi di dati sono sottrappresentati (ad esempio, minoranze di classe in un problema di classificazione), la data augmentation può essere utilizzata per creare più esempi di queste classi minoritarie, aiutando a bilanciare il dataset e migliorare le prestazioni del modello su tutte le classi.

## 3.2 Tecniche di Data Augmentation

La data augmentation comprende una vasta gamma di tecniche che trasformano le immagini esistenti per generare nuovi esempi di addestramento.

### 3.2.1 Tecniche Base di Data Augmentation

Queste tecniche mirano a incrementare la variabilità dei dati di addestramento attraverso modifiche semplici e intuitive, permettendo al modello di imparare a riconoscere gli oggetti in una varietà di condizioni e configurazioni.

#### Trasformazioni Geometriche

Le trasformazioni geometriche sono tecniche che alterano la posizione o la forma degli oggetti nell'immagine senza modificarne il contenuto. Queste tecniche sono utili per rendere i modelli di computer vision più robusti alle variazioni spaziali.

- **Rotazione:** ruota l'immagine di un determinato angolo - aiuta il modello a riconoscere oggetti indipendentemente dalla loro orientazione.
- **Traslazione:** sposta l'immagine lungo gli assi X e Y - aiuta il modello a riconoscere oggetti che potrebbero essere presenti in posizioni diverse nell'immagine.
- **Ritaglio:** seleziona una porzione dell'immagine originale - migliora la capacità del modello di identificare oggetti che possono essere solo parzialmente visibili.
- **Zoom:** ingrandimento o riduzione dell'immagine - aiuta il modello ad identificare oggetti a diverse scale.

- **Flip:** ribalta l'immagine orizzontalmente, verticalmente o entrambe - aiuta il modello a imparare che gli oggetti possono apparire in orientazioni diverse.

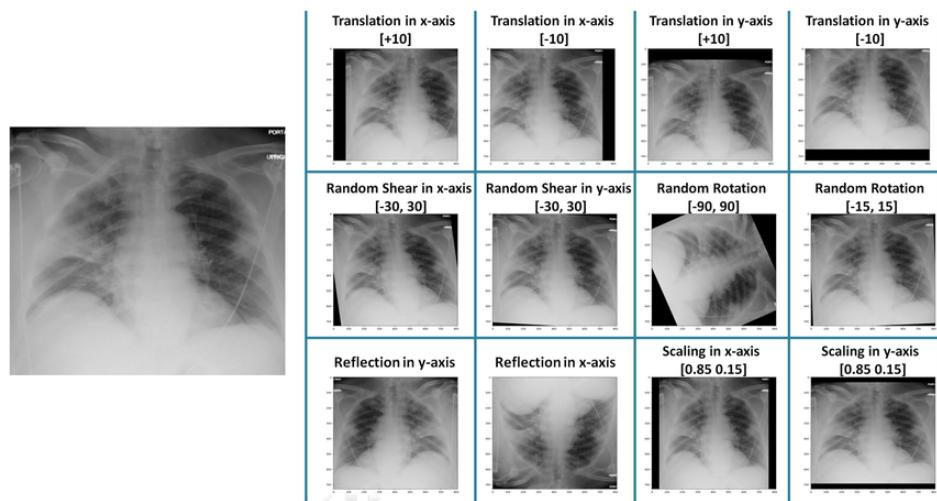


Figura 3.2: Esempi di trasformazioni geometriche applicate ad un'immagine X-ray [30]

## Modifiche dei Colori

Le modifiche dei colori alterano i valori cromatici nell'immagine, aiutando il modello a riconoscere gli oggetti sotto diverse condizioni di illuminazione e colore.

Possiamo andare a modificare:

- **Luminosità:** migliora la capacità del modello di adattarsi a condizioni di luce diverse
- **Saturazione:** aiuta il modello a riconoscere gli oggetti indipendentemente dall'intensità dei colori

- **Contrasto:** permette al modello di distinguere meglio gli oggetti in base alle differenze di luminosità
- **Colore:** aiuta il modello a riconoscere gli oggetti indipendentemente dalle condizioni cromatiche.

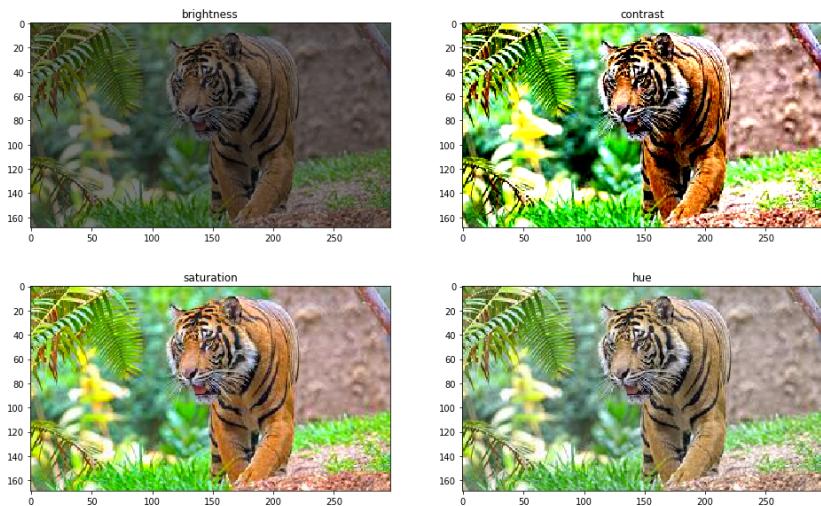


Figura 3.3: Esempi di trasformazioni di colore applicate ad una foto [35]

## Distorsioni e Rumore

Le distorsioni e l'aggiunta di rumore all'immagine possono migliorare la robustezza del modello, aiutandolo a imparare a riconoscere gli oggetti anche in condizioni non ideali o con interferenze.

- **Distorsione Elastica:** introduce deformazioni locali all'immagine, aumentando la varietà dei pattern geometrici presenti nei dati di addestramento.
- **Aggiunta di Rumore Gaussiano:** introduce variazioni casuali simili al rumore nei sensori delle fotocamere, migliorando la capacità del

modello di distinguere dettagli sottili e rendendolo più robusto contro i disturbi.

- **Aggiunta di Rumore Salt-and-Pepper:** simula interferenze come granelli di polvere o artefatti minori, rendendo il modello più robusto rispetto alle imperfezioni dell'immagine.
- **Blur:** sfoca l'immagine simulando condizioni di scarsa qualità, migliorando la capacità del modello di riconoscere oggetti in condizioni di bassa definizione o in movimento.
- **Distorsione da Lente:** introduce distorsioni ottiche simili a quelle causate da lenti fotografiche, migliorando la capacità del modello di adattarsi a variazioni nelle proporzioni degli oggetti.

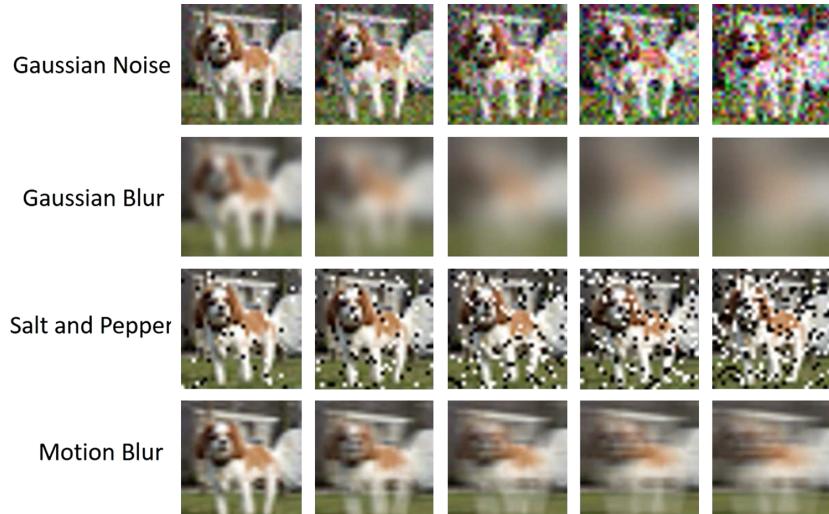


Figura 3.4: Esempi di distorsioni ed aggiunte di rumore [29]

### 3.2.2 Tecniche Avanzate di Data Augmentation

Le tecniche avanzate di data augmentation vanno oltre le trasformazioni geometriche ed i cambiamenti di colore, introducendo strategie più complesse per migliorare le prestazioni dei modelli. Queste tecniche sono progettate per generare dati sintetici più diversificati e realistici, migliorando così la capacità del modello di generalizzare su dati non visti durante l'addestramento.

#### Augmentation tramite GAN

L'Augmentation tramite Generative Adversarial Networks (GAN) è una tecnica che utilizza reti neurali generative e discriminative per generare dati sintetici ad alta fedeltà. Questi modelli apprendono la distribuzione dei dati di input e generano esempi che sono coerenti con tale distribuzione.

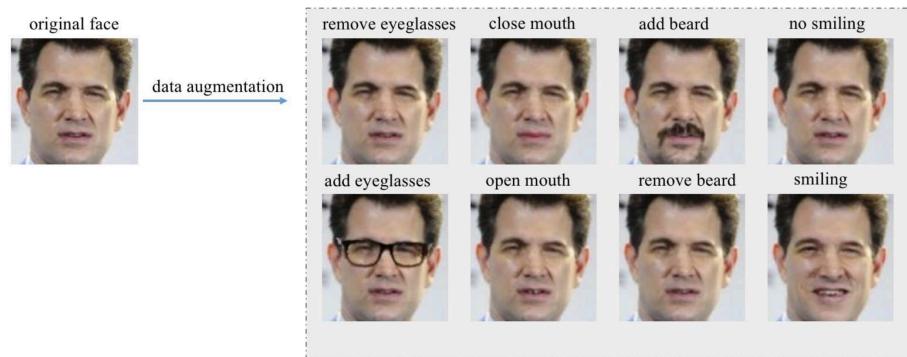


Figura 3.5: Esempi di immagini generate tramite GAN [23]

### Mixup, Cutout e Cutmix

Mixup è una tecnica di data augmentation che mescola coppie di campioni di addestramento per creare nuovi esempi. Questa tecnica combina le immagini e le etichette dei campioni originali per creare nuove istanze durante l'addestramento. D'altra parte, Cutout è una tecnica che consiste nel mascherare casualmente parti delle immagini di addestramento, incoraggiando il modello a focalizzarsi su caratteristiche meno sensibili al rumore. Infine CutMix è una tecnica che unisce le precedenti, ritagliando regioni delle immagini come nel Cutout, ma andandoci ad incollare pezzi di altri esempi. Questa tecnica forza il modello a generalizzare meglio, rendendolo robusto alle distorsioni localizzate e migliorando la sua capacità di rilevare e classificare oggetti anche in presenza di parziali occlusioni.

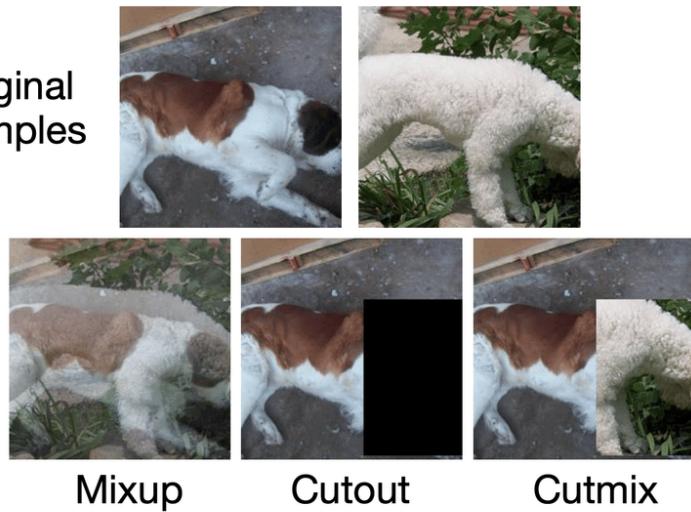


Figura 3.6: Esempi di Mixup, Cutout e Cutmix [44]

# Capitolo 4

## Sperimentazione e Risultati

### 4.1 Obiettivo

L’obiettivo di questa tesi sperimentale è confrontare le prestazioni di tre modelli avanzati di Object Detection applicati alle immagini termiche: YOLOv8, YOLO-World e RT-DETR. Questi modelli sono stati scelti per rappresentare tre diversi approcci al problema del riconoscimento degli oggetti, consentendo così un confronto dettagliato delle loro capacità su questo specifico tipo di dati.

Per massimizzare le prestazioni di questi modelli, espanderemo il dataset di immagini termiche utilizzando tecniche di augmentation, valutando le più efficaci offerte dalla libreria Albumentations. Successivamente, testeremo i tre modelli in questi scenari per identificare quello che meglio si adatta al riconoscimento di oggetti in immagini termiche e, soprattutto, ottiene il miglioramento più significativo grazie all’addestramento su dataset aumentato.

## 4.2 Preparazione del dataset

### 4.2.1 Dataset termico FLIR

Il dataset termico utilizzato è fornito da Teledyne FLIR [3], un'azienda leader nel settore delle tecnologie di rilevamento termico e di imaging a infrarossi, ed è stato progettato per consentire agli sviluppatori di avviare l'addestramento di reti neurali convoluzionali, finalizzate alla creazione della prossima generazione di sistemi avanzati di assistenza alla guida (ADAS) e veicoli autonomi più sicuri ed efficienti.

Il dataset è composto da immagini catturate sia di giorno (60%) che di notte (40%) su strade e autostrade nell'area di Santa Barbara (California) da Novembre a Maggio, con condizioni meteorologiche sia limpide che nuvolose.



Figura 4.1: Immagine del dataset termico FLIR [3]

Le immagini sono state acquisite utilizzando una termocamera con le seguenti specifiche [3]:

- **Termocamera IR:** Tau2 640 x 512, 13 mm f/1,0 (HFOV 45°, VFOV 37°)
- **Telecamera FLIR BlackFly:** (BFS-U3-51S5C-C) 1280 x 1024, ottica megapixel Computar 4-8 mm f/1,4-16 (FOV impostato per Tau2)

#### 4.2.2 Estrazione e Formattazione dei subset

Scaricando il dataset dalla apposita pagina di Teledyne FLIR [3], ci troviamo di fronte alle seguenti raccolte di immagini:

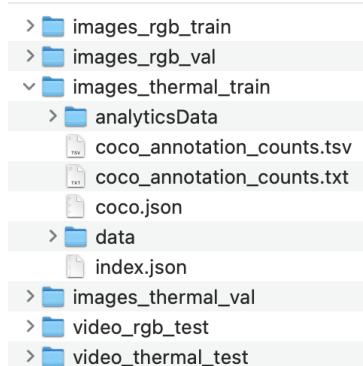


Figura 4.2: Contenuto di FLIR\_ADA挟\_v2.zip

I dati di nostro interesse sono contenuti nelle cartelle relative alle immagini termiche ("thermal") per i 3 subset di train, val e test; le immagini termiche di ogni subset si trovano della cartella **data**, mentre le annotazioni nei file **coco.json** (formato MSCOCO) e **index.json** (formato di Conservator, un tool proprietario di Teledyne FLIR per il management dei dataset).

Per addestrare i modelli YOLO, è necessario organizzare la cartella del dataset con la seguente struttura:

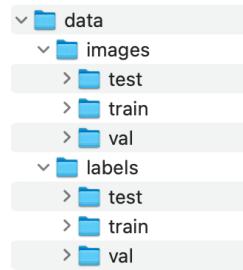


Figura 4.3: Struttura cartella dataset per i modelli YOLO

Tramite uno script Python sono quindi andato a copiare le immagini dei tre subset nelle relative cartelle in `data/images`, e successivamente sono andato a generare le annotazioni in formato YOLO di ciascuna immagine partendo da quelle presenti in `index.json`, inserendole in file di testo salvati in `data/labels`.

Perciò ciascuna immagine `(frameID).jpg` in `data/images/(subset)`, avrà il corrispettivo file di testo `(frameID).txt` in `data/labels/(subset)`, il quale contiene le relative annotazioni nel formato YOLO: ogni riga corrisponde ad una Bounding Box e contiene id della classe, coordinata X del centro, coordinata Y del centro, larghezza ed altezza.

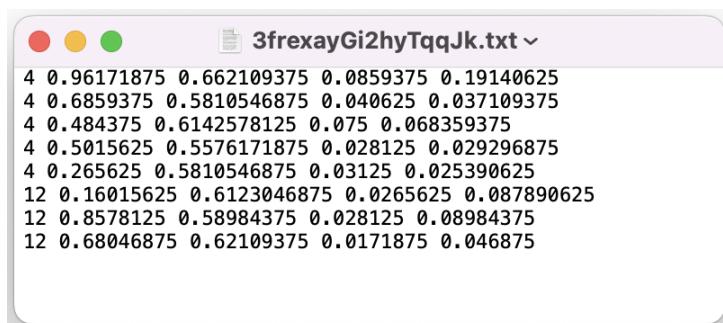


Figura 4.4: File contenente le annotazioni del frame "3frexayGi2hyTqqJk"

### 4.2.3 Analisi delle classi

Il dataset originario contiene le annotazioni relative a 20 classi di oggetti, segue l'analisi dettagliata del numero di annotazioni presenti nei 3 subset per ciascuna delle classi:

TRAIN DATA			VAL DATA			TEST DATA		
Class	Count	%	Class	Count	%	Class	Count	%
animal	8	0,00%	animal	0	0,00%	animal	0	0,00%
bike	7237	4,11%	bike	170	1,01%	bike	113	0,18%
bird	1	0,00%	bird	0	0,00%	bird	0	0,00%
bus	2245	1,28%	bus	179	1,07%	bus	0	0,00%
car	73623	41,82%	car	7133	42,49%	car	30517	48,86%
dog	4	0,00%	dog	0	0,00%	dog	25	0,04%
face	752	0,43%	face	73	0,43%	face	142	0,23%
hydrant	1095	0,62%	hydrant	94	0,56%	hydrant	277	0,44%
license plate	270	0,15%	license plate	17	0,10%	license plate	0	0,00%
light	16198	9,20%	light	2005	11,94%	light	6758	10,82%
motor	1116	0,63%	motor	55	0,33%	motor	3314	5,31%
other vehicle	1373	0,78%	other vehicle	63	0,38%	other vehicle	696	1,11%
person	44527	25,29%	person	4309	25,67%	person	11242	18,00%
rider	5951	3,38%	rider	161	0,96%	rider	1081	1,73%
scooter	15	0,01%	scooter	0	0,00%	scooter	0	0,00%
sign	20770	11,80%	sign	2472	14,73%	sign	5660	9,06%
skateboard	29	0,02%	skateboard	3	0,02%	skateboard	0	0,00%
stroller	15	0,01%	stroller	6	0,04%	stroller	0	0,00%
train	5	0,00%	train	0	0,00%	train	0	0,00%
truck	829	0,47%	truck	46	0,27%	truck	2634	4,22%
<b>Total</b>	<b>176063</b>		<b>Total</b>	<b>16786</b>		<b>Total</b>	<b>62459</b>	
<b>Filtered</b>	<b>4439</b>		<b>Filtered</b>	<b>372</b>		<b>Filtered</b>	<b>444</b>	
<b>Total Filtered</b>	<b>171624</b>		<b>Total Filtered</b>	<b>16414</b>		<b>Total Filtered</b>	<b>62015</b>	

Figura 4.5: Analisi delle annotazioni per ciascuna classe

Possiamo notare che alcune classi sono sottorappresentate nelle annotazioni rispetto al totale, il che potrebbe compromettere l'efficacia dell'addestramento del modello e deteriorare le metriche di valutazione complessive. Per questo motivo, ho deciso di filtrare le classi in questione dalle annotazioni del dataset (quelle evidenziate in rosso).

#### 4.2.4 Nuovo dataset filtrato

Ho quindi generato un nuovo dataset chiamato `filtered-data`, che contiene le stesse immagini dell'originale, ma con i file .txt delle annotazioni aggiornati. Le annotazioni sono state filtrate tramite uno script Python che ha scansionato le righe dei file di annotazione originali, rimuovendo quelle relative alle classi da escludere.

Così facendo, ho ottenuto il dataset su cui addestrare i modelli, suddiviso come segue:

- **Train subset:** 10742 esempi
- **Val subset:** 1144 esempi
- **Test subset:** 3749 esempi

## 4.3 Modelli di partenza

### 4.3.1 Ultralytics

Ultralytics [10] è un'azienda leader nello sviluppo di soluzioni avanzate di Computer Vision e Deep Learning. Conosciuta principalmente per il suo contributo alla famiglia di modelli YOLO, Ultralytics si distingue per la creazione di strumenti di rilevamento oggetti ad alte prestazioni, ampiamente utilizzati in varie applicazioni industriali e di ricerca. La piattaforma di Ultralytics fornisce modelli pre-addestrati e strumenti di facile utilizzo per l'implementazione rapida di algoritmi di object detection, garantendo prestazioni elevate e scalabilità.

### 4.3.2 Modelli di Ultralytics utilizzati

Per questo esperimento sono stati scelti i modelli più piccoli per ognuna delle tre tipologie di modello, caratterizzati da una ridotta complessità in termini di numero di parametri da dover addestrare.

I modelli in questione sono:

- **yolov8n**: modello YOLOv8 "nano" [13]
- **yolov8s-world**: modello YOLOv8 "small-world" [12]
- **rtdetr-l**: modello RT-DETR "large" [9]

La scelta di questi modelli è stata guidata dalla necessità di avere modelli leggeri e veloci da addestrare, in modo da ottimizzare il processo di sperimentazione e ottenere risultati rapidi senza compromettere significativamente la precisione.

### 4.3.3 Valutazione dei modelli

Per valutare le prestazioni dei modelli ho utilizzato il metodo `val` [8] fornito da Ultralytics, il quale analizza il comportamento di un determinato modello sul subset specificato nel file di configurazione, fornendo una serie di metriche utili. Il test avviene tramite il seguente script Python, in cui il file di configurazione `test-config.yaml` specifica al metodo di valutare il modello sul subset di test:

```

1 from ultralytics import YOLO, YOLOWorld, RTDETR
2
3 model = YOLO('yolov8n.pt')
4 metrics = model.val(data='test-config.yaml')
5
6 print("map50-95: ", metrics.box.map)
7 print("map50: ", metrics.box.map50)
8 print("map75: ", metrics.box.map75)
9 print("maps: ", metrics.box.maps)

```

Figura 4.6: Script per il testing di 'yolov8n'

## Metriche

Valutare l'efficacia dei modelli di Object Detection richiede l'uso di metriche precise che misurino la capacità del modello di individuare e localizzare correttamente gli oggetti all'interno delle immagini. Le seguenti metriche fondamentali sono essenziali per questa valutazione:

- **Precision:** misura la proporzione di oggetti predetti correttamente rispetto a tutti gli oggetti predetti dal modello.
- **Recall:** misura la proporzione di oggetti veri positivi rilevati correttamente rispetto a tutti gli oggetti veri positivi presenti nel dataset.
- **Average Precision (AP):** calcola l'area sotto la curva Precision-Recall (PR curve), riflettendo la precisione del modello al variare della soglia di confidenza delle predizioni

- **Intersection over Union (IoU)**: misura la sovrapposizione tra la bounding box predetta e quella ground-truth, ed è calcolato come area dell'intersezione divisa per area dell'unione delle due bounding box.

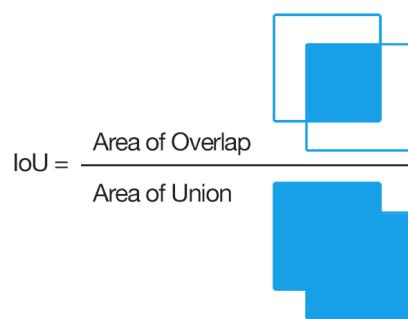


Figura 4.7: Intersection over Union [32]

Per valutare più efficacemente le prestazioni dei modelli utilizziamo le metriche di Mean Average Precision (mAP), che rappresentano la media delle Average Precision calcolate per ogni classe:

- **mAP<sub>50-95</sub>**: media dell'Average Precision (AP) calcolata su diverse soglie di IoU (Intersection over Union), dall'50% al 95%
- **mAP<sub>50</sub>**: media dell'Average Precision (AP) calcolata con IoU al 50%
- **mAP<sub>75</sub>**: media dell'Average Precision (AP) calcolata con IoU al 75%

Queste metriche ci forniranno una misura dettagliata delle capacità del modello nel rilevare e localizzare correttamente gli oggetti all'interno delle immagini, considerando diverse soglie di sovrapposizione tra le predizioni del modello e le ground-truth bounding box.

#### 4.3.4 Test dei modelli non addestrati

Prima di procedere con l’addestramento dei tre modelli, ho valutato le loro prestazioni iniziali sul dataset filtrato, ottenendo i seguenti risultati:

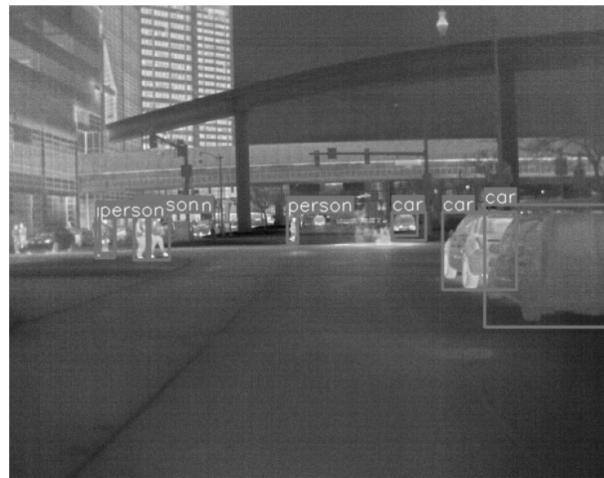
yolov8n		yolov8s-world		rtdetr-l	
mAP50-95	0,0092	mAP50-95	0,0086	mAP50-95	0,0026
mAP50	0,0160	mAP50	0,0153	mAP50	0,0066
mAP75	0,0079	mAP75	0,0086	mAP75	0,0017
AP per class		AP per class		AP per class	
bike	0,0621	bike	0,0537	bike	0,0000
car	0,0000	car	0,0000	car	0,0000
light	0,0092	light	0,0086	light	0,0026
motor	0,0158	motor	0,0109	motor	0,0236
other vehicle	0,0000	other vehicle	0,0000	other vehicle	0,0000
person	0,0000	person	0,0000	person	0,0000
rider	0,0000	rider	0,0000	rider	0,0000
sign	0,0092	sign	0,0086	sign	0,0026
truck	0,0000	truck	0,0000	truck	0,0001

Figura 4.8: Risultati test dei modelli non addestrati

Possiamo osservare che senza un’ottimizzazione specifica (fine-tuning) sul dataset di immagini termiche, i modelli presentano prestazioni inadeguate nell’Object Detection di questi dati, evidenziando metriche significativamente inferiori rispetto a quelle ottenute sui dataset standard (come COCO) riportate da Ultralytics.

mAP50-95 on COCO	
yolov8n	0,373
yolov8s-world	0,374
rtdetr-l	0,53

Figura 4.9: mAP50-95 dei modelli su dataset COCO



yolov8n



yolov8s-world



rtdetr-l

Figura 4.10: Detections effettuate dai modelli non addestrati

## 4.4 Addestramento su dataset filtrato

### 4.4.1 Addestramento dei modelli

L’addestramento dei tre modelli è stato effettuato tramite uno script Python che impiega la libreria di Ultralytics per caricare ed addestrare il modello:

---

```

1 from ultralytics import YOLO, YOLOWorld, RTDETR
2
3 model = YOLO('yolov8n.pt')
4 results = model.train(data='train-config.yaml', device=0, epochs=50, batch=8, verbose=True)

```

Figura 4.11: Script per l’addestramento di ‘yolov8n’

### Parametri

Il metodo `train` [7] dei modelli di Ultralytics permette di configurare una vasta varietà di parametri, quelli di nostro interesse sono:

- **data**: path al file di configurazione del dataset, in cui vengono specificati il path dei dati e le classi su cui addestrare il modello. In questo caso `train-config.yaml` indica i path ai subset train e val
- **device**: specifica il dispositivo/i computazionali per l’addestramento
- **epochs**: numero totale delle epochs di addestramento. Ciascuna epoch rappresenta un passaggio completo sull’intero dataset, perciò modificare questo valore influenza la durata dell’addestramento e le prestazioni del modello.
- **batch**: numero di campioni di addestramento processati simultaneamente durante ogni iterazione. Aumentare il batch size può migliorare l’efficienza computazionale, ma richiede più memoria.

- **verbose:** abilita un output dettagliato durante l’addestramento, fornendo log dettagliati e aggiornamenti sul progresso

## Augmentations

Il metodo `train` di ultralytics implementa internamente delle tecniche di augmentation, essenziali per migliorare la robustezza e le prestazioni dei modelli YOLO. Durante l’addestramento vengono quindi eseguite le seguenti tecniche:

- Adjustment of Hue
- Adjustment of Saturation
- Adjustment of Brightness
- Translation
- Scaling
- Horizontal Flipping
- Mixup
- Cutout
- Crop

#### 4.4.2 Test dei modelli addestrati sul dataset filtrato

Il primo addestramento dei modelli è stato condotto sul dataset `filtered-data` per un totale di 50 epochs, in modo da ottenere un buon equilibrio tra velocità di addestramento e capacità del modello; il batch size è stato impostato ad 8 per evitare problemi di memoria sui dispositivi.

Dopo aver completato l'addestramento, ho testato i modelli utilizzando il metodo `val` sulla migliore configurazione dei parametri interni ottenuta durante le 50 epochs di addestramento, ottenendo i seguenti risultati:

yolov8n		yolov8s-world		rtdetr-l	
Dataset	filtered-data	Dataset	filtered-data	Dataset	filtered-data
Epochs	50	Epochs	50	Epochs	50
Batch	8	Batch	8	Batch	8
mAP50-95	0,1421	mAP50-95	0,1689	mAP50-95	0,1820
mAP50	0,2617	mAP50	0,3063	mAP50	0,3348
mAP75	0,1344	mAP75	0,1617	mAP75	0,1682
AP per class		AP per class		AP per class	
bike	0,0009	bike	0,0019	bike	0,0038
car	0,4596	car	0,5141	car	0,5170
light	0,0532	light	0,0767	light	0,1343
motor	0,1103	motor	0,1400	motor	0,1046
other vehicle	0,0498	other vehicle	0,0880	other vehicle	0,0423
person	0,4350	person	0,4731	person	0,4900
rider	0,0069	rider	0,0199	rider	0,0274
sign	0,0644	sign	0,0992	sign	0,1153
truck	0,0992	truck	0,1074	truck	0,2033

Figura 4.12: Risultati test dei modelli addestrati sul dataset filtrato

Possiamo osservare un notevole miglioramento delle prestazioni rispetto ai test precedenti dei modelli, già con questo primo addestramento effettuato su `filtered-data`. Questo dimostra come il fine-tuning sia essenziale per ottimizzare i modelli su tipologie di dati specifiche come le nostre immagini termiche.

yolov8n			
Dataset: filtered-data		Epochs: 50	Batch: 8
mAP50-90	0,1421	+1531,43%	<i>compared to untrained</i>
mAP50	0,2617	+1439,76%	<i>compared to untrained</i>
mAP75	0,1344	+1595,20%	<i>compared to untrained</i>

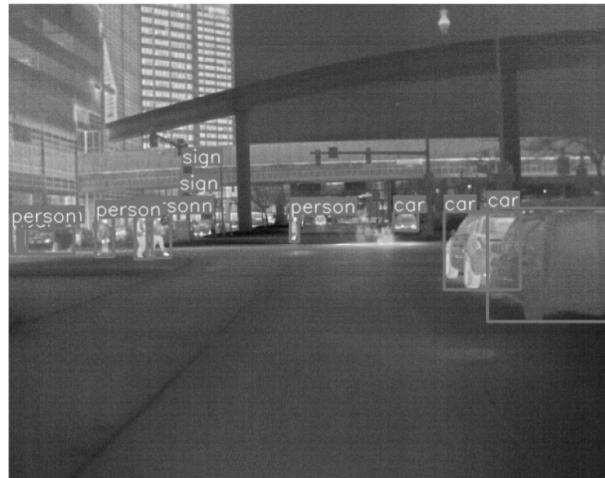
  

yolov8s-world			
Dataset: filtered-data		Epochs: 50	Batch: 8
mAP50-90	0,1689	+1852,79%	<i>compared to untrained</i>
mAP50	0,3063	+1897,01%	<i>compared to untrained</i>
mAP75	0,1617	+1777,70%	<i>compared to untrained</i>

rtdetr-l			
Dataset: filtered-data		Epochs: 50	Batch: 8
mAP50-90	0,1820	+6787,38%	<i>compared to untrained</i>
mAP50	0,3348	+3967,83%	<i>compared to untrained</i>
mAP75	0,1682	+9982,10%	<i>compared to untrained</i>

Figura 4.13: Confronto tra i risultati dei test dei modelli addestrati sul dataset filtrato e quelli dei precedenti test



yolov8n



yolov8s-world



rtdetr-l

Figura 4.14: Detections effettuate dai modelli addestrati sul dataset filtrato

## 4.5 Addestramento su dataset aumentato

Dopo aver ottenuto risultati significativi dal fine-tuning sul dataset filtrato, il passo successivo è stato individuare una serie di tecniche di augmentation da applicare al dataset. Per questo scopo, ho scelto di utilizzare la libreria Albumentations, che mi ha permesso di generare una versione aumentata del dataset. L'obiettivo era ampliare il subset di training e migliorare ulteriormente le prestazioni dei modelli.

### 4.5.1 Albumentations

Albumentations [1] è una libreria open-source per l'augmentation delle immagini, progettata principalmente per il deep learning e l'addestramento di modelli di intelligenza artificiale. Essa offre un'ampia gamma di trasformazioni per le immagini, permettendo agli sviluppatori di migliorare la diversità dei dati di addestramento senza modificare manualmente le immagini originali. Le trasformazioni supportate includono modifiche di colore, traslazioni, rotazioni, crop, cambiamenti prospettici, aggiunta di rumore e molto altro ancora. Albumentations è particolarmente apprezzata per la sua velocità e flessibilità, essendo ottimizzata per gestire grandi quantità di dati e integrarsi senza problemi con framework di deep learning come PyTorch e TensorFlow. Nel nostro caso la libreria è stata utilizzata per generare direttamente tramite uno script nuovi dataset aumentati.

### 4.5.2 Prima Data Augmentation

#### Primo dataset aumentato

Ho fatto un primo tentativo di data augmentation sul dataset filtrato, andando a generare una versione "augmented" di ciascuna immagine presente in `filtered-data/images/train`. Questo è stato fatto tramite uno script Python che applica le seguenti trasformazioni di Albumentations ad ogni immagine:

```
transform = A.Compose([
    A.Blur(blur_limit=7, p=0.5),
    A.RandomBrightnessContrast(p=0.5),
    A.RandomResizedCrop(size=(dataset_height, dataset_width), scale=(0.2, 1), p=1),
    A.Rotate(limit=(-180, 180), p=1),
    A.Sharpen(p=0.5),
    A.VerticalFlip(p=0.5)
], bbox_params=A.BboxParams(format='yolo', label_fields=['class_labels']))
```

Figura 4.15: Definizione della trasformazione di Albumentations da applicare alle immagini

Il metodo `transform` di Albumentations consente l'applicazione di un insieme specifico di trasformazioni su un'immagine, gestendo probabilità e intervalli definiti. Questo processo genera una nuova immagine trasformata con relative annotazioni aggiornate, poiché le coordinate delle bounding box possono variare o essere rimosse a seguito delle trasformazioni. A questo punto, le nuove immagini e annotazioni vengono utilizzate per creare il nuovo dataset aumentato `augmented-data-1`, il quale avrà esattamente il doppio degli esempi nel subset di training rispetto a `filtered-data`: 10.742 immagini del dataset originale e 10.742 immagini trasformate. I subset val e test rimangono invariati.

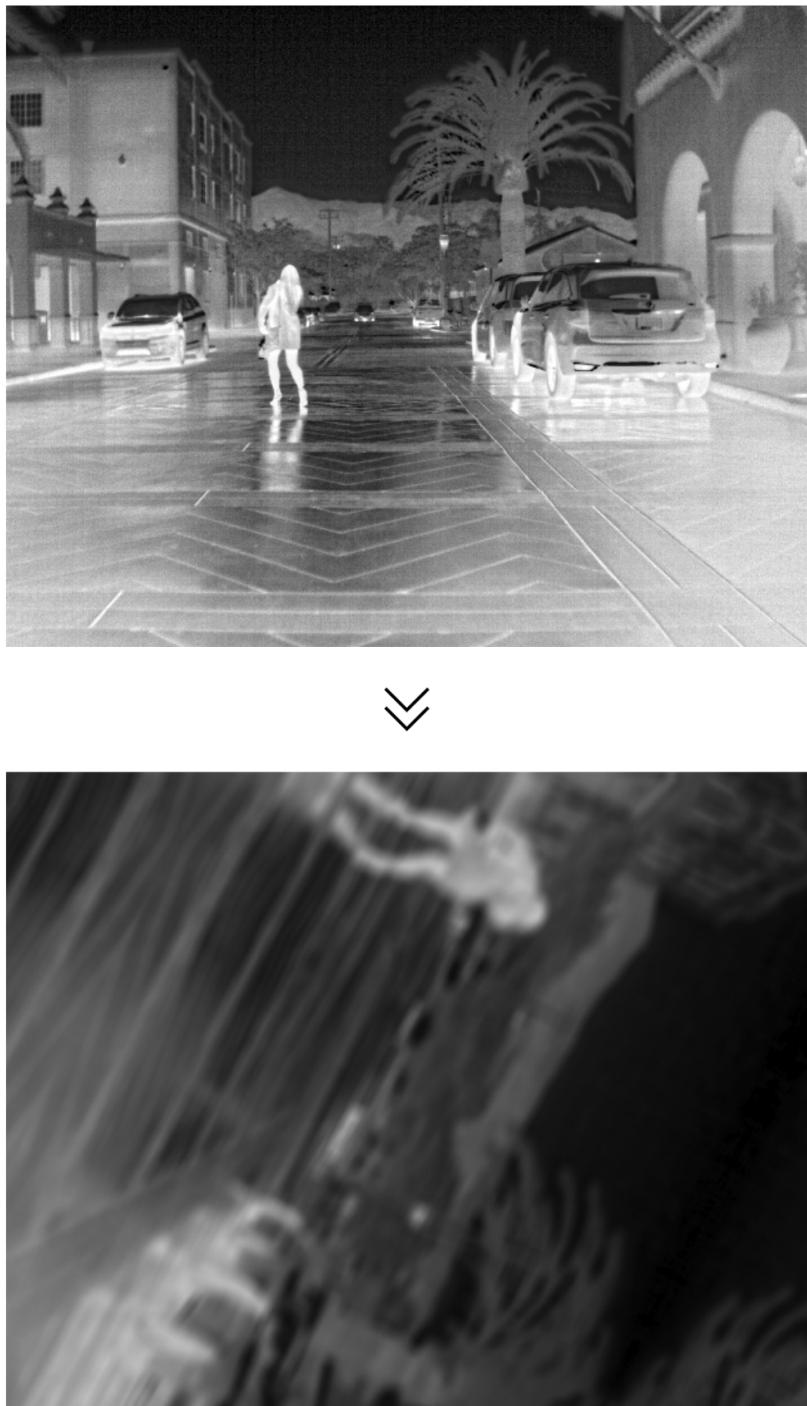


Figura 4.16: Esempio di immagine trasformata con la prima data augmentation

### Test modelli addestrati sul primo dataset aumentato

Il secondo addestramento dei modelli è stato quindi condotto sul dataset **augmented-data-1**, mantenendo le stesse condizioni del primo, con 50 epochs e un batch size di 8. I successivi test hanno prodotto i seguenti risultati:

yolov8n		yolov8s-world		rtdetr-l	
Dataset	augmented-data-1	Dataset	augmented-data-1	Dataset	augmented-data-1
Epochs	50	Epochs	50	Epochs	50
Batch	8	Batch	8	Batch	8
mAP50-95	0,1378	mAP50-95	0,1693	mAP50-95	0,1384
mAP50	0,2539	mAP50	0,3055	mAP50	0,2684
mAP75	0,1277	mAP75	0,1606	mAP75	0,1228
AP per class		AP per class		AP per class	
bike	0,0005	bike	0,0006	bike	0,0001
car	0,4616	car	0,5082	car	0,4261
light	0,0607	light	0,0947	light	0,0925
motor	0,0939	motor	0,1384	motor	0,1231
other vehicle	0,0493	other vehicle	0,0705	other vehicle	0,0412
person	0,4182	person	0,47966	person	0,40331
rider	0,0053	rider	0,0178	rider	0,0110
sign	0,0760	sign	0,1150	sign	0,0749
truck	0,0750	truck	0,0988	truck	0,0734

Figura 4.17: Risultati test dei modelli addestrati sul primo dataset aumentato

Notiamo che tutte le metriche, ad eccezione della mAP50-90 di YOLO-World, sono peggiorate con l’addestramento sul dataset aumentato; da questo evinciamo che la combinazione di trasformazioni utilizzate come augmentations è stata inefficace. Ciò è dovuto al fatto che alcune o tutte le trasformazioni erano inadeguate per il contesto delle immagini termiche: potrebbero aver modificato eccessivamente l’immagine o applicato trasformazioni incoerenti rispetto al dataset originario, come il Vertical Flip o la Rotation con un range di 180 gradi, compromettendo il "significato" delle immagini (ad esempio ribaltando macchine o altri elementi stradali, cosa che non accade nei dati catturati dalle termocamere).

yolov8n				
		Dataset: augmented-data-1	Epochs: 50	Batch: 8
mAP50-90	0,1378	+1393,09%	compared to untrained	
		-3,03%	compared to filtered-data training	
mAP50	0,2539	+1482,54%	compared to untrained	
		-3,00%	compared to filtered-data training	
mAP75	0,1277	+1510,81%	compared to untrained	
		-4,98%	compared to filtered-data training	

yolov8s-world				
		Dataset: augmented-data-1	Epochs: 50	Batch: 8
mAP50-90	0,1693	+1857,28%	compared to untrained	
		+0,23%	compared to filtered-data training	
mAP50	0,3055	+1897,01%	compared to untrained	
		-0,25%	compared to filtered-data training	
mAP75	0,1606	+1765,05%	compared to untrained	
		-0,67%	compared to filtered-data training	

rtdetr-l				
		Dataset: augmented-data-1	Epochs: 50	Batch: 8
mAP50-90	0,1384	+5137,44%	compared to untrained	
		-23,96%	compared to filtered-data training	
mAP50	0,2684	+3967,83%	compared to untrained	
		-19,81%	compared to filtered-data training	
mAP75	0,1228	+7259,16%	compared to untrained	
		-27,01%	compared to filtered-data training	

Figura 4.18: Confronto tra i risultati dei test dei modelli addestrati sul primo dataset aumentato e quelli dei precedenti test



yolov8n



yolov8s-world



rtdetr-l

Figura 4.19: Detections effettuate dai modelli addestrati sul primo dataset aumentato

### 4.5.3 Seconda Data Augmentation

#### Secondo dataset aumentato

Visto il peggioramento delle prestazioni a seguito della prima data augmentation, sono andato a generare un secondo dataset aumentato tramite una nuova combinazione di trasformazioni mirata ad essere più coerente con il contesto delle immagini termiche:

```
transform = A.Compose([
    A.Blur(blur_limit=3, p=0.5),
    A.CLAHE(clip_limit=2, p=0.5),
    A.GaussianBlur(blur_limit=(3, 7), p=0.5),
    A.GaussNoise(var_limit=(10.0, 50.0), p=0.5),
    A.MultiplicativeNoise(multiplier=(0.9, 1.1), p=0.5),
    A.RandomBrightnessContrast(brightness_limit=0.2, contrast_limit=0.2, p=0.5),
    A.Rotate(limit=10, p=0.5),
    A.Sharpen(alpha=(0.2, 0.5), lightness=(0.5, 1.0), p=0.5),
    A.ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.1, rotate_limit=10, p=0.5)
], bbox_params=A.BboxParams(format='yolo', label_fields=['class_labels']))
```

Figura 4.20: Definizione della nuova trasformazione di Albumentations da applicare alle immagini

Perciò, grazie allo stesso script Python utilizzato precedentemente, ho generato il dataset `augmented-data-2`.



Figura 4.21: Esempio di immagine trasformata con la seconda data augmentation

### Test modelli addestrati sul secondo dataset aumentato

A seguito del terzo addestramento, eseguito su `augmented-data-2` nelle medesime condizioni (50 epochs e batch size di 8), ho ottenuto i seguenti risultati dai test:

yolov8n		yolov8s-world		rtdetr-l	
Dataset	augmented-data-2	Dataset	augmented-data-2	Dataset	augmented-data-2
Epochs	50	Epochs	50	Epochs	50
Batch	8	Batch	8	Batch	8
mAP50-95	0,1526	mAP50-95	0,1707	mAP50-95	0,1599
mAP50	0,2826	mAP50	0,3110	mAP50	0,3065
mAP75	0,1431	mAP75	0,1608	mAP75	0,1462
AP per class		AP per class		AP per class	
bike	0,0026	bike	0,0014	bike	0,0004
car	0,4715	car	0,5166	car	0,4672
light	0,0503	light	0,0832	light	0,1147
motor	0,1097	motor	0,1251	motor	0,1318
other vehicle	0,0964	other vehicle	0,0983	other vehicle	0,0366
person	0,4417	person	0,48072	person	0,44891
rider	0,0082	rider	0,0177	rider	0,0219
sign	0,0715	sign	0,0871	sign	0,1077
truck	0,1215	truck	0,1265	truck	0,1103

Figura 4.22: Risultati test dei modelli addestrati sul secondo dataset aumentato

Qua possiamo notare che i tre modelli si comportano diversamente:

- YOLOv8 ha un significativo miglioramento delle metriche rispetto ad entrambi gli addestramenti precedenti.
- YOLO-World ha un leggero incremento generale della mAP50-90 e della mAP50, mentre la mAP75 migliora rispetto a quella ottenuta a seguito dell fine-tuning con la prima data augmentation, ma rimane comunque inferiore a quella ottenuta con l'addestramento su dataset non aumentato.
- RT-DETR si comporta come nel precedente caso, mantendo tutte le metriche inferiori al primo addestramento su `filtered-data`, pur mostrando miglioramenti rispetto alla prima data augmentation.

yolov8n				
		Dataset: augmented-data-2	Epochs: 50	Batch: 8
mAP50-90	0,1526	+1553,22%	compared to untrained	
		+7,37%	compared to filtered-data training	
		+10,72%	compared to augmented-data-1 training	
mAP50	0,2826	+1661,49%	compared to untrained	
		+7,97%	compared to filtered-data training	
		+11,31%	compared to augmented-data-1 training	
mAP75	0,1431	+1704,28%	compared to untrained	
		+6,43%	compared to filtered-data training	
		+12,01%	compared to augmented-data-1 training	

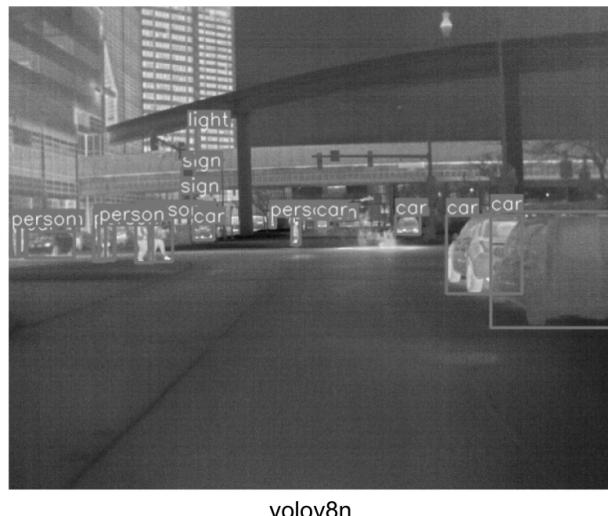
  

yolov8s-world				
		Dataset: augmented-data-2	Epochs: 50	Batch: 8
mAP50-90	0,1707	+1874,16%	compared to untrained	
		+1,09%	compared to filtered-data training	
		+0,86%	compared to augmented-data-1 training	
mAP50	0,3110	+1932,54%	compared to untrained	
		+1,52%	compared to filtered-data training	
		+1,78%	compared to augmented-data-1 training	
mAP75	0,1608	+1767,76%	compared to untrained	
		-0,53%	compared to filtered-data training	
		+0,15%	compared to augmented-data-1 training	

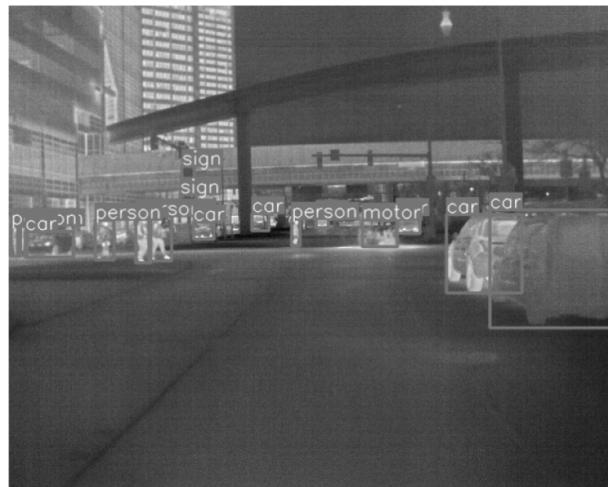
  

rtdetr-l				
		Dataset: augmented-data-2	Epochs: 50	Batch: 8
mAP50-90	0,1599	+5951,82%	compared to untrained	
		-12,13%	compared to filtered-data training	
		+15,55%	compared to augmented-data-1 training	
mAP50	0,3065	+4544,11%	compared to untrained	
		-8,46%	compared to filtered-data training	
		+14,17%	compared to augmented-data-1 training	
mAP75	0,1462	+8661,82%	compared to untrained	
		-13,10%	compared to filtered-data training	
		+19,06%	compared to augmented-data-1 training	

Figura 4.23: Confronto tra i risultati dei test dei modelli addestrati sul secondo dataset aumentato e quelli dei precedenti test



yolov8n



yolov8s-world



rtde-tr-l

Figura 4.24: Detections effettuate dai modelli addestrati sul secondo dataset aumentato

## Ulteriore addestramento sul secondo dataset aumentato

Per confermare i risultati ottenuti nei test precedenti, ho effettuato un ulteriore addestramento sul dataset `augmented-data-2`, aumentando il numero di epochs a 100. I test effettuati al termine di esso hanno prodotto i seguenti risultati:

yolov8n		yolov8s-world		rtdetr-l	
Dataset	augmented-data-2	Dataset	augmented-data-2	Dataset	augmented-data-2
Epochs	100	Epochs	100	Epochs	100
Batch	8	Batch	8	Batch	8
mAP50-95	0,1618	mAP50-95	0,1732	mAP50-95	0,1428
mAP50	0,3092	mAP50	0,3176	mAP50	0,2879
mAP75	0,1594	mAP75	0,1647	mAP75	0,1247
AP per class		AP per class		AP per class	
bike	0,0059	bike	0,0009	bike	0,0062
car	0,4849	car	0,5170	car	0,4119
light	0,0705	light	0,0882	light	0,0791
motor	0,1312	motor	0,1358	motor	0,1449
other vehicle	0,0953	other vehicle	0,1093	other vehicle	0,0286
person	0,4657	person	0,4807	person	0,3918
rider	0,0138	rider	0,0162	rider	0,0238
sign	0,0748	sign	0,0829	sign	0,0848
truck	0,1139	truck	0,1276	truck	0,1140

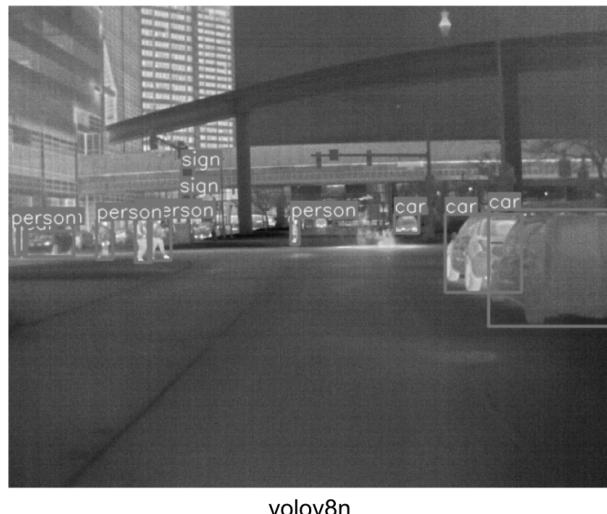
Figura 4.25: Risultati test dei modelli addestrati per 100 epochs sul secondo dataset aumentato

Dai risultati ottenuti possiamo osservare che:

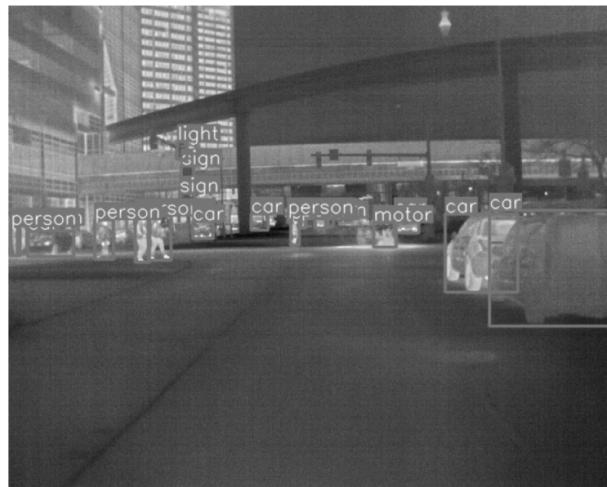
- YOLOv8 incrementa ulteriormente tutte le metriche di diversi punti percentuale rispetto ai precedenti addestramenti.
- YOLO-World incrementa tutte e tre le mAP, a differenza del precedente test in cui la mAP75 risultava inferiore a quella ottenuta con l'addestramento su `filtered-data`. Tuttavia, l'incremento è più contenuto rispetto a quello ottenuto con YOLOv8
- RT-DETR invece peggiora tutte le sue metriche rispetto al precedente addestramento di 50 epochs su `augmented-data-2`

yolov8n				
		Dataset: augmented-data-2	Epochs: 100	Batch: 8
mAP50-90	0,1618	+1653,15%	compared to untrained	
		+13,86%	compared to filtered-data training	
		+17,42%	compared to augmented-data-1 training	
		+6,04%	compared to augmented-data-2 training (50 epochs)	
mAP50	0,3092	+1826,99%	compared to untrained	
		+18,12%	compared to filtered-data training	
		+21,76%	compared to augmented-data-1 training	
		+9,40%	compared to augmented-data-2 training (50 epochs)	
mAP75	0,1594	+1909,98%	compared to untrained	
		+18,57%	compared to filtered-data training	
		+24,78%	compared to augmented-data-1 training	
		+11,40%	compared to augmented-data-2 training (50 epochs)	
yolov8s-world				
		Dataset: augmented-data-2	Epochs: 100	Batch: 8
mAP50-90	0,1732	+1902,75%	compared to untrained	
		+2,56%	compared to filtered-data training	
		+2,32%	compared to augmented-data-1 training	
		+1,45%	compared to augmented-data-2 training (50 epochs)	
mAP50	0,3176	+1975,52%	compared to untrained	
		+3,67%	compared to filtered-data training	
		+3,93%	compared to augmented-data-1 training	
		+2,11%	compared to augmented-data-2 training (50 epochs)	
mAP75	0,1647	+1812,50%	compared to untrained	
		+1,85%	compared to filtered-data training	
		+2,54%	compared to augmented-data-1 training	
		+2,40%	compared to augmented-data-2 training (50 epochs)	
rtdetr-l				
		Dataset: augmented-data-2	Epochs: 100	Batch: 8
mAP50-90	0,1428	+5303,50%	compared to untrained	
		-21,54%	compared to filtered-data training	
		+3,17%	compared to augmented-data-1 training	
		-10,71%	compared to augmented-data-2 training (50 epochs)	
mAP50	0,2879	+4263,09%	compared to untrained	
		-13,99%	compared to filtered-data training	
		+7,26%	compared to augmented-data-1 training	
		-6,05%	compared to augmented-data-2 training (50 epochs)	
mAP75	0,1247	+7377,41%	compared to untrained	
		-25,83%	compared to filtered-data training	
		+1,61%	compared to augmented-data-1 training	
		-14,66%	compared to augmented-data-2 training (50 epochs)	

Figura 4.26: Confronto tra i risultati dei test dei modelli addestrati per 100 epochs sul secondo dataset aumentato e quelli dei precedenti test



yolov8n



yolov8s-world



rtdeTR-l

Figura 4.27: Detections effettuate dai modelli addestrati per 100 epochs sul secondo dataset aumentato

## 4.6 Modello Ottimale

Analizzando i risultati di tutte le sperimentazioni, possiamo evincere che YOLOv8 è il modello che ottiene il maggior incremento nelle prestazioni a seguito della data augmentation effettuata sul dataset termico. Fino a questo punto, abbiamo addestrato la versione "Nano" del modello; quindi per avere una visione più completa delle potenzialità di YOLOv8, ho deciso di addestrare la versione "Large" del modello (**yolov8l** [13]) su `augmented-data-2`, mantenendo le 100 epochs. Seguono i risultati ottenuti dal test:

yolov8l		
Dataset	augmented-data-2	
Epochs	100	
Batch	8	
mAP50-95	0,2076	+28,25%
mAP50	0,3555	+14,99%
mAP75	0,1966	+23,35%
AP per class	bike	0,0082
	car	0,5714
	light	0,1128
	motor	0,1495
	other vehicle	0,1212
	person	0,5351
	rider	0,0374
	sign	0,1188
	truck	0,2138

Figura 4.28: Risultati test di **yolov8l** addestrato per 100 epochs sul secondo dataset aumentato a confronto con le metriche ottenute da **yolov8n** con lo stesso addestramento

Notiamo che, a parità di addestramento, con la versione "Large" del modello riusciamo ad ottenere un ulteriore incremento di tutte le metriche rispetto a quelle ottenute dalla "Nano".

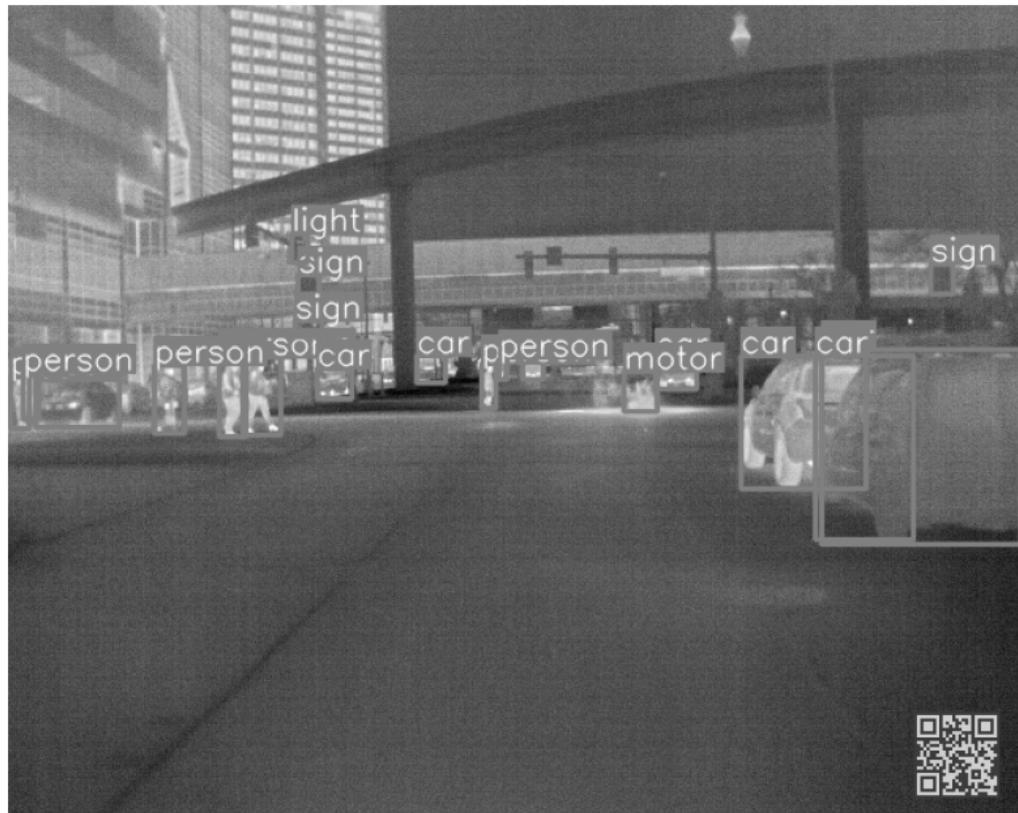


Figura 4.29: Detections effettuate da yolov8l addestrato per 100 epochs sul secondo dataset aumentato

# Conclusioni

A seguito degli esperimenti condotti, abbiamo osservato che tutti e tre i modelli (YOLOv8, YOLO-World e RT-DETR) hanno mostrato miglioramenti significativi nelle prestazioni quando sottoposti a un normale fine-tuning sul dataset termico. Tuttavia, un processo di data augmentation incoerente rispetto al dataset ha prodotto un generale abbassamento delle prestazioni, nonostante l'aumento della dimensione del subset di training.

Effettuando una data augmentation corretta e mirata per il dataset termico, solo YOLOv8 ha mostrato miglioramenti sostanziali, ragione per cui ritengo che la causa principale delle peggiori prestazioni di YOLO-World e RT-DETR sia la loro maggiore complessità, la quale potrebbe richiedere una gestione più sofisticata delle augmentations o una maggiore quantità di dati per ottenere miglioramenti significativi. Escludo che le performance inferiori dei due modelli siano dovute alle trasformazioni usate nella seconda data augmentation, in quanto si sarebbero dovuti manifestare peggioramenti anche nelle metriche di YOLOv8.

Grazie ai risultati ottenuti con la versione "Nano" di YOLOv8, abbiamo determinato che l'ottava versione di YOLO è la tipologia di modello che meglio si adatta all'Object Detection di immagini termiche. Pertanto, ho addestrato la versione "Large" di YOLOv8, ottenendo così il modello ottimale per l'obiettivo che ci eravamo prefissati.

# Bibliografia

- [1] Albumentations. <https://albumentations.ai/>. Accessed: 2024-07-07.
- [2] Comprehensive guide to ultralytics yolov5. <https://docs.ultralytics.com/yolov5/>. Accessed: 2024-07-07.
- [3] Dataset termico flir per l'addestramento di algoritmi. <https://www.flir.it/oem/adas/adas-dataset-form/>. Accessed: 2024-07-07.
- [4] Exploring object detection applications. <https://deeplobe.ai/exploring-object-detection-applications-and-benefits>. Accessed: 2024-07-07.
- [5] Getting started with ssd multibox detection. <https://it.mathworks.com/help/vision/ug/getting-started-with-ssd.html>. Accessed: 2024-07-07.
- [6] How single-shot detector (ssd) works? <https://developers.arcgis.com/python/guide/how-ssd-works>. Accessed: 2024-07-07.
- [7] Model training with ultralytics yolo. <https://docs.ultralytics.com/modes/train/>. Accessed: 2024-07-07.

- [8] Model validation with ultralytics yolo. <https://docs.ultralytics.com/modes/val/>. Accessed: 2024-07-07.
- [9] Rt-detr. <https://docs.ultralytics.com/models/rtdetr/>. Accessed: 2024-07-07.
- [10] Ultralytics. <https://www.ultralytics.com/it>. Accessed: 2024-07-07.
- [11] What is feature extraction? <https://www.educative.io/answers/what-is-feature-extraction>. Accessed: 2024-07-07.
- [12] Yolo-world model. <https://docs.ultralytics.com/models/yolo-world/>. Accessed: 2024-07-07.
- [13] Yolov8. <https://docs.ultralytics.com/models/yolov8/>. Accessed: 2024-07-07.
- [14] Tuning AI. Introduction to yolo: Revolutionizing real-time object detection. <https://www.linkedin.com/pulse/introduction-yolo-revolutionizing-real-time-object-detection>, 2023. Accessed: 2024-07-07.
- [15] Hong-Yuan Mark Liao Alexey Bochkovskiy, Chien-Yao Wang. Yolov4: Optimal speed and accuracy of object detection. 2020.
- [16] Shilpa Ananth. R-cnn for object detection. <https://towardsdatascience.com/r-cnn-for-object-detection-a-technical-summary-9e7bfa8a557c>, 2019. Accessed: 2024-07-07.
- [17] Ankushsharma. Yolo v1 architecture. <https://medium.com/@ankushsharma2805/yolo-v1-v2-v3-architecture-1ccac0f6206e>, 2020. Accessed: 2024-07-07.

- [18] Andrea Capuano. Extracting features from convolutional neural networks for image retrieval. <https://medium.com/arocketman/extracting-features-from-convolutional-neural-networks-for-image-retrieval-87b5127f8a92>, 2018. Accessed: 2024-07-07.
- [19] Hong-Yuan Mark Liao Chien-Yao Wang, Alexey Bochkovskiy. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. 2022.
- [20] Hongliang Jiang-Kaiheng Weng Chuyi Li, Lulu Li. Yolov6: A single-stage object detection framework for industrial applications. 2022.
- [21] Jacqueline Hong-Ahmad Daoudi Dillon Reis, Jordan Kupec. Real-time flying object detection with yolov8. 2023.
- [22] Rohit Gandhi. R-cnn, fast r-cnn, faster r-cnn, yolo — algoritmi di rilevamento oggetti. <https://deeplearningitalia.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-algoritmi-di-rilevamento-oggetti>. Accessed: 2024-07-07.
- [23] Shiguang Shan-Xilin Chen Gang Zhang, Meina Kan. Generative adversarial network with spatial attention for face attribute editing. 2018.
- [24] Ross Girshick. Fast r-cnn. 2015.
- [25] Ali Farhadi Joseph Redmon. Yolo9000: Better, faster, stronger. 2016.
- [26] Ali Farhadi Joseph Redmon. Yolov3: An incremental improvement. 2018.
- [27] Ross Girshick-Ali Farhadi Joseph Redmon, Santosh Divvala. You only look once: Unified, real-time object detection. 2016.

- [28] Rohit Kundu. Yolo: Algorithm for object detection explained. <https://www.v7labs.com/blog/yolo-object-detection>, 2023. Accessed: 2024-07-07.
- [29] Feng-Tsun Chien Liang-Yao Wang, Sau-Gee Chen. Robust deep convolutional neural network against image distortions. 2021.
- [30] Qunfeng Tan Mohamed Elgendi, Muhammad Umer Nasir. The effectiveness of image augmentation in deep learning networks for detecting covid-19: A geometric transformation perspective. 2021.
- [31] Joseph Redmon. How computers learn to recognize objects instantly. [https://www.ted.com/talks/joseph\\_redmon\\_how\\_computers\\_learn\\_to\\_recognize\\_objects\\_instantly](https://www.ted.com/talks/joseph_redmon_how_computers_learn_to_recognize_objects_instantly), 2017. Accessed: 2024-07-07.
- [32] Adrian Rosebrock. Intersection over union (iou) for object detection. <https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>, 2016. Accessed: 2024-07-07.
- [33] Trevor Darrell Jitendra Malik Ross Girshick, Jeff Donahue. Rich feature hierarchies for accurate object detection and semantic segmentation. 2014.
- [34] shafu.eth. R-cnn (object detection). <https://medium.com/@selfouly/r-cnn-3a9beddf55a>, 2019. Accessed: 2024-07-07.
- [35] Deval Shah. The essential guide to data augmentation in deep learning. <https://www.v7labs.com/blog/data-augmentation-guide>, 2022. Accessed: 2024-07-07.

- [36] Ross Girshick Jian Sun Shaoqing Ren, Kaiming He. Faster r-cnn: Towards real-time object detection with region proposal networks. 2016.
- [37] Sumit Singh. What is data augmentation. <https://www.labellerr.com/blog/what-is-data-augmentation-techniques-examples-benefits/>, 2022. Accessed: 2024-07-07.
- [38] Sumit Singh. Leveraging yolo object detection for accurate and efficient visual recognition. <https://www.labellerr.com/blog/why-is-the-yolo-algorithm-important/>, 2023. Accessed: 2024-07-07.
- [39] Jacob Solawetz. What is yolov5? a guide for beginners. <https://blog.roboflow.com/yolov5-improvements-and-evaluation/>, 2020. Accessed: 2024-07-07.
- [40] Jacob Solawetz. What is yolov8? the ultimate guide. <https://blog.roboflow.com/whats-new-in-yolov8/#yolov8-architecture-a-deep-dive>, 2023. Accessed: 2024-07-07.
- [41] Yixiao Ge Wenyu Liu Xinggang Wang Ying Shan Tianheng Cheng, Lin Song. Yolo-world: Real-time open-vocabulary object detection. 2024.
- [42] Dumitru Erhan Christian Szegedy Scott Reed Cheng-Yang Fu Alexander C. Berg Wei Liu, Dragomir Anguelov. Ssd: Single shot multibox detector. 2016.
- [43] Shangliang Xu Jinman Wei Guanzhong Wang Qingqing Dang Yi Liu Jie Chen Yian Zhao, Wenyu Lv. Detrs beat yolos on real-time object detection. 2023.

- [44] QiuHong Ke James Bailey Zihan Yang, Richard Sinnott. Individual feral cat identification through deep learning. 2021.