Politecnico di Torino

Corso di Laurea Magistrale in
Ingegneria Meccanica

A.A. 2025/2026
Sessione di Laurea settembre 2026

Metodologia e criteri di creazione di un dataset custom per object detection in un sistema di controllo in anello chiuso con feedback ottico

Relatori:
Prof. SORLI MASSIMO
Ing. MARRA VINCENZO
Ing. BERTOLINO ANTONIO CARLO
Ing. DE MARTIN ANDREA

Candidato:
Candita Daniele

SOMMARIO

La disponibilità di un dataset di immagini adeguatamente costruito rappresenta, in ogni applicazione di object detection basata su deep learning, il presupposto imprescindibile per le prestazioni del sistema finale: è attraverso di esso che la rete apprende a riconoscere l'oggetto di interesse in tutte le sue condizioni di presentazione. Quando l'oggetto da riconoscere è specifico di un'applicazione, come nel caso di un componente meccanico all'interno di un banco di prova, i dataset generici disponibili risultano insufficienti, rendendo necessaria la costruzione di un dataset custom.

Definire tale metodologia, dall'acquisizione delle immagini alla scelta delle strategie di annotazione, costituisce il nucleo centrale del presente lavoro di tesi, propedeutico allo sviluppo di un sistema di controllo posizione con feedback ottico basato su un algoritmo di object detection, quale alternativa ai tradizionali trasduttori di posizione.

L'attività è condotta su un banco prova a guida lineare, sul quale è montata una slitta libera di muoversi solo longitudinalmente, rispetto al quale si analizza la sostituzione del trasduttore con un sistema di visione basato su una rete YOLO v8n eseguita su hardware Raspberry Pi Compute Module 5. Tramite il framework di inferenza NCNN.

Si definiscono anzitutto tutti i passi seguiti per la creazione del dataset custom, dedicato al riconoscimento della slitta, comprendente le procedure di acquisizione delle immagini e la relativa strategia di annotazione. Si effettua quindi una selezione sistematica degli iperparametri di addestramento del modello, individuando la configurazione più efficiente in termini di accuratezza e costo computazionale. Si definisce e si implementa una pipeline di conversione e ottimizzazione del modello per l'esecuzione su hardware embedded, articolata su tre livelli di precisione a partire da quella di default FP32 fino a quella più aggressiva INT8, evidenziando le criticità introdotte dal processo stesso di quantizzazione.

Si effettua una valutazione sperimentale delle prestazioni del sistema così configurato, sia in ambiente PC sia sulla piattaforma target, analizzando vari aspetti, tra cui la stabilità delle stime di posizione o i possibili errori sistematici introdotti dal sensore camera.

RINGRAZIAMENTI

Desidero ringraziare il Prof. Massimo Sorli, l'Ing. Andrea De Martin e l'Ing. Antonio Bertolino per avermi offerto l'opportunità di partecipare a questo progetto di tesiche ha suscitato il mio interesse sin dall'inizio.

Ringrazio particolarmente l'Ing. Vincenzo Marra che mi ha accompagnato per tutta la durata della tesi, per la sua costante disponibilità e per le piacevoli chiacchiere prima dei laboratori.

Un ringraziamento speciale va ai miei genitori per avermi sostenuto e accompagnato in tutto il percorso di laurea con immenso sostegno e amore.

Un ringraziamento va a mio fratello che mi supporta e sopporta costantemente con il suo affetto e la streak di "gelatoo?" dopo cena.

Ringrazio zia Maria e Zia Rita per darmi sempre un appoggio morale e mangereccio una volta tornato a Torino (le polpette sono troppo buone).

Ringrazio Sonia per essermi stata a fianco dall'inizio della magistrale e aver condiviso quelle domeniche alla ricerca di libri.

Un ringraziamento va ai miei amici Valerio e Vittorio per aver accompagnato tutte le vacanze giù, e non solo, con i caffè e le partite di tennis e ping pong ai vantaggi interminabili.

Ringrazio Demian per essere stato un ottimo amico e collega universitario.

Ringrazio anche tutti i miei amici di villa, con cui ho iniziato questo percorso alla magistrale e aver condiviso ogni giorno i pasti in mensa.

Ringrazio soprattutto Mahdi e Masha con cui ho sempre speso del tempo di qualità assieme in tutti questi anni, dai laghi ad Avigliana al pasticciotto in centro a Lecce.

Infine, un grazie enorme va a Giusy e Silvio che mi hanno accolto fin dall'inizio e mi hanno accompagnato e sostenuto in tutto il percorso universitario. Mi hanno sempre fornito una prospettiva diversa della vita, affetto e conforto e spronato a diventare una versione migliore. Sono diventati la mia seconda famiglia e mi hanno dato sempre consigli utilissimi su ogni aspetto della vita.

A tutti vi voglio bene.

INDICE

INDICE DELLE FIGURE 7

INDICE DELLE TABELLE 9

1. INTRODUZIONE 10

1.1 Contesto 10

1.2 Descrizione banco prova 11

1.3 Breve storia di YOLO 13

1.4 Metriche di valutazione 13

1.4.1 Intersection over union (IoU) 14
1.4.2 Mean average precision (mAP) 14
1.4.3 Precision 14
1.4.4 Recall 15
1.4.5 Fitness 15
1.4.6 Funzioni di perdita 15
1.4.7 Tempo di inferenza o inference time (IT) 15
1.4.8 Giga Floating Point Operations (GFLOPs) 16

1.5 Obiettivi 16

2. CREAZIONE DEL DATASET CUSTOM 17

2.1 Analisi preliminare 17

2.1.1 Scelta del/dei target 17
2.1.2 Analisi della forma e variabilità del target 18
2.1.3 Analisi delle condizioni di ripresa video 18
2.1.4 Analisi delle condizioni operative 18
2.1.5 Formazione della tabella di acquisizione 19

2.2 Acquisizione video 20

2.2.1 Acquisizione effettiva 20

2.3 Fasi post acquisizione 23

2.3.1 Estrapolazione immagini da video e pulizia 23
2.3.2 Labelling delle immagini 24
2.3.3 Divisione tra split 27
2.3.4 Export del dataset 28

3. FASE DI TRAINING 29

3.1 Introduzione 29

3.2 Ambiente e architettura di training 30

3.2.1 Modello 30

3.3 Pianificazione della fase di training 30

3.4 Iperparametri 31

3.4.1 Batch 31
3.4.2 Imgsz 31
3.4.3 Device 32
3.4.4 Optimizer 32
3.4.5 Cache 32

3.4.6 Workers 32
3.4.7 Amp 32
3.4.8 Epochs 33
3.4.9 Patience 33
3.4.10 Deterministic 33
3.4.11 Seed 33

3.5 Analisi preliminare 34

3.6 Scelta dell'ottimizzatore 35
3.6.1 Dimensione 160x160 35
3.6.2 Dimensione 320x320 39
3.6.3 Dimensione 480x480 42
3.6.4 Dimensione 512x512 45
3.6.5 Dimensione 640x640 48

3.7 Fase di tuning 51
3.7.1 Learning rate iniziale (lr0) 51
3.7.2 Frazione finale del learning rate 52
3.7.3 Weight decay 52
3.7.4 Mosaic 52
3.7.5 Warm-up epochs 52
3.7.6 Risultati 53

3.8 Training finale e scelta del modello migliore pre-deploy 53
3.8.1 Dimensione 160x160 54
3.8.2 Dimensione 320x320 57
3.8.3 Dimensione 480x480 59
3.8.4 Dimensione 512x512 62
3.8.5 Dimensione 640x640 64
3.8.6 Conclusioni 66

4. FASE SPERIMENTALE (DEPLOY) 67

4.1 Stato dell'arte 67

4.2 Analisi su imgsz 68

4.3 Quantizzazione 70
4.3.1 Analisi da inferenza su PC Windows 75
4.3.2 Analisi da inferenza su Raspberry Pi 78

5. STATISTICHE DATASET E RISULTATI FINALI 85

5.1 Statistiche dataset nuovo 85

5.2 Differenze rispetto al dataset originario 87

5.3 Differenze nel training a 50 epoche (nuovo vs originario) 88

5.4 Differenze per l'inferenza (nuovo vs originario) 89

6. CONCLUSIONI 91

INDICE DELLE FIGURE

Figura 1 banco prova in esame ____ *11*
Figura 2 ingrandimento slitta con relativa guida ____ *12*
Figura 3 cattura delle tre angolazioni del target ____ *20*
Figura 4 posizionamento del sensore rispetto al banco prova ____ *21*
Figura 5 sfondi usati per le acquisizioni ____ *21*
Figura 6 slitta nelle posizioni di fine corsa sinistro, fine corsa destro e posizione centrale ____ *22*
Figura 7 scatti a diversa risoluzione ____ *22*
Figura 8 schermata scelta frame rate ____ *24*
Figura 9 esempio di bounding box sul target ____ *25*
Figura 10 esempio schermata per la creazione del bounding box ____ *25*
Figura 11 esempio di bounding box con occlusione parziale del target ____ *26*
Figura 12 esempio di occlusione totale del target ____ *26*
Figura 13 esempio di bounding box con illuminazione intensa e scarsa ____ *27*
Figura 14 esempio di target fuori inquadratura ____ *27*
Figura 15 training dfl/loss a imgsz 160 ____ *37*
Figura 16 training box/loss a imgsz 160 ____ *37*
Figura 17 validation dfl/loss a imgsz 160 ____ *38*
Figura 18 validation box/loss a imgsz 160 ____ *38*
Figura 19 training dfl/loss a imgsz 320 ____ *40*
Figura 20 training box/loss a imgsz 320 ____ *40*
Figura 21 validation dfl/loss a imgsz 320 ____ *41*
Figura 22 validation box/loss a imgsz 320 ____ *41*
Figura 23 training box/loss a imgsz 480 ____ *43*
Figura 24 training dfl/loss a imgsz 480 ____ *43*
Figura 25 validation box/loss a imgsz 480 ____ *44*
Figura 26 validation dfl/loss a imgsz 480 ____ *44*
Figura 27 training box/loss a imgsz 512 ____ *46*
Figura 28 training dfl/loss a imgsz 512 ____ *46*
Figura 29 validation box/loss a imgsz 512 ____ *47*
Figura 30 validation dfl/loss a imgsz 512 ____ *47*
Figura 31 training box/loss a imgsz 640 ____ *49*
Figura 32 training dfl/loss a imgsz 640 ____ *49*
Figura 33 validation box/loss a imgsz 640 ____ *50*
Figura 34 validation dfl/loss a imgsz 640 ____ *50*
Figura 35 F1-confidence per imgsz 160 ____ *54*
Figura 36 precision-confidence per imgsz 160 ____ *54*
Figura 37 recall-confidence per imgsz 160 ____ *55*
Figura 38 precision-recall per imgsz 160 ____ *55*
Figura 39 F1-confidence per imgsz 320 ____ *57*
Figura 40 precision-confidence per imgsz 320 ____ *57*
Figura 41 precision-recall per imgsz 320 ____ *58*
Figura 42 recall-confidence per imgsz 320 ____ *58*
Figura 43 F1-confidence per imgsz 480 ____ *59*
Figura 44 precision-confidence per imgsz 480 ____ *60*
Figura 45 precision-recall per imgsz 480 ____ *60*
Figura 46 recall-confidence per imgsz 480 ____ *61*
Figura 47 F1-confidence per imgsz 512 ____ *62*
Figura 48 precision-confidence per imgsz 512 ____ *62*
Figura 49 recall-confidence per imgsz 512 ____ *63*
Figura 50 F1-confidence per imgsz 640 ____ *64*

Figura 51 precision-confidence per imgsz 640 ____ *64*
Figura 52 recall-confidence per imgsz 640 ____ *65*
Figura 53 matrice di confusione per imgsz 640 ____ *65*
Figura 54 boxplot del tempo di inferenza al variare della risoluzione ____ *69*
Figura 55 boxplot del tempo di inferenza con modifica hardware ____ *70*
Figura 56 bounding box con modello a 32-bit floating point ____ *72*
Figura 57 bounding box con modello a 8-bit integer ____ *73*
Figura 58 boxplot IoU vs quantizzazione rispetto a risoluzione massima 640x640 ____ *75*
Figura 59 plot IoU-confidence ____ *76*
Figura 60 plot FPS vs IoU ____ *77*
Figura 61 influenza della distanza dal target e risoluzione finestra video ____ *80*
Figura 62 andamento della larghezza della bbox rispetto alla posizione longitudinale del centro ____ *83*
Figura 63 andamento dell'altezza della bbox rispetto alla posizione longitudinale del centro ____ *84*
Figura 64 esempio immagini con sfocatura lieve dovuta all'autofocus ____ *85*
Figura 65 esempio immagini con illuminazione intensa ____ *86*
Figura 66 esempio immagini con illuminazione debole ____ *86*
Figura 67 esempio immagini con occlusione parziale ____ *86*
Figura 68 confronto finale tra modello nuovo e originario ____ *90*

INDICE DELLE TABELLE

Tabella 1 riassunto condizioni operative 20
Tabella 2 analisi preliminare: mAP@50-95 e GLFOPs ad imgsz e modelli differenti 34
Tabella 3 metriche finali a diversi optimizer a diversi patience a imgsz 160 36
Tabella 4 metriche finali a diversi optimizer a diversi patience a imgsz 320 39
Tabella 5 metriche finali a diversi optimizer a diversi patience a imgsz 480 42
Tabella 6 metriche finali a diversi optimizer a diversi patience a imgsz 512 45
Tabella 7 metriche finali a diversi optimizer a diversi patience a imgsz 640 48
Tabella 8 metriche risultanti in fase di training per imgsz 51
Tabella 9 risultati tuning iperparametri per imgsz 53
Tabella 10 influenza della distanza sulla detection 79
Tabella 11 influenza della distanza sulle tre fasi della detection modello fp16 80
Tabella 12 influenza della distanza sulle tre fasi della detection modello mixed fp16 81
Tabella 13 confronto dei risultati del training a 50 epoche mediante i due dataset 88
Tabella 14 confronto finale nuovo vs originario 90

1.INTRODUZIONE

1.1 Contesto

L'object detection costituisce uno dei compiti fondamentali della computer vision. Il suo obiettivo consiste nell'individuare e localizzare gli oggetti presenti in un'immagine, assegnando a ciascuno una classe semantica, definita in fase di preparazione del dataset, e un bounding box che ne identifichi la posizione. La creazione di un dataset costituisce la base di ogni progetto in questo ambito, poiché la qualità e la completezza dei dati di addestramento influenzano direttamente le prestazioni del modello.

In letteratura si distinguono due categorie principali di dataset. La prima comprende raccolte di uso generale, costruite nel tempo da numerosi gruppi di ricerca a livello mondiale e contenenti milioni di immagini annotate: un esempio emblematico è il dataset COCO (common objects in context) [1], ampiamente adottato per applicazioni eterogenee quali il rilevamento di persone, il controllo del traffico e la sorveglianza e dataset di default per l'addestramento delle reti YOLO di cui si fa uso in questo lavoro di tesi. La seconda categoria include dataset custom, sviluppati ad hoc per domini applicativi specifici non ancora coperti da raccolte esistenti: casi tipici sono il monitoraggio dello stato sanitario di una coltivazione [2], la classificazione della qualità di prodotti ortofrutticoli durante la raccolta meccanizzata o l'ispezione visiva in linee di produzione industriale [3].

Indipendentemente dalla categoria, tutti i dataset di rilievo condividono due caratteristiche fondamentali: completezza e rigore metodologico nella fase di acquisizione e annotazione. La definizione di una procedura strutturata per la creazione di un dataset custom è pertanto essenziale e dipende da molteplici fattori: la natura del target da rilevare, il numero di classi da distinguere, i vincoli hardware imposti dal dispositivo di deployment e le possibili estensioni future dello studio. In particolare, la complessità visiva del target determina il numero di immagini necessarie: soggetti ad elevata variabilità, come la figura umana, che può differire per statura, corporatura, abbigliamento e aspetto, richiedono raccolte più ampie e diversificate per garantire una copertura adeguata della distribuzione reale, si fa l'esempio di COCO che possiede circa 330 000 immagini riguardanti oggetti di tutti i giorni, animali e persone, mentre soggetti semplici e poco variabili richiedono campioni limitati per ottenere una detection affidabile, con dataset che possono variare da poche centinaia a qualche migliaia di immagini [1] [4] [5].

Il presente lavoro si inserisce in questo contesto e propone la metodologia adottata per la costruzione di un dataset custom finalizzato alla detection di una slitta collegata ad una motore elettrico tramite cinghia dentata, vincolata a traslare esclusivamente lungo l'asse orizzontale tramite una guida lineare, tutto ricavato da un meccanismo di scorrimento di una stampante. Vengono illustrati i criteri di acquisizione, annotazione e selezione delle immagini, nonché le ottimizzazioni apportate al modello e alla pipeline di inferenza per soddisfare i requisiti di latenza necessari a garantire una detection in tempo reale.

1.2 Descrizione banco prova

Il banchetto iniziale in questione è quello rappresentato nella figura sottostante.

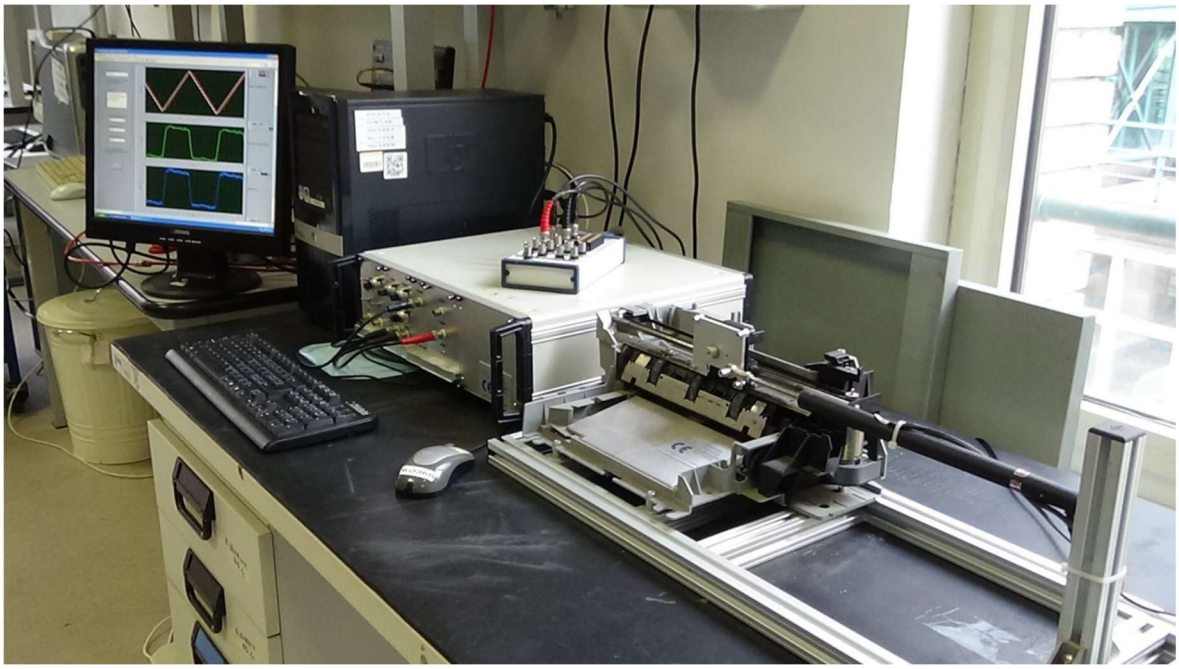


Figura 1 banco prova in esame

Dall'immagine si possono distinguere i vari componenti:

- la slitta montata su una guida orizzontale derivante da una classica stampante capace di scorrere da destra verso sinistra e viceversa grazie ad una trasmissione a cinghia dentata attuata da un motore elettrico;
- un trasduttore di posizione potenziometrico solidale alla slitta capace di rilevare la posizione della slitta;
- un sistema di acquisizione e controllo costituito dal rack di controllo analogico, una scheda NI myRio e un pc con interfaccia Labview.

Il target di questo lavoro di tesi e target del dataset creato è la slitta di cui ne viene riportato un ingrandimento.

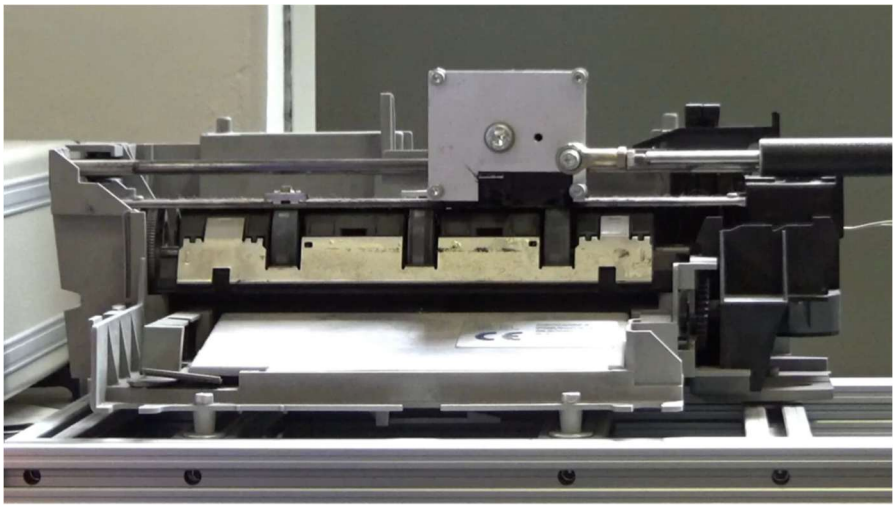


Figura 2 ingrandimento slitta con relativa guida

Oltre al banco riportato, viene impiegato un ulteriore dispositivo hardware per far girare l'algoritmo di detection YOLO ovvero il Raspberry Pi CM5 affiancato dal relativo sensore per le acquisizioni video e foto.

1.3 Breve storia di YOLO

Negli ultimi due decenni, il campo dell'object detection ha attraversato una trasformazione radicale, passando cronologicamente da metodi basati su feature manuali come SIFT (Scale-Invariant Feature Transform), sviluppato nel 1999 [6], e HOG (Histogram of Oriented Gradients), proposto nel 2005 [7], a framework di deep learning, in grado di apprendere direttamente le rappresentazioni gerarchiche dei dati.

HOG e SIFT sono due algoritmi classici di estrazione delle caratteristiche nel computer vision per il riconoscimento e il rilevamento di oggetti. Entrambi si basano sull'analisi dei gradienti dell'immagine, ma differiscono per approccio e applicazione.

- HOG è un descrittore che conta l'orientamento dei gradienti in celle locali di una griglia densa, normalizzando localmente il contrasto per essere invariante all'illuminazione. È particolarmente efficace per oggetti rigidi e rilevamento pedonale, poiché preserva le informazioni spaziali sulla forma dell'oggetto.
- SIFT è un descrittore che identifica punti chiave distinti in diverse scale e orientamenti, risultando invariante a scala, rotazione, illuminazione e deformazioni affini. È idealmente adatto per l'accoppiamento di regioni locali tra immagini diverse, come nel riconoscimento di oggetti in ambienti complessi o nella ricostruzione 3D.

Successivamente nel 2012, le reti CNN (Convolutinal Neural Network) rappresentano la svolta in questo settore grazie ad Alexnett che ridusse nettamente il tasso di errore nella classificazione di Imagenet rispetto ai metodi precedenti [8]. Le inferenze eseguite con questa architettura non erano tuttavia sufficientemente rapide tali da poter essere usate in condizioni di tempo reale.

Le successive Fast R-CNN [9] e Faster R-CNN [10] abbattono questo limite e abbassano i tempi di inferenza. Questi ultimi due detector a due stadi (two-stage detector) rimangono i più accurati per ottenere inferenze su oggetti piccoli e scene complesse se i tempi di inferenza non sono troppo stringenti.

Per aumentare ulteriormente la velocità di inferenza, sono stati sviluppati i detector a singolo stadio (one-stage detector), reti come SSD [11], RetinaNet [12] e tutte le reti della famiglia YOLO sviluppate a partire dal 2016 [13]. Tutti i detector a singolo stadio riscrivono il riconoscimento di un target in un'immagine come un problema di regressione su una griglia spaziale, producendo come output una localizzazione e classificazione in un'unica passata in avanti. Ciò ha reso le reti YOLO estremamente veloci. Questo ha portato ad un utilizzo massiccio di queste reti che man mano, con le varie versioni sino alla più recente YOLO 26 [14], hanno ottenuto valori sempre più vicini, in accuratezza, ai detector a due stadi rendendosi molto utili o necessari per deploy su dispositivi embedded.

1.4 Metriche di valutazione

La valutazione di un modello di object detection richiede l'impiego di metriche specifiche in grado di quantificare sia la correttezza della classificazione sia la precisione della localizzazione. Nel presente lavoro, le metriche considerate si distinguono in base alla fase in cui vengono impiegate: durante il training sono state utilizzate la intersection over union

(IoU), la mAP@50, la mAP@50-95, la precision, il recall, la fitness e le curve di loss; in fase di deploy e inferenza sono state invece considerate la latenza, espressa in millisecondi, e gli FPS (frame per second).

1.4.1 Intersection over union (IoU)

Il concetto alla base delle metriche di localizzazione è l'Intersection over Union (IoU), che misura il grado di sovrapposizione tra il bounding box predetto dal modello e quello reale, noto come ground truth. È definita come:

$$IoU = \frac{area\ di\ intersezione}{area\ totale}$$

dove l'intersezione rappresenta l'area comune tra i due bounding box e l'unione rappresenta l'area totale coperta da entrambi. Il valore di IoU è compreso tra 0 e 1, dove 0 indica assenza di sovrapposizione e 1 indica una corrispondenza perfetta. Una rilevazione viene considerata corretta (true positive) solo se il valore di IoU supera una soglia prestabilita: la convenzione standard, introdotta dal protocollo PASCAL VOC, fissa questa soglia tipicamente a 0,50 [15].

1.4.2 Mean average precision (mAP)

La mAP@50 (mean Average Precision con IoU uguale a 0.50) rappresenta la precisione media del modello calcolata considerando corrette le rilevazioni con un valore di Intersection over Union (IoU) pari o superiore a 0,50 rispetto alla ground truth [15]. Si tratta di una metrica ampiamente utilizzata in letteratura per una valutazione sommaria delle prestazioni di un modello di object detection.

La mAP@50-95 costituisce una valutazione più stringente, introdotta dal protocollo di valutazione COCO: si ottiene mediando l'Average Precision su un insieme di soglie IoU comprese tra 0,50 e 0,95 con incrementi di 0,05. A differenza della mAP@50, questa metrica valuta non solo la corretta identificazione dell'oggetto ma anche la precisione geometrica della bounding box predetto rispetto a quello reale, assumendo pertanto un peso maggiore nella valutazione complessiva del training.

1.4.3 Precision

La precision misura la percentuale di rilevazioni corrette sul totale delle rilevazioni effettuate dal modello. È definita come:

$$P = \frac{TP}{TP + FP}$$

dove TP (true positive) rappresenta il numero di rilevazioni corrette e FP (false positive) rappresenta il numero di falsi positivi riconosciuti. Valori prossimi all'unità indicano che il modello riconosce correttamente la quasi totalità dei target, con un numero contenuto di falsi positivi.

1.4.4 Recall

Il recall misura la capacità del modello di individuare tutti gli oggetti effettivamente presenti nelle immagini. È definita come:

$$R = \frac{TP}{TP + FN}$$

dove FN (false negative) rappresenta il numero di oggetti presenti ma non riconosciuti. Valori attorno all'unità significano che il modello tende a riconoscere correttamente la quasi totalità dei target cioè si hanno pochi falsi negativi.

1.4.5 Fitness

Il parametro di fitness è una combinazione pesata delle metriche descritte precedentemente e fornisce un'inquadratura sintetica della condizione della rete.

È calcolata secondo la seguente formula:

$$fitness = precision \times 0 + recall \times 0 + mAP@0.5 \times 0.1 + mAP@0.5{:}\,0.95 \times 0.9$$

In base all'applicazione della rete, i pesi possono essere modificati all'occorrenza.

1.4.6 Funzioni di perdita

Durante il training, le principali decisioni sono state guidate dall'analisi delle funzioni di perdita. La "box loss" quantifica l'errore nella localizzazione dei bounding box; la "classification loss" misura l'errore associato alla classificazione degli oggetti; la "distribution focal loss" (DFL) è impiegata per monitorare la precisione della regressione dei bounding box. L'andamento decrescente di queste curve nel corso delle epoche indica un training corretto, mentre deviazioni da questo comportamento possono segnalare fenomeni di underfitting o overfitting.

1.4.7 Tempo di inferenza o inference time (IT)

Infine, in fase di inferenza, la latenza per singolo frame o inference time (IT) rappresenta il tempo impiegato dal modello per elaborare un'immagine e restituire il risultato. Viene espressa in millisecondi ed è calcolata come somma dei tempi di pre-processing, inferenza e post-processing.

$$IT = preprocess + inference + postprocess$$

Il suo reciproco, moltiplicato per un fattore mille, fornisce il valore di FPS (frame per second), che esprime il numero di frame elaborati dal modello nell'unità di tempo.

$$FPS = \frac{1000}{IT}$$

1.4.8 Giga Floating Point Operations (GFLOPs)

I GFLOPs (Giga Floating Point Operations) quantificano il costo computazionale di una rete neurale, esprimendo il numero di miliardi di operazioni in virgola mobile (soprattutto addizioni e moltiplicazioni) necessarie per produrre un singola immagine. A differenza del tempo d'inferenza, che dipende dall'architettura dell'hardware a disposizione, i GFLOPs rappresentano una misura intrinseca e fissa dell'architettura, indipendente quindi dal dispositivo di esecuzione.

Nel presente lavoro tale metrica è stata utile per valutare la leggerezza computazionale della rete in vista del successivo deploy su dispositivo quale il Raspberry Pi facente parte del banco prova.

Nella fase di training, hanno avuto maggiore rilevanza la mAP@50-95, la fitness e le curve di loss per motivi che verranno riportati nel paragrafo dedicato completamente al training della rete.

1.5 Obiettivi

Il presente lavoro di tesi si propone di definire un metodo per la creazione di un dataset custom, impiegato per addestrare un modello YOLO v8n utilizzato come strumento per la stima della posizione del target, al fine di chiudere il loop di controllo posizione del banco prova in esame.

Il loop di controllo posizione è stato inizialmente realizzato mediante un trasduttore di posizione. Il presente lavoro si inserisce all'interno di uno studio di fattibilità riguardante la possibilità di sostituire tale trasduttore con un algoritmo di object detection basato su rete neurale YOLO, sfruttando la rilevazione di una slitta per ricavarne la posizione puntuale. A tal fine, oltre alla creazione del dataset, si rende necessaria un'analisi dei tempi di inferenza della rete, congiuntamente allo studio di alcuni parametri di training e di quantizzazione del modello.

L'analisi dei tempi di inferenza assume particolare rilevanza in quanto il ritardo introdotto dalla rete nel processo di detection si traduce direttamente in un ritardo nel loop di controllo, con conseguente degradazione delle prestazioni del sistema complessivo in termini di risposta in frequenza. In particolare, all'aumentare del tempo di inferenza, il sistema risulta progressivamente meno capace di seguire il segnale di riferimento in ingresso.

2.CREAZIONE DEL DATASET CUSTOM



La creazione di un dataset segue precisi passaggi, che portano a una raccolta di immagini il più possibile rappresentative della situazione che si intende far apprendere al modello incaricato di eseguire l'inferenza. Di seguito si riportano tutte le fasi seguite nella costruzione del dataset custom impiegato in questo lavoro.

2.1Analisi preliminare

Prima di procedere con le acquisizioni in laboratorio, è stata condotta un'analisi preliminare volta a identificare le condizioni che al meglio descrivono l'utilizzo del banco e in grado di migliorare le prestazioni di apprendimento della rete.

2.1.1Scelta del/dei target

Il target da detectare in questo lavoro di tesi è una slitta motorizzata tramite una cinghia dentata, azionata da un motore elettrico e vincolata a traslare esclusivamente lungo l'asse orizzontale per mezzo di una guida lineare. In vista del deployment su un dispositivo con risorse computazionali limitate, si è scelto di limitare il dataset a un'unica classe, corrispondente alla sola slitta, quindi, non sono state introdotte classi aggiuntive per la distinzione degli altri elementi del banco, come, ad esempio, la guida orizzontale oppure il trasduttore solidale alla slitta. Risulterebbe comunque interessante da questo punto di vista valutare il trade-off introdotto dall'aggiunta di una seconda classe che ha come target l'unico altro elemento traslante ovvero il trasduttore. L'aumento del numero delle classi, quindi degli oggetti da rilevare simultaneamente, comporta un lieve aumento dei tempi di inferenza e del carico computazionale assieme ad una riduzione della precisione media della rete [16].

Per questo motivo, nella scelta del numero delle classi occorre tenere conto delle risorse computazionali disponibili del dispositivo di deploy a disposizione, infatti, se occorresse estendere il dataset con classi aggiuntive, sarebbe necessario prevedere un corrispondente aumento della potenza di calcolo.

2.1.2Analisi della forma e variabilità del target

La slitta può essere descritta, da un punto di vista geometrico, come un parallelepipedo di metallo, descrivendola in modo semplicistico. Si tratta di un oggetto rigido: durante il normale funzionamento del banco non sono previste variazioni della sua forma, delle sue dimensioni o della sua colorazione. Questa caratteristica di invarianza geometrica e cromatica del target riduce sensibilmente la variabilità visiva da cui il modello deve generalizzare, coerentemente con quanto riportato nel paragrafo precedente circa la complessità visiva del target e numerosità del dataset richiesta.

2.1.3Analisi delle condizioni di ripresa video

Le acquisizioni sono state effettuate tramite sensori di immagine richiamati dal Raspberry Pi. In questo caso, i sensori a disposizione sono tre:

- IMX477, sensore da 12,3 MP, risoluzione massima 4056x3040 pixel [17]
- OG02B10, sensore a otturatore globale da 2 MP, risoluzione 1600x1300 pixel [18]
- IMX708, sensore da 12 MP, risoluzione massima 4608x2592 pixel integrato nella Camera Module 3 ufficiale di Raspberry Pi [19]

Una condizione da tenere in conto è la risoluzione operativa: ciascun sensore supporta diverse modalità di acquisizione, riportate nei rispettivi datasheet, ciascuna con una combinazione propria di risoluzione e frame rate massimo. Quindi, il dittongo camera-risoluzione può essere sfruttato per introdurre ulteriore variabilità nel dataset, ampliando la diversità delle condizioni di acquisizione a parità di target ripreso.

2.1.4Analisi delle condizioni operative

In questa sezione, sono state prese in considerazione tutte le possibili condizioni operative che possono verificarsi durante l'utilizzo del banco di prova.

Le condizioni principali individuate come rilevanti in questo caso sono le seguenti:

1. angolazione del sensore di acquisizione: il sensore deve essere posizionato in modo tale da riprendere il profilo destro, centrale e sinistro della slitta seguendo lo studio geometrico sviluppato nel precedente lavoro di tesi [20];
2. sfondo di inquadratura: gli sfondi impiegati durante le acquisizioni sono stati scelti tra quelli comunemente presenti in laboratorio, così da garantire una maggiore aderenza alle condizioni operative reali come monitor, foglio bianco, parete bianca o colorata, persona, mano, finestra, ecc.
3. presenza o meno del target: per un miglior apprendimento della rete, il dataset deve includere anche immagini in cui la slitta non è presente [21]
4. livelli di illuminazione: al fine di simulare le diverse condizioni di utilizzo del banco di lavoro durante la giornata, sono stati definiti tre livelli di illuminazione: scarsa, normale e intensa;
5. livelli di occlusione: pur non rappresentando condizioni operative tipiche, sono stati considerati diversi gradi di occlusione del target (25%, 50% e 75% della sua area

visibile dal sensore) con l'obiettivo di fornire alla rete esempi caratterizzati da differenti livelli di visibilità;

6. posizione della slitta: operativamente, devono essere introdotte acquisizioni della slitta distribuite lungo tutta la sua escursione lungo la relativa guida, dalla posizione di estrema sinistra a quella di estrema destra, al fine di rappresentare l'intero spazio di movimento del sistema;
7. distanza camera-target: per ciascun sensore sono state considerate le distanze operative individuate mediante lo studio geometrico del sistema e l'analisi dell'orientamento delle camere sviluppati nel lavoro di tesi precedente [20]. Tali distanze costituiscono i valori di riferimento da adottare durante la fase di acquisizione

Le condizioni operative riportate sono state pensate per favorire nella rete una migliore comprensione del target da rilevare, la slitta, e per massimizzare la varietà del dataset, migliorandone così la capacità di generalizzazione. A questo obiettivo contribuiscono anche le tecniche di data augmentation, applicate sia di default sia con configurazione manuale durante il training della rete YOLO v8n: tra le augmentation presenti si contano le trasformazioni geometriche (flip orizzontale, traslazione, scalatura), variazioni HSV di tonalità, saturazione e luminosità e la tecnica del mosaic, che combina quattro immagini in un unico frame composto durante le prime fasi del training [22].

Anche in presenza di immagini tra loro simili, queste tecniche risultano quindi utili per arricchire ulteriormente la variabilità osservata dal modello durante l'apprendimento.

2.1.5 Formazione della tabella di acquisizione

Le condizioni di acquisizione sono state raggruppate per formare una matrice delle condizioni operative riportata di seguito.

SENSORI	IMX 477	IMX 708	OG 02B10
RISOLUZIONE CAMERA	2028x1080 2028x1520 1332x990	1600x1300 1600x1080 1280x720	1080 720 480
LIVELLI LUCE	SCARSA	NORMALE	FORTE
ORIENTAMENTO	PROFILO SINISTRO	PROFILO CENTRALE	PROFILO DESTRO
DISTANZA CAMERA SLITTA	VICINO (45 cm)	NORMALE (60-75 cm)	LONTANO (125 cm)

SFONDO	MONITOR	MANO	BUSTO	FINESTRA	FOGLIO BIANCO

POSIZIONE SLITTA	ESTREMA SINISTRA	QUALSIASI POSIZIONE CENTRALE	ESTREMA DESTRA
OCCLUSIONE	TOTALE	PARZIALE	ASSENTE

Tabella 1 riassunto condizioni operative

2.2 Acquisizione video

Le acquisizioni per la creazione del dataset custom, funzionale all'addestramento della rete YOLO per svolgere il rilevamento della slitta, sono state eseguite in laboratorio utilizzando direttamente il terminale del banco di rilevamento ottico ovvero il Raspberry Pi CM5 a cui è collegato il sensore camera.

2.2.1 Acquisizione effettiva

A seguito dell'analisi preliminare delle condizioni utili alla creazione del dataset, le acquisizioni definitive sono state eseguite secondo una procedura parzialmente diversa rispetto a quanto inizialmente previsto.

Le acquisizioni sono state effettuate riprendendo comunque i tre profili (destro, sinistro e centrale), ma a distanze approssimative pari a 0,7 m, 1 m e 1,3 m, non rispettando quindi quanto riportato nel lavoro di tesi citato [20].

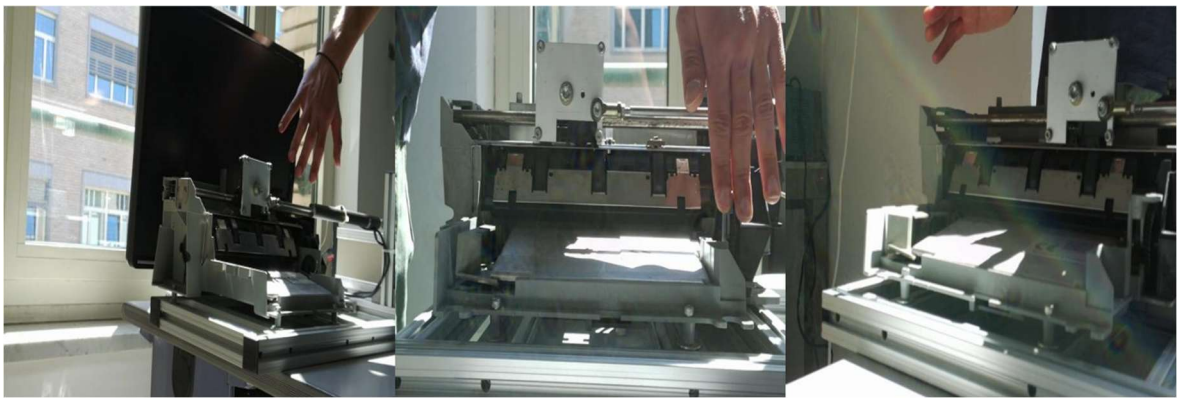


Figura 3 cattura delle tre angolazioni del target

Le acquisizioni sono state effettuate utilizzando un solo sensore, ovvero l'IMX708; gli altri due sensori a disposizione non sono stati impiegati a causa dell'assenza, al momento delle riprese, del supporto fisico necessario per il loro posizionamento sul tavolo.



Figura 4 posizionamento del sensore rispetto al banco prova

Le acquisizioni sono state eseguite utilizzando differenti tipologie di sfondo, tra cui una finestra, un foglio bianco, due persone con abiti di colore diverso (riprese limitatamente al busto) e un monitor. Durante le riprese, le mani sono state movimentate in modo casuale in prossimità del target (la slitta), sia frontalmente sia lateralmente, al fine di incrementare la variabilità del dataset.



Figura 5 sfondi usati per le acquisizioni

La maggior parte delle acquisizioni è stata effettuata impartendo al banco prova un comando con andamento sinusoidale, al fine di ottenere immagini con il target (la slitta) posizionato lungo l'intera estensione guida. Successivamente, sono state effettuate acquisizioni con la

slitta statica e un background in movimento per aumentare la variabilità delle acquisizioni stesse e evitare di dover successivamente scartare molteplici immagini doppioni.

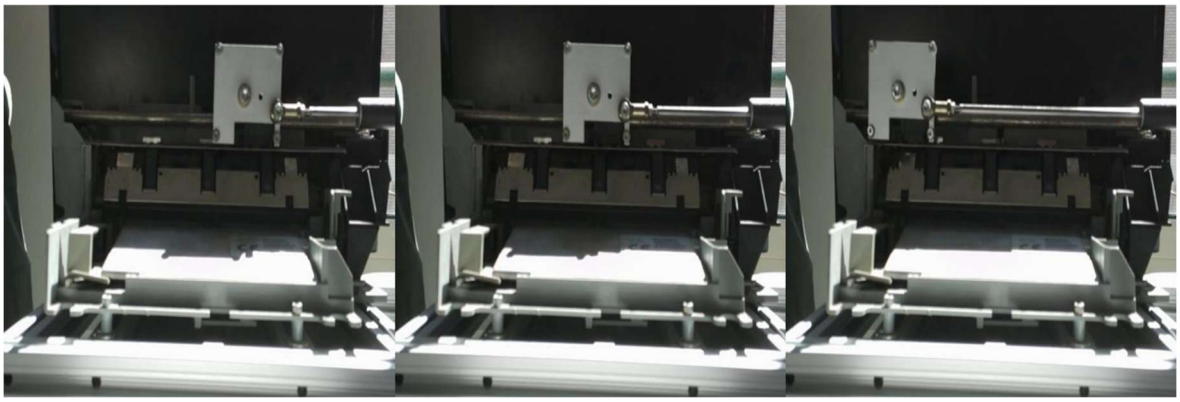


Figura 6 slitta nelle posizioni di fine corsa sinistro, fine corsa destro e posizione centrale

Tra le risoluzioni disponibili, per le acquisizioni in movimento è stata selezionata esclusivamente la risoluzione massima, in quanto una risoluzione inferiore avrebbe reso i bordi del target meno definiti nell'immagine acquisita, compromettendo la precisione dell'annotazione delle bounding box e, di conseguenza, le prestazioni del training della rete in termini di accuratezza di localizzazione [23]. Successivamente, sono stati effettuati alcuni scatti a risoluzione media e bassa.

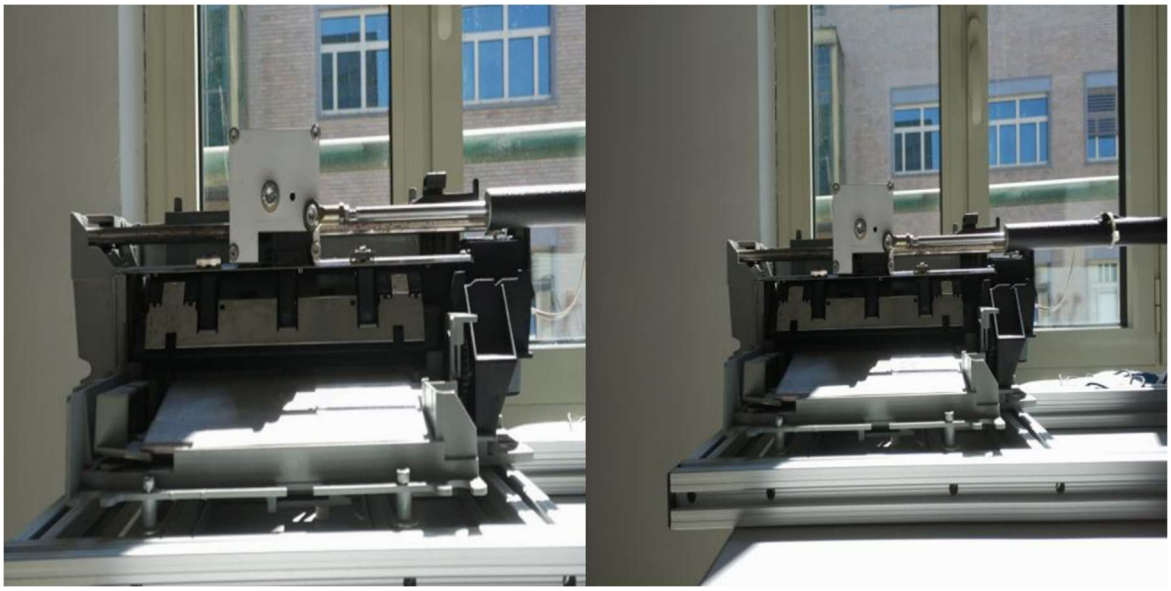


Figura 7 scatti a diversa risoluzione

Sono state tuttavia incluse nel dataset immagini in cui il target risulta sfocato, in quanto tale condizione riflette uno scenario operativo realistico: il sensore impiegato per le acquisizioni

è dotato di autofocus, che durante il funzionamento può non mantenere costantemente il fuoco sul target.

2.3 Fasi post acquisizione

Una volta completata l'acquisizione dei video, è stata definita una pipeline di generazione del dataset mediante la piattaforma Roboflow.

I video generati sono stati tutti post-processati frame per frame per ottenere le immagini necessarie per la costruzione del dataset.

Roboflow è una piattaforma end-to-end per la computer vision che consente di raccogliere e organizzare immagini, esplorare e ricercare dati visivi non strutturati, annotare immagini e creare dataset, esportare, addestrare e distribuire modelli di computer vision, nonché sfruttare l'active learning per migliorare il dataset nel tempo [24].

2.3.1 Estrapolazione immagini da video e pulizia

La prima fase ha riguardato l'estrazione dei frame dai video acquisiti, effettuata con frequenza variabile in funzione del contenuto informativo delle sequenze. In particolare, per segmenti caratterizzati da elevata variabilità del target, in termini di posizionamento e grado di occlusione, è stato adottato un frame rate di estrazione fino a 30 fps, al fine di preservare la massima quantità di informazione. Viceversa, per sequenze a bassa variabilità spaziale e temporale è stato impiegato un frame rate ridotto (fino a 5 fps), così da limitare la presenza di immagini con un contenuto informativo ridondante. Questa strategia di campionamento adattivo è coerente con un approccio consolidato nella costruzione di dataset da sorgenti video, secondo cui un frame rate di estrazione fisso tende a produrre immagini consecutive con differenze minime, a livello informativo, ai fini dell'addestramento, mentre un campionamento variabile, calibrato sull'effettiva dinamica di un video, consente di ridurre la ridondanza e di ottenere una rappresentatività più equilibrata del dataset [25].

La Figura 8 schermata scelta frame rate riporta un esempio di schermata derivante da Roboflow in fase di scelta del frame rate.

Una volta ottenute le immagini dai vari video, è stata condotta un'operazione di pulizia per tutte quelle immagini duplicate poiché, con un input di forma sinusoidale, è infatti frequente acquisire più frame con posizioni della slitta pressoché identiche.

Sono state eliminate, a questo scopo, circa cinquanta immagini di questo tipo.

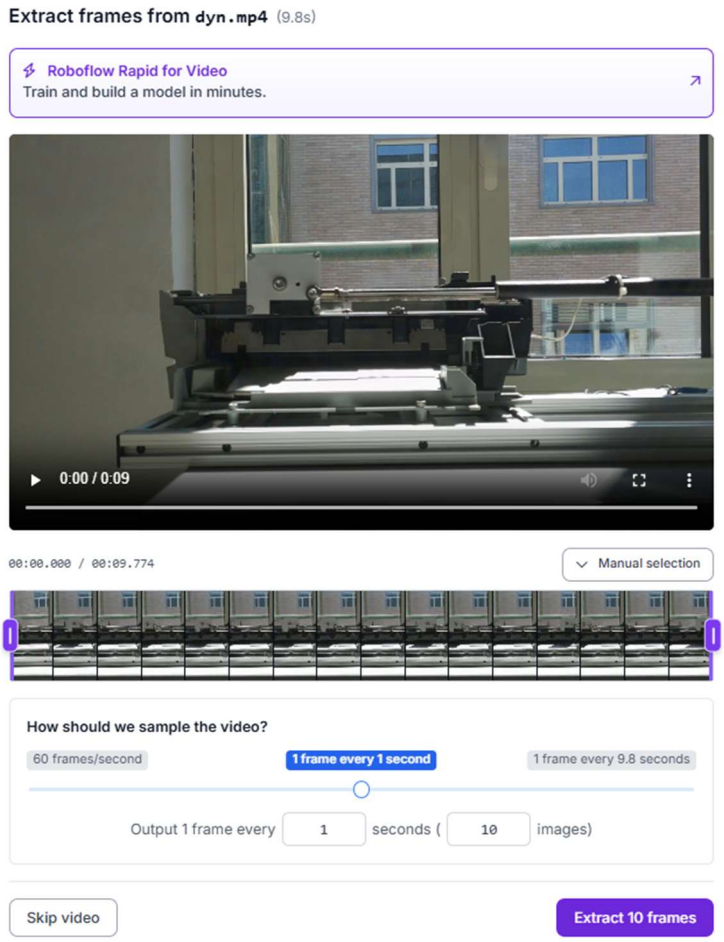


Figura 8 schermata scelta frame rate

2.3.2 Labelling delle immagini

La fase successiva ha riguardato l'annotazione del dataset. Per ciascuna immagine è stata definita una bounding box il più possibile aderente al target, a cui è stata associata la relativa etichetta di classe con nome" sled" derivante dall'etichetta scelta per il dataset su cui è stata allenata precedentemente la rete [20]. Tale fase è stata condotta sia manualmente sia mediante gli strumenti di assistenza basati su modelli di intelligenza artificiale integrati nella piattaforma Roboflow, con successiva validazione umana delle annotazioni generate automaticamente.

Il dataset è formato da una sola classe. La bounding box segue fedelmente l'intero profilo superiore della slitta, mentre il profilo inferiore è stato considerato come un semplice segmento rettilineo tale da formare la bounding box come in Figura 9 esempio di bounding box sul target.

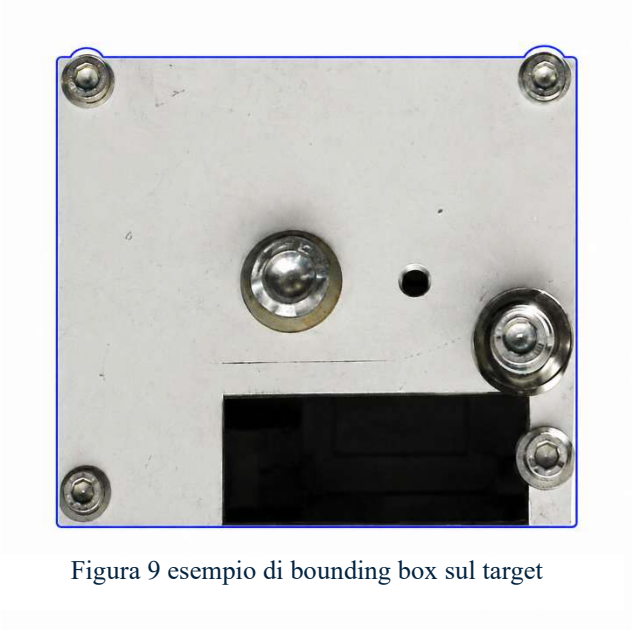


Figura 9 esempio di bounding box sul target

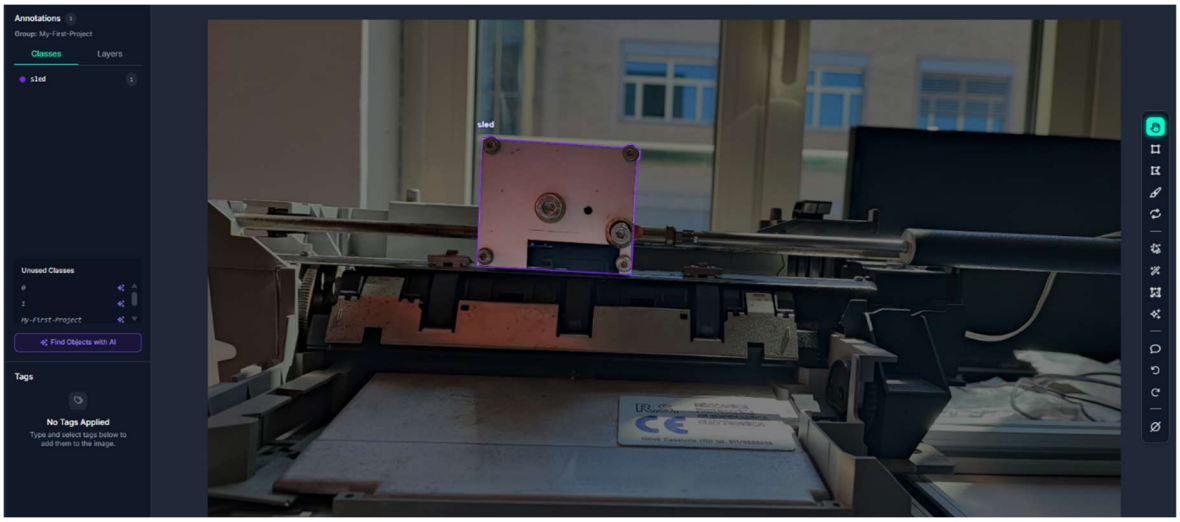


Figura 10 esempio schermata per la creazione del bounding box

Durante la fase di annotazione sono state adottate le seguenti semplici regole operative essenziali, volte a garantire coerenza nel dataset in presenza delle diverse condizioni descritte nell'analisi preliminare:

- in presenza di occlusione parziale del target, la bounding box è stata definita attorno alla forma intera della slitta, includendo quindi anche la porzione non visibile, al fine di migliorare la robustezza del modello anche in condizioni di visibilità ridotta (Figura 11 esempio di bounding box con occlusione parziale del target);

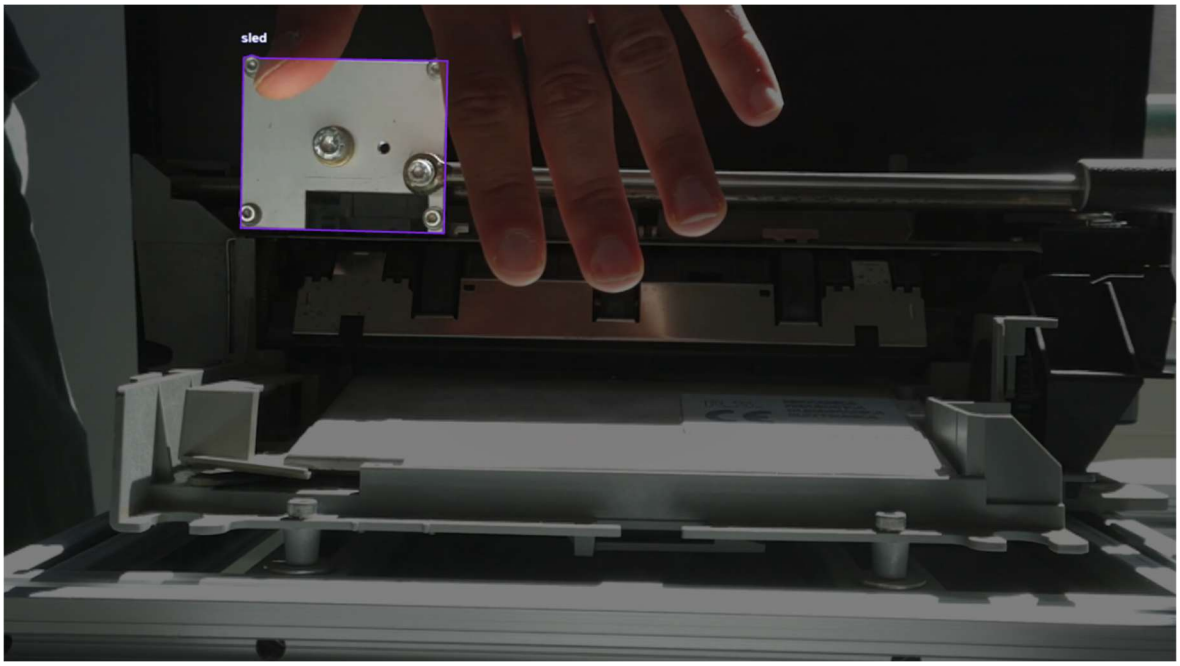


Figura 11 esempio di bounding box con occlusione parziale del target

- nei casi di occlusione totale, non è stato inserito alcun bounding box (Figura 12);



Figura 12 esempio di occlusione totale del target

- in condizioni di illuminazione non uniforme, la bounding box mantiene la forma intera della slitta, semplificando il profilo nelle zone in cui i contorni non sono chiaramente distinguibili a causa delle condizioni di luce avverse (Figura 13);

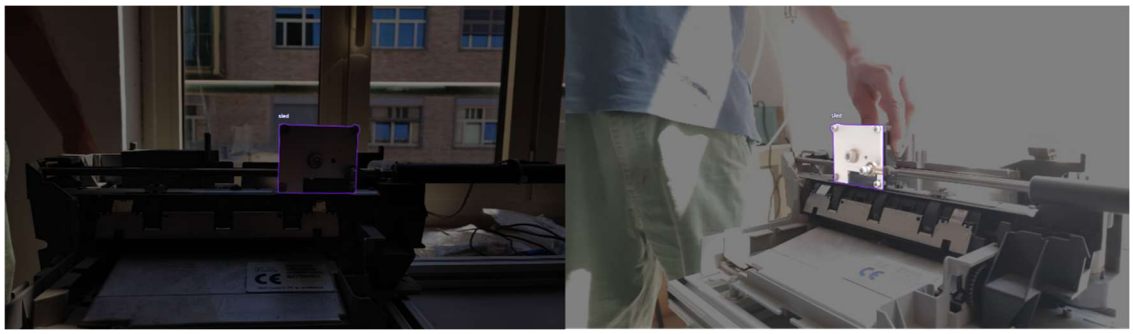


Figura 13 esempio di bounding box con illuminazione intensa e scarsa

- nei casi in cui la slitta risultasse parzialmente fuori dall'inquadratura, la bounding box è stata costruita attorno alla sola parte effettivamente visibile nell'immagine (Figura 14).

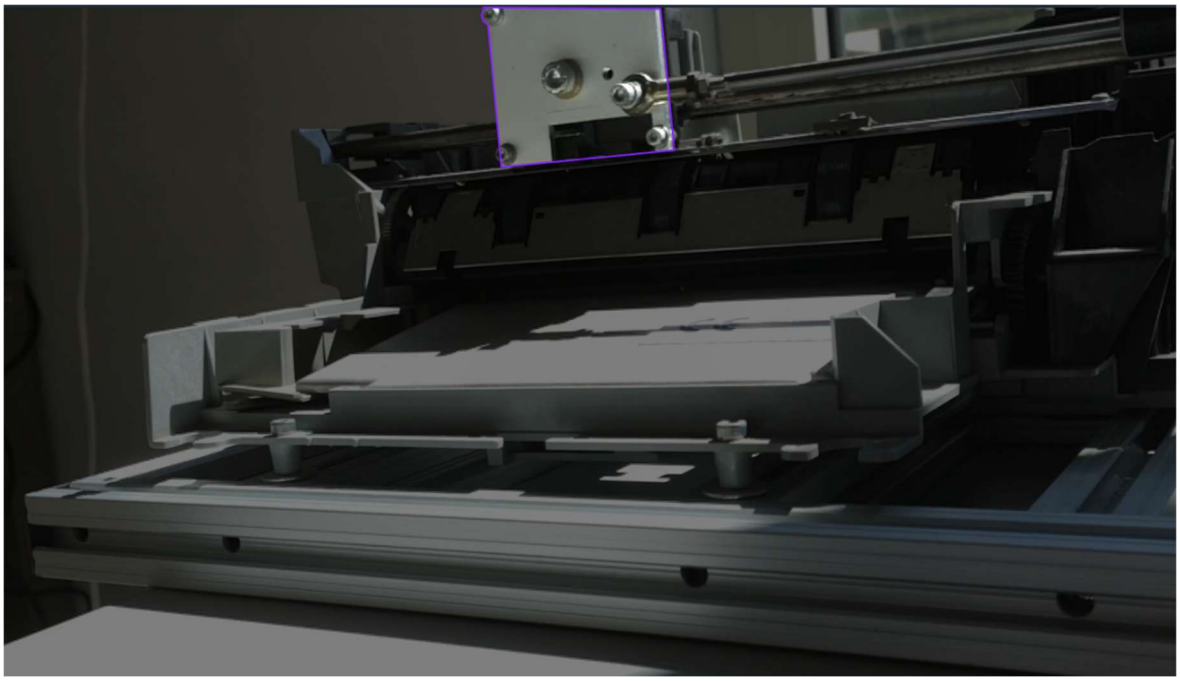


Figura 14 esempio di target fuori inquadratura

Le prime tre regole seguono una logica di annotazione amodale (l'annotatore ha stimato l'estensione completa del target anche quando è parzialmente occluso), mentre l'ultima regola, relativa alla troncatura ai bordi dell'inquadratura, adotta una logica di tipo modale, limitando la bounding box alla sola porzione visibile realmente.

2.3.3 Divisione tra split

Le ripartizioni più comuni per suddividere un dataset in training, validation e test per l'addestramento di modelli YOLO e più in generale di modelli di deep learning per la visione artificiale, sono tipicamente le seguenti:

- 60-80% per il training
- 10-20% per validation

- 10-20% per il test

Le suddivisioni 70/15/15 e 80/10/10 sono spesso valutate come standard di riferimento, ma la loro efficacia può rivelarsi tanto ottimale quanto anche controproducente a seconda del contesto: diversi studi che hanno confrontato più rapporti di split hanno rilevato variazioni significative delle metriche di accuratezza al variare della ripartizione, evidenziando che non esiste un rapporto universalmente ottimale e che una scelta inadeguata può portare a fenomeni di overfitting o underfitting [26] [27] [28]. La scelta dipende infatti dalla dimensione e complessità del dataset: per dataset piccoli e semplici, quindi nel caso da 500 fino a qualche migliaio di immagini, una ripartizione di 80/10/10 risulta generalmente preferibile. Per dataset di dimensioni maggiori e più complessi, quando si parla di centinaia di migliaia di immagini, è invece possibile raggiungere anche una ripartizione più spinta verso il training come 98/1/1, poiché anche una piccola percentuale di un dataset di grandi dimensioni è sufficiente a garantire stime affidabili delle metriche in fase di validazione e test.

Quando sono in gioco dataset con numerose immagini con molte più classi, la ripartizione delle classi nei vari split deve essere anche statisticamente uniforme, cioè che la proporzione di immagini appartenenti a ciascuna classe sia sostanzialmente la stessa nei tre split (campionamento stratificato). Questa pratica è particolarmente importante in presenza di classi sbilanciate, poiché previene una scarsa generalizzazione del modello: split non rappresentativi possono infatti portare a un deterioramento delle metriche su immagini "nuove", quindi mai osservate durante 'addestramento [29].

Nel caso in esame, considerato quanto riportato precedentemente, il dataset è stato suddiviso nei sottoinsiemi standard di training, validation e test secondo una ripartizione 80/10/10 cioè l'80% è stato destinato al training, il 10% alla validation e il restante 10% al test.

2.3.4 Export del dataset

Infine, per concludere il dataset deve essere esportato in formato compatibile con la rete che dovrà essere addestrata.

Nel caso in esame, il dataset annotato è stato esportato in formato compatibile con il framework di training della rete neurale YOLO v8n.

Le condizioni operative adottate durante le acquisizioni sono state scelte con l'obiettivo di massimizzare la variabilità del dataset, favorendo la capacità di generalizzazione del modello.

Le tecniche di data augmentation, un insieme di tecniche che consentono di ampliare artificialmente la diversità delle immagini disponibili, migliorando l'apprendimento del modello, non sono state applicate durante la fase di costruzione del dataset. La totalità delle immagini inserite nel dataset ha esclusivamente un solo target e di conseguenza una sola bounding box.

Le tecniche di augmentation sono state applicate esclusivamente durante la fase di training riportate nella sezione seguente.

3.FASE DI TRAINING

3.1 Introduzione

La fase di training costituisce un ruolo fondamentale di qualsiasi progetto di object detection poiché è in questa fase che il modello apprende, a partire dalle immagini annotate del dataset, a individuare e localizzare il target nelle immagini o, come in questo caso, nello streaming video.

Il processo di addestramento consiste nell'ottimizzazione iterativa dei pesi della rete attraverso la minimizzazione delle tre funzione di perdita precedentemente descritte, aggiornando i parametri ad ogni iterazione tramite un algoritmo di ottimizzazione fino alla convergenza.

La natura e la dimensione del dataset influenzano in modo determinante le scelte metodologiche adottate durante il training. Nel caso di dataset di grandi dimensioni, il modello ha accesso a una quantità di esempi sufficiente a generalizzare autonomamente, e gli unici limiti riguardano il costo computazionale e i tempi di addestramento necessario. Al contrario, quando si lavora con dataset di dimensioni ridotte, come avviene frequentemente in campo industriale o scientifico dove non esistono raccolte di immagini pubbliche [30], il rischio predominante è l'overfitting: il modello tende a "memorizzare" i campioni di training senza acquisire la capacità di generalizzare su immagini mai viste. In questi casi risulta fondamentale adottare strategie specifiche quali il trasferimento di apprendimento a partire da pesi pre-addestrati su dataset generali, l'impiego della data augmentation per ampliare artificialmente la variabilità del dataset, e una regolazione attenta degli iperparametri per evitare che il modello si adatti eccessivamente ai dati disponibili [31].

In un contesto caratterizzato da potenza di calcolo limitata e da un approccio di training privo di data augmentation, la qualità e la progettazione del dataset assumono un'importanza ancora maggiore. Le risorse computazionali disponibili costituiscono infatti un ulteriore vincolo significativo del presente lavoro: le scelte adottate nelle fasi di training sono state orientate verso il minimo impiego possibile delle risorse hardware, anche in vista di una possibile applicabilità della metodologia a lavori futuri e ad altri progetti.

L'addestramento è stato condotto su una GPU di fascia consumer con memoria video limitata, una configurazione non concepita per il deep learning professionale ma che può essere rappresentativa della realtà di molti progetti embedded e di ricerca applicata, in cui l'accesso a infrastrutture di calcolo dedicate rimane spesso un vincolo pratico rilevante [32]. Operare in un contesto di risorse limitate ha imposto scelte conservative su alcuni parametri di configurazione e sulla pianificazione del training, descritti nel dettaglio nelle sezioni successive, con l'obiettivo di rendere il processo di training stabile e riproducibile entro i vincoli hardware disponibili.

3.2 Ambiente e architettura di training

3.2.1 Modello

I modelli presenti nella famiglia YOLOv8 sono cinque, distinte per profondità e larghezza della rete: "n" (nano), "s" (small), "m" (medium), "l" (large) e "x" (extra-large). Le varianti si differenziano sostanzialmente per numero di parametri, velocità, precisione e capacità computazionale: la variante nano conta circa 3,2 milioni di parametri e raggiunge un mAP del 37,3% sul dataset COCO, mentre all'estremo opposto la variante extra-large conta circa 68,2 milioni di parametri con un mAP del 53,9% [33]. Le varianti intermedie small, medium e large si posizionano lungo questo spettro, offrendo gradualmente maggiore accuratezza a fronte di un crescente costo computazionale. La variante nano risulta pertanto la più leggera e veloce, ma con precisione ridotta rispetto alle altre, risultando la scelta più indicata per il deployment su dispositivi edge con risorse di calcolo e memoria limitate.

Il modello preso in considerazione per l'addestramento è YOLO v8n; è stato il modello e la variante scelta anche nel precedente lavoro di tesi su questo banchetto [20]. Nei paragrafi successivi si indaga anche sulla possibilità di un aggiornamento del modello alla versione più recente.

Tutte le fasi di training e di tuning sono state effettuate su un hardware consumer, ovvero una GPU NVIDIA GeForce GTX 1650 con 4 GB di memoria video dedicata.

3.3 Pianificazione della fase di training

La fase di training si sviluppa su tre differenti step.



Il training iniziale è stato necessario per avere una visione su quale versione del modello portare avanti nella fase di tuning: la scelta ricade tra il modello di default, con i pesi derivanti dal pre-training sul dataset di base COCO, oppure il modello “best” ottenuto come output del training iniziale, e ha inoltre riguardato alcuni parametri da impostare per il tuning (optimizer ed epoche), descritti nelle sezioni successive;

Il tuning degli iperparametri è necessario per individuare l’insieme di iperparametri che meglio si adattano al dataset custom creato. In questo lavoro tale fase è stata condotta sfruttando l’approccio nativo di Ultralytics, che adotta un algoritmo per l’ottimizzazione degli iperparametri: a partire da una configurazione di base, a ogni iterazione vengono generate nuove combinazioni di iperparametri tramite mutazione, applicando cioè piccole variazioni casuali ai parametri esistenti, selezionando di volta in volta le configurazioni con fitness più elevata [34].

Il training finale è eseguito utilizzando gli iperparametri forniti dal tuning, con un numero di epoche maggiorato al fine di ottenere una precisione finale superiore e permettere al modello di apprendere in modo più completo.

3.4 Iperparametri

La configurazione degli iperparametri di training rappresenta una fase delicata del processo di addestramento, poiché incide direttamente sulla stabilità della convergenza, sui tempi di esecuzione e, in ultima analisi, sulla qualità del modello ottenuto.

A differenza dei pesi della rete, che vengono appresi automaticamente durante il training, gli iperparametri devono essere impostati a priori dall'operatore e richiedono un compromesso tra le esigenze del processo di apprendimento e i vincoli imposti dall'hardware disponibile. Nel presente lavoro, la scelta di alcuni parametri è stata fortemente condizionata dalla memoria video limitata della GPU utilizzata, rendendo necessarie configurazioni conservative. Di seguito vengono illustrati i parametri adottati e le relative motivazioni.

3.4.1 Batch

La dimensione del batch (batch) è stata impostata a 8 elementi per iterazione. Un valore così contenuto è stato reso necessario dalla quantità di memoria video disponibile: durante alcuni training iniziali, batch size superiori hanno causato errori di esaurimento della memoria GPU (out-of-memory), in particolare alle risoluzioni di immagine più elevate.

3.4.2 Imgsz

La dimensione delle immagini in ingresso alla rete (imgsz) è stata variata inizialmente tra 320, 512 e 640 pixel nel corso della campagna sperimentale, al fine di valutarne l'impatto su accuratezza e velocità di inferenza. Successivamente, è stato reso necessario ampliare il range ai valori 160, 320, 480, 512 e 640 pixel, per disporre di una maggiore granularità di scelta.

Il valore di 160 pixel rappresenta il limite inferiore utile per YOLO v8n: nell’architettura, il backbone applica una fattore di downsampling cumulativo pari a 32, determinato dalla sequenza di cinque convoluzioni con stride 2. Di conseguenza, la feature map al livello di rilevamento più profondo (P5/32) presenta dimensioni pari a imgsz/32x imgsz/32; per un input di 160x160 pixel, tale feature map risulta essere 5x5, ovvero 25 celle della griglia disponibili per la detection a quella scala.

YOLO v8n adotta un design anchor-free: a differenza delle versioni precedenti, la rete non si basa su bounding box di dimensione fissa, ma predice direttamente, per ciascuna cella della griglia, le coordinate della bounding box e la relativa classe. Il numero di celle disponibili a ciascun livello rappresenta quindi il fattore limitante per la capacità di localizzazione della rete a quella scala [35].

Al di sotto della soglia di 160 pixel, il numero di celle disponibili al livello P5 diventa insufficiente per la localizzazione di oggetti di grandi dimensioni, mentre i livelli P4/16 (10x10 celle) e P3/8 (20x20 celle) subiscono una perdita di risoluzione spaziale che compromette il rilevamento di oggetti di medie e piccole dimensioni. Il valore di 160 pixel

rappresenta pertanto, in questo lavoro, la soglia minima al di sotto della quale la densità di celle disponibili su tutti e tre i livelli di detection non è più ritenuta sufficiente a garantire prestazioni accettabili di localizzazione.

3.4.3 Device

Il dispositivo di calcolo (device) è stato impostato su 0, corrispondente alla GPU dedicata disponibile nel sistema, in modo da sfruttare l'accelerazione hardware offerta da CUDA per l'intero processo di training. Il training eseguito su CPU è tendenzialmente più lento, dato l'elevato numero di operazioni in parallelo che l'architettura di una rete neurale convoluzionale richiede e che una GPU è progettata per gestire in modo nativo.

3.4.4 Optimizer

L'ottimizzatore (optimizer) è stato selezionato di volta in volta tra SGD, AdamW, Adam, RAdam e NAdam nel corso della campagna di confronto descritta nella sezione a esso dedicata, al fine di identificare la configurazione più efficace per questo caso di studio specifico.

3.4.5 Cache

Il caching del dataset in memoria (cache)è stato inizialmente abilitato impostandolo a True per ottenere una riduzione dei tempi di training. Successivamente, a causa di problemi di occupazione di memoria è stato disabilitato impostandolo a False. Sebbene il caching consenta tipicamente di velocizzare le epoche successive alla prima evitando la lettura ripetuta da disco, la sua attivazione avrebbe comportato un ulteriore consumo di RAM di sistema non sostenibile in combinazione con gli altri processi attivi durante il training.

3.4.6 Workers

Il numero di worker dedicati al caricamento dei dati (workers) è stato inizialmente impostato a 6 data la vasta disponibilità iniziale di memoria. A causa della molteplicità dei training eseguiti, la memoria disponibile è stata progressivamente occupata e ciò ha portato alla riduzione dei worker a 2 e successivamente a 0. Valori più elevati di questo parametro, in questo contesto, si sono infatti rivelati controproducenti: un tentativo con workers=6 ha causato un errore di esaurimento del file di paging del sistema operativo, dovuto all'eccessivo numero di processi worker generati in parallelo rispetto alla memoria disponibile.

3.4.7 Amp

Come nel caso di cache, l'addestramento in precisione mista (amp, Automatic Mixed Precision) è stato inizialmente abilitato, impostando il parametro a True. Questa tecnica introdotta da Micikevicius et al. [36], consiste nell'eseguire parte delle operazioni in precisione a 16 bit anziché a 32 bit, mantenendo una copia dei pesi in precisione FP32 per l'aggiornamento del gradiente, così da ridurre l'occupazione di memoria video e velocizzando il calcolo a fronte di una perdita di accuratezza generalmente trascurabile. Successivamente, solo per le due fasi finali di tuning e training lungo, il parametro è stato impostato a False.

3.4.8 Epochs

Il numero massimo di epoche (epochs) è stato impostato in funzione della tipologia di esperimento: 50 epoche per le run iniziali di confronto, 85 o 50 epoche per la fase di tuning degli iperparametri, con valori differenziati in base alla dimensione dell'immagine impiegata, e 150 epoche per il training finale del modello selezionato.

3.4.9 Patience

Il parametro di pazienza (patience), che determina il numero di epoche consecutive senza miglioramento delle metriche di validazione prima dell'arresto anticipato del training (early stopping) [37], non è stato introdotto nella fase iniziale degli esperimenti. È stato successivamente impiegato, prima della fase di tuning, con un duplice obiettivo: da un lato verificare quante epoche fossero effettivamente necessarie per raggiungere il modello migliore al variare del valore di patience, dall'altro individuare il valore più adatto da adottare durante il tuning. Questa scelta ha permesso di evitare addestramenti inutilmente prolungati una volta raggiunta la convergenza del modello, contribuendo al contempo a contenere i tempi complessivi della campagna sperimentale.

3.4.10 Deterministic

Il comportamento deterministico (deterministic) è stato attivato, impostando il parametro a True. Questa opzione forza l'utilizzo di algoritmi deterministici da parte del backend di calcolo, garantendo che, a parità di seed e configurazione, due esecuzioni successive del medesimo esperimento producano risultati identici.

3.4.11 Seed

Infine, il seed casuale (seed) è stato fissato al valore 42 per tutti gli esperimenti condotti, in combinazione con il parametro deterministic descritto in precedenza: i due parametri trovano infatti pieno significato solo se utilizzati insieme, poiché fissare il seed senza imporre un comportamento deterministico non garantirebbe comunque la riproducibilità esatta dei risultati per run successive. Come riportato nella documentazione ufficiale di PyTorch, la riproducibilità completa non è garantita tra release, commit o piattaforme diverse, e alcune funzioni CUDA possono costituire una fonte di non-determinismo anche a parità di seed impostato [38]. Per questo motivo, accanto al parametro seed, è necessario usare esplicitamente un comportamento deterministico di calcolo. Fissare il seed garantisce dunque la riproducibilità dei risultati tra run differenti, condizione necessaria per poter confrontare in modo affidabile le diverse configurazioni di iperparametri e ottimizzatori testate nelle sezioni successive.

La scelta del valore 42, pur arbitraria dal punto di vista tecnico, richiama la celebre "risposta alla domanda fondamentale sulla vita, l'universo e tutto quanto" del romanzo *Guida galattica per autostoppisti* di Douglas Adams [39], divenuto negli anni un riferimento ricorrente e quasi un easter egg nella comunità di sviluppatori e ricercatori in ambito informatico.

3.5 Analisi preliminare

L'analisi preliminare si è resa necessaria per determinare vari punti:

- quale modello risultasse preferibile, tra il modello "best" e il modello di default, da portare a un tuning correttamente impostato
- se fosse opportuno ampliare o meno il range di valori utilizzati per imgsz

I primi training sono stati eseguiti unicamente a scopo esplorativo per testare sia l'hardware a disposizione sia il comportamento del modello YOLO v8n sul nuovo dataset. Questa fase preliminare è stata condotta eseguendo un primo training di 20 epoche, un successivo tuning senza alcuna specifica da codice e un training finale di 150 epoche.

Una volta individuata la configurazione adatta per ottenere un training completo, sono stati eseguiti in sequenza: un training breve di 50 epoche, un tuning di sole 20 iterazioni e un training finale per 150 epoche. Al termine di questi training, i risultati ottenuti sono stati raccolti e analizzati congiuntamente, permettendo di trarre le prime conclusioni. In questo contesto è stata usata la mAP@50-95 come metrica principale. I risultati della prima fase preliminare sono riportati nella tabella seguente.

IMGSZ	mAP@50-95	GFLOPs
320	0,9899	2,0
320 (con modello best)	0,9876	2,0
512	0,9898	5,2
512 (con modello best)	0,9879	5,2
640	0,9913	8,2
640 (con modello best)	0,9864	8,2

Tabella 2 analisi preliminare: mAP@50-95 e GFLOPs ad imgsz e modelli differenti

Dalla tabella sopra riportata si deduce che è preferibile eseguire le fasi successive di training a partire dal modello di default, piuttosto che dal modello best: a parità di imgsz, il modello di default ottiene sistematicamente valori di mAP@50-95 leggermente superiori rispetto al corrispondente modello best.

Inoltre, anche la colonna dei GFLOPs restituisce un'indicazione chiara: il numero di operazioni si riduce notevolmente al diminuire della dimensione di immagine di input. Poiché il numero di GFLOPs è direttamente correlato al costo computazionale, e quindi, indicativamente, al tempo di inferenza, richiesto dal modello, addestrare la rete con una dimensione di immagine ancora più ridotta può tradursi in un beneficio concreto in termini di latenza in fase di deploy.

A partire da questo punto, i training iniziali, i tuning e i training finali più lunghi sono stati eseguiti utilizzando esclusivamente il modello di default.

Il range dei valori considerato per imgsz è stato ampliato includendo i valori 160 e 480 pixel: il valore 160 è stato inserito in quanto rappresenta il valore minimo accettato dall'architettura della rete, come discusso in precedenza, mentre il valore 480 è stato introdotto unicamente come valore intermedio tra 320 e 640.

3.6 Scelta dell'ottimizzatore

Il successivo passo è stato la scelta dell'optimizer, effettuato nell'ottica di risparmio di risorse computazionali e di tempo in vista della successiva fase di tuning dato che il tuning rappresenta essere una moltitudine di training, dato dal numero di iterazioni, con una combinazione di iperparametri ogni volta differente. Questi step preliminari sono stati quindi condotti con l'obiettivo di individuare una rete capace di raggiungere, dopo un numero contenuto di epoche, un buon livello di riconoscimento del target. È in questa fase che entra in gioco il parametro di patience: sono stati eseguiti diversi training con diversi valori di patience (da 5 fino a 40) per ogni optimizer (NAdam, Adam, RAdam, Adamw e SGD inizialmente) tutti limitati a 150 epoche.

Da alcuni training preliminari sulla dimensione 160x160, è emerso che gli ottimizzatori Adam e Radam riportavano un andamento instabile in tutte le metriche osservate. Quindi, non sono stati introdotti nello studio per la scelta finale dell'optimizer; quindi, i dati si baseranno solo sui tre optimizer rimanenti: NAdam, Adamw e SGD.

La scelta viene operata vedendo due aspetti: la posizione della best epoch (preferibile quanto più precoce) e l'andamento delle funzioni di perdita durante l'intero training privilegiando l'ottimizzatore che ha prodotto un training più pulito e valori di perdita più bassi.

Verranno riportati, in una tabella, i dati riguardanti l'optimizer, la best epoch e la best mAP@50-95 raggiunta; successivamente, in forma grafica, l'andamento di tutte le funzioni di perdita utili. La classification loss non viene riportata perché è una funzione di perdita che ha poco valore informativo in questo contesto data la presenza nel dataset di una sola classe: per ogni combinazione imgsz-optimizer, tale funzione raggiunge infatti un plateau quasi immediatamente durante le prime epoche. Analogamente, non viene considerata la mAP@50 in quanto metrica poco restrittiva (soglia di IoU del 50%) raggiunge il suo massimo (0.995) dopo poche epoche, coerentemente con il peso ridotto attribuitole anche nella formula della fitness.

Questa operazione è stata ripetuta per ciascuna dimensione di immagine considerata.

3.6.1 Dimensione 160x160

La dimensione di immagine 160x160 è quella che ha ricevuto maggiore attenzione, poiché rappresenta la configurazione da cui ci si attende la latenza di inferenza più bassa.

Si riporta di seguito la tabella con le metriche principali, relative a tutti i training eseguiti con i diversi optimizer considerati.

OPTIMIZER	PATIENCE	EPOCA BEST	BEST mAP@50-95
AdamW	10	32	0.9332
AdamW	20	77	0.9701
AdamW	25	77	0.9701
AdamW	30	77	0.9701
NAdam	10	30	0.9103
NAdam	15	52	0.9567
NAdam	20	52	0.9567
NAdam	25	52	0.9567
NAdam	30	137	0.9880
SGD	10	59	0.9778
SGD	15	59	0.9778
SGD	20	59	0.9778
SGD	25	82	0.9841
SGD	30	82	0.9841

Tabella 3 metriche finali a diversi optimizer a diversi patience a imgsz 160

Da questi dati emerge immediatamente che la mAP@50-95 migliore viene raggiunta da SGD con best epoch alla 82^a iterazione. AdamW raggiunge la sua best epoch alla 77^a epoca ma con n valore di mAP@50-95 nettamente inferiore.

Sulla base di questi risultati, per il tuning con imgsz=160 è stata scelta una patience pari a 30, impostando un massimo di 85 epoche.

Nel grafico seguente sono riportati solo gli andamenti relativi ai training condotti con patience=30.



Figura 16 training box/loss a imgsz 160



Figura 15 training dfl/loss a imgsz 160

Per quanto riguarda la fase di training, le curve di loss di SGD e AdamW risultano pressoché sovrapposte, mentre quella di NAdam è costantemente al di sopra delle altre due curve. Per questo motivo, l'ottimizzatore NAdam non viene preso in considerazione in questa fase.

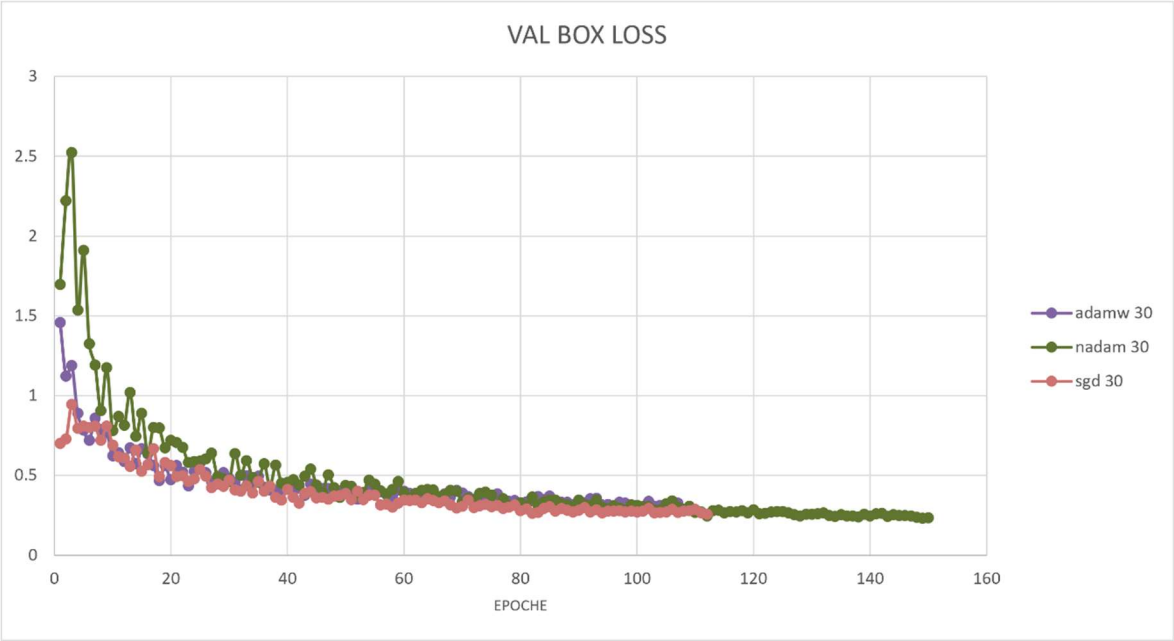


Figura 18 validation box/loss a imgsz 160

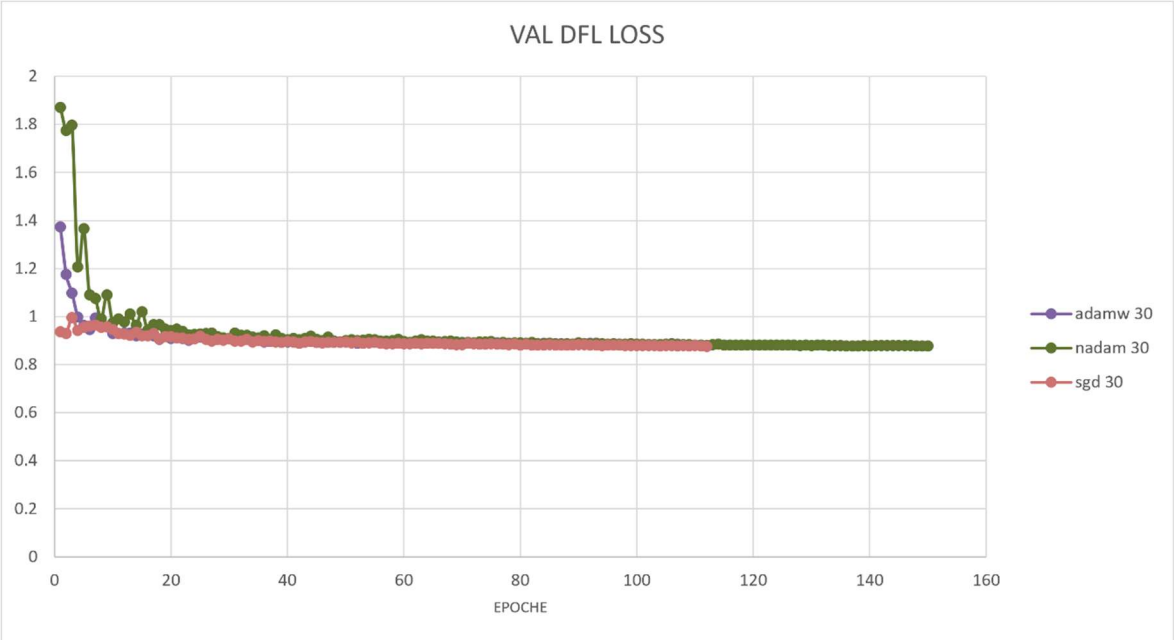


Figura 17 validation dfl/loss a imgsz 160

Il grafico della validation loss fornisce un quadro più definito: Nadam risulta sempre da scartare a causa della costante instabilità di apprendimento osservata; SGD si conferma la migliore mostrandosi il più stabile tra i tre; AdamW presenta invece un andamento pressoché identico a quello di SGD.

Alla luce della migliore mAP@50-95 raggiunta e del comportamento più stabile delle funzioni di perdita, per la dimensione di immagine 160x160 viene scelto quindi SGD come ottimizzatore.

3.6.2 Dimensione 320x320

Con l'aumento della dimensione di immagine, la mAP@50-95 cresce: questo andamento è coerente con la maggiore quantità di informazione visiva a disposizione a risoluzioni più elevate, che consente di apprendere rappresentazioni più accurate del target.

Tale incremento comporta, a sua volta, un allungamento dei tempi di training necessari per raggiungere una determinata soglia di convergenza.

Si riporta la tabella riassuntiva dei dati più rilevanti.

OPTIMIZER	PATIENCE	EPOCA BEST	BEST mAP@50-95
AdamW	20	75	0.98597
AdamW	30	104	0.98991
AdamW	40	104	0.98991
NAdam	20	128	0.9878
NAdam	30	183	0.98991
NAdam	40	183	0.98991
SGD	20	85	0.99171
SGD	30	85	0.99171
SGD	35	85	0.99171
SGD	40	85	0.99171

Tabella 4 metriche finali a diversi optimizer a diversi patience a imgsz 320

I dati mostrano una divisione netta sia rispetto la best epoch sia rispetto la best mAP@50-95 raggiunta. In particolare, SGD si distingue per la stabilità del risultato: la best epoch si attesta sempre alla 85^a iterazione, indipendentemente dal valore di patience impostato (20, 30, 35, 40), a conferma che il modello raggiuge il proprio massimo in modo precoce e non trae ulteriore beneficio da ulteriori iterazioni.

AdamW e NAdam, al contrario, mostrano una best epoch che si sposta in avanti all'aumentare della patience (rispettivamente da 75 a 104 epoche per AdamW e da 128 a 183 epoche per NAdam), pur raggiungendo, con patience più elevate, un mAP@50-95 comunque inferiore a quello di SGD.

Anche in questo caso, considerando questi dati, SGD si conferma la scelta migliore tra gli ottimizzatori testati.

Per il tuning successivo, vengono adottati quindi un massimo di 85 epoche assieme a 30 come valore di patience.

Si riportano gli andamenti delle funzioni di perdita.

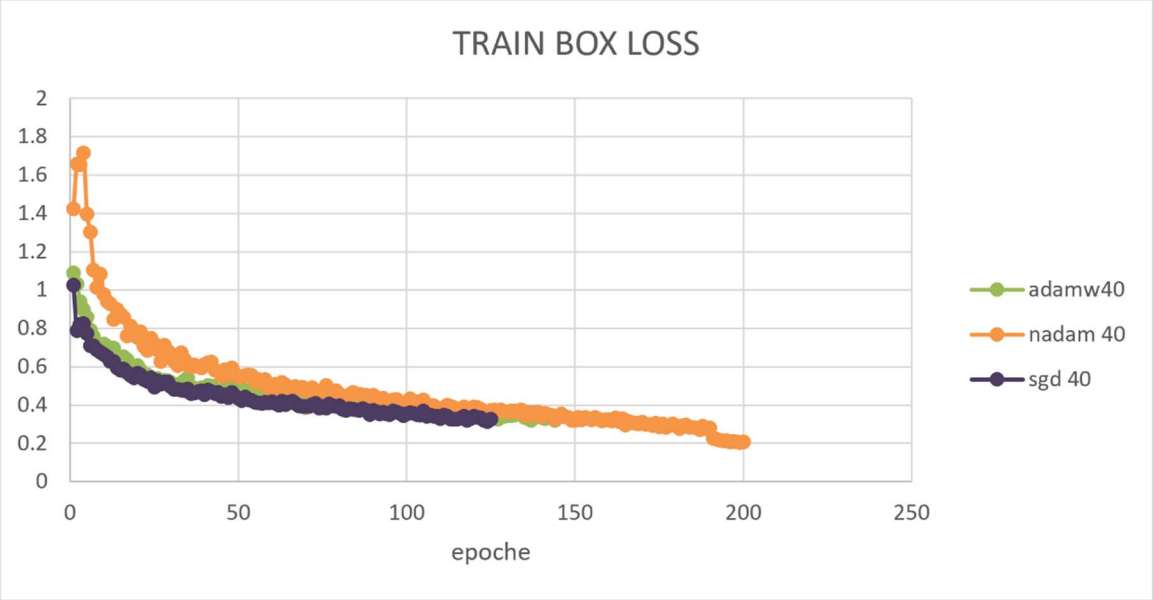


Figura 20 training box/loss a imgsz 320

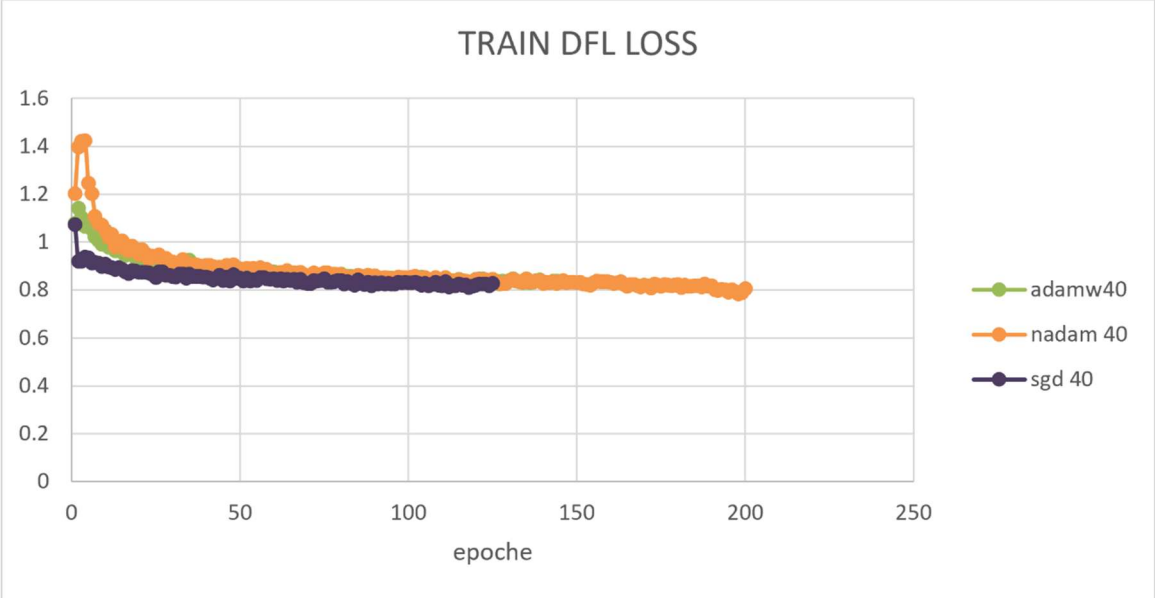


Figura 19 training dfl/loss a imgsz 320

Le funzioni di perdita riportano un apprendimento più lento quando viene impiegato l'optimizer NAdam: la relativa curva si colloca costantemente al di sopra delle altre due curve. Mentre, la situazione risulta invece migliore per gli altri due ottimizzatori, per i quali l'apprendimento procede in modo soddisfacente.

In ogni caso, per la fase di training l'optimizer migliore rimane SGD poiché si attesta fin da subito su valori molto bassi e raggiunge il plateau più rapidamente rispetto agli altri due.

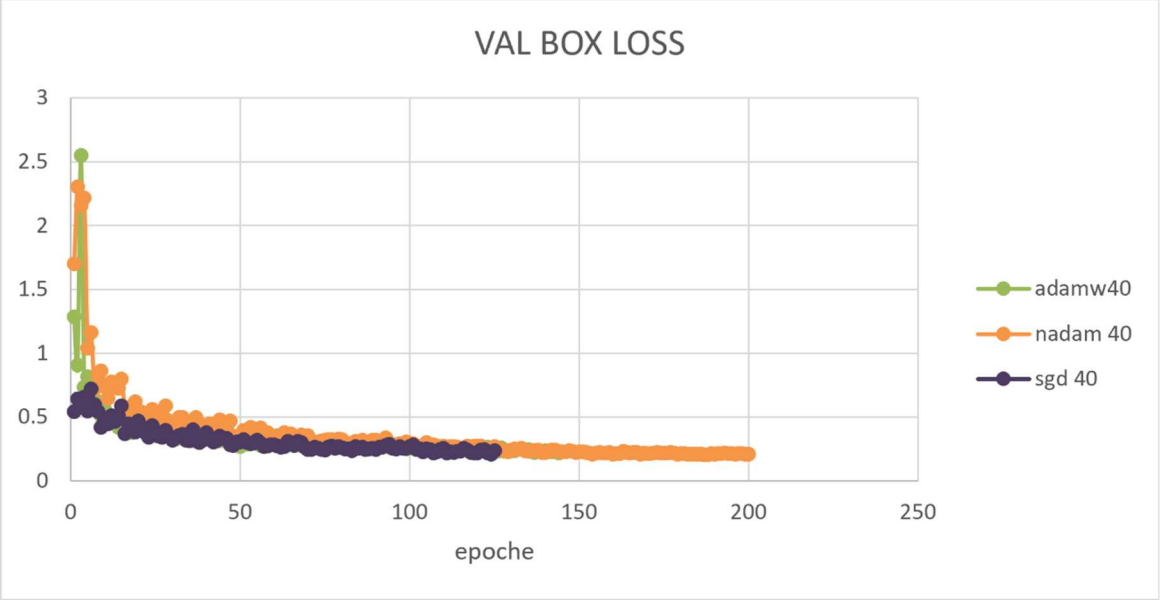


Figura 22 validation box/loss a imgsz 320

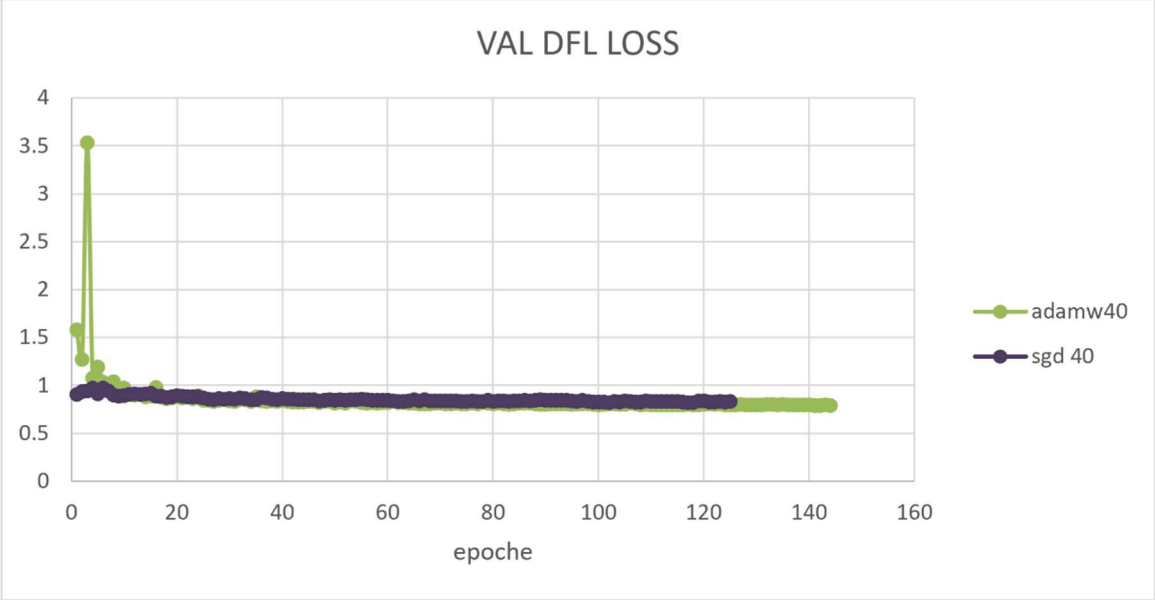


Figura 21 validation dfl/loss a imgsz 320

Nella fase di validation, si osserva lo stesso comportamento visto durante la fase di training: la curva relativa a NAdam si colloca al di sopra delle altre due, che si mantengono invece su valori inferiori.

Nel grafico della DFL loss (Figura 21 validation dfl/loss a imgsz 320), la curva relativa a NAdam non viene riportata a causa di alcuni spike che non consentono una rappresentazione corretta e ben visibile dell'intero grafico.

Sia in fase di training che in fase di validation, SGD è stato il miglior ottimizzatore; per il criterio di scelta della best epoch, anche per la imgsz=320 è stato selezionato SGD come ottimizzatore.

Per le successive dimensioni di immagine, le curve riguardanti l'optimizer NAdam non vengono più riportate in alcun grafico dato che in tutti i casi precedenti come in quelli successivi, si è dimostrato non adatto all'addestramento per il task in esame.

3.6.3 Dimensione 480x480

Si riportano i dati riassuntivi per le prove effettuate a dimensione di immagine 480x480.

OPTIMIZER	PATIENCE	EPOCA BEST	BEST mAP@50-95
AdamW	20	54	0.98978
AdamW	30	54	0.98978
AdamW	40	54	0.98978
NAdam	20	69	0.98919
NAdam	30	69	0.99189
NAdam	40	121	0.99175
SGD	20	54	0.99131
SGD	30	77	0.99186
SGD	40	77	0.99186

Tabella 5 metriche finali a diversi optimizer a diversi patience a imgsz 480

Anche in questo caso, la configurazione migliore risulta quella con SGD poiché riporta un valore di mAP@50-95 superiore rispetto a AdamW, infatti, le best epoch è uguale per entrambi.

Le best epoch osservate in questo caso (54 per AdamW e SGD a patience 20, 77 per SGD a patience più elevate) risultano tuttavia sensibilmente più basse rispetto a quanto riscontrato nelle configurazioni precedenti, dove la best epoch si attestava tipicamente intorno alle 80-90 epoche.

Proprio per questo motivo, pur avendo dati che suggerirebbero una convergenza più precoce, per la successiva fase di tuning sono stati scelti un valore di patience pari a 35 e un numero massimo di 85 epoche. Questa scelta, leggermente più conservativa rispetto a quanto strettamente indicato dai dati preliminari, è stata adottata per introdurre un margine di sicurezza rispetto alla best epoch osservata.

Per questo dimensione di immagine, vengono scelti un valore di patience pari a 35 e un

Si riportano i grafici delle funzioni di perdita.

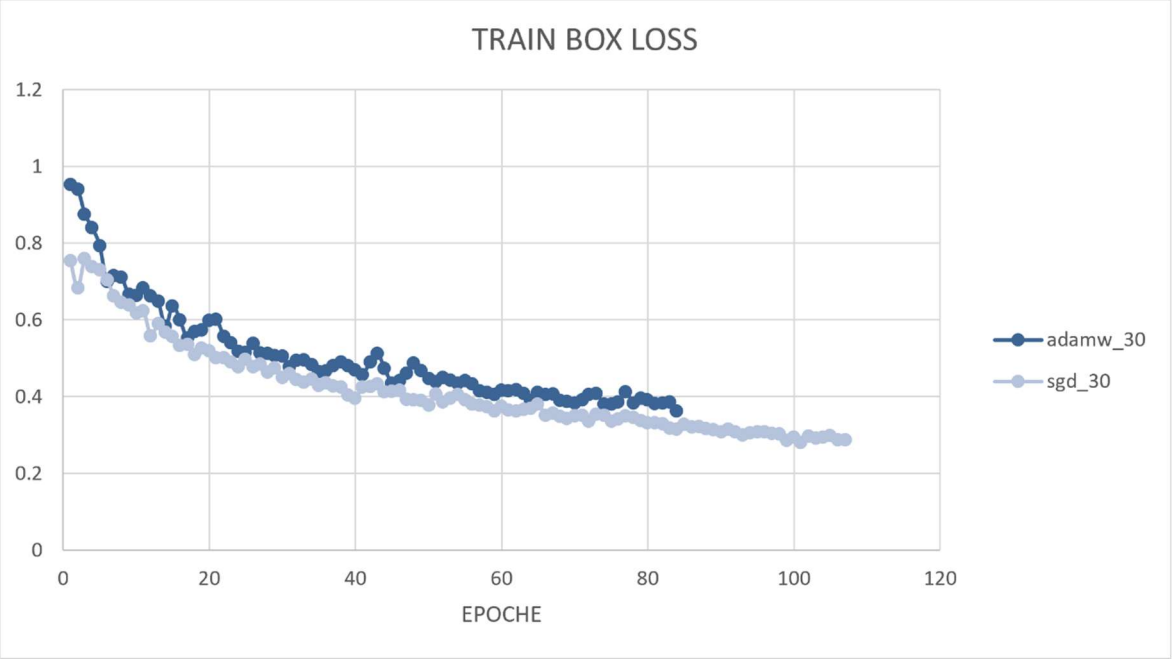


Figura 23 training box/loss a imgsz 480

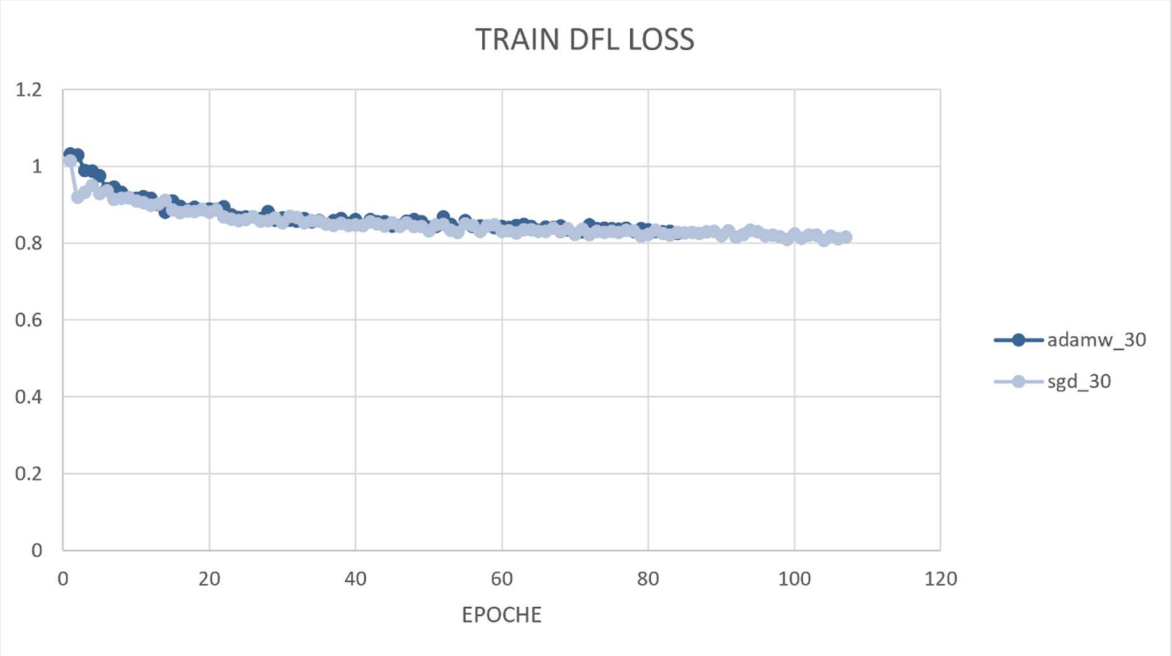


Figura 24 training dfl/loss a imgsz 480

Per quanto riguarda la fase di training, l'ottimizzatore SGD riesce a localizzare una bounding box in modo nettamente migliore rispetto ad AdamW: la curva della box loss di quest'ultimo è chiaramente al di sopra per l'intera durata del training. Mentre, per la dfl loss, il comportamento è pressoché simile per i due optimizer.

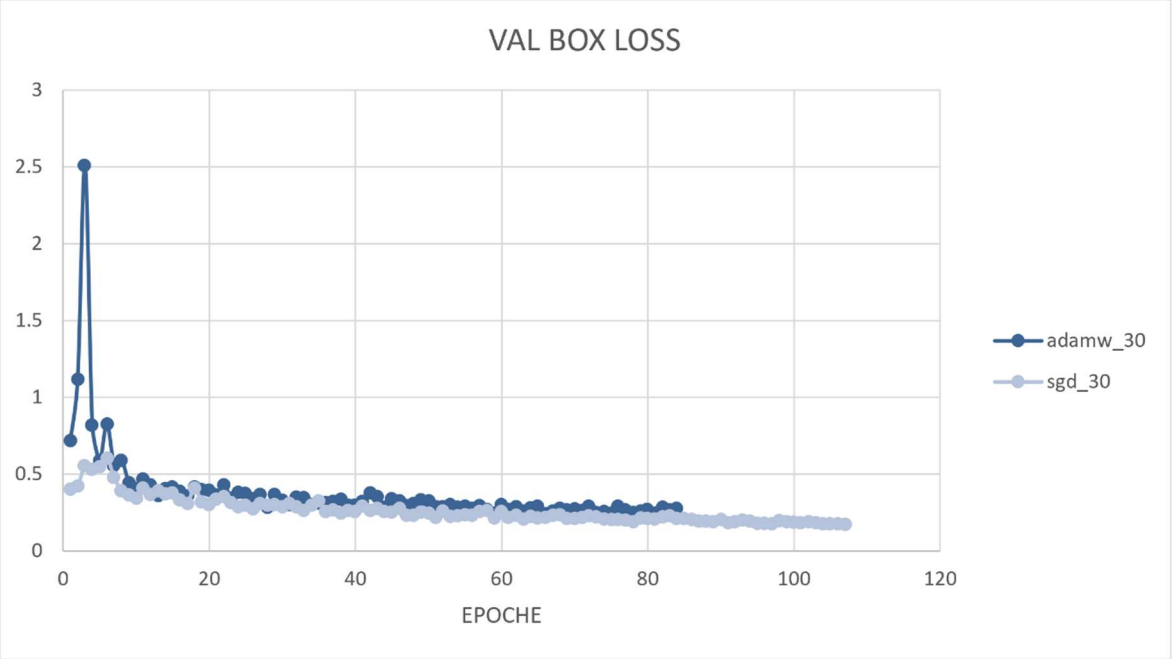


Figura 25 validation box/loss a imgsz 480

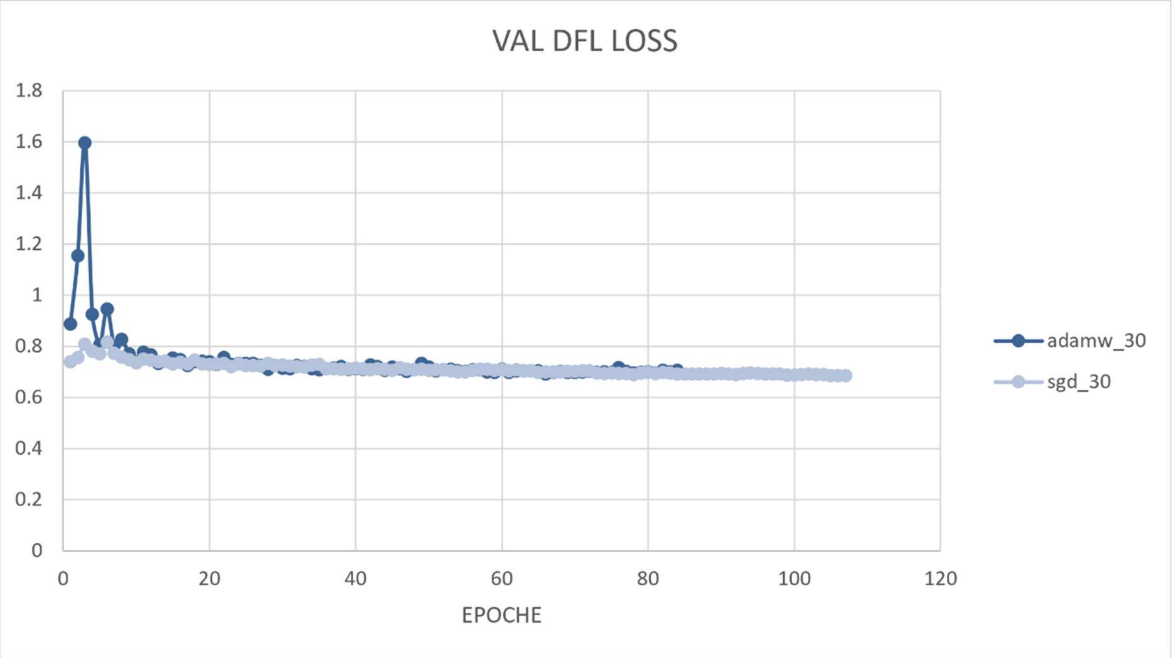


Figura 26 validation dfl/loss a imgsz 480

Il caso si ripete nella fase di validation: SGD continua a trovare bounding box più accurate rispetto ad AdamW, con le due curve nettamente distinguibili e separate l'una dall'altra sebbene in modo meno marcato rispetto alla fase di training.

Anche per la imgsz 480, viene quindi scelto SGD come ottimizzatore, data la netta differenza riscontrata sia in fase si training che in fase di validation.

3.6.4 Dimensione 512x512

Si riportano i dati riassuntivi per le prove effettuate a dimensione di immagine 512x512.

OPTIMIZER	PATIENCE	EPOCA BEST	BEST mAP@50-95
AdamW	20	92	0.994
AdamW	30	92	0.994
AdamW	40	92	0.994
SGD	20	73	0.9918
SGD	30	73	0.9918
SGD	40	73	0.9918

Tabella 6 metriche finali a diversi optimizer a diversi patience a imgsz 512

Dai dati riportati emerge, in questo caso, una situazione di compromesso: SGD ottiene la sua best epoch dopo 73 iterazioni, contro le 92 iterazioni di AdamW; AdamW raggiunge un valore di mAP@50-95 di poco superiore rispetto a SGD (0.994 contro 0.9918). La differenza di accuratezza tra i due risulta contenuta tale da passare in secondo piano relativamente al vantaggio in termini di best epoch ottenuta. Anche in base ai risultati delle configurazioni precedenti, viene quindi scelto SGD, privilegiando la maggiore capacità di convergenza.

In questo caso la scelta della patience è stata guidata principalmente dall'esigenza di mantenere una coerenza tra le varie run di training già condotte alle altre dimensioni di immagine: viene quindi scelto un valore di patience pari a 35, analogamente al numero delle epoche massime impostate a 85.

Si riportano i grafici delle funzioni di perdita.

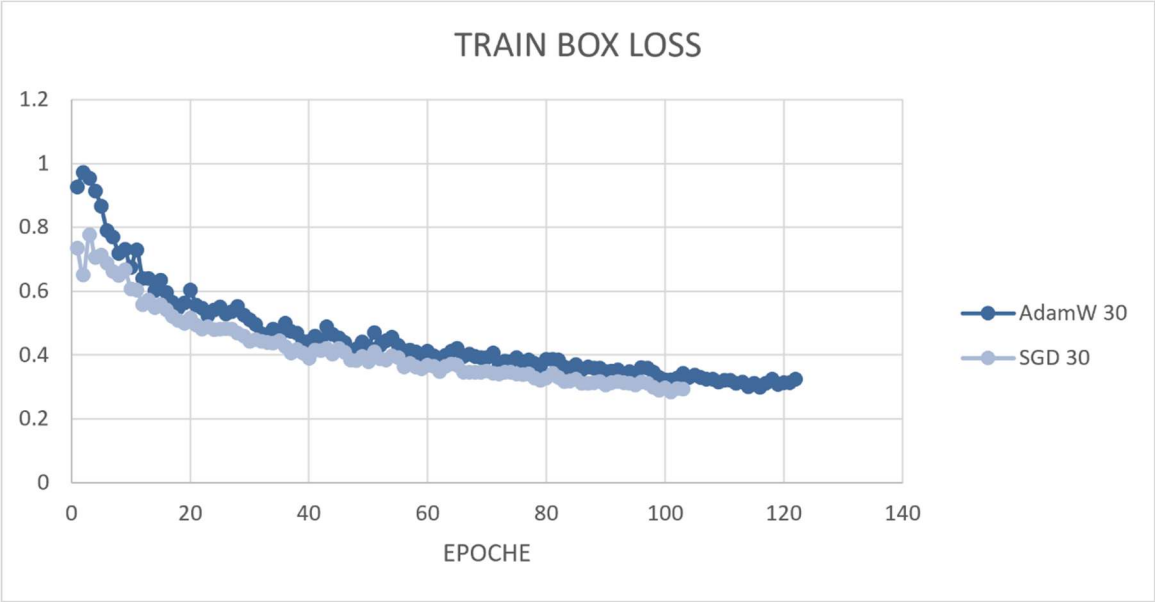


Figura 27 training box/loss a imgsz 512

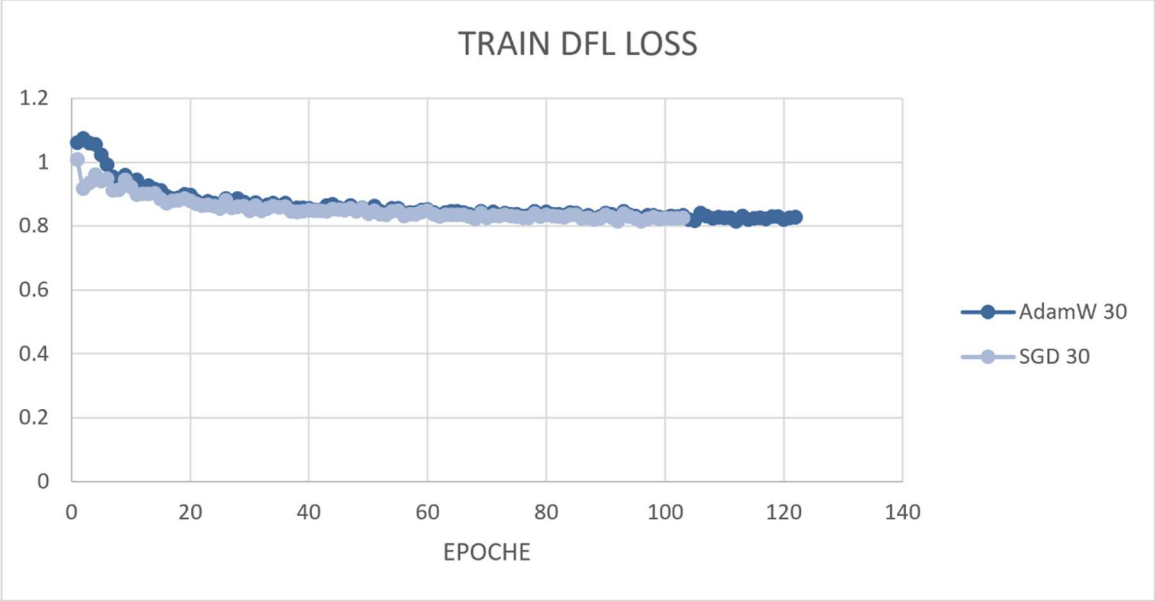


Figura 28 training dfl/loss a imgsz 512

Durante la fase di training, SGD ottiene ancora una volta risultati migliori nella loss relativa al posizionamento della bounding box. Mentre, in questo caso, le curve dei due optimizer sono praticamente sovrapponibili per quanto riguarda la dfl loss.

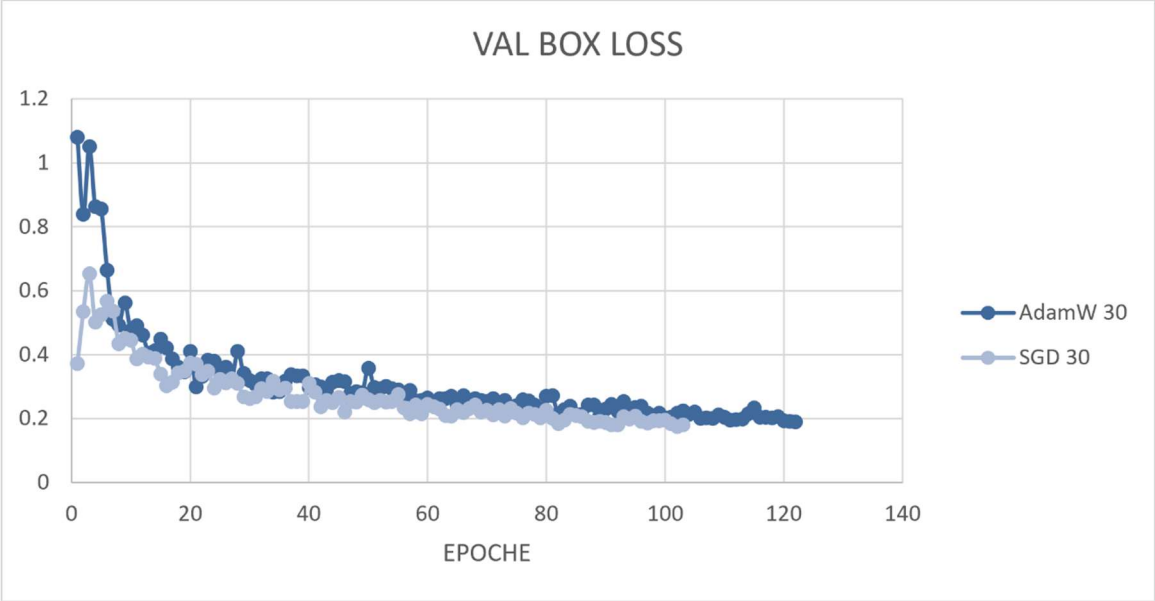


Figura 29 validation box/loss a imgsz 512

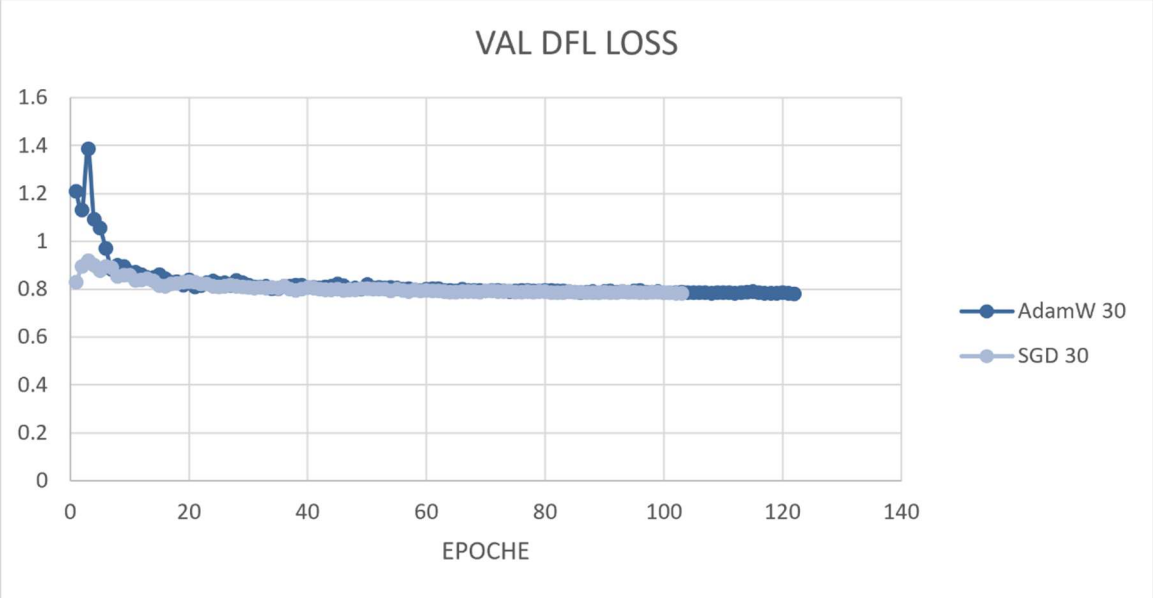


Figura 30 validation dfl/loss a imgsz 512

Si può inoltre notare, nel grafico della box loss, un andamento di AdamW leggermente più instabile in riferimento alle run condotte a imgsz differente. In ogni caso, la curva di SGD si mantiene costantemente al di sotto della curva di AdamW quindi anche a questa imgsz SGD è migliore.

Considerando quindi, la prestazioni in entrambe le funzioni di perdita in training e validation e tenendo conto del compromesso discusso in precedenza, viene scelto SGD come ottimizzatore anche per imgsz=512.

3.6.5 Dimensione 640x640

Si riportano i dati riassuntivi per la dimensione 640x640.

OPTIMIZER	PATIENCE	EPOCA BEST	BEST mAP@50-95
AdamW	20	73	0.98938
AdamW	25	73	0.98938
AdamW	30	134	0.99196
SGD	20	71	0.994
SGD	25	39	0.98947
SGD	30	71	0.994
SGD	35	71	0.994
SGD	40	71	0.994
SGD	45	71	0.994

Tabella 7 metriche finali a diversi optimizer a diversi patience a imgsz 640

L'ottimizzatore SGD risulta, anche in questo caso, il migliore sia in termini di best epoch che mAP@50-95: raggiunge la propria best epoch prima (71 epoche contro le 73 delle run più brevi di AdamW) e raggiunge un punteggio di mean average precision nettamente superiore rispetto al punteggio raggiunto da AdamW anche dalla run più lunga (134 epoche al best con 0.99196 di mAP@50-95), nonostante, in linea teorica, un training più prolungato dovrebbe tendenzialmente favorire il raggiungimento di punteggi migliori nelle metriche.

Si segnala che la run con SGD e patience=25 rappresenta un'eccezione rispetto a comportamento costante osservato per lo stesso ottimizzatore alle altre patience testate: la best epoch si attesta a 39 iterazioni con una mAP@50-95 di 0.98947. Tale scostamento è verosimilmente riconducibile ad una variabilità intrinseca al processo di training dato che valori di patience inferiore e superiori a 25 restituiscono valori tra loro identici. Questo dato non è stato considerato quindi rappresentativo ai fini della scelta finale dell'ottimizzatore.

In questo caso, per il tuning sono state scelti i valori 30 per il parametro di patience e 75 come epoche massime per ogni iterazione. Per concludere, si riportano i grafici delle funzioni di perdita.

.

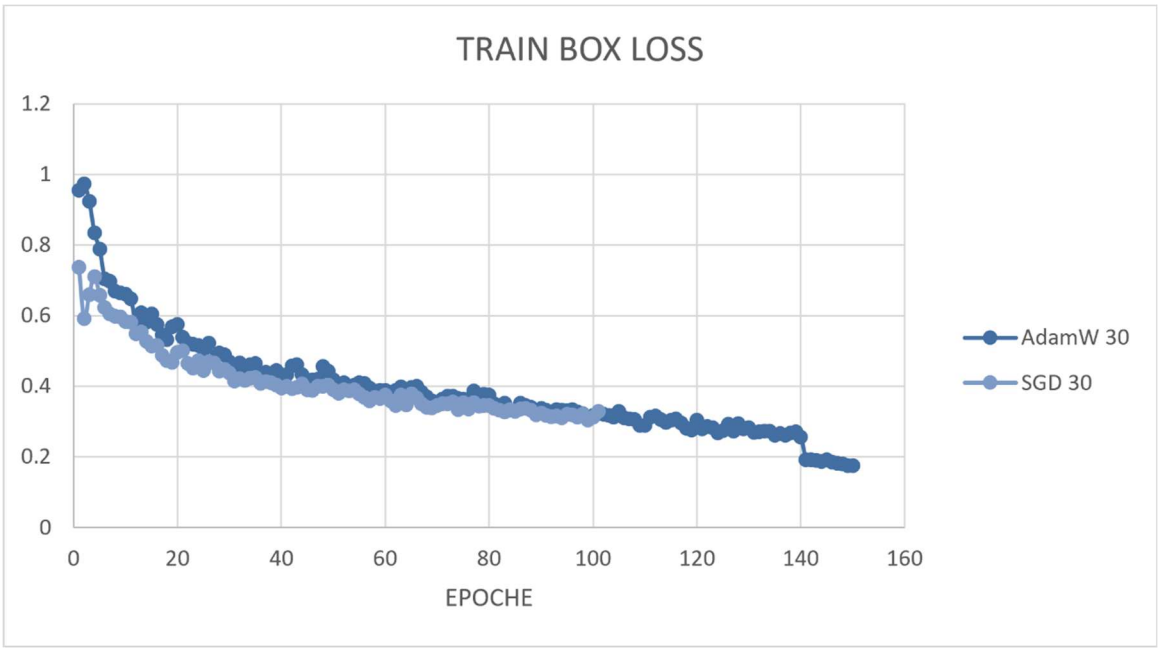


Figura 31 training box/loss a imgsz 640



Figura 32 training dfl/loss a imgsz 640

Dalle funzioni di loss osservate durante il training, si conferma SGD come migliore anche in questo caso: la curva di SGD è sempre al di sotto della corrispettiva curva di AdamW sebbene entrambe le curve si attestino allo stesso valore dopo essere state addestrate per circa cento epoche. Lo stesso comportamento si osserva per la DFL loss con l'unica differenza riscontrabile nella maggiore linearità delle due curve.

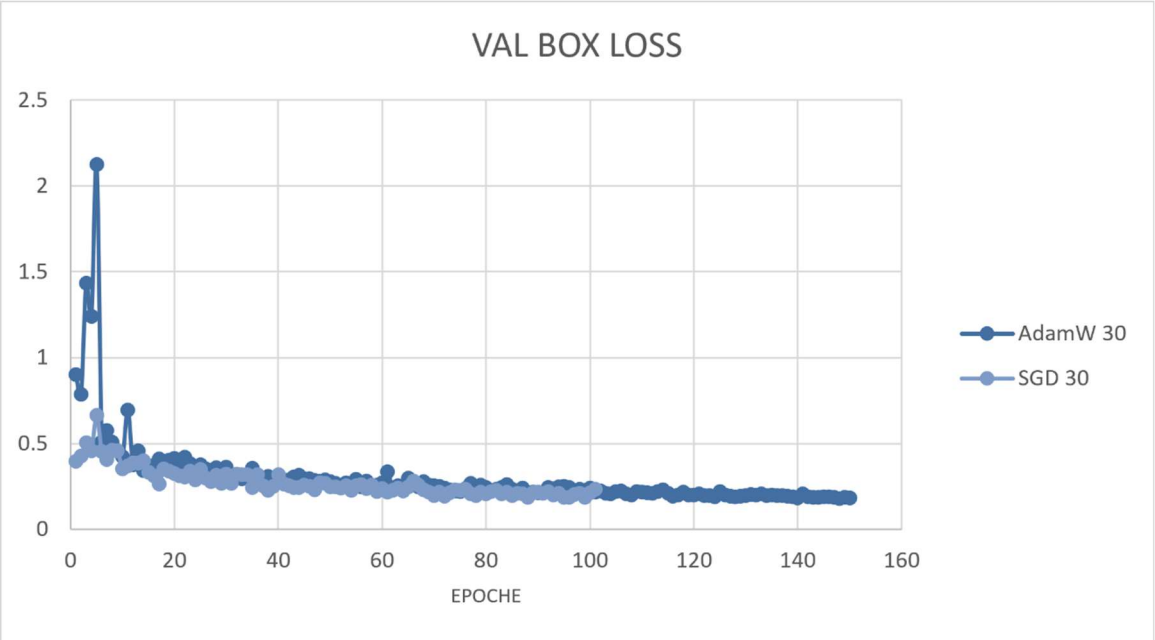


Figura 33 validation box/loss a imgsz 640

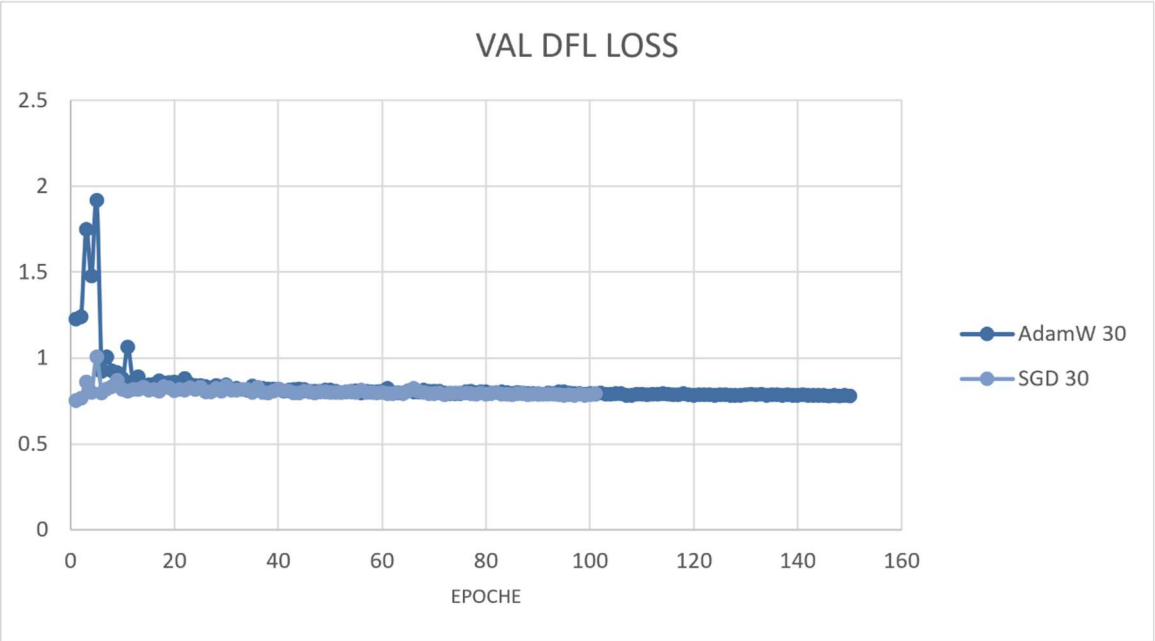


Figura 34 validation dfl/loss a imgsz 640

Anche in fase di validation, SGD è risultato migliore; entrambi gli ottimizzatori raggiungono comunque il plateau alla medesima epoca, intorno alle 100 epoche.

Concludendo, anche per la imgsz 640, l'ottimizzatore ideale è SGD.

Dai dati riportati nei paragrafi precedenti possono essere estrapolate le seguenti conclusioni generali:

- al crescere della dimensione di immagine, il modello tende a raggiungere il proprio best con un numero minore di epoche;
- al diminuire della imgsz, il tempo necessario per raggiungere il best diminuisce;
- all'aumentare della imgsz, la mAP@50-95 tende ad aumentare.

Si riporta una tabella riassuntiva con i vari valori a supporto di quanto appena esposto.

IMGSZ	EPOCA BEST	BEST mAP@50-95	TEMPO AL BEST (s)
160	82	0.984	534.1
320	85	0.9917	496.9
480	77	0.9919	675.1
512	73	0.9918	696.5
640	71	0.994	938.1

Tabella 8 metriche risultanti in fase di training per imgsz

3.7 Fase di tuning

Il passo successivo è il tuning degli iperparametri.

Al fine di ottimizzare le prestazioni del modello YOLO v8n sul dataset acquisito, sono stati ricercati i valori di alcuni iperparametri tramite il comando "tune" già offerto dalla piattaforma di Ultralytics. La selezione degli iperparametri da ottimizzare e la definizione dei relativi intervalli sono state individuate grazie ad alcuni processi di tuning preliminari .

Sono stati impiegati due valori di learning rate denominati "lr0" e "lrf", mosaic, weight decay e warm-up epochs.

3.7.1 Learning rate iniziale (lr0)

Il tasso di apprendimento iniziale o "lr0" è stato esplorato nell'intervallo racchiuso tra 10^{-3} e 10^{-1}. Tale scelta tiene conto del fatto che SGD, a differenza degli ottimizzatori come AdamW, richiede un learning rate iniziale più elevato per garantire una più facile convergenza, pur risultando al contempo sensibile a valori eccessivamente alti che possono compromettere la stabilità del training. Studi su architetture YOLOv8 addestrate con SGD riportano un learning rate iniziale pari a 0.01, con valori finali che decadono fino a 0.0001 [40]; il valore di partenza è già il valore di default utilizzato dalla piattaforma di Ultralytics e rappresenta quindi un punto di partenza già consolidato.

3.7.2 Frazione finale del learning rate

Il parametro “lrf” definisce la frazione finale del learning rate. Insieme a “lr0” (il learning rate iniziale), determina come il valore decresce durante l'addestramento. Questo parametro influenza direttamente la capacità del modello di stabilizzarsi nelle ultime epoche di training. Ricerche su architetture YOLO hanno identificato lr0 e lrf come gli strumenti più opportuni per l'ottimizzazione dello schedule del learning rate, assieme a batch size, epoche e weight decay [41]. Se impostato ad esempio al valore di 0.01, il learning rate finale sarà l'1% di quello iniziale.

Il fattore di decadimento del learning rate è stato limitato al range di valori compresi tra 0.01 e 0.2. Un limite superiore di 0.2 è stato ritenuto sufficiente, in quanto valori più elevati impedirebbero un decadimento adeguato del learning rate, riducendo l'effetto di fine-tuning nelle epoche conclusive.

3.7.3 Weight decay

Il weight decay (decadimento dei pesi) è un parametro di regolarizzazione utilizzato durante l'addestramento dei modelli per prevenire l'overfitting. Con un dataset di dimensioni ridotte, il rischio di overfitting è infatti una delle principali criticità da affrontare, come già discusso nella sezione dedicata alla fase di training.

La regolarizzazione tramite weight decay (weight_decay) è stata esplorata nell'intervallo compreso tra 10^{-4} e $5x10^{-3}$. L'intervallo scelto consente al tuner di esplorare sia configurazioni con regolarizzazione più debole che più intensa.

Il weight decay è stato usato per due soli tuning: ad imgsz 160, ovvero per il modello che verrà usato molto più frequentemente, e quello ad imgsz 640, ovvero il modello che presenta la migliore localizzazione in termini di mAP@50-95.

3.7.4 Mosaic

Il Mosaic è una tecnica di data augmentation che combina quattro immagini di addestramento in una sola, affiancandole insieme in un'unica griglia. Invece di mostrare al modello singole immagini, il Mosaic crea campioni artificiali che costringono la rete a riconoscere gli oggetti in contesti diversi, scale ridotte e posizioni varie all'interno di un singolo frame. Il range di valori è tra 0 e 1, dove mosaic=1 implica che tutte le immagini vengano combinate mentre mosaic=0 disattiva completamente la trasformazione [42].

3.7.5 Warm-up epochs

Infine, Il warmup_epochs è un parametro che definisce il numero di epoche iniziali durante le quali il learning rate aumenta gradualmente partendo da un valore molto basso fino a raggiungere quello impostato (lr0). Serve a prevenire l'instabilità dell'addestramento nelle prime fasi, evitando che gradienti troppo elevati danneggino i pesi del modello appena inizializzati.

Il numero di epoche di warm-up è stato esplorato nell'intervallo di valori compreso tra 1 e 5. In particolare, il warmup previene l'instabilità del training limitando gli aggiornamenti ai pesi dei layer profondi nelle prime fasi dell'ottimizzazione. Con SGD, che non dispone di

meccanismi adattativi interni (a differenza di ottimizzatori come Adam o le sue varianti), questa fase assume un'importanza particolare: un numero insufficiente di epoche di warm-up può causare oscillazioni o divergenza nelle iterazioni iniziali, mentre un warm-up eccessivamente lungo riduce le epoche effettivamente utili per la convergenza, specialmente in presenza di dataset ridotti.

3.7.6 Risultati

Si riportano i risultati dei vari tuning degli iperparametri per ogni dimensione di immagine in una tabella riassuntiva.

IMGSZ	WARM-UP EPOCHS	lr0	lrf	MOSAIC	WEIGHT DECAY
160	2,94979	0,0101	0,01044	0,763	0,0001
320	2,99846	0,02182	0,01205	0,951	-
480	5	0,00722	0,01235	0,999	-
512	3,00146	0,01134	0,01575	1	-
640	3,00297	0,0157	0,1754	0,982	0,0001

Tabella 9 risultati tuning iperparametri per imgsz

Uno dei risultati cha emerge immediatamente è il valore del parametro di mosaic a imgsz=160: a questa dimensione le immagini sono già molto piccole e un'applicazione troppo aggressiva del mosaic (ad esempio per valori attorno allo 0,9) dividerebbe ulteriormente lo spazio disponibile, rendendo gli oggetti (già composti da pochi pixel) difficili da distinguere. Il valore individuato dal tuning, 0.76 permette al modello di vedere immagini intere con maggior frequenza; inoltre, considerando un dataset con un numero di immagini così ridotto, valori più elevati di mosaic rischierebbero di introdurre troppa confusione durante il training, portando il modello ad apprendere prevalentemente rumore piuttosto che caratteristiche utili alla detection. Per le altre dimensioni di immagine, al contrario, un valore di mosaic più elevato risulta più giustificato data la maggiore densità di informazioni portata dalla maggiore quantità di pixel.

Gli altri valori degli iperparametri forniti dai vari tuning si aggirano tutti attorno ai valori classici già riportati di default.

3.8 Training finale e scelta del modello migliore pre-deploy

Successivamente alla fase di tuning, è stato eseguito un training finale con i valori degli iperparametri forniti dal tuning, con il numero di epoche di training settato a 150. In questo modo, si è data la possibilità al modello di apprendere appieno il dataset, maggiorandone il numero di epoche rispetto alle fasi di training precedenti

Si riportano i grafici di precision-recall, F1-confidence, recall-confidence e precision-confidence forniti dal training per ogni imgsz.

3.8.1 Dimensione 160x160

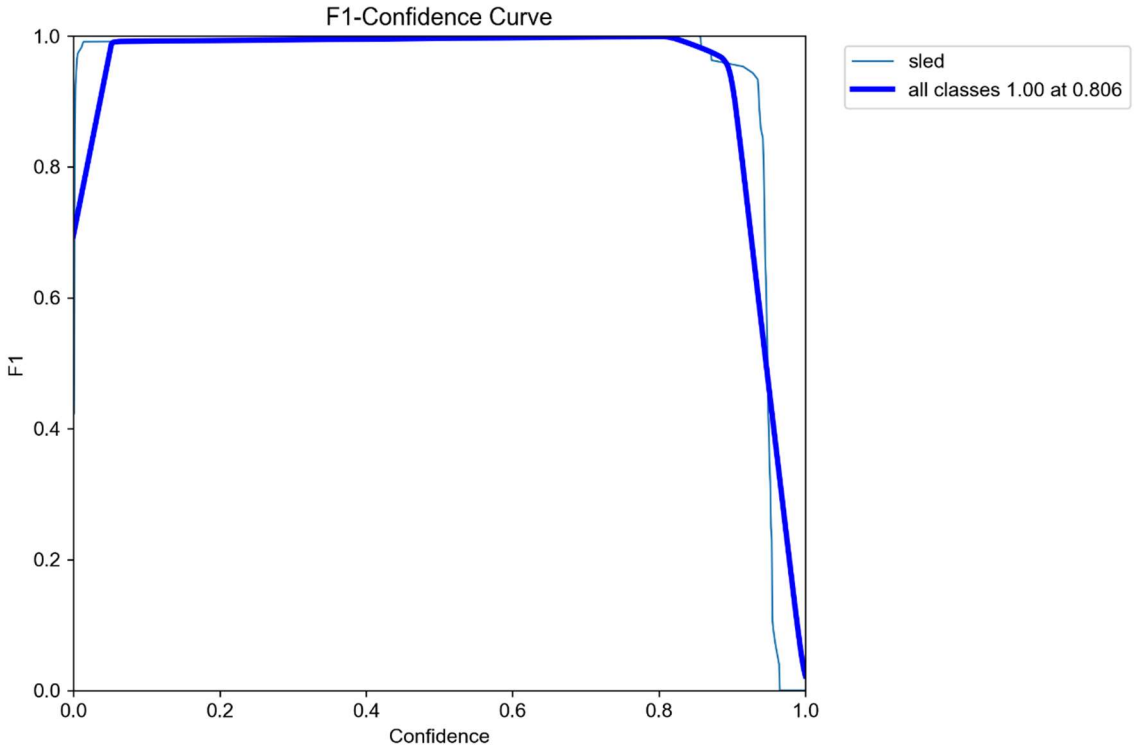


Figura 35 F1-confidence per imgsz 160

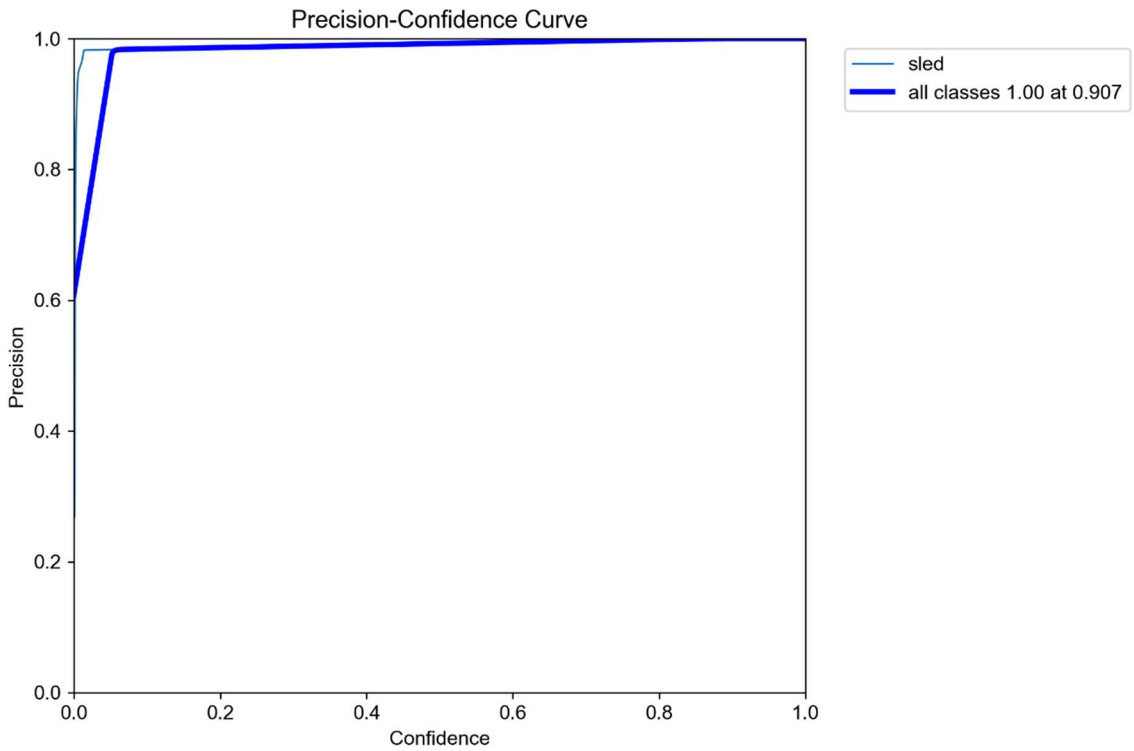


Figura 36 precision-confidence per imgsz 160

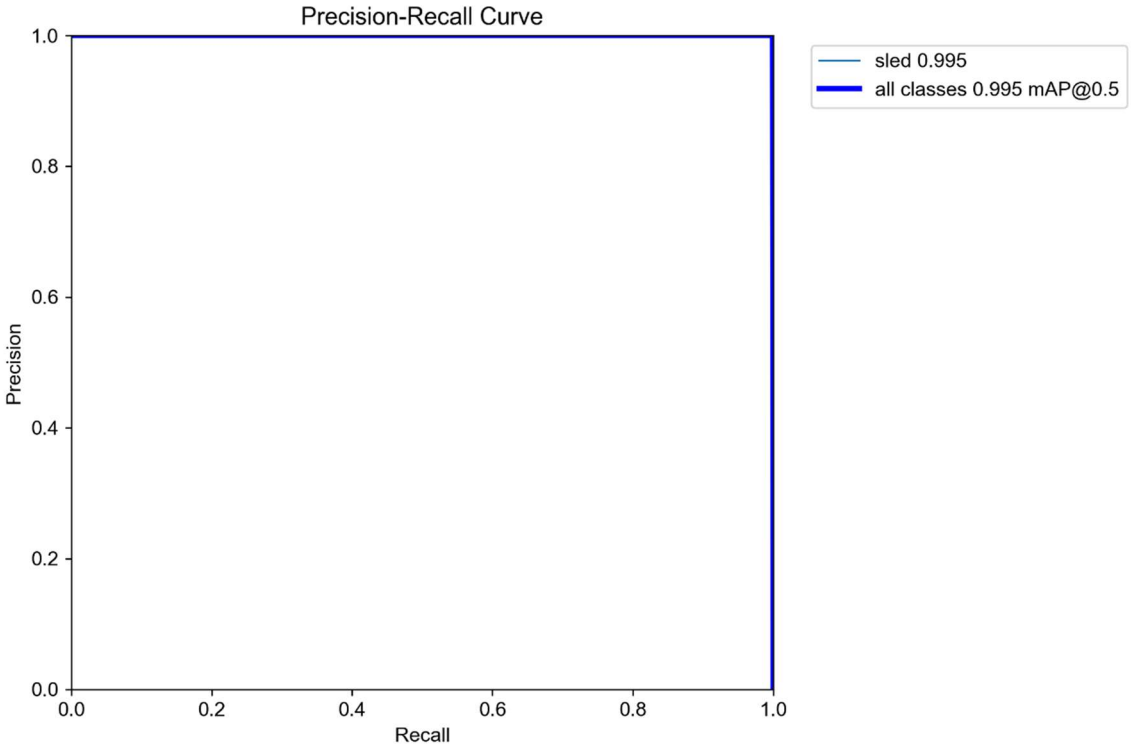


Figura 38 precision-recall per imgsz 160

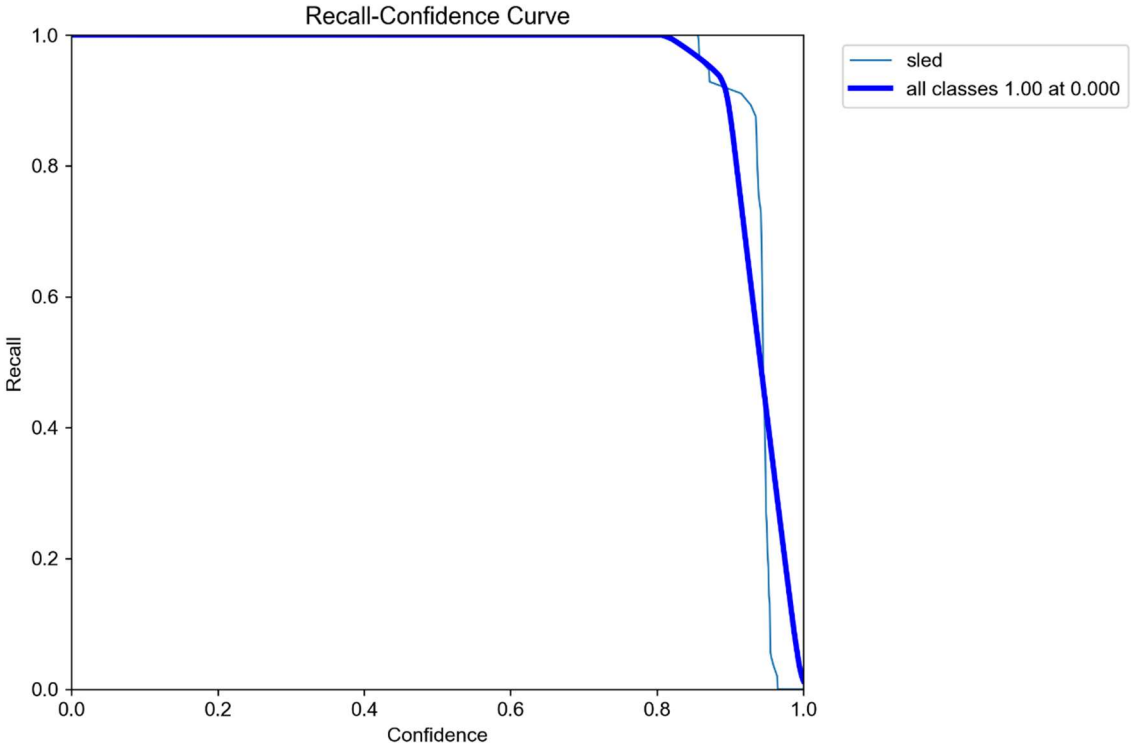


Figura 37 recall-confidence per imgsz 160

L'analisi delle curve di valutazione prodotte al termine del training conferma le ottime prestazioni del modello YOLO v8n sul task di rilevamento del target. La curva Precision-Recall, riportata in Figura 38 precision-recall per imgsz 160, evidenzia un valore di mAP@0.5 pari a 0.995, con un andamento che approssima un rettangolo quasi perfetto: ciò conferma quanto riportato nella sezione dedicata alla scelta dell'ottimizzatore. La precisione si mantiene costante e prossima all'unità per l'intero intervallo di recall, fino a un crollo verticale in prossimità dell'estremo destro del range. Tale comportamento rappresenta l'indicatore più diretto di una classificazione quasi ideale, in cui il compromesso tra falsi positivi e falsi negativi risulta trascurabile a qualsiasi soglia operativa, difatti il modello trova il target nel 100% delle immagini.

Questa osservazione trova ulteriore conferma nelle curve derivate dalla confidence. La curva Precision-Confidence, Figura 36 precision-confidence per imgsz 160, mostra come la precisione superi già valori prossimi a 0.95 per soglie di confidenza molto basse (a partire da circa 0.05), per poi raggiungere il valore massimo di 1 per valori di confidence di 0.907: il modello genera dunque pochissimi falsi positivi anche quando vengono considerate detection a bassa confidenza. Speculare è il comportamento della curva Recall-Confidence, Figura 37 recall-confidence per imgsz 160, che si mantiene costante al valore massimo (1.00) fino a una soglia di confidenza di circa 0.8, per poi decrescere rapidamente; questo indica che la maggior parte delle detection corrette viene prodotta dal modello con un livello di confidenza medio-alto.

La sintesi di questi due comportamenti si riflette nella curva F1-Confidence, Figura 35 F1-confidence per imgsz 160, che raggiunge il valore massimo (F1=1,00) in corrispondenza del valore di confidence 0.806, mantenendo un plateau ampio e stabile approssimativamente tra 0.1 e 0.8. L'ampiezza di tale plateau costituisce un indicatore di robustezza del modello rispetto alla scelta della soglia di confidenza operativa: eventuali variazioni della soglia all'interno di questo intervallo non comportano un deterioramento significativo delle prestazioni. Inoltre, è utile sottolineare che per valori di confidence prossimi allo zero, le predizioni del modello sono molteplici, aspetto che può essere spiegato dalla dimensione d'immagine molto ridotta.

È tuttavia opportuno contestualizzare criticamente risultati così prossimi alla perfezione. Un mAP@0.5 di 0.995 e un F1-score massimo pari a 1.00, ottenuti su un dataset monoclasse di dimensioni contenute, riflettono verosimilmente l'elevata omogeneità del dominio di acquisizione, stesso sensore (IMX708), stesse condizioni di illuminazione, stesso sfondo e medesima configurazione meccanica tra i sottoinsiemi di training, validazione e test, più che un'effettiva capacità di generalizzazione del modello a scenari visivamente differenti. Queste ultime considerazioni vanno intese come limite metodologico, poiché queste prestazioni vanno intese come ottenute in un dominio chiuso e controllato caratteristico di poca variabilità.

3.8.2 Dimensione 320x320

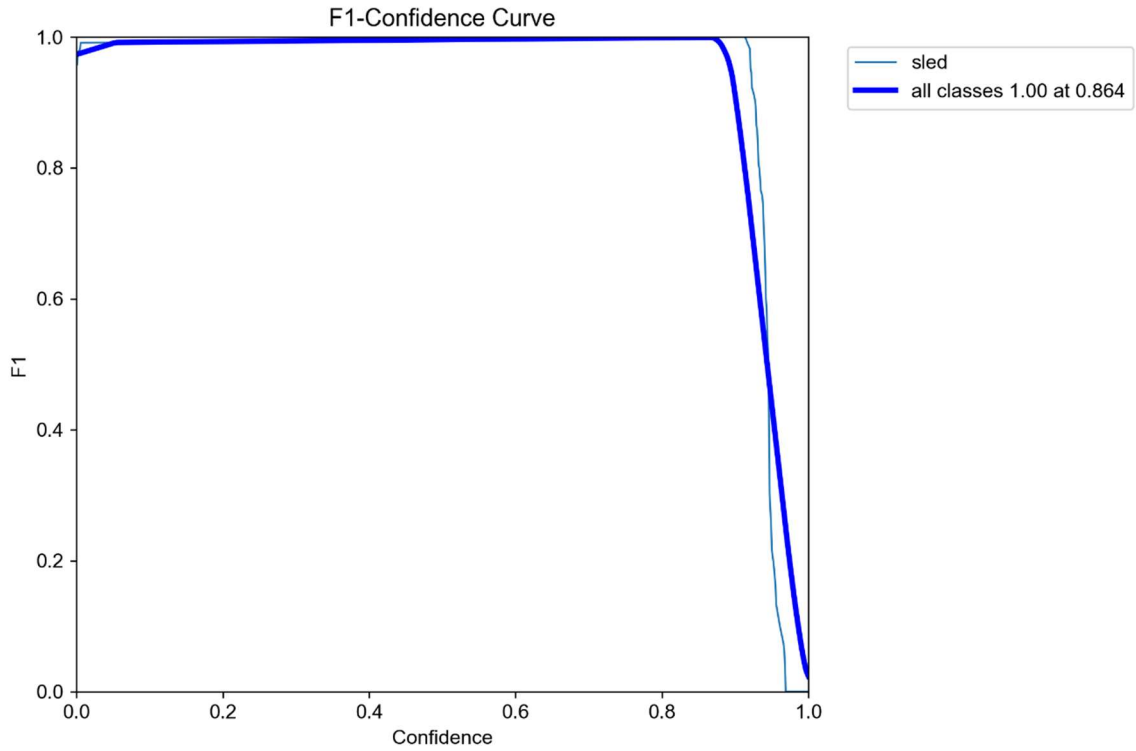


Figura 39 F1-confidence per imgsz 320

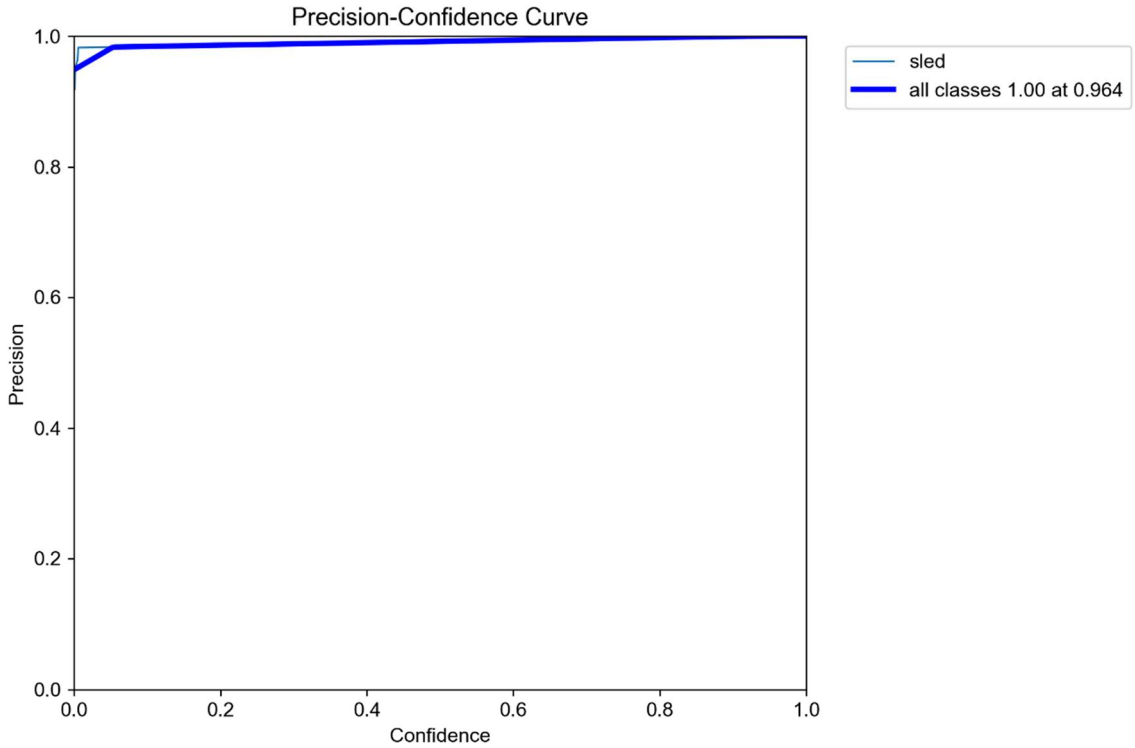


Figura 40 precision-confidence per imgsz 320

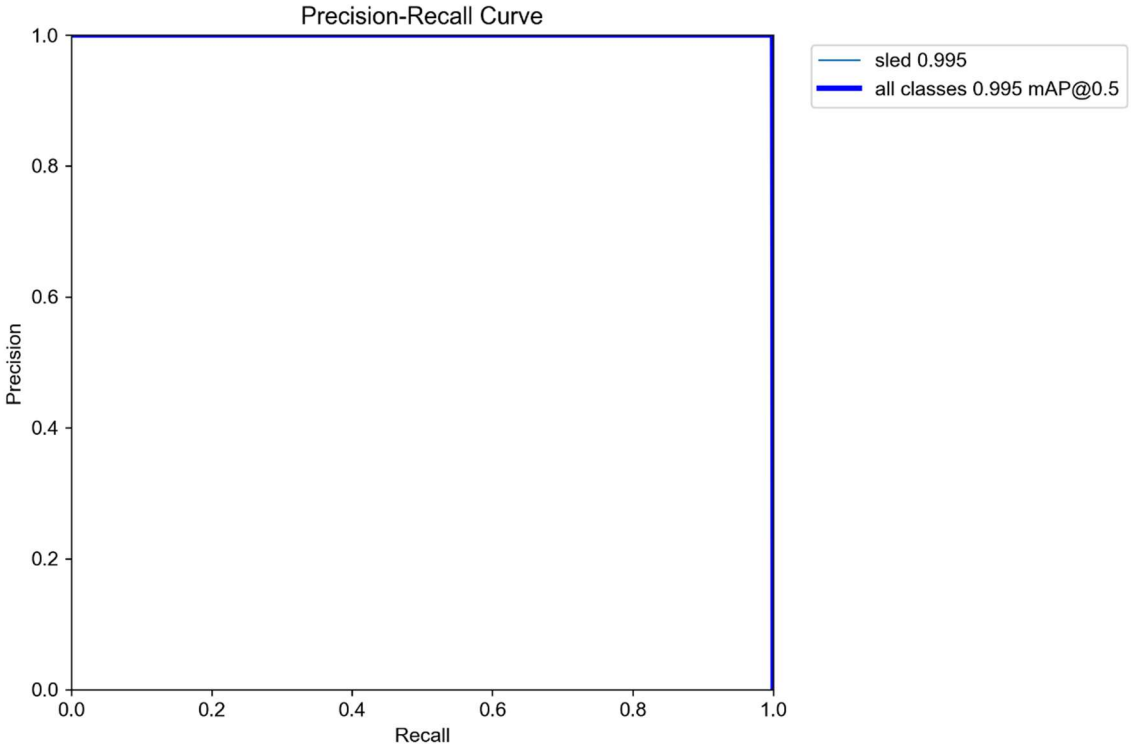


Figura 41 precision-recall per imgsz 320

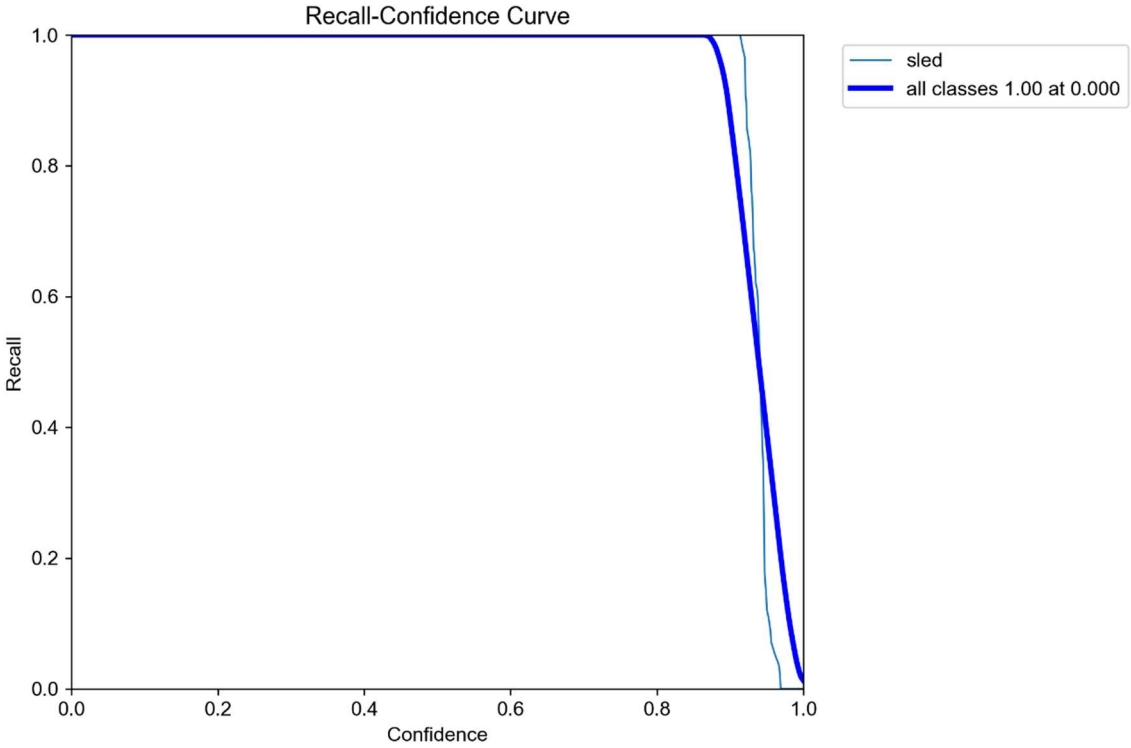


Figura 42 recall-confidence per imgsz 320

I risultati ottenuti dall'addestramento a imgsz 320 conducono alle stesse conclusioni già osservate per il caso a imgsz 160, con una differenza sostanziale: in Figura 39 F1-confidence per imgsz 320, la F1 si mantiene a valori prossimi all'unità già a partire da valori di confidenza nulli, sintomo che l'aumento della quantità di pixel presenti nell'immagine di input porti il modello verso una classificazione più precisa. Viene infine riconfermata la stabilità al valore massimo della mAP@50 questo a riprova del fatto che il compito che il modello deve raggiungere è senza dubbio molto facile soprattutto se viene impostata una confidence bassa.

3.8.3 Dimensione 480x480

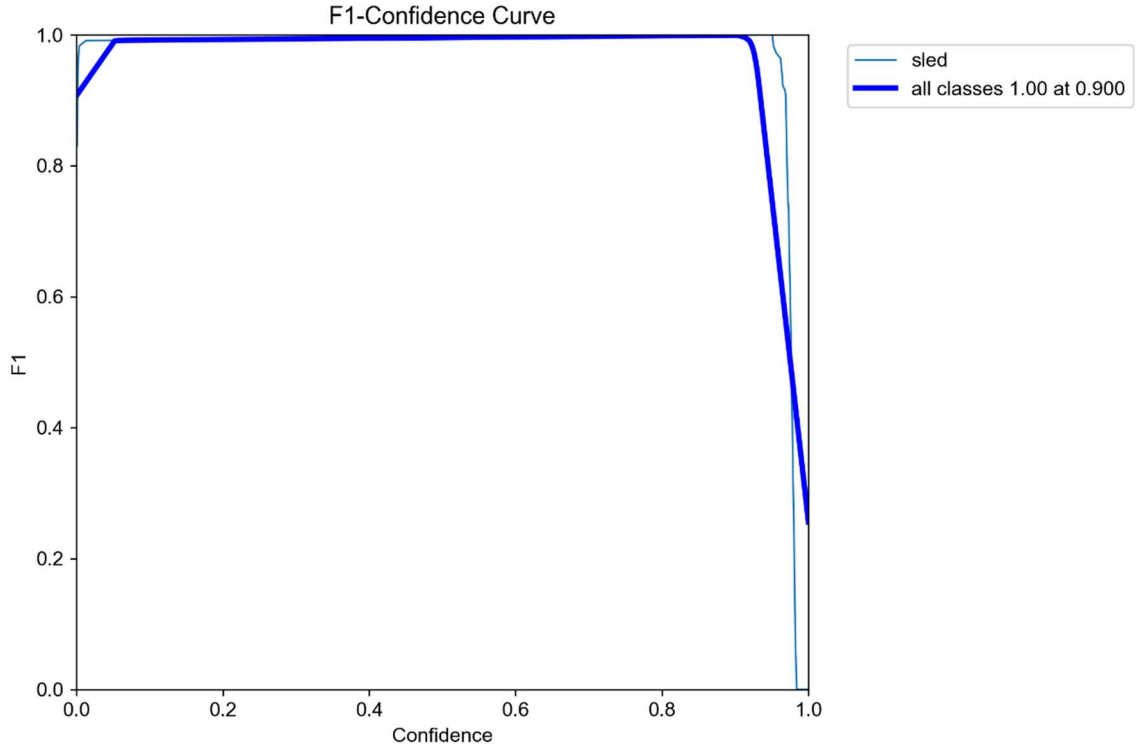


Figura 43 F1-confidence per imgsz 480

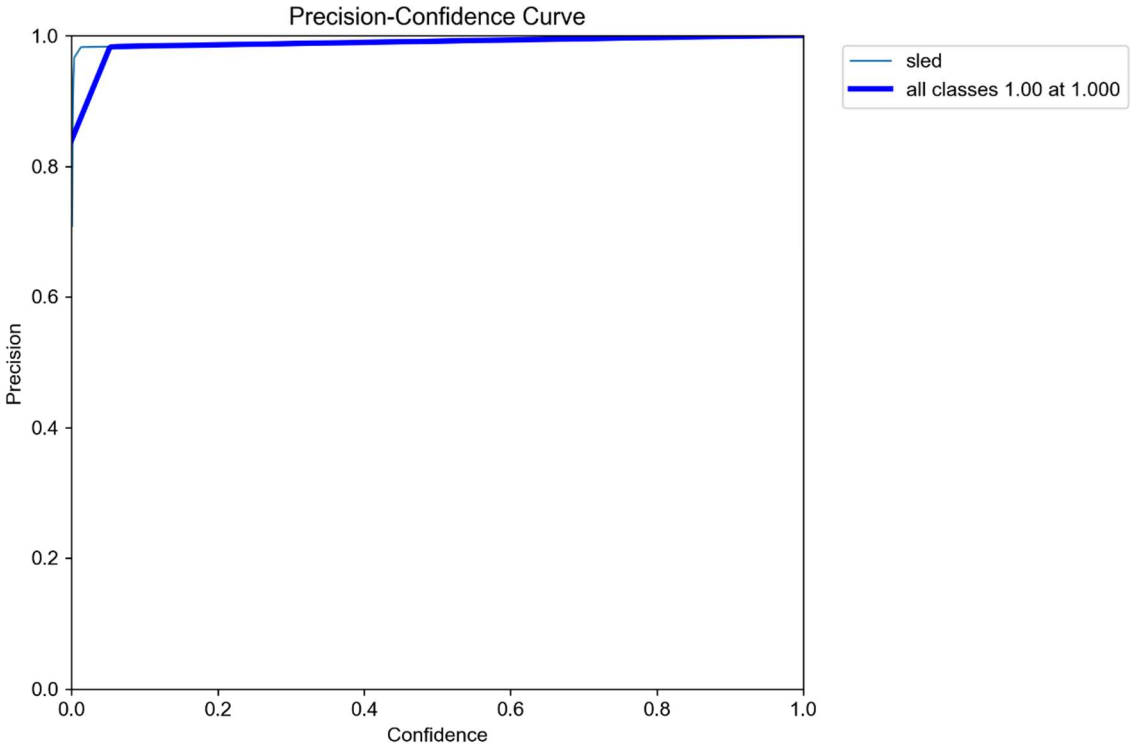


Figura 44 precision-confidence per imgsz 480

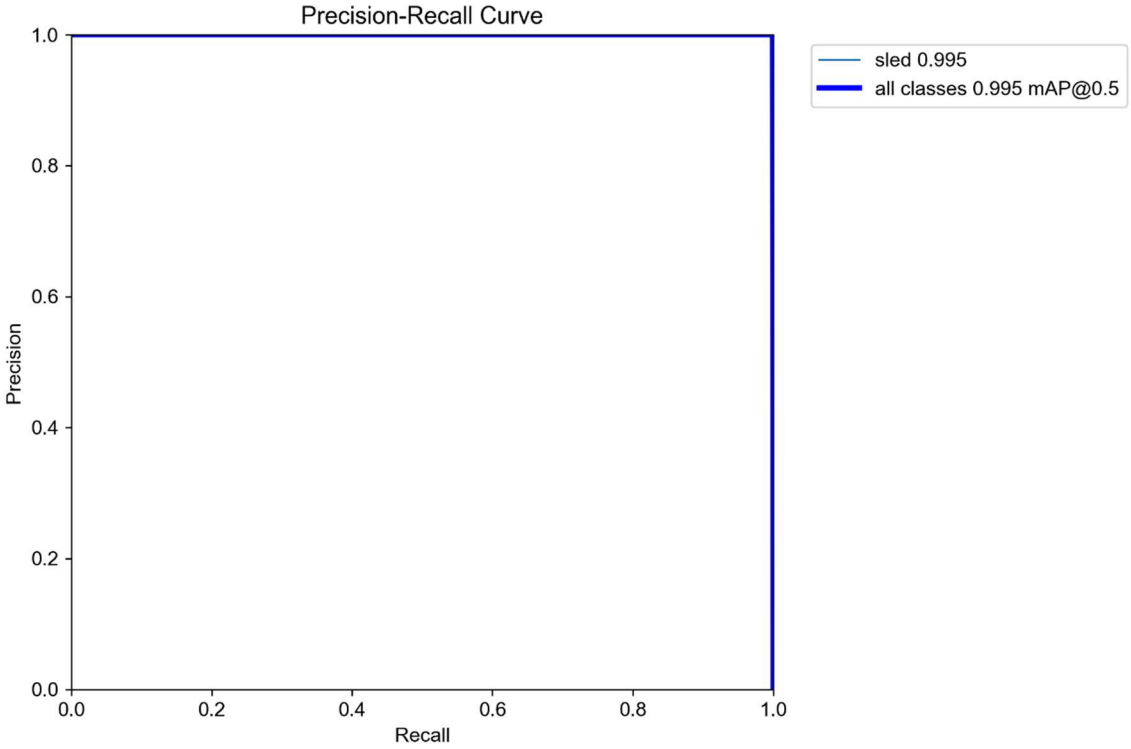


Figura 45 precision-recall per imgsz 480

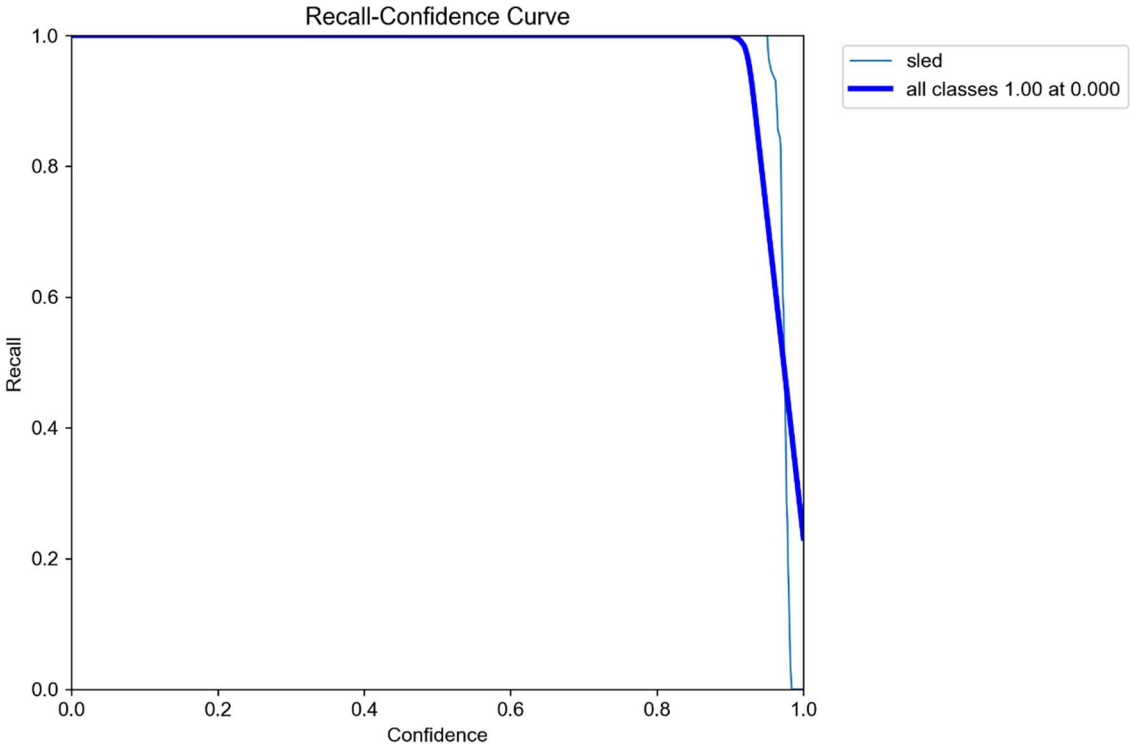


Figura 46 recall-confidence per imgsz 480

Dai dati relativi alla dimensione 480x480, si conferma una tendenza verso valori di soglia di confidenza ancora più alti: il valore massimo di F1 viene raggiunto al valore di confidenza di 0.900 mentre il valore massimo di precision si ottiene a confidenza 1, come osservato nella Figura 43 F1-confidence per imgsz 480 e Figura 44 precision-confidence per imgsz 480. È da notare, in Figura 46 recall-confidence per imgsz 480, che per valori di confidenza molto bassi, il modello produce delle detection corrette questo confermato anche dalla Figura 45 precision-recall per imgsz 480.

3.8.4 Dimensione 512x512

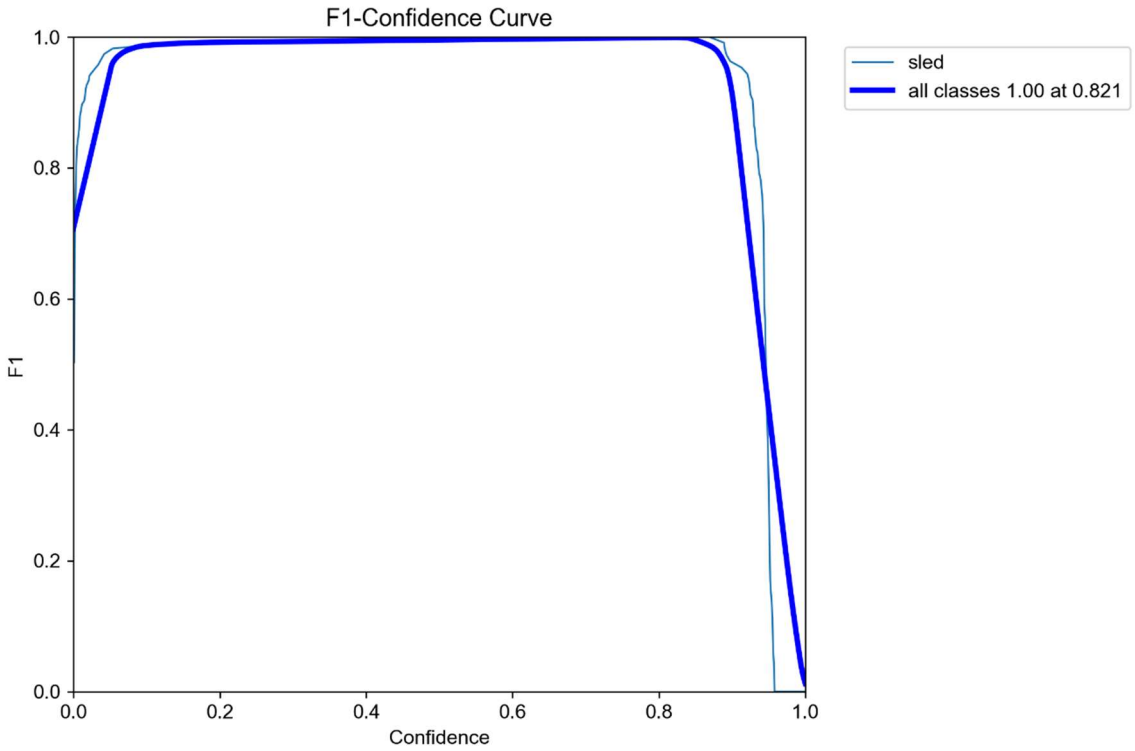


Figura 47 F1-confidence per imgsz 512

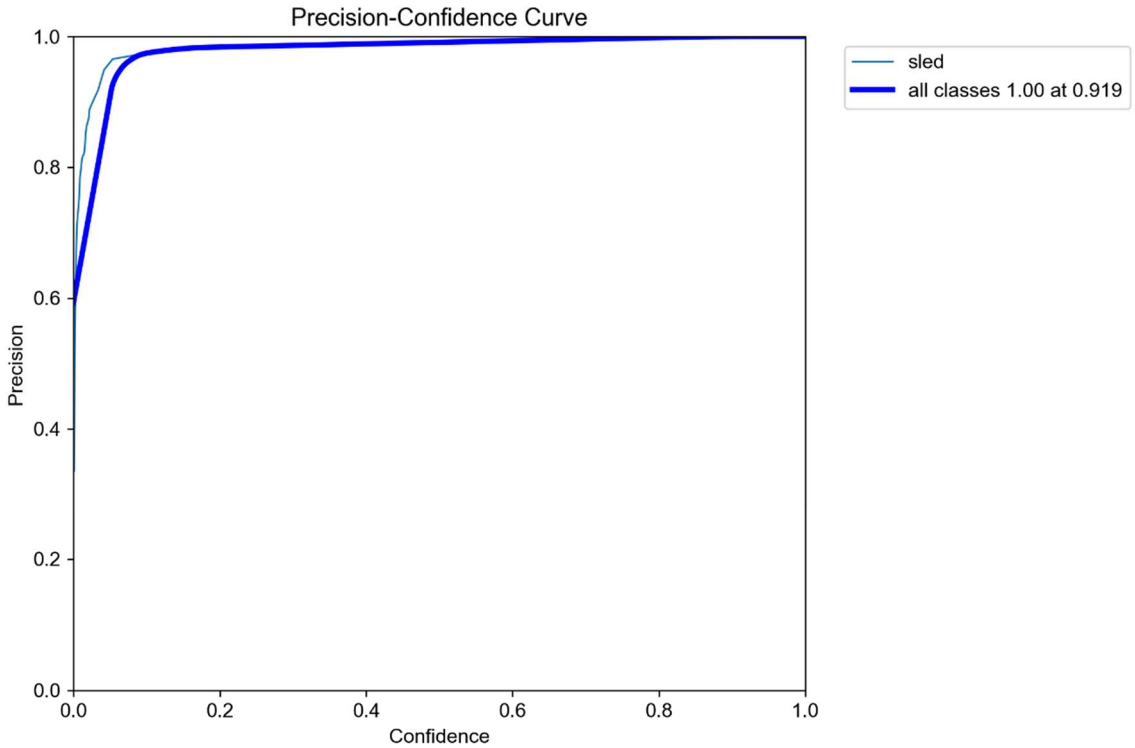


Figura 48 precision-confidence per imgsz 512

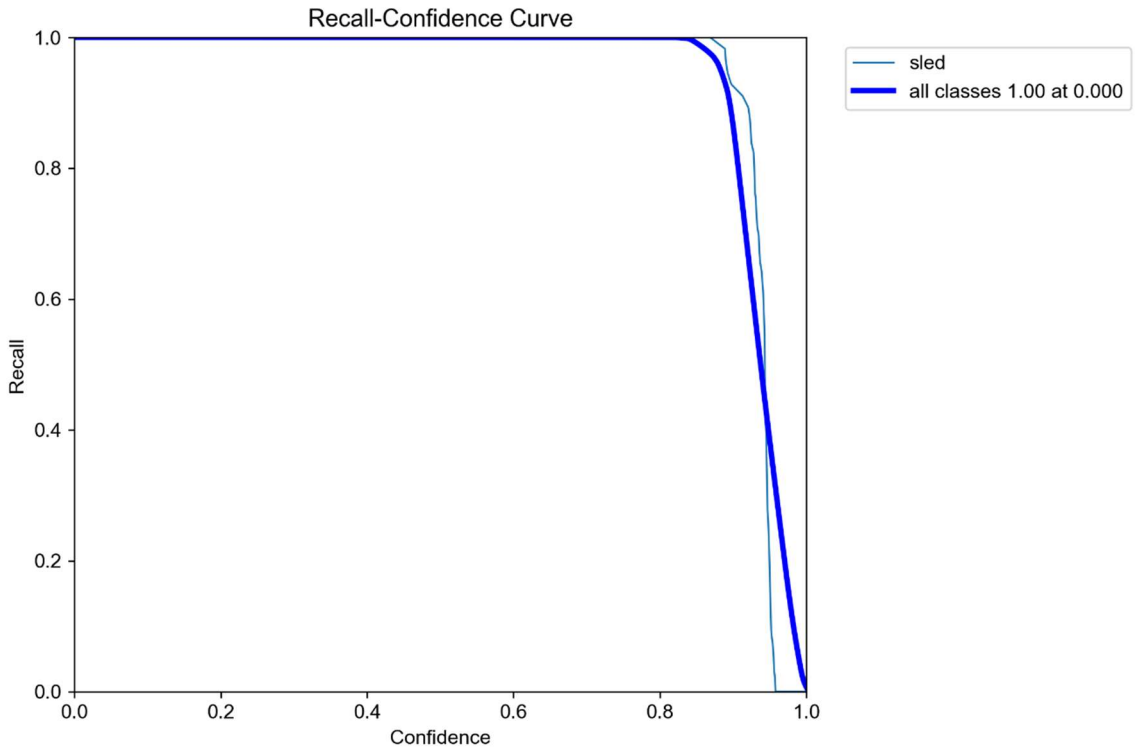


Figura 49 recall-confidence per imgsz 512

I valori riportati nelle figure sopra riportate, non confermano la tendenza lineare che è stata ipotizzata precedentemente: i valori massimi si attestano a 0.821 per la F1 (Figura 47)e 0.919 per la precision (Figura 48), risultando quindi inferiori rispetto ai valori osservati per imgsz 480.

3.8.5 Dimensione 640x640

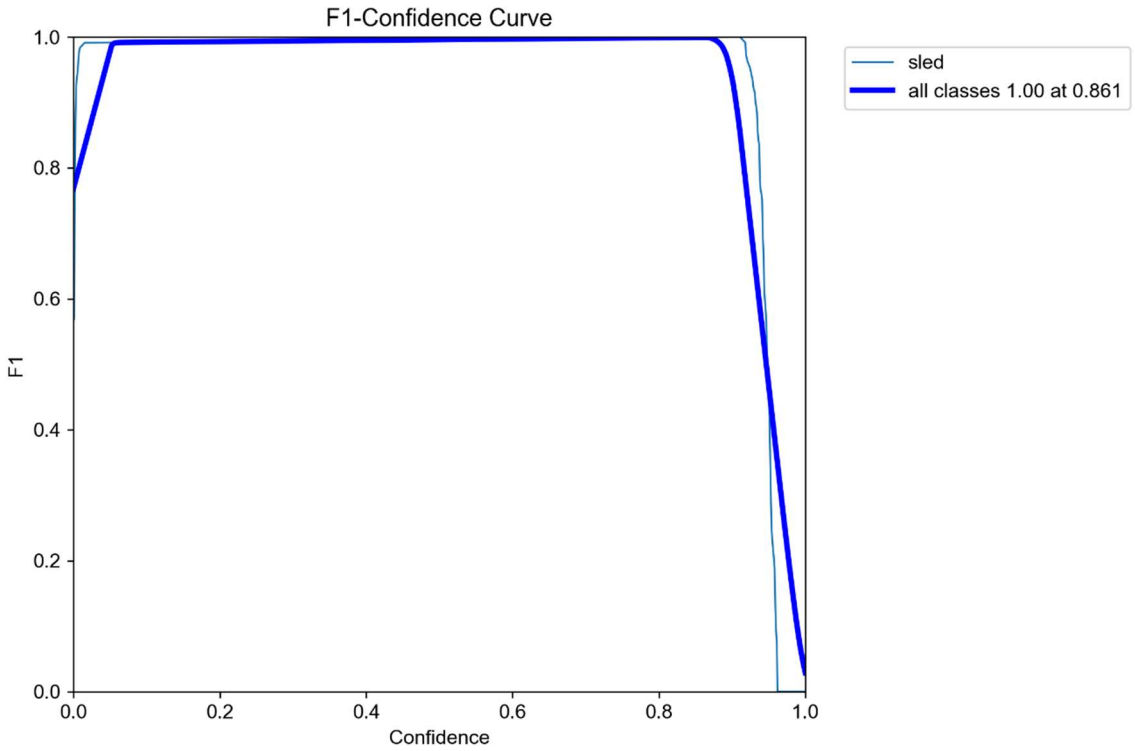


Figura 50 F1-confidence per imgsz 640

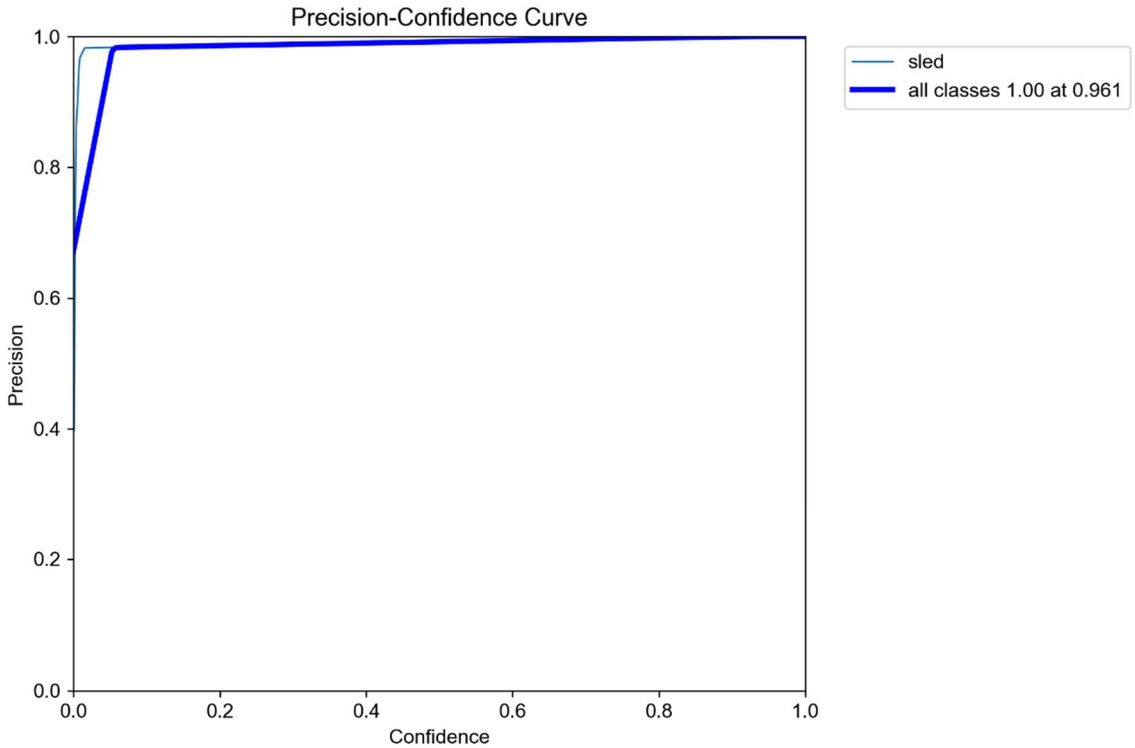


Figura 51 precision-confidence per imgsz 640

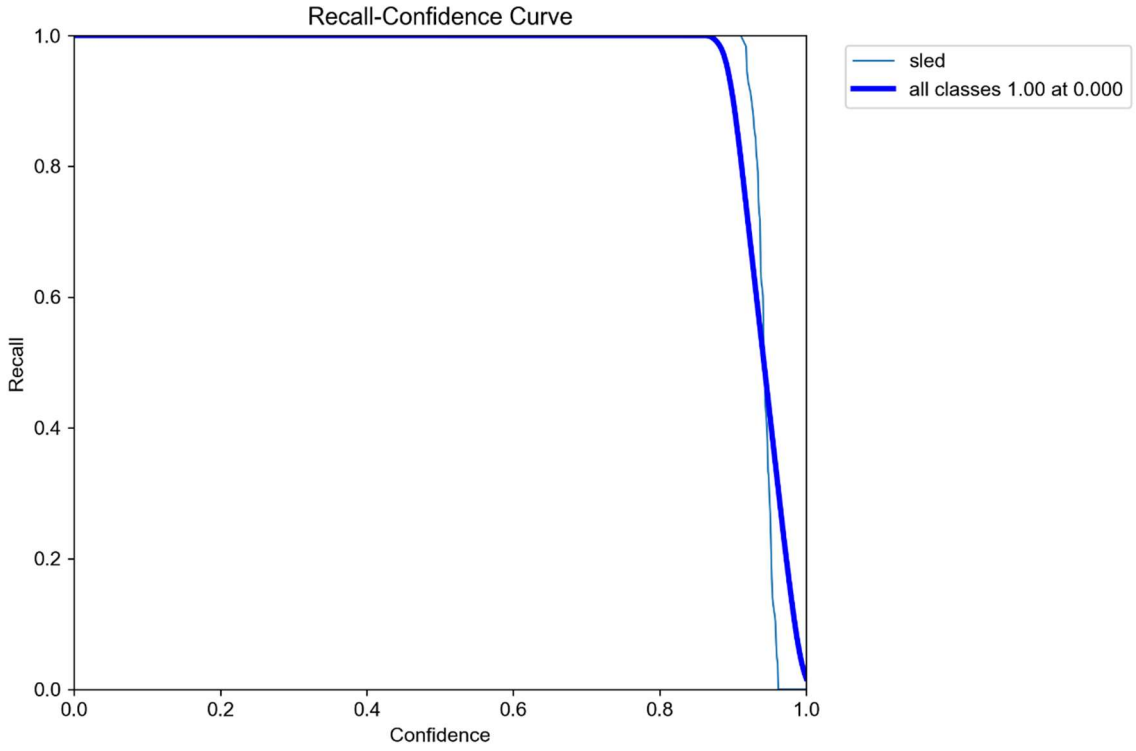


Figura 52 recall-confidence per imgsz 640

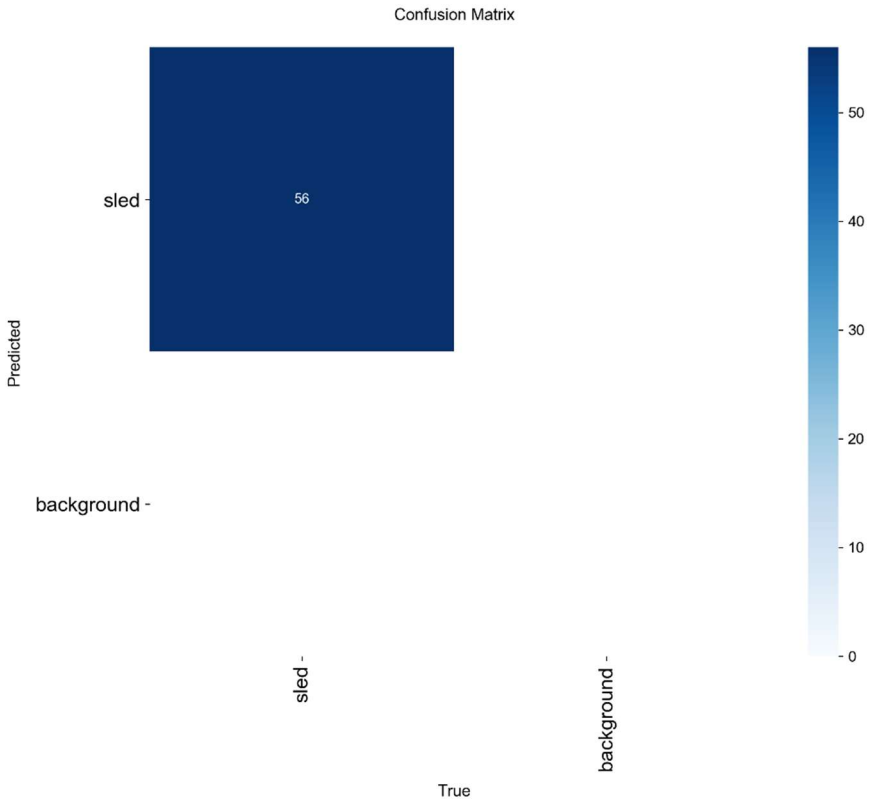


Figura 53 matrice di confusione per imgsz 640

Si riporta, in questo caso, la matrice di confusione. Il punteggio di detection del target nel corso della validazione è ottimo: il modello trova il target nel 100% delle immagini del gruppo di validazione. Questo risultato vale per tutti i modelli a imgsz diverse; una possibile spiegazione di questo comportamento risiede anche nella modalità di costruzione del dataset: il gruppo di training è stato volutamente reso più impegnativo da apprendere in modo da favorire un riconoscimento più agevole e affidabile quando al modello viene successivamente presentata un'immagine relativamente più semplice, come tipicamente accade nel gruppo di validazione.

Anche in questo caso, la curva di precision-recall non viene riportata perché non fornisce nessun riscontro aggiuntivo in quanto il modello ottiene sistematicamente il punteggio massimo di 0.995 di mAP@0.50 indipendentemente dalla dimensione di immagine considerata.

3.8.6 Conclusioni

I dati riportati permettono di trarre alcune considerazioni rilevanti. In primo luogo, il modello riesce a rilevare correttamente la totalità dei target presenti nelle immagini del set di validazione, indipendentemente dalla risoluzione di input adottata: non si riscontra infatti alcuna presenza di falsi positivi o falsi negativi in nessuna delle configurazioni testate. Va sottolineato che le prestazioni pressoché perfette ottenute a tutte le risoluzioni non sono casuali, ma costituiscono una diretta conseguenza della relativa semplicità del problema di detection affrontato, coerentemente con la scelta di un target geometricamente e cromaticamente invariante. Questo conduce a un'ulteriore considerazione: il "collo di bottiglia" delle prestazioni non risiede nella capacità del modello di distinguere la slitta dallo sfondo, compito già risolto in modo pressoché ottimale anche alla risoluzione minima, bensì nella necessità di massimizzare l'efficienza computazionale per soddisfare i vincoli di latenza imposti dall'hardware embedded di destinazione.

In secondo luogo, i parametri effettivamente da prendere in considerazione nel confronto tra le diverse risoluzioni sono F1, precision e recall in funzione della confidence, in particolare i rispettivi valori di soglia, in quanto risultano essere gli unici indicatori a mostrare una variazione apprezzabile al variare dell'imgsz.

Sulla base di queste premesse, la configurazione più adatta per il deployment su Raspberry Pi CM5 risulta essere quella a risoluzione più bassa, ovvero 160 pixel, per due motivi principali. Il primo è strettamente legato al carico computazionale: maggiore è la dimensione dell'immagine in input, maggiore è il numero di operazioni richieste per ciascuna detection, con un impatto diretto sulla latenza di inferenza [23]. Il secondo motivo è legato ai valori di soglia a cui si attestano F1, precision e recall: il modello a risoluzione minima ottiene infatti valori comparabili a quelli alla risoluzione massima, senza un miglioramento sistematico all'aumentare dell'imgsz. Di conseguenza, a fronte di prestazioni sostanzialmente equivalenti, tra le diverse configurazioni testate, è stato scelto il modello più leggero dal punto di vista computazionale, in vista dell'applicazione della rete in un contesto di detection in tempo reale.

4.FASE SPERIMENTALE (DEPLOY)

Una volta completata la fase di addestramento del modello YOLO v8n, la sperimentazione si è concentrata sul trasferimento del modello verso la piattaforma embedded, rappresentata dal Raspberry Pi CM5. Questa fase riveste un'importanza critica ai fini del progetto, poiché è in questo passaggio che le prestazioni teoriche del modello, misurate in termini di accuratezza sul dataset di validazione, devono confrontarsi con i vincoli reali imposti dall'hardware a risorse limitate e dai requisiti di funzionamento in tempo reale del sistema di controllo

Ai fini del deploy, il modello addestrato in formato PyTorch è stato sottoposto a un processo di conversione verso il formato NCNN, un framework di inferenza open-source sviluppato da Tencent specificamente per l'esecuzione efficiente di reti neurali su dispositivi embedded [43].

La scelta di NCNN come target di export è stata motivata dalla sua ottimizzazione per architetture ARM, pienamente compatibile con il processore del Raspberry Pi CM5, nonché dal supporto a diverse precisioni numeriche, elemento che ha reso possibile la successiva fase di quantizzazione. L'intera pipeline di conversione, dal modello PyTorch originale fino alle varianti quantizzate, è stata sviluppata e validata in ambiente Windows, per poi essere trasferita ed eseguita sulla piattaforma target.

La fase sperimentale si sviluppa su diversi punti:

- confronto iniziale del frame rate (FPS**),** o equivalentemente del tempo di inferenza, ottenuto in fase di inferenza tra i vari modelli a risoluzione diversa, quale indicatore delle prestazioni in termini di velocità di posizionamento del target con successiva scelta della imgsz
- confronto successivo sia della confidenza associata alle rilevazioni, per verificare l'eventuale degrado introdotto dalla quantizzazione, sia del posizionamento delle bounding box tutto ciò per validare la scelta di un livello di quantizzazione rispetto ad un altro

4.1 Stato dell'arte

I dispositivi attualmente disponibili per l'inferenza edge si collocano su un ampio spettro di prestazioni e costi. Tra le soluzioni più diffuse vi è la famiglia NVIDIA Jetson Orin, che offre moduli con prestazioni AI comprese tra 67 TOPS per la serie Orin Nano e 275 TOPS per la serie AGX Orin [44]. In una posizione più accessibile si colloca il Google Coral Edge TPU, un acceleratore dedicato che eroga 4 TOPS con un consumo di circa 0,5 W per TOPS, risultando particolarmente efficiente seppure vincolato all'esecuzione di reti quantizzate in INT8 [45]. Un ulteriore attore rilevante nel panorama è rappresentato dalla NPU Hailo, disponibile in due varianti: la Hailo-8, che raggiunge fino a 26 TOPS e la sua versione entry-level Hailo-8L, che offre fino a 13 TOPS, pensata per applicazioni con requisiti computazionali più contenuti [46].

Un aspetto rilevante per i sistemi basati su single-board computer come il Raspberry Pi CM5, impiegato nel presente lavoro, è la possibilità di estenderne le capacità computazionali tramite acceleratori hardware esterni. Un esempio concreto è il Raspberry Pi AI Kit, che integra un acceleratore per l'inferenza di reti neurali da 13 TOPS basato sul chip Hailo-8L, oppure la più recente AI HAT+, che è disponibile in due varianti, una da 13 TOPS basata su Hailo-8L e una da 26 TOPS basata su Hailo-8. Questa possibilità di espansione modulare rappresenta una direzione promettente per sviluppi futuri del presente lavoro: mantenendo invariata l'architettura software sviluppata, l'aggiunta di un acceleratore dedicato potrebbe consentire l'esecuzione di modelli più complessi o l'aumento della frequenza di inferenza, avvicinandosi ulteriormente ai requisiti di un sistema di controllo di posizione in tempo reale [46].

È opportuno sottolineare che l'estensione hardware sopra descritta riguarda esclusivamente la fase di inferenza e non quella di training. Condurre l'addestramento della rete direttamente sul dispositivo embedded risulterebbe infatti lungo e tedioso: le NPU dedicate come quelle descritte sono ottimizzate per l'esecuzione efficiente di reti già addestrate e quantizzate, non per il calcolo dei gradienti e l'aggiornamento iterativo dei pesi tipico della fase di training, che richiede risorse di calcolo, memoria e tempi di elaborazione difficilmente sostenibili su una piattaforma a basso consumo. Per questo motivo, in linea con la prassi consolidata nello sviluppo di sistemi embedded, il training del modello è stato condotto su un dispositivo desktop dotato di GPU dedicata, riservando al Raspberry Pi CM5 il solo compito di eseguire l'inferenza sul modello già addestrato e ottimizzato.

4.2 Analisi su imgsz

Prima di scegliere il modello definitivo per il deploy sul dispositivo Raspberry Pi CM5, è stata condotta una prima fase di analisi volta alla scelta della risoluzione d’immagine o “imgsz”.

L'obiettivo di questa analisi è fornire una stima della differenza in termini di tempo di inferenza alle varie risoluzioni.

La risoluzione influisce direttamente sulla velocità del modello: un modello con risoluzione più elevata ha più informazioni da processare quindi sarà necessario un tempo di elaborazione maggiore per ciascun frame; vale il discorso inverso per risoluzioni più basse, per cui il modello dovrebbe risultare più veloce.

In questa fase, viene valutato solo il tempo di inferenza come principale caratteristica poiché è il parametro che risente pesantemente del restringimento della finestra di input al modello.

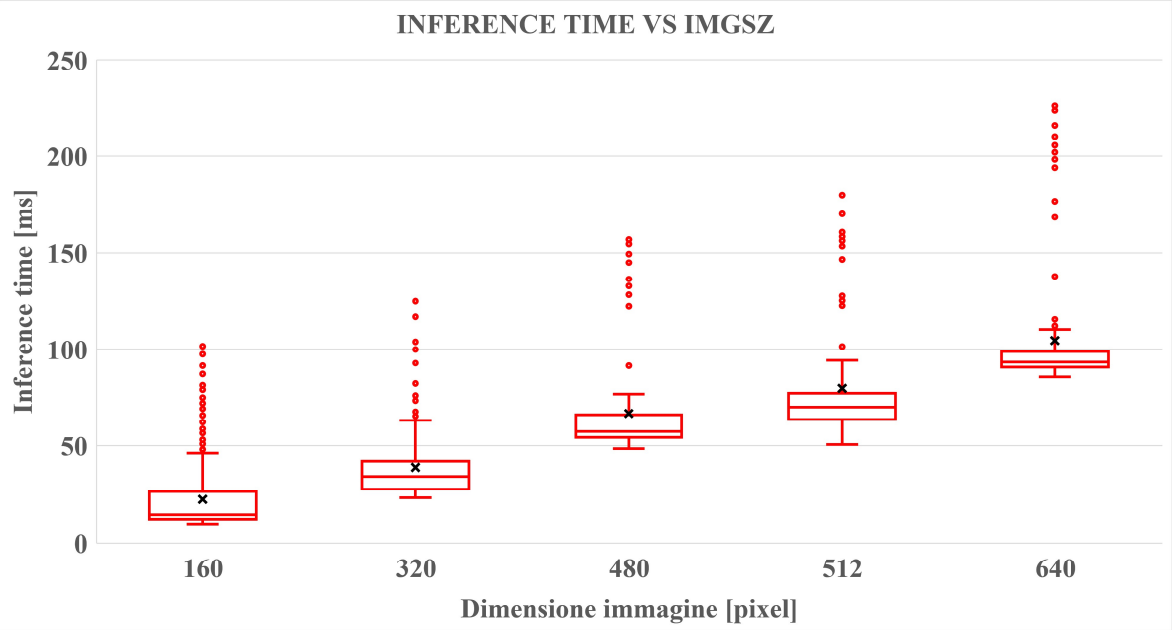


Figura 54 boxplot del tempo di inferenza al variare della risoluzione

La risoluzione ha un impatto notevole sul tempo di inferenza: come già anticipato il modello addestrato con dimensione di immagine 640x640 impiega molto più tempo per la stessa inferenza. In particolare, i tempi di inferenza si riducono di circa sei volte passando da circa 104 ms a imgsz 640 a circa 22 ms di media a imgsz 160. Questi risultati orientano la scelta verso la risoluzione più bassa come base su cui operare ed eseguire le successive inferenze, ovvero si vuole indagare sulla configurazione 160x160; gli altri modelli restano comunque disponibili per ottenere un confronto.

Si nota, anche, la presenza di molti outlier ad ogni risoluzione: per le dimensioni 480x480, 512x512 e 640x640 sono distribuiti verso valori molto più distanti dal massimo di rispettivo box, mentre per le altre risoluzioni sono molto vicini al massimo delle relative distribuzioni. I valori più estremi sono stati comunque riportati nel grafico ma sono valori non rappresentativi poiché sono valori ottenuti all'inizio dell'inferenza ovvero quando il sistema deve eseguire il codice di inizializzazione (apertura della finestra video, caricamento del modello, ecc.).

Quindi, la conclusione naturale portata dai dati in Figura 54 boxplot del tempo di inferenza al variare della risoluzione è l'adozione della risoluzione più bassa, 160x160 sia per il maggiore raggruppamento degli outlier attorno al massimo sia per il tempo di inferenza nettamente inferiore rispetto alle altre dimensioni.

La scelta di una risoluzione 160 comporta tuttavia un trade-off ovvero la presenza di una maggior numero di outlier rispetto ai modelli a risoluzione maggiore. In questo frangente, le inferenze sono state ottenute senza operare alcuna modifica a livello hardware il che spiega la presenza dei molteplici valori esterni al box. Per sfruttare a pieno le capacità del dispositivo, tre modifiche sostanziali sono state introdotte: il governor di gestione della frequenza è stato impostato su "performance" per bloccare il processore alla frequenza massima e ottenere una maggiore stabilità di inferenza; è stato eliminato il multithreading

per passare al single thread attraverso il comando OMP_NUM_THREADS=1; infine, il tempo di blocktime è stato ridotto con il comando OMP_WAIT_POLICY=PASSIVE.

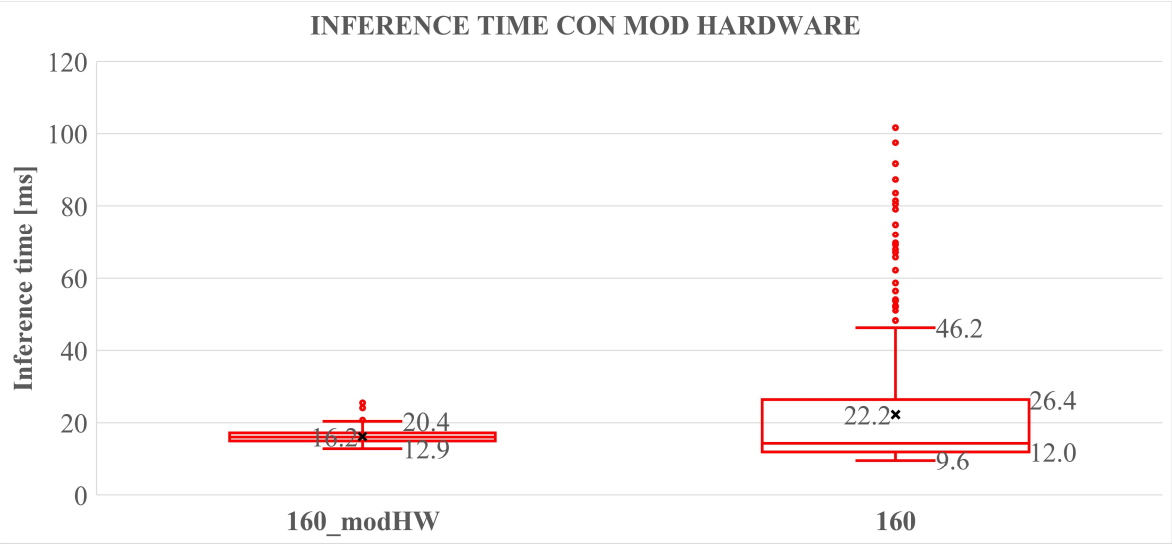


Figura 55 boxplot del tempo di inferenza con modifica hardware

L'inferenza che si ottiene con questa modifica risulta decisamente migliore: gli outlier sono quasi completamente scomparsi osservando solo gli outlier dovuti all'inizializzazione dell'inferenza. La media del tempo di inferenza si attesta a 16,2 ms per il modello con le modifiche hardware, contro i 22,2 ms del modello senza modifica: solo con questa modifica si ottiene un miglioramento dell'inference time del 27% in deploy, senza perdita di accuratezza di posizionamento del target.

Inoltre, i dati si dimostrano essere distribuiti maggiormente attorno alla media: la deviazione standard per il modello 160 si attesta a 16,3 ms mentre per il modello 160_modHW si attesta a 1,7 ms, valore decisamente preferibile in termini di stabilità e prevedibilità del sistema.

I dati con cui sono stati ricavati i grafici riportati precedentemente sono stati ricavati usando i modelli a precisione completa, ovvero FP32.

Si tiene a precisare che le modifiche apportate non sono state effettuate nei lavori di tesi e paper precedenti [20] [47].

4.3 Quantizzazione

A partire dal modello esportato in formato NCNN, sono state individuate tre configurazioni di precisione numerica, ciascuna caratterizzata da un differente compromesso tra accuratezza, occupazione di memoria e velocità di inferenza:

- FP32 (precisione singola): rappresenta la configurazione di riferimento, corrispondente al modello NCNN così come ottenuto dal processo di export, priva di qualsiasi ulteriore quantizzazione. Mantiene inalterata l'accuratezza del modello

originale ma a scapito di un maggiore impiego computazionale e di tempi di inferenza più elevati.

- FP16 (mezza precisione): ottenuta mediante dimezzamento della precisione numerica dei pesi tramite gli strumenti offerti dal framework NCNN (comando "ncnnoptimize"); consente approssimativamente di ridurre del 50% le dimensioni del modello con una perdita di accuratezza generalmente trascurabile, di solito minore dell'1% [36], rappresentando un compromesso intermedio adatto a piattaforme con supporto hardware per operazioni in virgola mobile a precisione ridotta.
- INT8 (quantizzazione intera): la configurazione più aggressiva, che converte i pesi in numeri interi a 8 bit attraverso un processo di calibrazione basato su un sottoinsieme rappresentativo di immagini; vengono adottati i comandi "ncnn2table" e "ncnn2int8". Questa tecnica consente la riduzione dimensionale più marcata, quantificabile in circa il 75% rispetto al modello FP32, ma introduce un errore di quantizzazione che richiede una validazione puntuale. È stata validata la fattibilità di deploy di questo modello quantizzato così come è stato creato e successivamente è stata effettuata una calibrazione tramite confronto frame-by-frame tra le bounding box prodotte dal modello FP32 e quelle prodotte dal modello INT8.

La disponibilità di tutte e tre le configurazioni, tutte derivate dal medesimo export NCNN, all'interno della pipeline di inferenza sviluppata per il CM5 ha permesso un confronto sistematico delle prestazioni, condotto utilizzando i vari modelli a risoluzione di input (imgsz) diversa, al fine di identificare la combinazione ottimale in termini di accuratezza e latenza per l'applicazione di controllo in tempo reale. Particolare attenzione è stata posta sul modello quantizzato INT8 poiché la fattibilità di entrambi i modelli con configurazione in virgola mobile (FP32 e FP16) è stata già verificata essere utile nel corso del precedente lavoro di tesi, che ha impiegato con successo un modello con pesi salvati in virgola mobile (FP) [20].

Viene valutato il comportamento delle configurazioni a precisione in virgola mobile (FP32 e FP16), il funzionamento della configurazione a quantizzazione INT8, la più critica tra quelle esplorate a causa dell'errore di quantizzazione introdotto dal processo di calibrazione, e alcune modifiche a livello hardware per migliorare le prestazioni direttamente del CM5.

L'architettura del modello base in uscita dal training è Pytorch. È già noto che tale formato non è adatto al deploy su un dispositivo edge come il Raspberry Pi. Ne consegue un successivo export verso l'architettura NCNN molto più efficiente e pensata per i dispositivi usati in questo lavoro. L'export verso NCNN con precisione a 32-bit in virgola mobile (FP32) è stato eseguito tramite comando "model.export" sfruttando direttamente le funzionalità del framework Ultralytics.

Nativamente, questa architettura utilizza pesi gestiti in 32 bit in virgola mobile che su richiesta, tramite richiamo della flag "half" impostata su True in sede di esportazione, possono essere quantizzati direttamente a 16 bit in virgola mobile. In alternativa, la stessa conversione può essere effettuata a posteriori sul modello NCNN già esportato, tramite il comando "ncnnoptimize" con l'opportuno flag di ottimizzazione FP16.

In questa fase si vuole indagare una possibile quantizzazione più severa, ovvero riducendo i pesi della rete a 8 bit in numeri interi. Una quantizzazione di questo genere può portare ad

un crollo notevole delle metriche del modello, fino al punto in cui il target non risulta più correttamente riconosciuto.

In una prima verifica qualitativa, osservando le bounding box presentate a schermo durante l'inferenza, il riconoscimento del target spesso è risultato spesso errato: le dimensioni delle bounding box non rispecchiano fedelmente quelle reali del target, risultando spesso più piccole di quest'ultimo con un conseguente posizionamento del centro delle bounding box anch'esso sbagliato. Inoltre, si osserva un calo notevole della confidenza attestandosi attorno a 0,5 quasi costantemente rispetto al picco di 0.9 ottenuto quasi sempre per i modelli privi di quantizzazione.

Si riportano di seguito due frame utili a mostrare la differenza tra una bounding box trovata dalla rete a FP32 e una bounding box trovata dal modello a quantizzazione INT8.

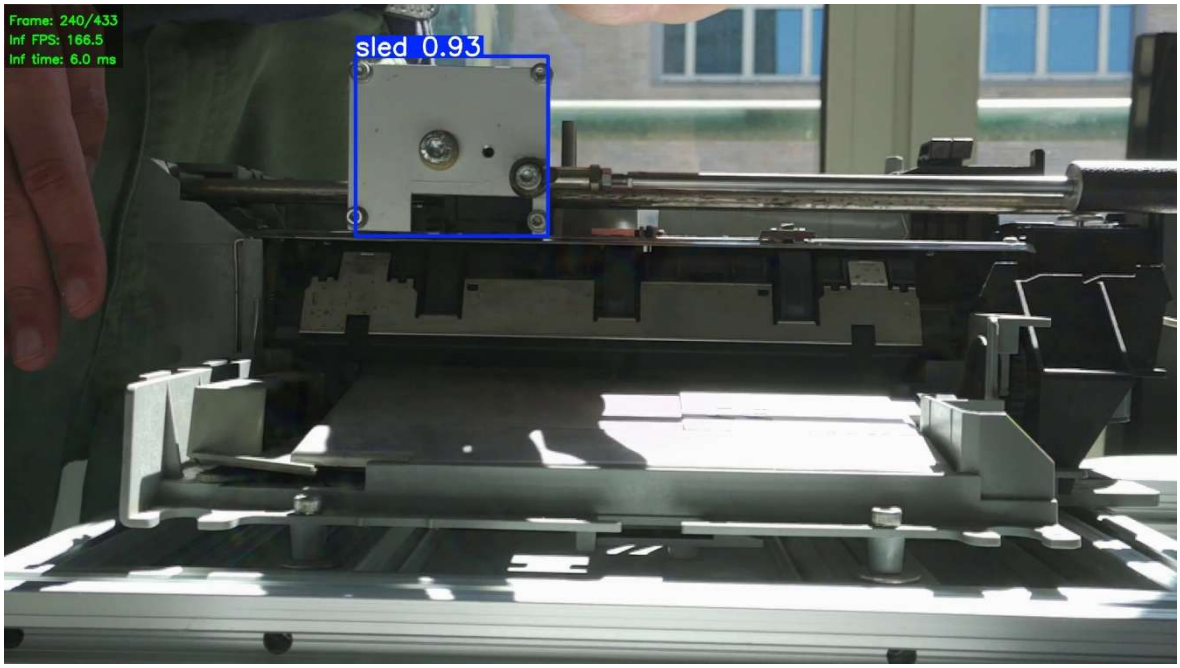


Figura 56 bounding box con modello a 32-bit floating point

Nell'immagine sopra riportata si vede chiaramente una bounding box che racchiude la slitta completamente, seppure non perfettamente.

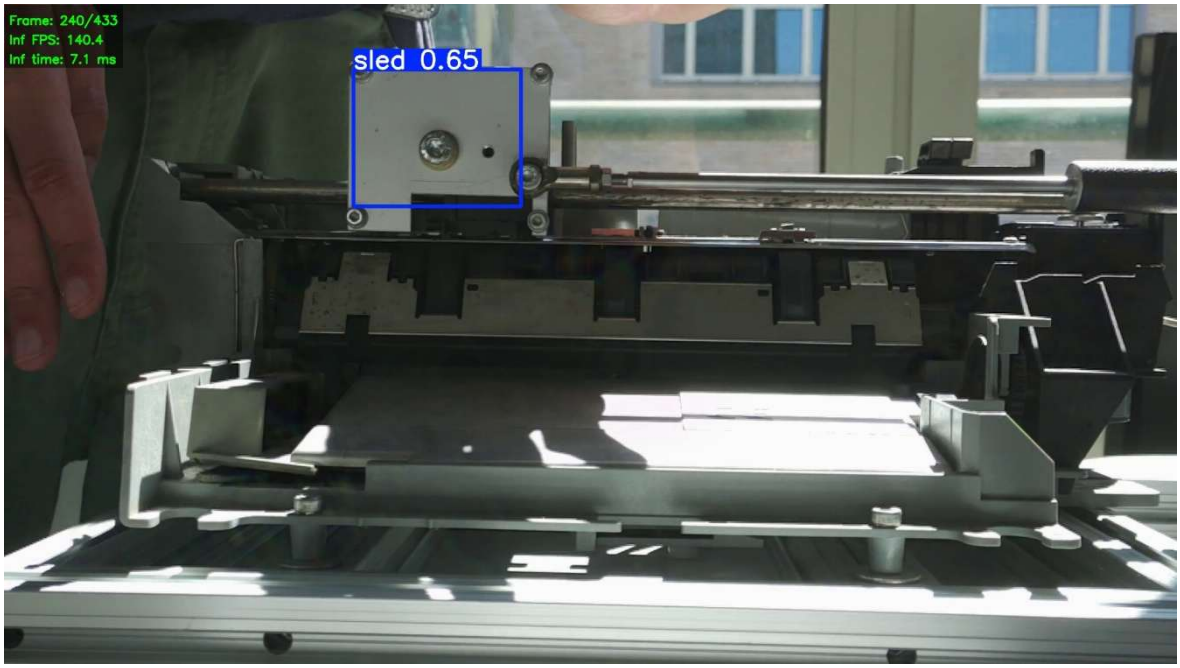


Figura 57 bounding box con modello a 8-bit integer

In Figura 57 bounding box con modello a 8-bit integer si nota chiaramente un degrado pesante del posizionamento delle bounding box con relativo abbassamento dello score di confidenza. Per le motivazioni riportate, quindi, il modello con quantizzazione a 8 bit pura non può essere usato in questo contesto.

L'instabilità delle bounding box predette, riconducibile alla propagazione dell'errore di quantizzazione lungo i percorsi più sensibili della testa di detection ha comportato lo sviluppo di un modello a quantizzazione mista. L'identificazione dei layer da escludere è stata condotta attraverso un'analisi diretta del grafo computazionale esportato in formato NCNN, ispezionando il file ".param" per ricostruire la topologia dei collegamenti tra input e output di ciascun layer convoluzionale [48]. La quantizzazione mista è operata lasciando solo alcuni layer, quelli incolpati dello scarso posizionamento della bounding box, a quantizzazione FP32 e FP16 mentre i restanti sono tutti quantizzati a INT8.

La rete yolo presenta tre sottoreti distinte: il backbone, il neck e la head. Il grafo presente nel file ".param" ottenuto alla fine dell'esportazione in NCNN, è strutturato su un elevato numero di layer distribuiti su 200 righe circa.

Nel file si possono distinguere le tre sottocategorie e le varie operazioni compiute, poiché per ciascun layer vengono riportati rispettivamente il tipo, il nome univoco e il conteggio di input/output di ciascuna operazione (ad esempio operazioni di convoluzione, addizione, sottrazione, concatenamento, ecc.) e successivamente vengono riportati i parametri specifici di quel layer indicati come mostrato di seguito.

$[layer\ type]\ [layer\ name]\ [input\ count]\ [output\ count]\ [input\ blobs]\ [output\ blobs]\ [layer\ specific\ parameters]$

Sono state identificate 63 convoluzioni distribuite nelle tre grandi sottoreti.

Una volta identificata la head della rete, l'analisi è stata volta a:

- individuare i layer responsabili alla classificazione e posizionamento della bounding box per escluderli dalla quantizzazione
- scegliere la precisione da usare per i layer individuati (FP16 o FP32)

Tale analisi si è conclusa inizialmente verso due sottoreti distinte e strutturalmente indipendenti all'interno della testa di detection di YOLO v8n: il ramo di regressione delle coordinate (box regression, identificato dai layer conv_47, conv_50, conv_53) e il ramo di classificazione (identificato dai layer conv_56, conv_59, conv_62).

A questi si aggiunge il layer conv_63, che implementa l'operazione di integrazione della Distribution Focal Loss (DFL).

La scelta di escludere proprio questi sette layer dalla quantizzazione è motivata da due considerazioni concorrenti, entrambe emerse dall'ispezione dei parametri del file ".param". In primo luogo, si tratta di layer topologicamente collocati immediatamente a monte di operazioni numericamente sensibili, la funzione Softmax nel caso della DFL e la funzione Sigmoid nel caso del ramo di classificazione, per le quali un errore di quantizzazione sull'input, anche di piccola entità, può tradursi in una distorsione significativa della distribuzione di probabilità in uscita, fino al collasso della predizione. In secondo luogo, a differenza dei layer convoluzionali del backbone e del collo (neck) della rete, che presentano tutti un'attivazione non lineare LeakyReLU fusa (parametro activation_type=2 nel file .param che corrisponde all'attivazione della quantizzazione int8), i sette layer in questione non presentano alcuna funzione di attivazione fusa: si tratta di trasformazioni lineari pure, il cui range dinamico di uscita non è vincolato da una soglia di saturazione, rendendoli intrinsecamente più esposti a errori di scala in fase di calibrazione INT8.

La correttezza di questa selezione è stata verificata inizialmente empiricamente attraverso un confronto tra due configurazioni di quantizzazione mista: una prima configurazione, "mixed-4", che escludeva unicamente i layer del ramo di regressione e la DFL (conv_47, conv_50, conv_53, conv_63), e una seconda configurazione," mixed-7", estesa a includere anche il ramo di classificazione (conv_56, conv_59, conv_62). La configurazione mixed-4 si è rivelata insufficiente, producendo un'assenza pressoché totale di bounding box in output, sintomo compatibile con un collasso della DFL e/o con una saturazione del punteggio di confidenza al di sotto della soglia di rilevamento attestatasi attorno a 0,2, comunque punteggio insufficiente, riconducibile alla quantizzazione residua del ramo di classificazione non ancora escluso in questa configurazione.

Solo con la configurazione mixed-7, che esclude congiuntamente entrambi i rami della testa di detection, il modello ha ripristinato un comportamento di rilevamento stabile, confermando che entrambe le sottoreti, regressione e classificazione, necessitano di essere mantenute in precisione piena, FP32 o FP16, per garantire l'affidabilità delle predizioni, mentre il backbone e il collo della rete, privi di questa sensibilità, possono essere quantizzati integralmente a INT8 senza una perdita apprezzabile di accuratezza.

Successivamente, l'analisi è stata divisa in due fasi in base alla piattaforma su cui sono state eseguite le inferenze. Nella prima fase, la classificazione e il posizionamento dei modelli a

quantizzazione mista (mixed_7 FP32 e mixed_7 FP16) sono stati confrontati con quelli del modello a precisione intermedia (FP16) e del modello a risoluzione 640x640, eseguendo l'inferenza su PC Windows a partire da un video acquisito in precedenza. Nella seconda fase, è stato invece eseguito il confronto tra le diverse quantizzazioni utilizzando inferenze real time, con streaming video acquisito direttamente da camera collegata al Raspberry Pi.

4.3.1 Analisi da inferenza su PC Windows

Nella prima fase, l'acquisizione è stata eseguita per i modelli a quantizzazione FP16, INT8 misto con i layer esclusi a FP32 e INT8 misto con i layer esclusi a FP16. Inoltre, in questo ambito, è stata esplorata anche l'influenza del passaggio a schermo dello streaming video, comprensivo dell'inferenza con relativa bounding box, acquisendo i dati di inference time anche in assenza della preview. La presenza della preview durante l'inferenza è stata sempre utilizzata durante tutte le acquisizioni precedenti e continuerà ad essere impiegata in seguito; in questo caso, però, l'unico obiettivo era solo quantificare quanti millisecondi venissero aggiunti a tempo di inferenza dalla renderizzazione a schermo dello streaming video.

Per eseguire il confronto, sono stati ricavati i dati di posizionamento spaziale delle bounding box: per ciascun frame, sono stati salvati i quattro punti del rettangolo della bounding box e il corrispettivo centro. Sono state quindi calcolate le aree di ogni bounding box per ogni frame del video e modello, ed è stato calcolato il IoU rispetto all'area della bounding box ottenuta dal modello a 640 così da avere un'indicazione chiara e quantitativa sull'affidabilità delle bounding box prodotte da ciascuna orientando di conseguenza la scelta del modello più adatto da impiegare.

Si riportano i grafici a scatole e baffi di IoU per i vari modelli.

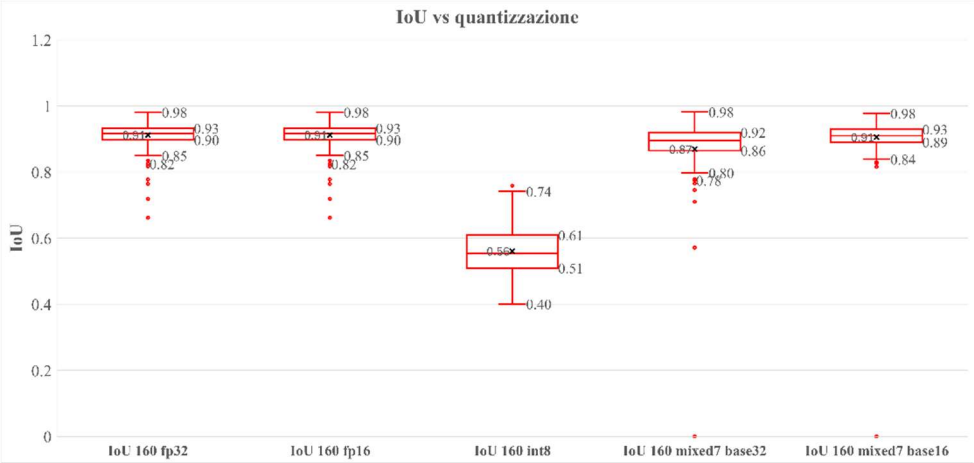


Figura 58 boxplot IoU vs quantizzazione rispetto a risoluzione massima 640x640

La distribuzione dei valori di IoU, riportata nel grafico a scatole, mostra una situazione chiara. Il modello a quantizzazione INT8 pura ha un comportamento peggiore: il valore di media di IoU rispetto alla bounding box di riferimento si posiziona di poco sotto 0.6, indicando che le bounding box individuate non rappresentano correttamente né la posizione né le dimensioni reali del target.

Entrambi i modelli a quantizzazione floating point (FP32 e FP16) riportano la medesima distribuzione con valori di media di IoU pari a 0.912 sebbene siano presenti numerosi outlier. Il modello con quantizzazione mista con layer esclusi a FP32 ha una distribuzione leggermente peggiore rispetto alla propria controparte a quantizzazione FP16: il valor medio di IoU si attesta a 0.869, con una distribuzione comunque eterogenea e non concentrata attorno alla media.

Il modello con quantizzazione mista con layer esclusi a FP16 mostra un comportamento migliore per due aspetti: i valori risultano maggiormente concentrati attorno la media pari a 0.906, comunque superiore rispetto al modello a quantizzazione mista a FP32, e la distribuzione presenta un numero limitato di outlier.

La IoU da sola non è sufficiente a caratterizzare l'idoneità di un modello al deploy, poiché quest'ultimo deve raggiungere una soglia di confidence adeguata a poter essere impiegato in un contesto di detection real time di un oggetto.

A questo scopo, si riporta uno un grafico a punti che mette in relazione la media dei dati di IoU con la media della confidence, per ciascuna configurazione.

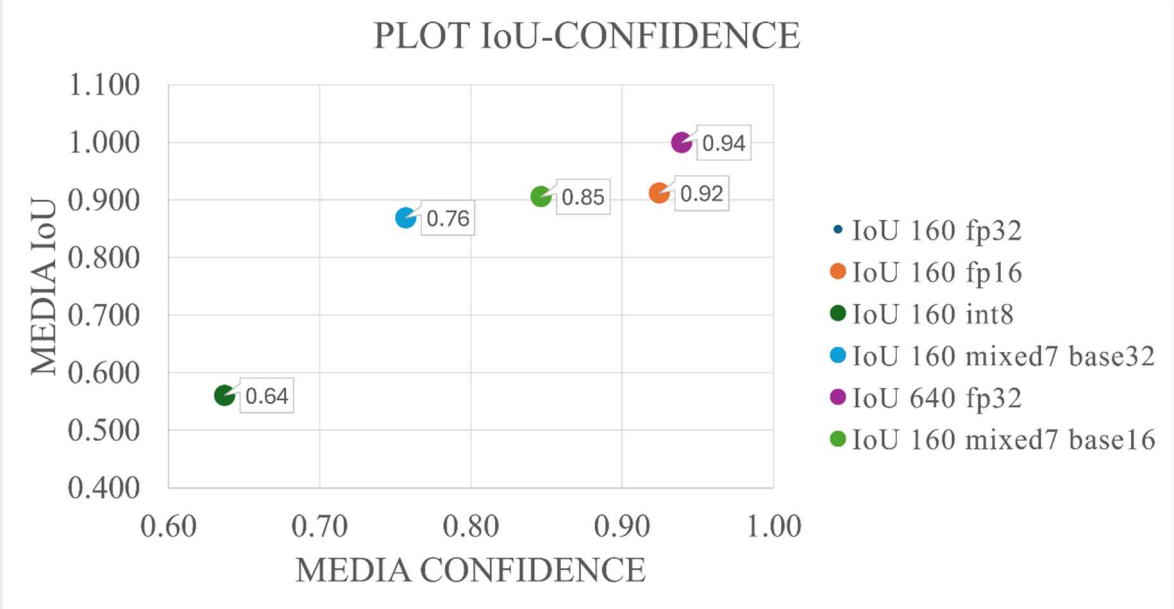


Figura 59 plot IoU-confidence

Il modello INT8 non solo si conferma incapace di posizionare le bounding box, ma anche caratterizzato da valori di confidence molto bassi: risulta pertanto da escludere a prescindere dal deploy.

Le configurazioni FP32 e FP16 presentano lo stesso comportamento, coerentemente con quanto osservato nel box plot precedente, con i due punti praticamente sovrapposti: ottengono ottimi valori di confidence entrambe, quasi al pari del modello di riferimento a risoluzione massima.

La quantizzazione mista porta una peggioramento della confidenza media con cui le bounding box vengono individuate. In particolare, il modello con i layer mantenuti a FP16

risulta migliore rispetto alla propria controparte con layer mantenuti a FP32, con una confidence media rispettivamente di 0.85 e 0.76.

Si vuole inoltre analizzare le prestazioni in termini di tempo di inferenza: dalle stesse run di inferenza da cui sono ottenuti i dati per ricavare i grafici precedentemente riportati, sono stati salvati anche il tempo di pre-process,il tempo di inferenza e il tempo di post-process. Tali valori sono stati mediati e riportati nel grafico seguente rapportati alla media di IoU.

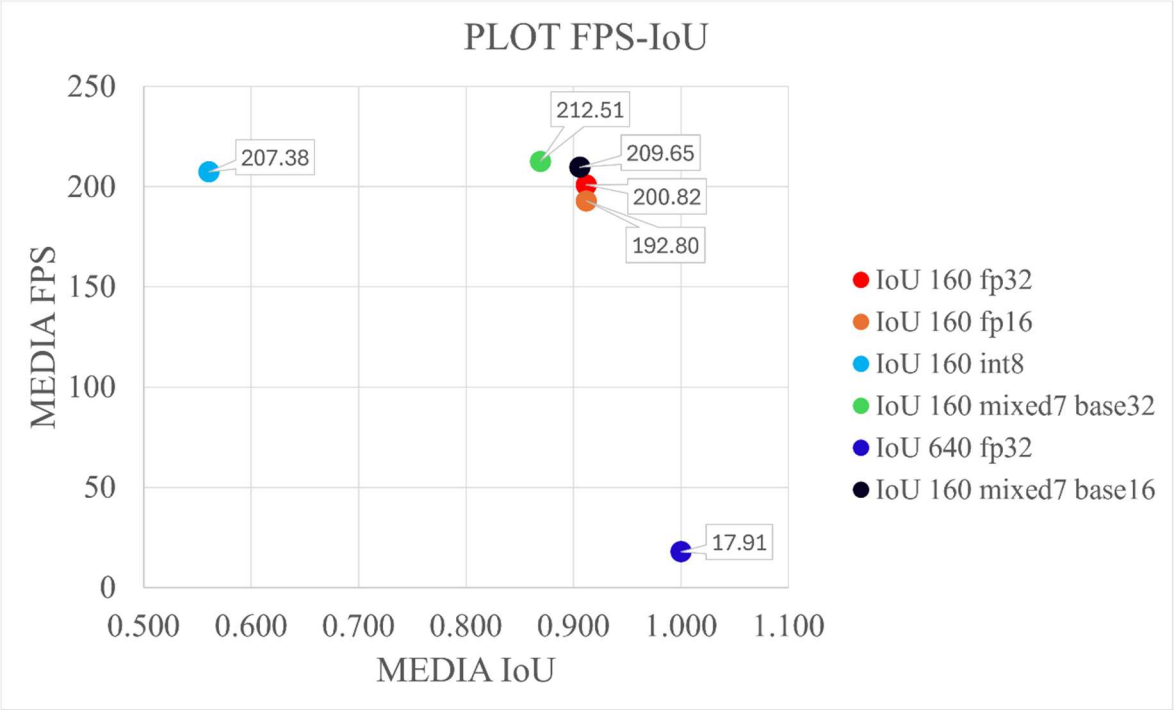


Figura 60 plot FPS vs IoU

La quantizzazione mista comporta mediamente un incremento del numero dei frame al secondo: si passa da 193 fps circa usando la precisione FP16 a 213 FPS circa con la quantizzazione mista a layer residui FP32, e a 210 FPS circa con la quantizzazione mista a layer residui FP16.

Il modello a quantizzazione INT8 pura risulta comparabile rispetto le altre configurazioni; tuttavia, in questo caso, dato che le medie di IoU e confidence sono pessime il modello viene comunque scartato.

Si riporta anche il valor medio del modello a risoluzione massima (imgsz 640) per evidenziare una considerazione: è presente una differenza sostanziale in termini di FPS con un incremento medio di circa 10x adottando la risoluzione più bassa e applicando la quantizzazione al modello.

Quindi, tenendo conto dei dati relativi a media IoU, media confidence e media inference time o FPS, il modello che risulta più promettente è il modello con quantizzazione mista a base INT8 con layer esclusi a quantizzazione FP16.

Nella fase successiva, l'attenzione si concentra quindi su questa configurazione.

I dati ricavati in questa sezione sono interamente ricavati da inferenze su GPU su PC Windows.

4.3.2Analisi da inferenza su Raspberry Pi

La seconda fase è stata costruita per determinare se sul Raspberry Pi, si riscontri lo stesso comportamento visto su PC Windows, indagando in particolare tre aspetti:

- l'individuazione dei modelli più validi per il deploy;
- l'influenza delle due risoluzioni di acquisizione,1280x720 e 640x360;
- l'influenza della distanza tra la camera e il target.

I modelli caricati sono i seguenti:

- precisione piena (FP32);
- quantizzazione FP16;
- quantizzazione mista con base INT8 e layer esclusi in FP32;
- quantizzazione mista con base INT8 e layer esclusi in FP16.

In questa fase, le inferenze sono state eseguite tramite streaming video real time, acquisito mediante la camera facente parte del banco prova in esame.

La slitta è stata posta in movimento attraverso un comando analogico, impostato per mezzo di potenziometri rotativi posizionati sul rack a disposizione. Il comando analogico ha le seguenti caratteristiche:

- forma d'onda sinusoidale;
- ampiezza pari a70 mm;
- frequenza del segnale pari a 1 Hz;
- guadagno nel controllo pari a 5.

Il comando analogico è stato scelto in modo tale da avere un movimento il più lineare possibile, tale da minimizzare l'influenza dell'attrito, un guadagno tale da non causare instabilità nel movimento della slitta, e un allineamento il più fedele possibile tra il segnale di comando (set) e segnale di retroazione (feedback), privo quindi di attenuazioni o perdite di segnale apprezzabili.

In queste inferenze, il Raspberry Pi ha operato a frequenza di lavoro costante, impostata al valore massimo consentito senza alcun valore di overclock impostato.

I dati salvati durante queste inferenze sono: la posizione dei quattro punti del rettangolo della bounding box e il suo relativo centro, la confidenza della detection e i valori del tempo di pre-process, inferenza e post-process.

Le inferenze sono state eseguite posizionando la camera in due posizioni distinte, adottando il metro come valore di distanza discriminante, con l'obiettivo di verificare se la rete fosse in grado di rilevare correttamente il target anche oltre tale soglia: una configurazione

“lontano”, a circa 1,3 m, e una configurazione “vicino”, a circa 0,8 m dal target. La configurazione con la camera ravvicinata è identificata, nel nome della serie corrispondente di dati, con la dicitura “vicino”.

Si presentano di seguito i dati ricavati dalle inferenze condotte con i vari modelli analizzati, valutati sotto il profilo del detection rate.

	Frames	Detected	Detection Rate DR (%)
1280_720_fp16_vicino	181	181	100.00
1280_720_fp_16_lontano	126	117	92.86
1280_720_mixed_fp16_lontano	141	78	55.32
1280_720_mixed_fp16_vicino	180	180	100.00
1280_720_mixed_fp32_vicino	237	237	100.00
640_360_fp32_vicino	187	187	100.00
640_360_mixed_fp16_vicino	154	124	80.52

Tabella 10 influenza della distanza sulla detection

Il detection rate si modifica di conseguenza con la distanza: quando il target si trova lontano dalla camera, il tasso di detection si riduce. Questo comportamento si nota per il modello FP16 o maggiormente per il mixed_FP16, per il quale una bounding box viene individuata in circa il 55% dei frame contro alla configurazione il 100% ottenuto nella configurazione ravvicinata.

Il valore ridotto di detection rate nei casi con target lontano è in parte riconducibile a un valore di soglia della confidence troppo restrittivo, tenendo conto di due aspetti importanti: da un lato, la rete, essendo già di dimensioni ridotte, fatica a mantenere sia una detection presente, quindi una bounding box a schermo, che un livello di confidence alto contemporaneamente; dall’altro, la risoluzione di immagine utilizzata è la più bassa possibile, per cui la rete deve individuare il target dalla distanza attraverso una finestra estremamente ridotta, con un livello di informazione, a livello di densità di pixel, basso. Infatti, per questo motivo, al fine di ottenere una detection in queste condizioni, è stato necessario ridurre il valore di confidence richiamata nel codice di inferenza inizialmente impostata a 0,8 e successivamente abbassata a 0,4.

Osservando poi i dati del modello mixed_FP16 in funzione della risoluzione della schermata video, si nota un decremento del detection rate quando si passa dalla risoluzione 1280x720 (DR pari a 100%) alla risoluzione 640x360 (DR pari a 81% circa). Risulta pertanto preferibile, a parità di altre condizioni, adottare la risoluzione video più elevata.

Volendo, infine, presentare i dati in funzione dell’inference time, si ottengono i risultati riportati nel seguente grafico a scatole e baffi.

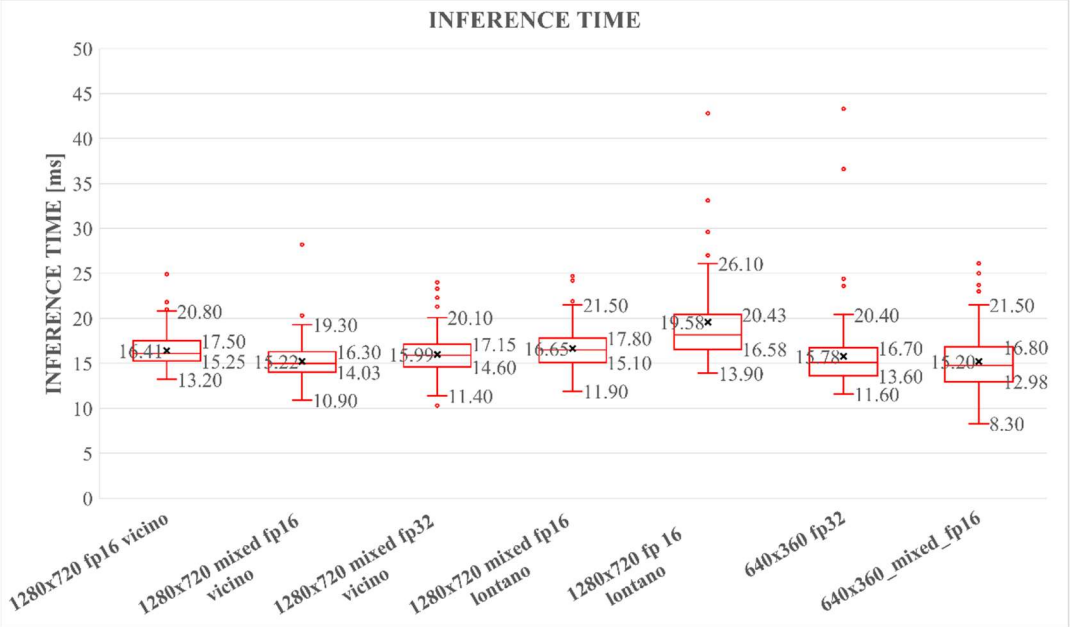


Figura 61 influenza della distanza dal target e risoluzione finestra video

L'analisi dei tempi di inferenza tra le sette configurazioni riportate in Figura 61, evidenzia un divario sistematico tra le prove condotte con la camera posizionata più vicina al target e quelle condotte a distanza maggiore, a parità di ogni altra condizione sperimentale (stessa sessione di acquisizione, medesime modifiche hardware).

Per la configurazione FP16 a risoluzione 1280×720, il tempo medio di inferenza totale passa da 16,41 ms nella condizione "vicino" a 19,58 ms nella condizione a distanza maggiore, con un incremento di circa il 19%.

Lo stesso andamento si osserva per la configurazione a quantizzazione mista con layer esclusi a FP16, sempre a risoluzione 1280x720: il tempo medio di inferenza passa da 15,22 ms nella condizione “vicino” a 16,65 ms nella condizione a distanza maggiore, con un incremento pari a circa il 9%.

1280x720 fp16 lontano			
	PRE-PROCESS	INFERENCE	POSTPROCESS
MEDIA [ms]	1.4	16.2	2.0
DEVIAZIONE ST [ms]	0.826	6.087	0.974
1280x720 fp16 vicino			
	PRE-PROCESS	INFERENCE	POSTPROCESS
MEDIA [ms]	1.2	13.4	1.8
DEVIAZIONE ST [ms]	0.43	1.68	0.51

Tabella 11 influenza della distanza sulle tre fasi della detection modello fp16

1280x720 mixed fp16 lontano			
	PRE-PROCESS	INFERENCE	POSTPROCESS
MEDIA [ms]	1.4	13.6	1.6
DEVIAZIONE ST [ms]	0.5	2.3	0.6
1280x720 mixed fp16 vicino			
	PRE-PROCESS	INFERENCE	POSTPROCESS
MEDIA [ms]	1.1	12.4	1.7
DEVIAZIONE ST [ms]	0.3	1.7	0.6

Tabella 12 influenza della distanza sulle tre fasi della detection modello mixed fp16

La scomposizione del tempo nelle tre fasi costitutive della pipeline, pre-process, inferenza (forward pass) e post-process, mostra che il divario tra le condizioni “vicino” e “lontano” è quasi interamente attribuibile alla fase di inferenza pura (13,4 ± 1,68 ms contro 16,2 ± 6,087 ms per il modello fp16), mentre le fasi di pre- e post-process contribuiscono in maniera marginale alla differenza complessiva (rispettivamente 0,2 ms e 0,2 ms su un totale di 3,2 ms).

Questo risultato è di per sé degno di nota, poiché il costo computazionale di una rete convoluzionale a dimensione di input fissa (il numero di operazioni eseguite dalla rete) è funzione esclusivamente della forma del tensore in ingresso e dei parametri della rete, non del contenuto semantico dei pixel elaborati [49]. A parità di risoluzione nominale del frame, il costo computazionale non dovrebbe pertanto, in linea teorica, essere alterato dalla distanza della camera dal target.

Un ulteriore elemento a supporto di questa osservazione è rappresentato dalla marcata asimmetria nella deviazione standard della fase di inferenza tra le due condizioni, quasi quattro volte superiore nella condizione a distanza maggiore (6,087 ms) rispetto alla condizione ravvicinata (1,68 ms). Una variabilità così pronunciata in un carico computazionale che dovrebbe essere deterministico suggerisce l'intervento di un fattore di sistema esterno al calcolo puro, piuttosto che un effetto legato al contenuto della scena. Il candidato più plausibile è una variazione dinamica della frequenza di clock della CPU dovuta a throttling termico gestito a livello di firmware dal SoC del CM5: tale meccanismo agisce in modo indipendente dal governor di frequenza impostato a livello di sistema operativo, che progressivamente riduce le frequenze operative quando la temperatura del core si colloca tra 80° C e 85° C, indipendentemente dalla politica di gestione della frequenza selezionata dall’utente, e può quindi manifestarsi anche con il governor fissato su performance [50].

La stessa analisi di scomposizione, condotta sulla configurazione a quantizzazione mista, conferma il pattern generale ma ne attenua sensibilmente l'entità: il tempo di inferenza pura passa da 12,4 ± 1,7 ms nella condizione "vicino" a 13,6 ± 2,3 ms nella condizione a distanza maggiore, con un incremento assoluto (1,2 ms) e un rapporto tra deviazioni standard (2,3/1,7 di circa 1,35) nettamente inferiori a quelli osservati per la configurazione FP16 pura (rapporto di circa 3,6). Il fatto che la configurazione a quantizzazione mista, computazionalmente meno intensiva per la presenza di diversi layer in precisione INT8, mostri una variabilità sensibilmente più contenuta e più stabile tra le due condizioni di distanza suggerisce che l'entità del fenomeno possa dipendere anche dal carico termico/energetico indotto dalla configurazione stessa, e non esclusivamente da un fattore di sistema fisso. Non essendo stato effettuato, durante la sessione di acquisizione, un logging della temperatura e dello stato di throttling correlato al singolo frame, non è stato possibile verificare direttamente questa ipotesi nell'ambito del presente lavoro; si raccomanda, per sviluppi futuri, l'integrazione del monitoraggio della temperatura, al fine di isolare con maggiore precisione la causa della variabilità osservata e di verificare se essa sia effettivamente modulata dal carico computazionale della configurazione in esecuzione.

Per quanto riguarda il confronto tra strategie di quantizzazione, la configurazione a quantizzazione mista mostra un vantaggio di latenza consistente rispetto alla configurazione FP16 pura in entrambe le condizioni di distanza: il guadagno è pari al 7% nella condizione ravvicinata (da 16,41 ms a 15,22 ms) e sale al 15% nella condizione a distanza maggiore (da 19,58 ms a 16,65 ms), un risultato che si aggiunge ai benefici già caratterizzati in termini di robustezza delle bounding box per questa configurazione. Le configurazioni a risoluzione di cattura 640×360 mostrano infine tempi medi (15,78 ms per la variante fp32 e 15,20 ms per la variante mista fp16) comparabili a quelli osservati nelle condizioni a 1280×720 ravvicinate, un aspetto che merita un'analisi separata rispetto al fenomeno distanza-dipendente qui discusso, per verificare se anche in questo caso il fattore determinante sia riconducibile a variabilità di sistema piuttosto che alla risoluzione di cattura in sé.

Concludendo, quanto appena riportato ci fornisce una situazione chiara:

- la camera deve essere posizionata quanto più vicino possibile al target pena l’abbassamento della confidence necessaria all’individuazione del target o l’assenza della bounding box a schermo e l’aumento conseguente dell’inference time; tale necessità risulta peraltro già coerente con quanto emerso dallo studio geometrico del sistema condotto nel precedente lavoro di tesi, secondo cui la camera non deve essere posizionata a distanza elevata dal target, bensì a una distanza contenuta [20];
- il modello più competitivo a livello di tempo d’inferenza è il modello a quantizzazione mista con base int8 e layer esclusi a quantizzazione fp16.

Si riportano, per completezza, i grafici riguardanti la larghezza della bounding box rispetto alla posizione longitudinale del centro ed inoltre si vuole specificare che le acquisizioni sono state eseguite facendo coincidere il centro della schermata video con il centro della slitta usata come target.

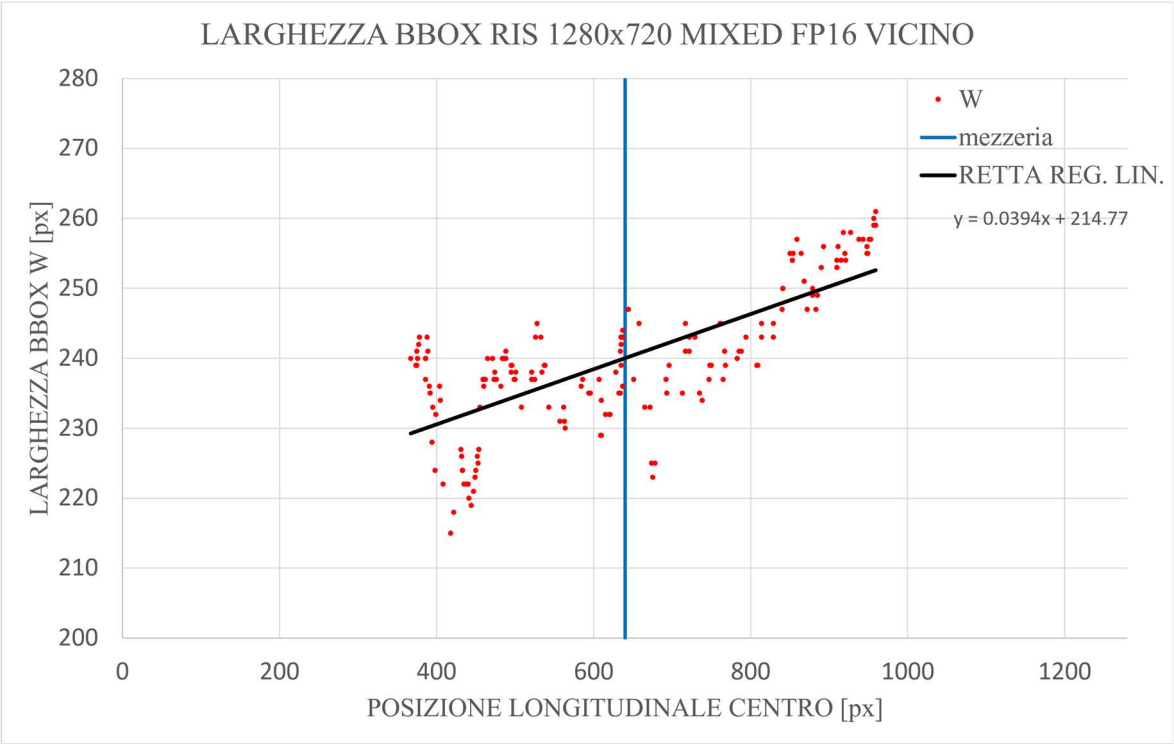


Figura 62 andamento della larghezza della bbox rispetto alla posizione longitudinale del centro

Il grafico mette in relazione la larghezza della bounding box (W) con la posizione longitudinale del centro slitta lungo la guida e rivela come tale dimensione non sia costante lungo l'intero range di posizione, ma presenti una tendenza di crescita procedendo verso destra: la slitta tende cioè ad essere rilevata con una larghezza maggiore quando si trova nella porzione destra della guida rispetto a quando si trova in posizione centrale o sinistra. Si osserva inoltre come la relazione tra larghezza del bounding box e posizione non sia strettamente monotona: nella porzione sinistra del grafico (posizione compresa indicativamente tra 400 e 460 px) è visibile un avvallamento locale in cui la retta di regressione lineare sovrastima sistematicamente i valori osservati. La causa più plausibile di tale fenomeno è la distorsione radiale introdotta dall'ottica della camera usata per le acquisizioni, un effetto tipico dei sistemi di visione con lenti grandangolari o comunque non perfettamente corrette, che risulta particolarmente marcato nella periferia del campo visivo e che, come osservato in letteratura anche nel contesto specifico dell'object detection, può compromettere l'affidabilità della rappresentazione della bounding box quando il target si allontana dal centro ottico [51].

È stato inoltre osservato, nel corso delle prove, come il banco di test tendesse a spostarsi progressivamente verso destra a causa delle vibrazioni indotte dal movimento della slitta stessa, un fenomeno che ha determinato una distribuzione asimmetrica dei campioni acquisiti, con una maggiore concentrazione di punti nella porzione destra della guida rispetto alla linea di mezzeria. Tale sbilanciamento nel campionamento non è da considerarsi una causa della tendenza osservata in W, quanto piuttosto un fattore che va tenuto in considerazione nell'interpretazione della robustezza statistica del fit nelle diverse porzioni della guida: la maggiore densità di campioni nella regione destra rende la stima della

relazione tra larghezza e posizione più affidabile in quella regione rispetto alla porzione sinistra, dove il numero più limitato di osservazioni potrebbe rendere la tendenza meno robusta.

Una prima soluzione, di natura strutturale, consisterebbe nell'irrigidire il banco alla base di appoggio (ad esempio tramite staffe supplementari, materiale antivibrante più efficace), agendo direttamente sulla causa fisica dello spostamento indotto dalle vibrazioni. Tale soluzione eliminerebbe il problema a monte, ma richiede un intervento sull'apparato meccanico non sempre disponibile nell'immediato, risultando più adatta a sviluppi futuri del banco.

Una seconda soluzione, di natura procedurale, consisterebbe nel suddividere la guida in fasce di posizione predefinite e verificare che ciascuna sia coperta da un numero comparabile di campioni, effettuando acquisizioni mirate nelle fasce sottorappresentate (come la porzione sinistra osservata nel presente lavoro). Questo approccio non richiede modifiche meccaniche, ma non elimina la causa fisica dello spostamento.

Per completezza si vuole riportare l'equivalente del grafico riportato in Figura 62 andamento della larghezza della bbox rispetto alla posizione longitudinale del centro, però riguardante la dimensione dell'altezza della bounding box sempre in funzione della lunghezza

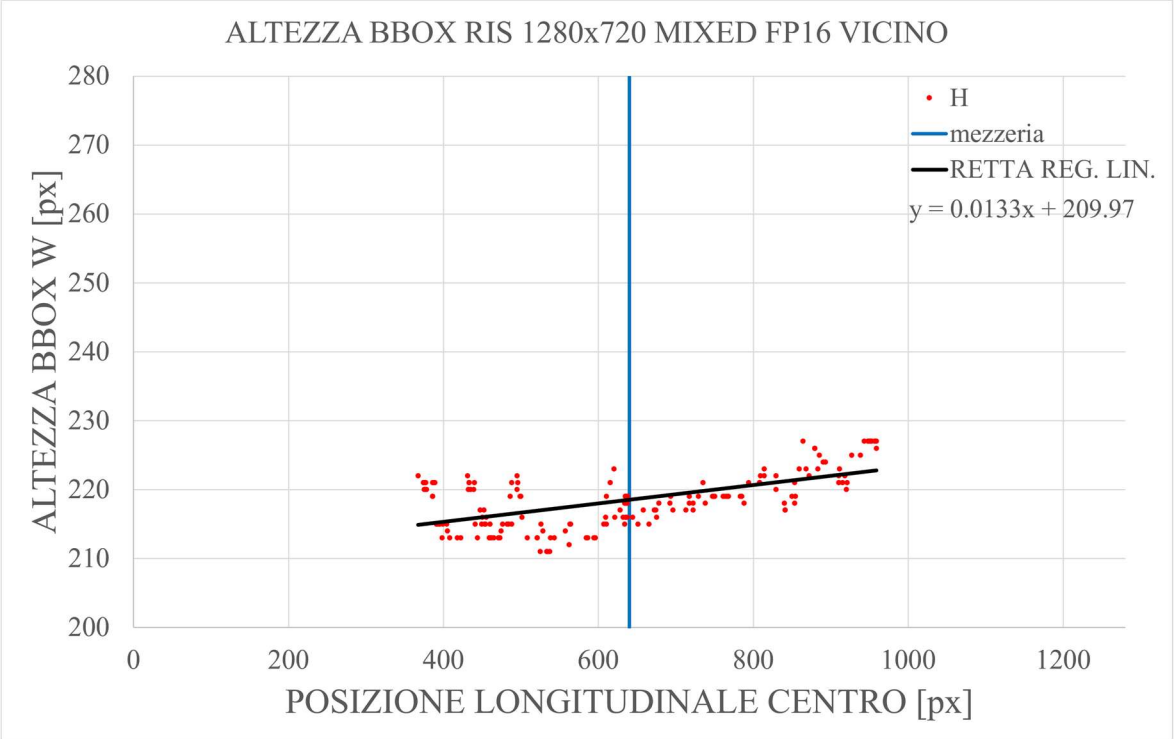


Figura 63 andamento dell'altezza della bbox rispetto alla posizione longitudinale del centro

longitudinale del centro slitta.

Il grafico mostra una tendenza prevalentemente lineare, con una variazione tra valore minimo e massimo dell'altezza del bounding box di circa 16 px. Tale contenuta escursione è coerente con la posizione verticale del centro della slitta (variazione di circa 9 px), che si mantiene sostanzialmente costante lungo l'intera prova: il coefficiente angolare della retta di

regressione, infatti, risulta prossimo allo zero e circa tre volte inferiore rispetto a quello osservato per la larghezza, come atteso dato che lo spostamento della slitta avviene prevalentemente in direzione orizzontale.

La lieve tendenza residua e la dispersione dei punti attorno alla retta sono verosimilmente riconducibili a fenomeni secondari quali il gioco meccanico tra guida e slitta o l'inerzia di quest'ultima nelle fasi di inversione del moto, che possono introdurre piccole oscillazioni verticali altrimenti assenti in un moto puramente orizzontale.

Non è inoltre da escludere un contributo di jitter nella stima del bounding box da parte del modello, un fenomeno già caratterizzato nell'analisi condotta in condizioni di slitta ferma, che si sovrapporrebbe alle variazioni fisiche reali sopra descritte rendendo difficile, sulla base del solo grafico in movimento, una netta separazione tra le due componenti di rumore.

5. STATISTICHE DATASET E RISULTATI FINALI

5.1 Statistiche dataset nuovo

Il dataset è composto da una singola classe, denominata come “sled” in fase di annotazione. Il dataset inizialmente contava circa 600 immagini; a seguito dell'eliminazione dei duplicati, la versione finale si è attestata a 554 immagini, ripartite in 443 per il training, 56 per la validation e 55 per il testing.

Analizzando la distribuzione per angolazione, 276 immagini ritraggono la slitta frontalmente, 229 ne catturano il profilo destro e sole 49 il profilo sinistro. Questa disparità è riconducibile al malfunzionamento del banco prova, causato dal blocco del motore deputato alla movimentazione della slitta: l'impossibilità di movimentare il target lungo la guida ha limitato la varietà di posizionamenti acquisibili per il profilo sinistro, le cui riprese sono state quindi prevalentemente impiegate per variare gli sfondi disponibili.

Classificando le immagini in base alle condizioni operative, il dataset include 26 immagini in cui il target risulta sfocato a causa dell'autofocus della camera, 12 acquisite in condizioni di illuminazione intensa, 6 in condizioni di illuminazione ridotta e 45 con occlusioni parziali del target, di cui una sola con occlusione totale.

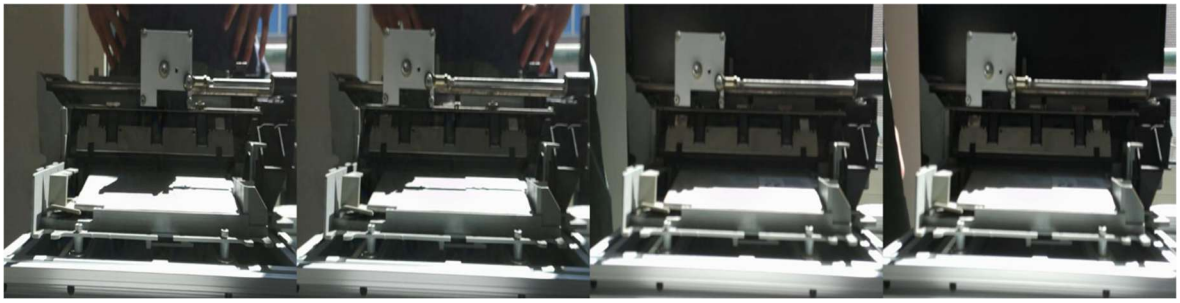


Figura 64 esempio immagini con sfocatura lieve dovuta all'autofocus

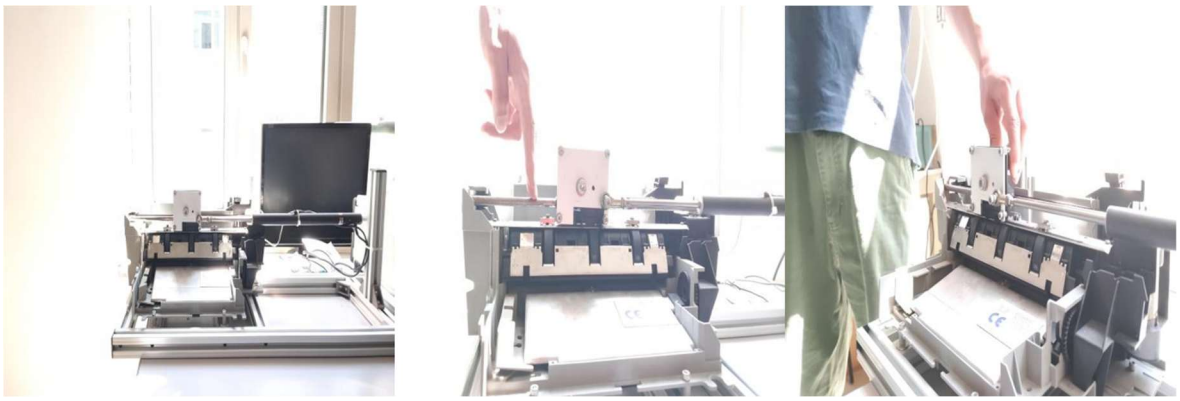


Figura 65 esempio immagini con illuminazione intensa



Figura 66 esempio immagini con illuminazione debole



Figura 67 esempio immagini con occlusione parziale

Le immagini precedentemente citate, acquisite in condizioni operative critiche, sono state interamente destinate al sottoinsieme di training, al fine di consentire al modello di

apprendere dalle situazioni più sfidanti, migliorandone la robustezza e facilitando il compito di detection in fase di rilascio sulla piattaforma Raspberry Pi.

Il dataset nella sua forma attuale presenta margini di miglioramento, in particolare riguardo al profilo sinistro, per il quale sarebbe opportuno eseguire ulteriori acquisizioni con la slitta in movimento così da colmare la disparità con gli altri profili. Un incremento del numero complessivo di immagini sarebbe inoltre auspicabile, considerando che per dataset a singola classe si raggiungono tipicamente circa mille o più immagini.

Tuttavia, due aspetti contestualizzano queste limitazioni. In primo luogo, l'architettura scelta, YOLO v8n, è una rete leggera che richiede un numero relativamente contenuto di immagini per essere addestrata efficacemente. In secondo luogo, il compito di detection in questo contesto è intrinsecamente semplice: il target è sempre il medesimo oggetto, privo di deformazioni e invariante in forma e colore. L'unica variabile significativa è la velocità di scorrimento della slitta lungo la guida, determinata dal comando iniziale impostato sul banco prova. Queste caratteristiche rendono il dataset attuale sufficientemente rappresentativo per gli scopi del presente lavoro.

Qualora le condizioni operative dovessero variare o il target subire modifiche, sarebbe sufficiente ripetere le fasi descritte in precedenza per aggiornare il dataset alle nuove condizioni operative.

5.2 Differenze rispetto al dataset originario

Il dataset creato nel precedente lavoro di tesi è stato creato seguendo un processo simile a quello descritto all'inizio del secondo capitolo.

Rispetto al nuovo dataset, una delle principali differenze riguarda la tipologia di forme d'onda di input: questo dataset è stato creato considerando la sola forma sinusoidale. Questo aspetto non ha inficiato sulla creazione del dataset poiché, durante il training, vengono fonite alla rete neurale singoli frame estratti dalle acquisizioni video il cui scopo è rappresentare le diverse posizioni della slitta lungo la propria guida. Risulta quindi maggiormente importante garantire un'adeguata quantità di "punti" o posizioni spaziali della slitta piuttosto che una specifica forma d'onda che sia sinusoidale o triangolare.

Il precedente dataset conta anche esso circa 500 immagini. Tale numerosità era principalmente dovuta alle limitazioni imposte dalla piattaforma Roboflow al momento della realizzazione del precedente lavoro di tesi. Mentre durante la realizzazione del nuovo dataset, la piattaforma ha offerto maggiore disponibilità in termini di quantità di immagini sviluppate gratuitamente, rendendo possibile l'impiego di più immagini per l'addestramento, aspetto rilevante soprattutto nel caso in esame con modelli di dimensioni ridotte e con un numero di parametri contenuto.

Un'ulteriore limitazione del precedente dataset, rispetto a quello realizzato nel presente lavoro, è la maggiore variabilità dell'angolo di inquadratura e dello sfondo inserito durante le acquisizioni. L'introduzione di differenti angolazioni di ripresa consente di presentare alla rete neurale il target da prospettive differenti, aumentando la variabilità dei dati disponibili durante l'addestramento. Ciò ha permesso al modello di apprendere caratteristiche

maggiormente robuste del target e ha consentito di migliorare la capacità di riconoscimento in presenza di variazioni di posizione della telecamera durante la fase di inferenza.

Il medesimo discorso può essere costruito introducendo variabilità dello sfondo. Durante il normale funzionamento del sistema, infatti, il banco di prova potrebbe essere collocato in contesti differenti, rendendo necessaria la presenza di background diversi da quelli osservati durante l'acquisizione dei dati. La presentazione alla rete di uno stesso target in presenza di sfondi variabili riduce pertanto la dipendenza dalle caratteristiche specifiche dello scenario di acquisizione e favorisce l'apprendimento di caratteristiche maggiormente associate al solo oggetto da rilevare.

Di fatti queste modifiche hanno portato ad una maggiore stabilità delle bounding box durante l'inferenza accompagnata da un livello di confidenza generalmente maggiore.

5.3 Differenze nel training a 50 epoche (nuovo vs originario)

Il training del modello originario è stato eseguito impostando 50 epoche totali. Il training del modello oggetto di questo lavoro di tesi è stato eseguito seguendo più fasi di training.

Al fine di effettuare un confronto diretto tra i due approcci, in questa sezione vengono analizzati i risultati ottenuti dai due modelli al termine di un training della durata di 50 epoche, mantenendo quindi costante il numero di epoche e variando esclusivamente il dataset utilizzato per l'addestramento.

	MODELLO NUOVO	MODELLO ORIGINARIO
mAP@50-95	0.97917	0.97522
Train box-loss	0.33409	0.2243
Train dfl-loss	0.79365	0.78048
Val box-loss	0.26567	0.27331
Val dfl-loss	0.88028	0.80475

Tabella 13 confronto dei risultati del training a 50 epoche mediante i due dataset

I dati riportati a parità di epoche evidenziano innanzitutto una differenza positiva, seppure contenuta, nella metrica mAP@50-95. Il modello addestrato con il nuovo dataset raggiunge infatti un valore pari a 0,97917, rispetto a 0,97522 ottenuto dal modello originario. Sebbene la variazione sia numericamente ridotta, essa indica una prestazione complessiva leggermente superiore del modello con il dataset nuovo.

L'analisi dei valori di loss richiede una distinzione tra la fase di training e validation. Entrambe le train loss risultano maggiori nel modello nuovo (0,33409 vs 0,2243 e 0,79365 vs 0,78048). Questo comportamento è coerente con la maggiore difficoltà presente nel nuovo dataset.

Considerando invece i risultati nel validation set, non si possono trarre le medesime conclusioni: la box-loss del modello nuovo è minore (0,26567 vs 0.27331), a differenza della DFL-loss per cui il modello nuovo riporta un valore maggiore (0.88028 vs 0.80475).

Nel complesso, il confronto suggerisce che il nuovo dataset sottopone il modello a condizioni di apprendimento più complesse, ma al contempo maggiormente rappresentative dello scenario operativo considerato. La maggiore eterogeneità delle immagini può infatti rendere più difficoltoso il processo di ottimizzazione durante il training, senza necessariamente tradursi in una riduzione delle capacità di generalizzazione.

5.4 Differenze per l'inferenza (nuovo vs originario)

Il confronto conclusivo da dover esporre riguarda l'inferenza del modello ottenuto nel precedente lavoro di tesi e di quello ottenuto nel presente lavoro.

Il modello originario conta un addestramento a imgsz di 320 successivamente impiegato in fase di inferenza con richiamo di imgsz impostato a 160. Esso non è stato sottoposto ad alcun processo di quantizzazione, aspetto verificato dalla flag "half" impostata su "false" all'interno del file metadati generato al termine dell'export in NCNN.

Di contro, il modello nuovo conta un training ad imgsz 160 con richiamo di imgsz impostato a 160 in fase di inferenza. A valle dell'esportazione, il modello è stato sottoposto a un processo di quantizzazione mista, nel quale le componenti selezionate sono state convertite in INT8 mentre quelle escluse sono state mantenute alla precisione FP16.

Per garantire una condizione di confronto il più possibile coerente con quella adottata nel precedente lavoro di tesi, le acquisizione utilizzate per la valutazione sono state eseguite adottando, per quanto possibile, la medesima configurazione di posizionamento: la telecamera è stata posizionata in modo tale da far coincidere approssimativamente il centro slitta con la mezzeria del campo visivo della telecamera stessa.

I dati che vengono riportati di seguito sono stati ricavati seguendo all'incirca la stessa configurazione di posizionamento seguita nel lavoro di tesi precedente quindi la camera è stata posizionata in mezzeria della slitta e di conseguenza il centro della slitta è stato fatto coincidere grossolanamente con la mezzeria della schermata video.

Si riporta quindi il grafico a scatola e baffi per saggiare la differenza tra i due modelli.

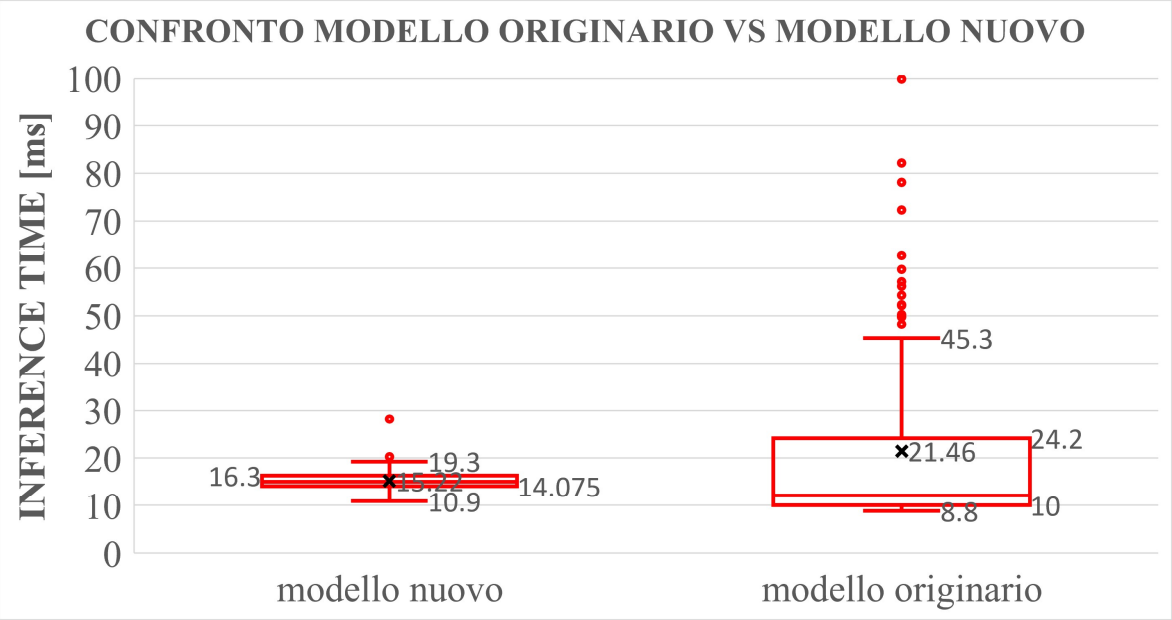


Figura 68 confronto finale tra modello nuovo e originario

I dati ricavati nei primi training hanno evidenziato la possibilità di esecuzione di ulteriori step di training. La Figura 68 conferma questa possibilità, mostrando l'evoluzione delle prestazioni a seguito di un training maggiormente prolungato.

Vengono riportati i principali dati di interesse rispetto ai due training.

	NUOVO	ORIGINARIO
Durata training [h]	2.53	2.57
Media tempo d'inferenza [ms]	15.22	21.46
Dev. St. tempo d'inferenza [ms]	1.94	19.26
Numero outlier	2	22

Tabella 14 confronto finale nuovo vs originario

La durata complessiva del training risulta pressoché equivalente nei due casi (2,53 h vs 2,57 h). Nel caso del modello addestrato sul dataset nuovo, tuttavia, il training è stato esteso oltre le 50 epoche mediante ulteriori step di addestramento come riportato nei capitoli precedenti, un'estensione resa possibile anche grazie all'impiego dell'hardware GPU dedicato.

Il tempo di inferenza subisce invece una diminuzione drastica del 30% (da 21.46s a 15.22s), un miglioramento che racchiude sia gli interventi al livello hardware sia le ottimizzazioni software descritte nei capitoli precedenti.

Il secondo dato rilevante, è il comportamento della variabilità del tempo di inferenza. La deviazione standard crolla di circa un ordine di grandezza ovvero passa da 19,26 s a 1,94 s.

Risultato coerente con il numero di outlier registrati che passa da 22 nel modello originario a soli 2 nel modello nuovo.

In un contesto applicativo come quello studiato in questa tesi, la riduzione della variabilità temporale è un aspetto rilevante quanto la riduzione del tempo medio di inferenza. La presenza di outlier si traduce direttamente in ritardi irregolari nell'aggiornamento della posizione del target, compromettendo il feedback di controllo.

6.CONCLUSIONI

Il presente lavoro di tesi si inserisce nel contesto di una sostituzione di un trasduttore di posizione, montato su un banco di prova a guida lineare, con un sistema di visione artificiale basato su un modello di object detection YOLO v8n eseguito su una piattaforma embedded, il Raspberry Pi Compute Module 5, in inferenza tramite il sensore Pi Camera Module 3, tramite il framework di inferenza NCNN.

Il percorso seguito ha attraversato un ciclo di sviluppo di un sistema di visione artificiale destinato al deployment su hardware embedded: dalla creazione del dataset, all'addestramento e alla selezione del modello, fino all'ottimizzazione e alla validazione sperimentale in condizioni realistiche.

La creazione di una dataset custom rappresenta un passaggio fondamentale in un'applicazione di object detection basata su architettura di deep learning come YOLO, poiché è attraverso di esso che la rete apprende non solo a riconoscere l'oggetto di interesse, ma anche a gestirne le diverse sfaccettature, come le variazioni dello sfondo o le variazioni angolari dell'inquadratura. Proprio per questo, disporre di una metodologia strutturata a cui fare riferimento ogni volta che sia necessario modificare un dataset esistente o costruirne uno completamente nuovo costituisce un valore aggiunto. È in questo ambito che si colloca parte del contributo del presente lavoro di tesi, che ha definito in modo puntuale tutti i passaggi eseguiti per la creazione del dataset relativo al caso di studio specifico.

Nella fase di costruzione del dataset, è stata definita una strategia di annotazione basata su bounding box che descrivono l'ingombro completo della slitta anche in presenza di occlusioni parziali di natura temporanea, scelta motivata dalla necessità di garantire coerenza geometrica nel tempo qualora uno di questi due casi occorra.

È stata successivamente condotta una selezione sistematica degli iperparametri di addestramento, confrontando diversi ottimizzatori (SGD, AdamW, Adam e NAdam) a diverse risoluzioni di input (da 640x640 fino 160x160 pixel), da cui SGD è emerso come la scelta più efficace in termini di velocità di convergenza e qualità delle metriche di rilevamento. La risoluzione di 160x160 pixel è stata individuata come configurazione di deployment ottimale, in quanto rappresenta il limite architetturale di YOLO v8n, garantendo così il minore costo computazionale possibile a parità di prestazioni.

Il passaggio dal modello addestrato in Pytorch al formato necessario per l'esecuzione su hardware embedded ha richiesto la costruzione di una pipeline di conversione ed ottimizzazione verso il formato NCNN, articolata su tre livelli di precisione numerica: FP32

come formato di default, FP16 come primo livello di compressione, e INT8 come livello massimo di compressione, ottenuto attraverso un processo di calibrazione dedicato eseguito sullo split di training. Quest'ultimo passaggio si è rivelato il più critico e ha richiesto un lavoro di messa a punto poiché una prima configurazione di quantizzazione aggressiva ha prodotto un collasso completo delle capacità di rilevamento del modello, riconducibile a un collasso della distribution focal loss e alla saturazione della confidenza di classificazione. Il problema è stato risolto individuando una configurazione di quantizzazione mista, che esclude selettivamente dalla conversione INT8 gli ultimi layer convoluzionali della testa di detection, preservandone la precisione in FP16. Questa soluzione ha permesso di ottenere un modello finale con una riduzione dimensionale di circa il 75% rispetto alla versione FP32, mantenendo al contempo capacità di rilevamento affidabili, a dimostrazione della fattibilità di un deployment efficiente su hardware con risorse limitate.

La validazione sperimentale, condotta sia in ambiente PC sia sul Raspberry Pi CM5, ha permesso di caratterizzare in modo approfondito del sistema sotto alcuni aspetti. È stato analizzato l'effetto della quantizzazione sulla stabilità spaziale delle bounding box, evidenziando come la degradazione introdotta dall'INT8 misto non sia uniforme tra le diverse componenti geometriche del rilevamento, ma colpisca in misura maggiore la larghezza delle bounding box piuttosto che la posizione del suo centro, una distorsione sistematica riconducibile alle caratteristiche ottiche del sensore camera.

Per far fronte al problema dell'instabilità delle bounding box sono state esplorate, in via preliminare, diverse strategie alternative: il freezing di una bounding box mediata dopo un certo numero di frame, da riprodurre inalterata nei frame successivi associandola al rispettivo centro della bounding box; l'introduzione di un tracker multi-oggetto ridotto a un filtro monodimensionale, giustificata dal fatto che, data la struttura del banco di prova, l'interesse ricade unicamente sul moto lungo l'asse x; l'applicazione di fattori di scala ricavati da una calibrazione eseguita su un modello a compressione minore, introducendo una media sulle coordinate; infine, una possibile implementazione di un filtro temporale sulla posizione del centro e sulla larghezza della bounding box, da utilizzare come criterio di accettazione o scarto della detection stessa. Queste strategie sono state unicamente testate in via esplorativa, senza un riscontro sperimentale sistematico, e per questo motivo non sono state riportate nel corpo principale del presente lavoro di tesi; rappresentano tuttavia una base di un possibile approfondimento futuro.

Il confronto tra diverse strategie di acquisizione e ridimensionamento dell'immagine ha mostrato benefici misurabili, sia in termini di tempo medio di inferenza sia di variabilità delle prestazioni, derivanti da una riduzione della risoluzione effettuata a monte della pipeline piuttosto che internamente al modello.

Un risultato da porre in particolare rilievo riguarda le prestazioni ottenute in termini di tempo di inferenza. La configurazione finale individuata, ovvero quantizzazione mista con base INT8 e layer esclusi a mantenuti a F16, a risoluzione 160x160, ha permesso di ridurre il tempo medio di inferenza a valori pienamente compatibili con un'applicazione in tempo reale, congiuntamente all'abbattimento della variabilità del tempo stesso di inferenza: la deviazione standard è stata ridotta di circa un ordine di grandezza rispetto alla configurazione di partenza, un risultato ottenuto attraverso una combinazione di ottimizzazione software (quantizzazione mirata, riduzione della risoluzione a monte della pipeline di inferenza) e

hardware (impostazione del governor di frequenza CPU su *perfomance*, passaggio al singolo thread e impostazione della policy di attesa dei thread su *passive*). Questo aspetto è tra i più significativi del presente lavoro di tesi: in un contesto di controllo posizione in retroazione, la stabilità temporale della misura è una condizione tanto rilevante quanto la sua accuratezza media, poiché una latenza fortemente variabile si traduce in aggiornamenti irregolari della stima della posizione, con un impatto diretto sulla qualità del controllo.

Alla luce, quindi, di quanto emerso dalle analisi condotte, il modello più equilibrato scelto tra le configurazioni esaminate è quello a quantizzazione mista con base INT8 e layer esclusi in FP16, addestrato a risoluzione 160x160 che risulta essere il modello raccomandato per il deployment finale sul Raspberry Pi CM5.

Le criticità individuate lungo il percorso ovvero la sensibilità della quantizzazione INT8 a determinati layer della rete, la distorsione introdotta dal sensore, la variabilità di latenza osservata in alcune configurazioni sulla piattaforma embedded, rappresentano aspetti che è stato possibile mitigare in gran parte.

La problematica del jitter può essere definitivamente risolta impiegando osservatori statistici basati sulla misura fornita dalla videocamera, quali il filtro di Kalman, il filtro di Kalman esteso (EKF) e il filtro particellare (*Particle Filter*), al fine di predire la posizione della slitta anche in assenza di una misura affidabile, filtrando al contempo il rumore di misura introdotto dalla quantizzazione e dalla distorsione ottica. Questi sviluppi permetterebbero, nel loro complesso, di rendere il deployment del modello YOLO più solido e compatibile con l'applicazione di feedback ottico sul banco di prova in esame, in vista di un'eventuale integrazione definitiva del sistema in un contesto operativo reale.

Restano aperte alcune direzioni di approfondimento che permetterebbero di consolidare ulteriormente i risultati ottenuti. Un benchmark esaustivo delle diverse configurazioni di modello YOLO fornirebbe una base quantitativa più solida per la scelta della configurazione ottimale. Un monitoraggio dei parametri termici e di utilizzo delle risorse del Raspberry Pi CM5 durante l'inferenza permetterebbe inoltre di stabilire con certezza il ruolo del throttling termico nella variabilità delle prestazioni osservata eventualmente accompagnato ad un passaggio a un sistema operativo con caratteristiche real-time in luogo della distribuzione standard attualmente in uso.

Infine, l'adozione di una struttura meccanica di supporto più rigida per il posizionamento della camera e di una struttura di ancoraggio fissa del banchetto stesso contribuirebbe a rendere le condizioni sperimentali più stabili e riproducibili nel tempo, riducendo ulteriori fonti di variabilità non direttamente legate al modello o all'algoritmo di visione. Questi sviluppi, pure non mettendo in discussione la validità delle conclusioni raggiunte, costituiscono un naturale proseguimento del lavoro svolto, in vista di un'eventuale integrazione definitiva del sistema in contesto operativo reale.

BIBLIOGRAFIA

[1] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick e P. Dollár, «Microsoft COCO: Common Objects in Context,» in *Computer vision – ECCV 2014: 13th European conference, Zurich, Switzerland, September 6–12, 2014, proceedings, Part V*, 2014.

[2] R. Kaur, U. Mittal, A. Wadhawan, A. Almogren, J. Singla, S. Bharany, S. Hussen e A. U. R. &. A. A. Al-Huqail, «YOLO-LeafNet: A robust deep learning framework for multispecies plant disease detection with data augmentation,» *Scientific Reports,* vol. 15, n. 28513, 2025.

[3] P. Klco, D. Koniar, L. Hargas, K. P. Dimova e M. Chnapko, «Quality inspection of specific electronic boards by deep neural networks,» *Scientific Reports,* vol. 13, n. 20657, 2023.

[4] V. A. A. Raj, P. Pathmanaban, G. M. Raja, C. Hariharan, M. P. Kumar, V. K. T. e R. Akash, «Lightweight deep learning models for visual weld defect inspection: balancing speed and accuracy with YOLO architectures,» *Welding International,* vol. 40, n. 2, pp. 70-84, 2025.

[5] H. Yaseen, «YOLO NETWORK FOR DEFECT DETECTION IN OPTICAL LENSES,» *arXiv,* 2025.

[6] D. G. Lowe, «Distinctive image features from scale-invariant keypoints,» *International Journal of Computer Vision,* vol. 60, n. 2, p. 91–110, 2004.

[7] B. T. Navneet Dalal, «Histograms of oriented gradients for human detection,» in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005.

[8] G. E. Hinton, K. A. e S. I., «ImageNet classification with deep convolutional neural networks,» in *Advances in Neural Information Processing Systems 25*, 2012.

[9] R. Girshick, «Fast R-CNN,» in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[10] S. Ren, K. He, R. Girshick e J. Sun, «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, n. 6, p. 1137–1149, 2017.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. F. Berg e A. C., «SSD: Single shot multibox detector,» in *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I*, 2016.

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He e P. Dollár, «Focal Loss for Dense Object Detection,» in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[13] J. Redmon, S. Divvala, R. Girshick e A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] G. Jocher, J. Qiu, M. Liu, S. Lyu, F. C. Akyon e M. E. Kalfaoglu, «Ultralytics YOLO26: Unified Real-Time End-to-End Vision Models,» *arXiv,* 2026.

[15] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn e A. Zisserman, «The Pascal Visual Object Classes (VOC) Challenge,» *International Journal of Computer Vision,* vol. 88, n. 2, p. 303–338, 2010.

[16] D. K. Alqahtani, M. A. Cheema, M. A. Rodriguez e A. N. Toosi, «arXiv,» 2026. [Online]. Available: https://arxiv.org/abs/2409.16808. [Consultato il giorno 31 Agosto 2026].

[17] Arducam, «Arducam 12MP 477P Autofocus High Quality Camera for Raspberry Pi,» [Online]. Available: https://www.arducam.com/arducam-12mp-imx477-motorized-focus-high-quality-camera-for-raspberry-pi.html. [Consultato il giorno 31 agosto 2026].

[18] Arducam, «Arducam 2MP Global Shutter OG02B10 Color Camera Modules Pivariety,» [Online]. Available: https://www.arducam.com/arducam-2mp-global-shutter-og02b10-color-camera-modules-pivariety-b0348.html. [Consultato il giorno 31 agosto 2026].

[19] R. P. Ltd., «Camera Module 3,» [Online]. Available: https://www.raspberrypi.com/products/camera-module-3/. [Consultato il giorno 31 agosto 2026].

[20] V. Marra, «WebThesis – Politecnico di Torino,» [Online]. Available: https://webthesis.biblio.polito.it/37587/. [Consultato il giorno 31 agosto 2026].

[21] A. Nogueira-Rodríguez, D. Glez-Peña, M. Reboiro-Jato e H. López-Fernández, «Negative Samples for Improving Object Detection—A Case Study in AI-Assisted Colonoscopy for Polyp Detection,» *Diagnostics,* vol. 13, n. 5, p. 966, 2023.

[22] Ultralytics, «Ultralytics Documentation,» [Online]. Available: https://docs.ultralytics.com/it/guides/yolo-data-augmentation. [Consultato il giorno 31 agosto 2026].

[23] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama e K. Murphy, «Speed/accuracy trade-offs for modern convolutional object detectors,» in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[24] Roboflow, «Roboflow,» [Online]. Available: https://roboflow.com/. [Consultato il giorno 01 09 2026].

[25] K. Li, M. I. Fathan, K. Patel, T. Zhang, C. Zhong, A. Bansal, A. Rastogi, J. S. Wang e G. Wang, «Colonoscopy polyp detection and classification: Dataset creation and comparative evaluations,» *PLOS ONE,* vol. 16, n. 8, p. e0255809, 2021.

[26] CloudFactory, «CloudFactory Wiki,» [Online]. Available: https://wiki.cloudfactory.com/docs/mp-wiki/splits/data-splitting-in-machine-learning. [Consultato il giorno 1 09 2026].

[27] H. Bichri, A. Chergui e M. Hain, «Investigating the Impact of Train / Test Split Ratio on the Performance of Pre-Trained Models with Custom Datasets,» *International Journal of Advanced Computer Science and Applications (IJACSA),* vol. 15, n. 2, 2024.

[28] M. Sivakumar, S. Parthasarathy e T. Padmapriya, «Trade-off between training and testing ratio in machine learning for medical image processing,» *PeerJ Computer Science,* p. e2245, 2024.

[29] H. Lee e S. Ahn, «arXiv,» 2023. [Online]. Available: https://arxiv.org/abs/2308.14466. [Consultato il giorno 01 09 2026].

[30] P. Kora, C. P. Ooi, O. Faust, U. Raghavendra, A. Gudigar, W. Y. Chan, K. Meenakshi, K. Swaraja, P. Plawiak e U. R. Acharya, «Transfer learning techniques for medical image analysis: A review,» *Biocybernetics and Biomedical Engineering,* vol. 42, n. 1, p. 79–107, 2022.

[31] S. Puttemans, T. Callemein e T. Goedemé, «Building Robust Industrial Applicable Object Detection Models Using Transfer Learning and Single Pass Deep Learning Architectures,» [Online]. Available: https://arxiv.org/abs/2007.04666.

[32] V. Kamath e A. Renuka, «Deep learning based object detection for resource constrained devices: Systematic review, future trends and challenges ahead,» *Neurocomputing,* vol. 531, p. 34–60, 2023.

[33] Ultralytics, «Esplora Ultralytics YOLOv8,» [Online]. Available: https://docs.ultralytics.com/it/models/yolov8. [Consultato il giorno 2 settembre 2026].

[34] Ultralytics, «Hyperparameter Tuning,» [Online]. Available: https://docs.ultralytics.com/it/guides/hyperparameter-tuning. [Consultato il giorno 02 settembre 2026].

[35] M. K. Ranjan Sapkota, «Ultralytics YOLO Evolution: An Overview of YOLO26, YOLO11, YOLOv8 and YOLOv5 Object Detectors for Computer Vision and Pattern Recognition,» [Online]. Available: https://arxiv.org/abs/2510.09653. [Consultato il giorno 02 settembre 2026].

[36] S. N. J. A. G. D. E. E. D. G. B. G. M. H. O. K. G. V. H. W. Paulius Micikevicius, «Mixed Precision Training,» in *International Conference on Learning Representations (ICLR)*, 2018.

[37] L. Prechelt, «Early Stopping — But When?,» in *Neural Networks: Tricks of the Trade*, Springer, Berlin, Heidelberg, 2012, p. 53–67.

[38] P. Contributors, «Reproducibility,» [Online]. Available: https://pytorch.org/docs/stable/notes/randomness. [Consultato il giorno 02 settembre 2026].

[39] D. Adams, The Hitchhiker's Guide to the Galaxy, Pan Books, 1979.

[40] B. Khalili e A. W.Smyth, «SOD-YOLOv8 -- Enhancing YOLOv8 for Small Object Detection in Traffic Scenes,» 2024. [Online]. Available: https://arxiv.org/abs/2408.04786. [Consultato il giorno 03 settembre 2026].

[41] M. U. Gill e P. Rajendran, «Hyperparameter optimization of YOLO using differential evolution, multi-fidelity optimization, and Bayesian optimization,» *Scientific Reports,* vol. 16, p. 16117, 2026.

[42] Ultralytics, «Aumento dei dati con Ultralytics YOLO,» [Online]. Available: https://docs.ultralytics.com/it/guides/yolo-data-augmentation. [Consultato il giorno 03 settembre 2026].

[43] Tencent, «ncnn: A High-Performance Neural Network Inference Framework,» [Online]. Available: https://github.com/Tencent/NCNN. [Consultato il giorno 03 settembre 2026].

[44] NVIDIA, «NVIDIA Jetson Modules,» [Online]. Available: https://developer.nvidia.com/embedded/jetson-modules. [Consultato il giorno 03 settembre 2026].

[45] Google, «Edge TPU FAQ,» [Online]. Available: https://coral.ai/docs/edgetpu/faq/. [Consultato il giorno 03 settembre 2026].

[46] R. P. Ltd., «AI HAT+,» [Online]. Available: https://www.raspberrypi.com/documentation/accessories/ai-hat-plus.html. [Consultato il giorno 03 settembre 2026].

[47] V. Marra, A. De Martin, A. C. Bertolino e M. Sorli, «Preliminary study on the development of a closed-loop control system using optical feedback,» *Machines,* vol. 13, n. 12, p. 1085, 2025.

[48] Tencent, «Param and Model File Structure,» [Online]. Available: https://ncnn.readthedocs.io/en/latest/developer-guide/param-and-model-file-structure.html. [Consultato il giorno 03 settembre 2026].

[49] P. Molchanov, S. Tyree, T. Karras, T. Aila e J. Kautz, «Pruning Convolutional Neural Networks for Resource Efficient Inference,» in *ICLR 2017 (International Conference on Learning Representations)*, 2017.

[50] R. P. Foundation, «Frequency management and thermal control,» [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#frequency-management-and-thermal-control. [Consultato il giorno 04 settembre 2026].

[51] G. Sistu e S. Yogamani, «FisheyeDetNet: 360° Surround view Fisheye Camera based Object Detection System for Autonomous Driving,» [Online]. Available: https://arxiv.org/abs/2404.13443. [Consultato il giorno 04 settembre 2026].