UNIVERSITÀ DEGLI STUDI DI CATANIA

DIPARTIMENTO DI MATEMATICA E INFORMATICA

CORSO DI LAUREA MAGISTRALE IN INFORMATICA

Matteo Imbrosciano e

Daniele Cocuzza

Coin Detection

Ralazione

Prof: Giovanni Maria Farinella

Prof: Rosario Leonardi

ANNO ACCADEMICO 2023/24

INDICE

1 INTRODUZIONE

La visione artificiale è un campo dell'informatica che simula la comprensione visiva umana attraverso le macchine. Si applica in un'ampia gamma di settori, inclusi la robotica, la sorveglianza, l'assistenza sanitaria e l'automazione industriale. Il rilevamento degli oggetti, una componente essenziale della visione artificiale, permette alle macchine di identificare e localizzare elementi visivi specifici nelle immagini o nei video.

Gli algoritmi di rilevamento degli oggetti non solo riconoscono la presenza di oggetti ma anche delineano la loro posizione esatta, solitamente tramite bounding box (rettangoli che incorniciano l'oggetto).

Il modello usato è YOLOv8 che analizza le immagini in una sola passata (da qui "You Only Look Once"), contrariamente ad altri approcci che richiedono passaggi multipli.

Le metriche sono essenziali per valutare l'efficacia di un algoritmo di rilevamento degli oggetti. Queste includono la precision, il recall, l'accuracy e la F1-score.

Il riconoscimento automatizzato delle monete presenta sfide uniche, inclusa la varietà di angolazioni, illuminazione, e sfondi sotto cui le monete possono essere presentate.

Utilizzando YOLOv8, sviluppiamo un sistema in grado di identificare efficacemente le monete euro da immagini catturate in condizioni variabili. Questo sistema dimostra l'adattabilità e la robustezza del modello in applicazioni reali.

Il successo del modello dipende dalla qualità e varietà del dataset utilizzato. Il nostro dataset include immagini di monete acquisite da diverse fonti e annotate per garantire un addestramento accurato.

L'accuratezza di YOLOv8, lo rendono ideale per applicazioni come il riconoscimento delle monete in contesti commerciali e finanziari.

Questa relazione offre una panoramica approfondita su come le tecniche di rilevamento degli oggetti possono essere applicate per risolvere problemi pratici. Si illustrerà il processo di acquisizione del dataset, l'addestramento del modello e l'utilizzo del modello addestrato per sviluppare un'applicazione capace di riconoscere le monete da un'immagine.

2 Riferimenti Teorici

Il rilevamento degli oggetti è una componente fondamentale dei sistemi di visione artificiale che consente l'identificazione e la localizzazione di specifici elementi all'interno di immagini o video. Questo processo è essenziale in molteplici applicazioni che vanno dalla sicurezza alla navigazione autonoma, dall'analisi del comportamento umano alla gestione dei contenuti multimediali. Nel contesto del sistema che abbiamo sviluppato ovvero l'object detection, il rilevamento degli oggetti assume un ruolo cruciale, specialmente per la sua integrazione con tecnologie di tracciamento in tempo reale e analisi video avanzata.

2.1 Tecnologie di Rilevamento

Il rilevamento degli oggetti nel nostro sistema è implementato tramite il modello YOLOv8, che rappresenta l'ultima evoluzione di una famosa serie di modelli di deep learning progettati per fornire un rilevamento ad alte prestazioni con tempi di elaborazione ridotti.

2.2 Funzionamento di YOLOv8

YOLOv8 rappresenta un punto di riferimento nel campo del rilevamento oggetti grazie alla sua architettura innovativa e alla sua efficienza operativa. Le caratteristiche di YOLOv8, e della serie YOLO in generale, sta nella sua capacità di analizzare l'immagine complessiva in una sola passata, differenziandosi sostanzialmente da altri approcci che analizzano parti dell'immagine in fasi separate. Il cuore dell'approccio YOLOv8 risiede nella suddivisione dell'immagine in una griglia, dove ogni cella della griglia è responsabile del rilevamento degli oggetti il cui centro cade all'interno di essa. Questa suddivisione consente al modello di mappare efficientemente lo spazio dell'immagine e di associare ogni rilevamento a una specifica area, semplificando così il compito di localizzazione. Per ogni cella della griglia, YOLOv8 prevede simultaneamente i bounding boxes – rettangoli che delimitano l'oggetto rilevato – e le probabilità associate a ciascuna classe di oggetto che il modello è stato addestrato a riconoscere. I bounding boxes sono caratterizzati da quattro attributi: le coordinate del centro (x, y), la larghezza (w) e l'altezza (h). Ogni box ha anche associata una probabilità

di classe, che indica la confidenza del modello nel rilevamento dell'oggetto all'interno di quel box. Questo approccio non solo accelera notevolmente il processo di rilevamento ma riduce anche la complessità computazionale, rendendo YOLOv8 particolarmente adatto per applicazioni in tempo reale e su dispositivi con capacità di elaborazione limitate.

2.3 La precisione e l'efficienza

La precisione e l'efficienza di YOLOv8, un modello all'avanguardia nel rilevamento degli oggetti, derivano da una serie di innovazioni tecniche e architetturali che ne ottimizzano le prestazioni mantenendo allo stesso tempo requisiti computazionali contenuti. Questi aspetti sono cruciali per applicazioni in tempo reale e su dispositivi con capacità di elaborazione limitate. L'architettura di YOLOv8 è progettata per massimizzare l'efficienza computazionale e la precisione di rilevamento. Sfrutta convoluzioni profonde, che sono operazioni matematiche applicate alle immagini per estrarre caratteristiche visive ad alto livello. Queste caratteristiche sono fondamentali per identificare oggetti all'interno delle immagini, poiché catturano dettagli complessi che distinguono un oggetto dall'altro.
La profondità delle convoluzioni permette al modello di apprendere rappresentazioni ricche e variegate, che sono essenziali per il rilevamento accurato di oggetti in scenari diversificati. Una delle chiavi dell'efficienza di YOLOv8 è l'impiego di AutoML (Automated Machine Learning) e pruning nelle fasi di sviluppo e ottimizzazione del modello. L'AutoML aiuta a identificare automaticamente le configurazioni ottimali della rete neurale, riducendo il tempo e le risorse necessarie per la sperimentazione manuale. Il pruning, d'altra parte, consiste nel rimuovere i pesi meno importanti dalla rete neurale, una pratica che porta a modelli più leggeri e veloci senza un calo significativo della precisione di rilevamento. Queste tecniche riducono la complessità del modello e i suoi requisiti di memoria, rendendo YOLOv8 particolarmente adatto per l'esecuzione su hardware con risorse limitate. Il pruning, in particolare, è cruciale per adattare il modello a dispositivi edge come telecamere di sicurezza, smartphone e sistemi embedded in veicoli autonomi, dove la rapidità di elaborazione è tanto critica quanto l'accuratezza del rilevamento. L'ottimizzazione di YOLOv8 per precisione ed efficienza si traduce in un equilibrio tra la qualità del rilevamento e la velocità di elaborazione. Ad esempio, in ambito di sorveglianza, un sistema in grado di identificare rapidamente un individuo sospetto può permettere un intervento tempestivo. Analogamente, nella guida autonoma, la capacità

di riconoscere in tempo reale pedoni o ostacoli è importante per la sicurezza.

2.4 Metriche di Qualità

Le metriche di qualità per le immagini e i video giocano un ruolo fondamentale nell'ambito della visione artificiale e del processing multimediale. Forniscono un ponte tra l'aspetto tecnico e quantitativo dell'elaborazione delle immagini e la percezione umana della qualità visiva. Eseguire una valutazione qualitativa efficace richiede una comprensione profonda sia degli aspetti tecnici che di quelli percettivi. Bisogna distinguere due diverse tipologie di qualità:

- Qualità assoluta: La qualità assoluta di un'immagine si riferisce alle sue proprietà oggettive, misurabili attraverso un insieme di attributi che possono variare da quelli semplici, come il contrasto o la luminosità, a quelli più complessi composti da sottocategorie. Alcuni di questi attributi sono direttamente quantificabili, mentre altri, per la loro natura intrinseca, restano al di fuori dell'ambito della misurazione diretta.
- Qualità Percepita: la qualità percepita incorpora la dimensione soggettiva della visione, il che significa che oltre alle caratteristiche fisiche dell'immagine, intervengono fattori come l'interpretazione personale, le aspettative e la familiarità dell'osservatore con il soggetto dell'immagine. Questo tipo di qualità è influenzato da vari elementi, tra cui le tecniche di elaborazione utilizzate, le specifiche tecnologiche del dispositivo di visualizzazione e le caratteristiche uniche del sistema visivo umano.

Dunque, le metriche di qualità assoluta vengono definite metriche oggettive, mentre quelle di qualità percepita vengono definite metriche soggettive. Le metriche di qualità possono essere a sua volta classificate in base alla presenza di una maggiore o minore disponibilità del segnale di riferimento, con il quale confrontare l’immagine distorta.

A seconda della disponibilità dell'immagine di riferimento, le metriche di qualità si dividono in:

- Full Reference (FR): Queste metriche presuppongono la disponibilità dell'immagine originale non distorta, consentendo un confronto diretto e completo. Offrono una valutazione precisa della qualità, confrontando l'immagine distorta con quella di riferimento su una base pixel per pixel o attraverso caratteristiche estratte.
- No-Reference (NR): Tali metriche sono cruciali quando l'immagine di riferimento non è disponibile. Utilizzano modelli predittivi o estraggono indicatori di qualità basandosi unicamente sull'immagine distorta, facendo affidamento su principi di qualità intrinseci o su assunzioni sulle caratteristiche delle immagini di alta qualità.
- Reduced Reference (RR): Questo approccio si colloca tra i due precedenti, utilizzando solo informazioni parziali estratte dall'immagine originale. Le metriche RR sono particolarmente utili in scenari in cui si desidera bilanciare la necessità di un riferimento con la limitata disponibilità di dati o con restrizioni di banda.

2.4.1 Precision

La precisione è una delle metriche più comuni utilizzate per valutare le prestazioni di un modello di classificazione. Rappresenta la frazione di istanze classificate correttamente come positivi rispetto a tutte le istanze classificate come positivi, indipendentemente dalla classificazione negativa.

$$\text{Precisione} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Dove:

- TP (True Positives) rappresenta il numero di casi positivi che sono stati correttamente classificati come positivi dal modello.
- FP (False Positives) rappresenta il numero di casi negativi che sono stati erroneamente classificati come positivi dal modello.

La precisione misura la proporzione di previsioni positive del modello che sono corrette. Una precisione più alta indica che il modello ha una bassa percentuale di falsi positivi, ossia ha meno casi negativi erroneamente classificati come positivi. Una precisione più bassa indica che il modello ha una percentuale più alta di falsi positivi.

2.4.2 ReCall

Il recall, conosciuta anche come sensitivity o true positive rate (TPR), è una metrica utilizzata per valutare le prestazioni di un modello di classificazione, specialmente in presenza di classi sbilanciate. Rappresenta la frazione di istanze positive correttamente identificate dal modello rispetto a tutte le istanze positive effettive. In formule si ha:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Dove:

- TP (True Positives) rappresenta il numero di casi positivi che sono stati correttamente classificati come positivi dal modello.

- FN (False Negatives) rappresenta il numero di casi positivi che sono stati erroneamente classificati come negativi dal modello.

Il recall misura la capacità del modello di identificare correttamente tutti i casi positivi presenti nel dataset. Un recall più alto indica che il modello ha una bassa percentuale di casi positivi non individuati (falsi negativi). Un recall più basso indica che il modello ha una percentuale più alta di casi positivi non individuati.

2.4.3 Accuracy

L'accuracy, o accuratezza, è una delle metriche più comuni per valutare le prestazioni di un modello di classificazione, ma può essere applicata anche in contesti come il rilevamento di oggetti. L'accuracy misura la frazione di predizioni corrette (sia positive che negative) rispetto al totale delle predizioni fatte dal modello.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Dove:

- TP (True Positives) sono i casi in cui il modello predice correttamente la presenza della classe positiva;
- TN (True Negatives) sono i casi in cui il modello predice correttamente l'assenza della classe positiva;
- FP (False Positives) sono i casi in cui il modello predice erroneamente la presenza della classe positiva;
- FN (False Negatives) sono i casi in cui il modello predice erroneamente l'assenza della classe positiva.

Nel rilevamento di oggetti, l'uso dell'accuracy può essere meno diretto perché non esiste sempre un concetto chiaro di "negativo vero" (TN). In molti sistemi di rilevamento di oggetti, ogni area dell'immagine potenzialmente contiene un oggetto di interesse, quindi il concetto di TN (aree correttamente identificate come prive di oggetti) spesso non è applicabile.
Di conseguenza, altre metriche come la Precision e il Recall diventano più rilevanti:

- Precision: indica quanto sono affidabili i positivi predetti dal modello.
- Recall: indica quanto bene il modello è in grado di identificare tutti i positivi rilevanti.

2.4.4 F1-Score

La F1-score è una metrica cruciale per valutare le prestazioni di un modello di rilevamento di oggetti, che equilibra la precision e il recall, offrendo una visione complessiva dell'accuratezza del modello quando entrambi questi aspetti sono importanti. La F1-score è particolarmente utile in contesti dove è fondamentale non solo identificare correttamente gli oggetti di interesse (recall), ma anche minimizzare il numero di falsi allarmi (precision).
La F1-score è calcolata come la media armonica tra precisione e richiamo.

La formula è la seguente:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

La F1-score aiuta a bilanciare queste due esigenze, fornendo una misura di successo che tiene conto di entrambi gli aspetti. Utilizzare la F1-score durante l'adjustment dei parametri del modello può guidare verso un miglior equilibrio tra rilevare correttamente gli oggetti (richiamo) e minimizzare gli allarmi falsi (precisione). Questo è particolarmente utile per affinare il modello in maniera che risponda meglio alle necessità specifiche dell'applicazione.

2.5 Grafici

2.5.1 Matrice di Confusione

Il grafico ci mostra come le predizioni del modello si confrontano con i dati reali.
L'asse delle ascisse rappresenta le classi reali mentre l'asse delle ordinate rappresenta le classi predette.
Le celle sulla diagonale principale rappresentano le classi in cui il modello ha fatto una previsione corretta, le celle fuori dalla diagonale principale rappresentano gli errori di classificazione.

2.5.2 Precision confidence Curve

Nell'asse x si trova la soglia di confidenza dei rilevamenti del modello, valore nell'intervallo [0, 1]. Per valori prossimi allo 0 si accettano quasi tutti i rilevamenti mentre per valori prossimi ad 1 si accettano solo i rilevamenti di cui il modello è sicuro.
Nell'asse y vi è la Precision, più alta è la Precision e meno errori il modello commette nei rilevamenti accettati. Il grafico mostra la precisione del modello al variare della soglia di confidenza, quando la confidenza è bassa il modello rileva molti oggetti, includendo molti falsi positivi, quindi la precisione è più bassa. All'aumentare della soglia di confidenza, il modello diventa più selettivo, rilevando meno oggetti ma con maggiore precisione.

2.5.2 ReCall confidence Curve

La Recall-Confidence Curve rappresenta la relazione tra il livello di fiducia delle previsioni di un modello e la recall (o sensitività) ottenuta a quel livello di fiducia. Questo tipo di grafico aiuta a capire come varia la recall del modello man mano che aumentiamo la soglia di fiducia necessaria affinché una previsione venga considerata positiva. L'asse x rappresenta il livello di fiducia. I valori sull'asse vanno da 0 a 1, dove 0 significa che il modello è completamente insicuro della sua previsione e 1 significa che il modello è completamente sicuro. L'asse y appresenta il recall, che misura la proporzione di veri positivi catturati dal modello. Il recall varia da 0 a 1, dove 0 significa che il modello non ha catturato nessun vero positivo e 1 significa che ha catturato tutti i veri positivi.

2.5.2 F1 confidence Curve

L'F1-Confidence Curve traccia la relazione tra il livello di fiducia delle previsioni di un modello e l'F1-score ottenuto a quel livello di fiducia. L'F1-score è una misura che combina precisione e recall in un'unica metrica, fornendo un'idea complessiva delle prestazioni del modello. L’asse x rappresenta la soglia di fiducia, con valori che vanno da 0 a 1. Una soglia di 0 significa che il modello classifica tutte le istanze come positive, mentre una soglia di 1 significa che il modello classifica solo le istanze di cui è completamente sicuro come positive. L’asse y rappresenta l'F1-score, che varia da 0 a 1. Un valore di 0 indica che il modello ha una pessima combinazione di precisione e recall, mentre un valore di 1 indica che il modello ha la massima precisione e recall possibili.

3 Applicazioni Pratiche

3.1 Problema

Si vuole proporre una soluzione per il problema del riconoscimento delle monete tramite algoritmi di machine learning. Questo tipo di problema è particolarmente rilevante in contesti in cui è necessario analizzare rapidamente e accuratamente un numero elevato di monete, come nei distributori automatici, nelle macchine di scambio valuta, o nei sistemi di pagamento automatizzati.
La valuta trattata nel progetto è l'euro.

L'applicazione deve essere in grado di identificare le monete da immagini catturate da diverse angolazioni e con variazioni di illuminazione e sfondo. La capacità del sistema nel gestire queste variazioni è importante per garantire un'alta affidabilità nelle operazioni di riconoscimento.

La soluzione al problema migliorerebbe l'efficienza nelle operazioni automatizzate di riconoscimento delle monete, riducendo l'errore umano.

3.2 Acquisizione del Dataset

Il dataset è fondamentale per l'addestramento di un modello robusto e preciso. Per garantire una varietà e una copertura ampia nel nostro dataset, abbiamo combinato immagini acquisite personalmente con altre scaricate dal web. Circa il 70% delle immagini sono state fotografate direttamente da noi, utilizzando diverse condizioni di illuminazione e angolazioni per simulare vari scenari realistici in cui il sistema finale dovrà operare. Il restante 30% delle immagini è stato ricavato da fonti online, permettendoci di arricchire ulteriormente il dataset con monete che non erano immediatamente disponibili nel nostro ambiente locale.

Le immagini sono etichettate con le seguenti classi:

['10c', '1c', '1e', '20c', '2c', '2e', '50c', '5c']

Ogni classe rappresenta il valore della moneta.

Il nostro approccio ha previsto la creazione di due distinti tipi di dataset, ciascuno con un obiettivo specifico:

- Dataset di monete singole per l'addestramento: Il primo dataset è composto esclusivamente da immagini di singole monete. Queste immagini sono state annotate manualmente utilizzando VGG Annotation Tool, un software popolare per l'annotazione di immagini che ci ha permesso di etichettare ogni moneta con grande precisione. Le etichette comprendono il tipo di moneta. Questo dataset è stato principalmente utilizzato per la fase di addestramento del modello, permettendo di insegnare al sistema le caratteristiche distintive di ciascuna moneta.

- Dataset di monete multiple per l'addestramento: Il secondo tipo di dataset consisteva in 200 immagini utilizzate per il training, contenenti diverse monete in ciascuna immagine. Queste sono state annotate utilizzando Roboflow, un altro strumento di annotazione che supporta la creazione di bounding box. Ogni bounding box era accuratamente posizionato attorno a ogni moneta e etichettato con il valore della moneta. Questo dataset è stato essenziale per l'addestramento del modello in compiti di rilevazione d'oggetti, dove la localizzazione è tanto importante quanto la classificazione.

- Dataset di immagini di gruppo per il test: Dopo l'addestramento iniziale, abbiamo utilizzato un set di immagini che includevano gruppi di monete. Questo ci ha permesso di valutare la capacità del modello di identificare e classificare le monete in scenari più complessi e vicini a situazioni reali.

L'uso combinato di questi dataset ha offerto una panoramica completa sulle capacità del nostro modello. Le fasi di testing con immagini raffiguranti più monete hanno evidenziato la robustezza del modello nel riconoscere e localizzare più monete in un'unica immagine. Si mostreranno le differenze nelle prestazioni durante l’addestramento con il primo dataset rispetto al secondo.

Struttura dataset per YOLO

```
├── train/
│   ├── Images/
│   │   ├── imageID_1.jpg
│   │   ├── imageID_2.jpg
│   │   ├── ...
│   │   └── imageID_n.jpg
│   ├── labels/
│   │   ├── imageID_1.txt
│   │   ├── imageID_2.txt
│   │   ├── ...
│   │   └── imageID_m.txt
├── valid/
│   ├── Images/
│   │   ├── imageID_1.jpg
│   │   ├── imageID_2.jpg
│   │   ├── ...
│   │   └── imageID_n.jpg
│   ├── labels/
│   │   ├── imageID_1.txt
│   │   ├── imageID_2.txt
│   │   ├── ...
│   │   └── imageID_m.txt
├──testing/
│   ├── Images/
│   │   ├── imageID_1.jpg
│   │   ├── imageID_2.jpg
│   │   ├── ...
│   │   └── imageID_n.jpg
│   ├── labels/
│   │   ├── imageID_1.txt
│   │   ├── imageID_2.txt
│   │   ├── ...
│   │   └── imageID_m.txt
└──data.yaml
```

Il file data.yaml contiene le informazioni riguardo la struttura del dataset utilizzato per addestrare il modello. Si specificano i percorsi delle cartelle contenenti le immagini di training, valid e test. Viene specificato il numero e il nome delle classi.

3.3 Metodi

Nel nostro progetto di rilevazione di monete tramite tecniche di machine learning, abbiamo impiegato il modello YOLO (You Only Look Once) per l'addestramento con l'obiettivo di sviluppare un sistema capace di identificare e localizzare oggetti nelle immagini.

Per l'addestramento, abbiamo utilizzato il dataset precedentemente descritto, che include immagini annotate di singole monete e gruppi di monete. Le annotazioni includevano bounding box e identificativi delle monete, preparate per essere compatibili con le specifiche di input di YOLO.

L’addestramento è stato effettuato sia per il dataset contenente immagini che raffigurano più monete in un’immagine e sia per il dataset con immagini che raffigurano una sola moneta in un’immagine. Sono stati condotti più esperimenti usando le versioni: YOLOv8n e YOLOv8s.

Le immagini di training sono state caricate nel sistema di addestramento di YOLO. Durante l'addestramento, YOLO elabora le immagini, apprendendo a riconoscere le varie caratteristiche e attributi delle monete, come dimensione, forma e dettagli distintivi.

Al termine dell'addestramento, YOLO genera una serie di output che sono salvati automaticamente in una cartella denominata “runs”. Questa cartella contiene vari file che rappresentano i risultati dell'addestramento, inclusi i pesi del modello addestrato, le metriche di prestazione, e le immagini di esempio con le predizioni del modello.

3.3.1 Train

```
from ultralytics import YOLO

model = YOLO('yolov8n.yaml')

results = model.train(data='data.yaml', epochs=500)
```

Dettagli del codice

1. *from ultralytics import YOLO*
   Con questo comando si importa il modulo YOLO dalla libreria ultralytics.
2. *model = YOLO('yolov8n.yaml')*
   Crea un'istanza del modello YOLO utilizzando un file di configurazione YAML denominato 'yolov8n.yaml'.
3. *results = model.train(data='data.yaml', epochs=500)*
   Avvia il processo di addestramento del modello YOLO.
   Parametri:
   - *data='data.yaml'*: specifica il percorso al file di configurazione dei dati, 'data.yaml'. Questo file contiene informazioni sul dataset utilizzato per l'addestramento, come i percorsi delle immagini e delle annotazioni, la suddivisione del dataset (training e validation set), e le classi degli oggetti da rilevare.
   - *epochs=500*: imposta il numero di epoche di addestramento a 500. Un'epoca rappresenta un ciclo completo attraverso tutto il dataset di addestramento.

   La variabile *results* contiene i risultati dell'addestramento, inclusi metriche di performance come la loss, la precisione e il recall.

Durante l'esecuzione del codice relativo al training, viene generata automaticamente la cartella 'runs', essa contiene i risultati dell'addestramento, i pesi del modello e altre informazioni

```
runs/
├── detect/
│   ├── train/
│   │   ├── weights/
│   │   │   ├── best.pt              # Pesi del modello che hanno ottenuto le migliori prestazioni
│   │   │   └── last.pt              # Pesi del modello dopo l'ultima epoca di addestramento
│   │   ├── events.out.tfevents...  # File di log di TensorBoard
│   │   ├── results.png             # Immagini di esempio con bounding box
│   │   ├── labels.jpg              # Immagine con etichette di esempio
│   │   ├── F1_curve.png            # Grafico della curva F1
│   │   ├── P_curve.png             # Grafico della precisione
│   │   ├── R_curve.png             # Grafico del richiamo
│   │   ├── PR_curve.png            # Grafico della curva Precision-Recall
│   │   ├── confusion_matrix.png    # Matrice di confusione
│   │   └── train_batch0.jpg        # Batch di addestramento di esempio
│   │   └── train_batch1.jpg        # Batch di addestramento di esempio
│   │   └── val_batch0_labels.jpg   # Batch di validazione di esempio
│   │   └── val_batch0_pred.jpg     # Batch di validazione con predizioni
│   ├── val/
│   │   ├── (similar structure to train directory)
│   └── ...
```

3.3.2 Validation

```
from ultralytics import YOLO
import numpy as np

model = YOLO('/content/gdrive/My Drive/Progetto/runs/yolov8n/moneta_multipla/runs/detect/train/weights/best.pt')

metrics = model.val(data='/content/gdrive/My Drive/Progetto/runs/yolov8n/moneta_multipla/data.yaml')
```

Il modello viene caricato in memoria utilizzando il percorso specificato. Questo modello è stato precedentemente addestrato per rilevare monete in immagini.

La funzione model.val() viene utilizzata per valutare le prestazioni del modello su un set di dati di validazione.

Il file data.yaml specifica i percorsi delle immagini di validazione e delle relative annotazioni.

La variabile metrics conterrà le metriche di valutazione del modello, come precisione, richiamo, F1 score, mAP (mean Average Precision) e altre statistiche che indicano quanto bene il modello riesce a rilevare e classificare le monete nel set di dati di validazione. Le metriche ottenute possono essere utilizzate per capire le prestazioni del modello e per fare confronti con altri modelli o versioni dello stesso modello.

3.4 Applicativo

3.4.1 Obiettivo

Si intende sviluppare un'applicazione in grado di elaborare immagini di monete caricate dagli utenti. Una volta caricata l'immagine, l'applicazione utilizzerà algoritmi di riconoscimento delle immagini, basati su tecniche di machine learning, per rilevare e identificare le monete presenti nella fotografia.

Il modello di riconoscimento delle monete esegue la rilevazione e la classificazione delle monete presenti nell'immagine. Ogni moneta rilevata viene contrassegnata con una bounding box e un'etichetta indicante il tipo di moneta (ad esempio, 1 centesimo, 2 centesimi, 5 centesimi, 10 centesimi, 20 centesimi, 50 centesimi, 1 euro, 2 euro). L'algoritmo calcola anche la somma dei valori di ciascuna moneta identificata.

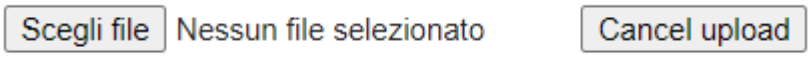
L'applicazione genera quindi una nuova versione dell'immagine originale, con le bounding box e le etichette sovrapposte sulle monete rilevate. Inoltre, in un'area specifica dell'immagine, viene visualizzato il valore totale delle monete identificate, sommando i valori individuali delle monete riconosciute.

3.4.2 Soluzione

L'applicazione utilizza diverse librerie Python tra cui YOLO per il rilevamento degli oggetti, Matplotlib per la visualizzazione grafica, OpenCV per l'elaborazione delle immagini e Google Colab per il caricamento dei file.

Viene caricato un modello YOLO pre-addestrato, specificamente addestrato per rilevare monete.

L'utente carica un'immagine di monete tramite la seguente interfaccia:



L'immagine è convertita in formato RGB, necessario per l'analisi successiva.

Il modello YOLO analizza l'immagine pre-processata per rilevare le monete presenti. Il modello genera delle "bounding box" attorno alle monete rilevate, insieme a una stima del tipo di moneta e un punteggio di confidenza per ciascuna rilevazione.

Per evitare duplicazioni, l'applicazione utilizza la Non-Maximum Suppression (NMS) per filtrare le rilevazioni sovrapposte, mantenendo solo quelle con il punteggio di confidenza più alto.

Utilizzando un dizionario che associa ciascun tipo di moneta al suo valore in euro, l'applicazione calcola il valore totale delle monete rilevate. Viene anche mantenuto un conteggio di ciascun tipo di moneta rilevata.

Le bounding box e le etichette contenenti il tipo di moneta e il punteggio di confidenza vengono disegnate sull'immagine originale. Viene inoltre aggiunta un'etichetta che mostra il valore totale delle monete rilevate.

L'immagine annotata viene visualizzata utilizzando Matplotlib, permettendo all'utente di vedere quali monete sono state rilevate e il loro valore totale.

Supponiamo che l'utente carichi la seguente immagine:

Dopo aver caricato l'immagine l'applicativo restituisce il detection e la somma delle monete.

Nel caso del dataset raffigurante immagini con una singola moneta:

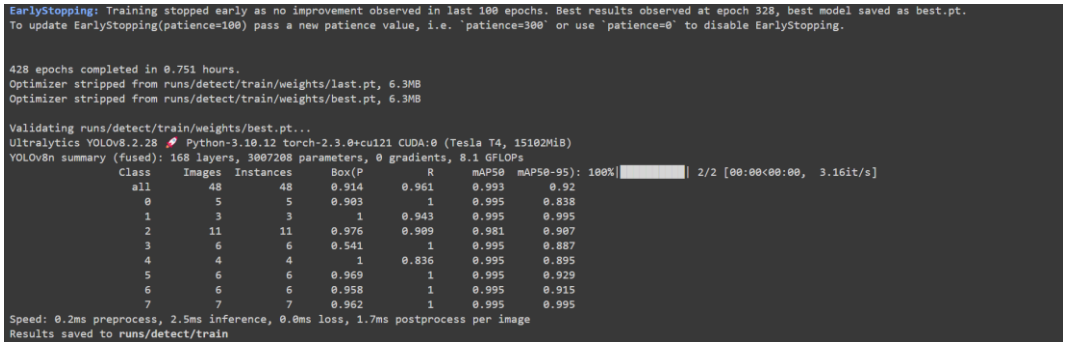
L’output è il seguente:

4.1 Dataset a singola moneta

Si mostrano i risultati ottenuti effettuando il training col dataset in cui è raffigurata una moneta in un'immagine.

4.1.1 Modello YOLOv8n

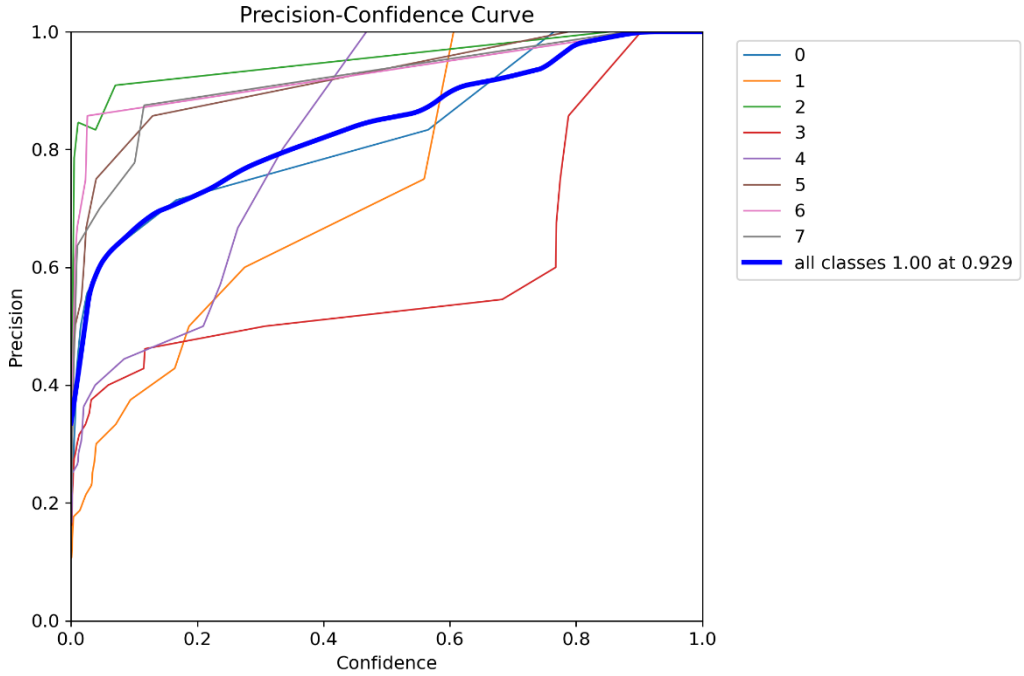
Il training effettuato utilizzando il modello YOLOv8n abbiamo usato 500 epoche ma il training si è fermato a 428 perché dall'epoca 328 non ci sono stati risultati migliori.



Il modello è stato valutato tramite le metriche: Precision, Recall, Accuracy ed F1 Score, di seguito i valori ottenuti:

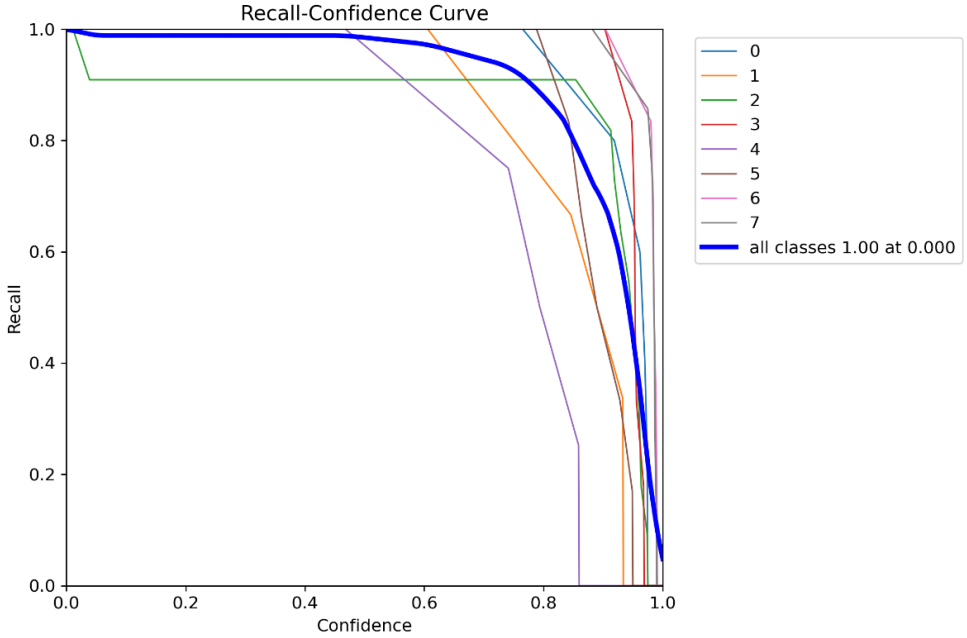
- Precision per classe: [0.90107 1 0.97594 0.54106 1 0.96927 0.95816 0.96163]
- Recall per classe: [1 0.94343 0.90909 1 0.83655 1 1 1]
- F1 score per classe: [0.94796 0.97089 0.94133 0.70219 0.911 0.98439 0.97863 0.98044]
- Precision media: 0.9134
- Recall media: 0.9611
- F1 Score media: 0.9271
- Accuracy: 0.4683

Il grafico seguente è una Precision-Confidence Curve.



Le classi: 5c, 50c, 1e, 2e hanno una precisione molto alta su diversi valori di confidenza, questo indica che il modello ha una buona capacità di rilevare queste monete.

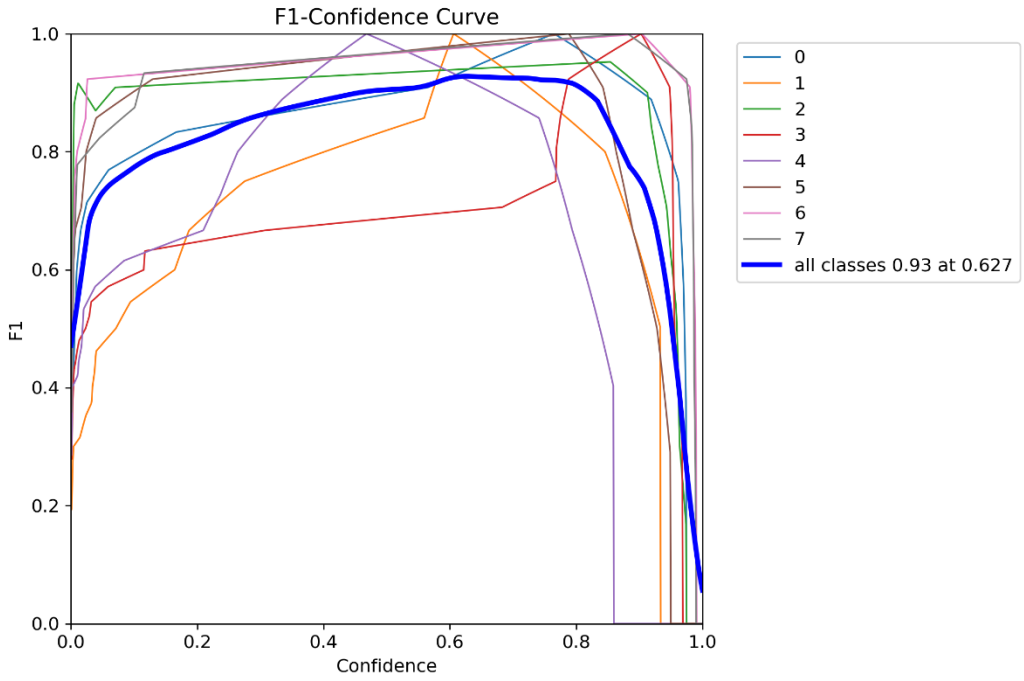
A seguire il grafico Recall-Confidence Curve del modello utilizzato.



Nell’asse x vi è la soglia di confidenza, mentre nell’asse y il valore di Richiamo corrispondente alla soglia di confidenza x.

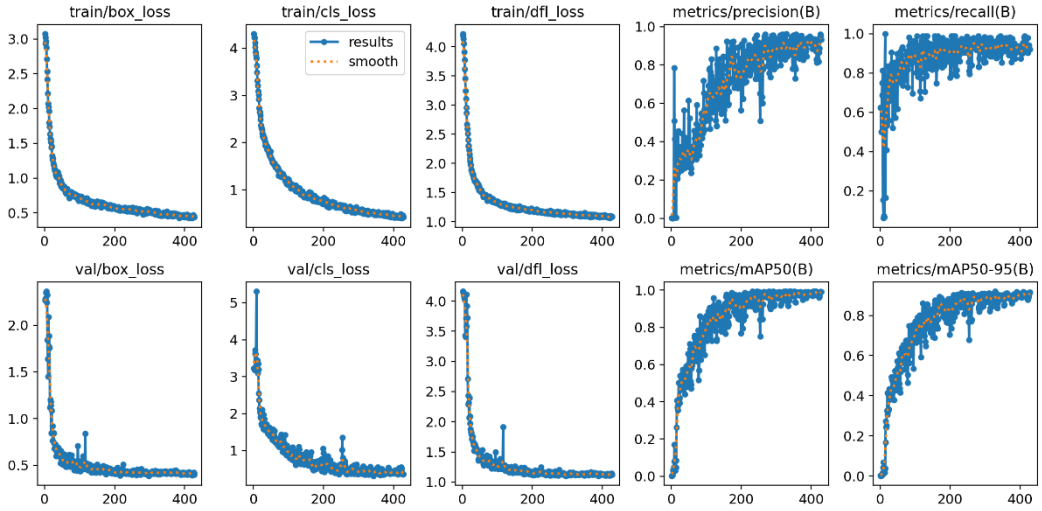
Si può notare che la Recall diminuisce all’aumentare della soglia di confidenza, aumentando la soglia di confidenza il modello diventa più selettivo, riducendo il numero di falsi positivi ma aumentando i falsi negativi.

Si mostra il grafico F1-Confidence Curve.



Il grafico mostra come cambia l’F1-Score al variare della soglia di confidenza.

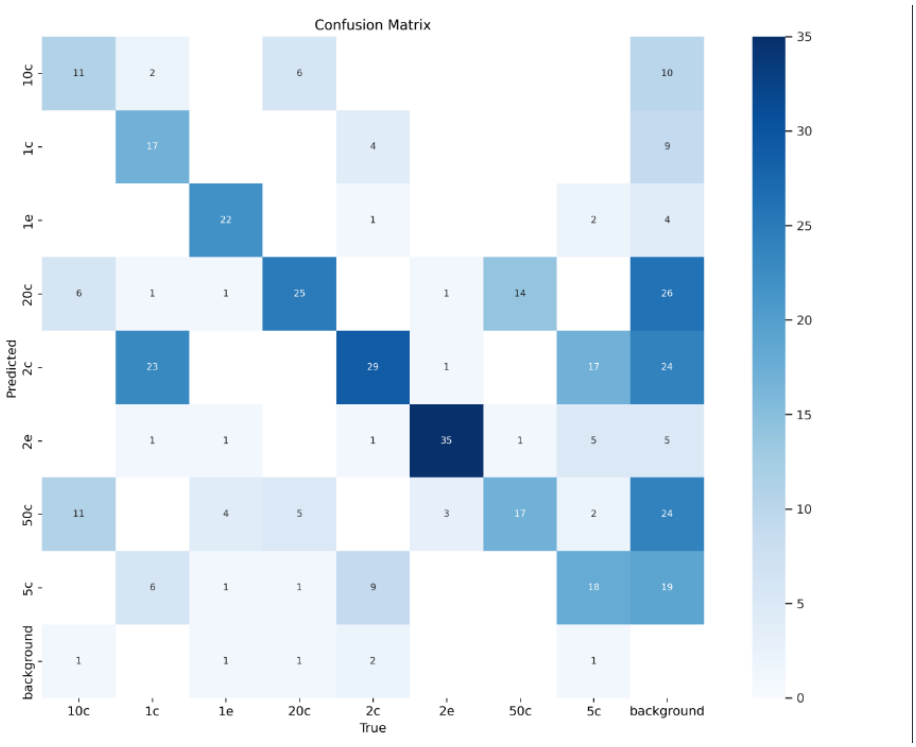
I grafici seguenti mostrano nella prima riga le rappresentazioni dei valori relativi al train mentre nella seconda riga le rappresentazioni relative al validation e alle metriche di performance.



- train/box_loss: perdita della localizzazione dei box durante il training. La perdita diminuisce rapidamente all'inizio per poi diminuire più lentamente. Questo indica che il modello sta imparando a localizzare i box in maniera più accurata.
- train/cls_loss: perdita di classificazione durante l'addestramento. La perdita diminuisce rapidamente, ciò significa che il modello sta migliorando nella classificazione degli oggetti rilevati.
- train/dfl_loss: perdita di distribuzione durante il train. La perdita diminuisce rapidamente per poi stabilizzarsi, il modello sta migliorando nella previsione della distribuzione degli oggetti.
- metrics/precision(B): precisione del modello durante il training. La precisione aumenta durante l'addestramento, il che significa che il modello diventa sempre più preciso nei suoi rilevamenti.
- metrics/recall(B): richiamo del modello durante il training. Il richiamo aumenta rapidamente all'inizio e per poi stabilizzarsi, indicando che il modello sta migliorando nella rilevazione degli oggetti presenti.
- val/box_loss: perdita della localizzazione dei box durante il validation. La perdita diminuisce rapidamente all'inizio per poi stabilizzarsi, rispetto al training i valori possono essere più alti a causa della differenza nei dati di validazione.
- val/cls_loss: perdita di classificazione durante la validazione. Andamento simile alla perdita del training.
- val/dfl_loss: perdita di distribuzione durante il validation. Come nel training diminuisce rapidamente per poi stabilizzarsi.

- metrics/mAP50(B): Media della precisione (mean Average Precision) al 50% di IoU. Aumenta con il procedere del training, mostrando che il modello sta migliorando la rilevazione corretta degli oggetti.
- metrics/mAP50-95(B): Media della precisione (mean Average Precision) su un intervallo di soglie di IoU (dal 50% al 95%). Andamento simile a metrics/mAP50(B).

Le perdite di training e validation diminuiscono nel tempo, questo indica che il modello sta imparando. Le metriche di precisione, richiamo e mAP aumentano, comporta che il modello migliora le sue previsioni.



Il grafico ci mostra come le predizioni del modello si confrontano con i dati reali.
L'asse delle ascisse rappresenta le classi reali mentre l'asse delle ordinate rappresenta le classi predette.
Le celle sulla diagonale principale rappresentano le classi in cui il modello ha fatto una previsione corretta, le celle fuori dalla diagonale principale rappresentano gli errori di classificazione.
Se il modello predice "background" per una determinata immagine, significa che non ha rilevato alcuna delle classi target (10c, 1c, 1e, 20c, 2c, 2e, 50c, 5c) in quella immagine.

- 10c: il modello ha predetto "background" per 6 immagini che erano realmente "10c".
- 1e: il modello ha predetto "background" per 6 immagini che erano realmente "1e".
- 20c: il modello ha predetto "background" per 4 immagini che erano realmente "20c".
- 2c: il modello ha predetto "background" per 3 immagini che erano realmente "2c".
- 2e: il modello ha predetto "background" per 8 immagini che erano realmente "2e".

4.1.1 Modello YOLOv8s

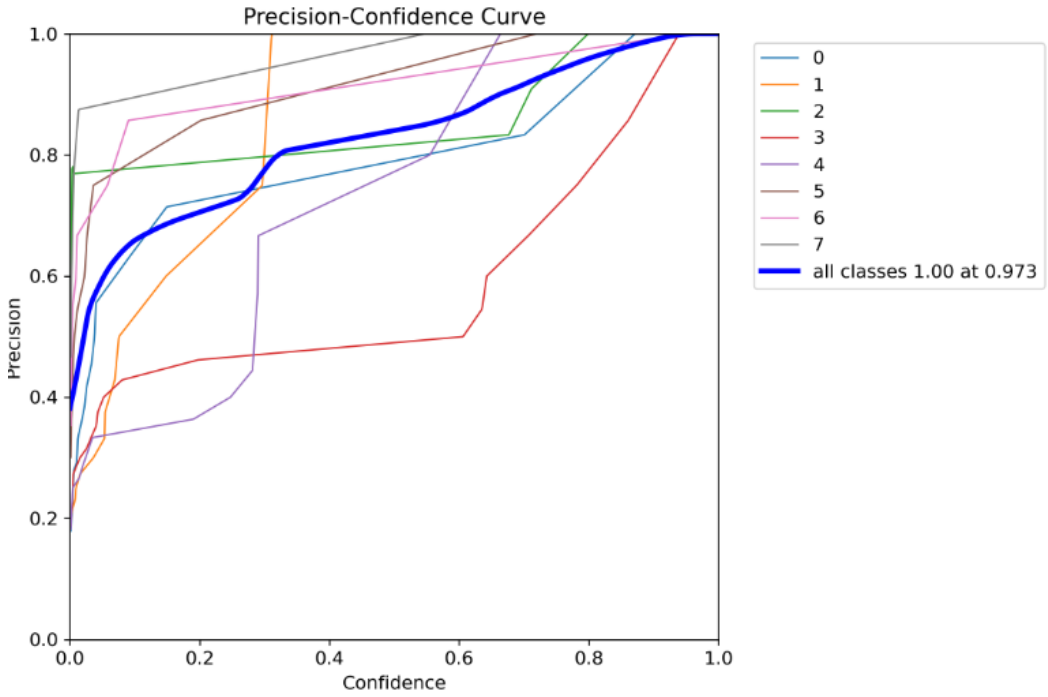
Il training effettuato utilizzando il modello YOLOv8s, ha dato dei discreti risultati, abbiamo usato 400 epoche.

```
400 epochs completed in 0.804 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 22.6MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 22.6MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics YOLOv8.2.28 🚀 Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8s summary (fused): 168 layers, 11128680 parameters, 0 gradients, 28.5 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:00<00:00,  2.29it/s]
                   all         48         48      0.903       0.93      0.993      0.912
                     0          5          5      0.826          1      0.995      0.807
                     1          3          3          1      0.587      0.995      0.995
                     2         11         11      0.832      0.909      0.976      0.925
                     3          6          6      0.624          1      0.995      0.838
                     4          4          4          1      0.981      0.995      0.939
                     5          6          6      0.985          1      0.995      0.903
                     6          6          6      0.953          1      0.995      0.933
                     7          7          7          1       0.96      0.995      0.955
Speed: 0.3ms preprocess, 4.5ms inference, 0.0ms loss, 1.9ms postprocess per image
Results saved to runs/detect/train2
```

Il modello è stato valutato tramite le metriche: Precision, Recall, Accuracy ed F1 Score, di seguito i valori ottenuti:

- Precision per classe: [0.82596 1 0.83224 0.62238 1 0.98501 0.95289 1]
- Recall per classe: [1 0.58637 0.90909 1 0.99488 1 1 0.95999]
- F1 score per classe: [0.90469 0.73926 0.86897 0.76724 0.99743 0.99245 0.97588 0.97959]
- Precision media: 0.9023
- Recall media: 0.9313
- F1 Score media: 0.9032
- Accuracy: 0.4583

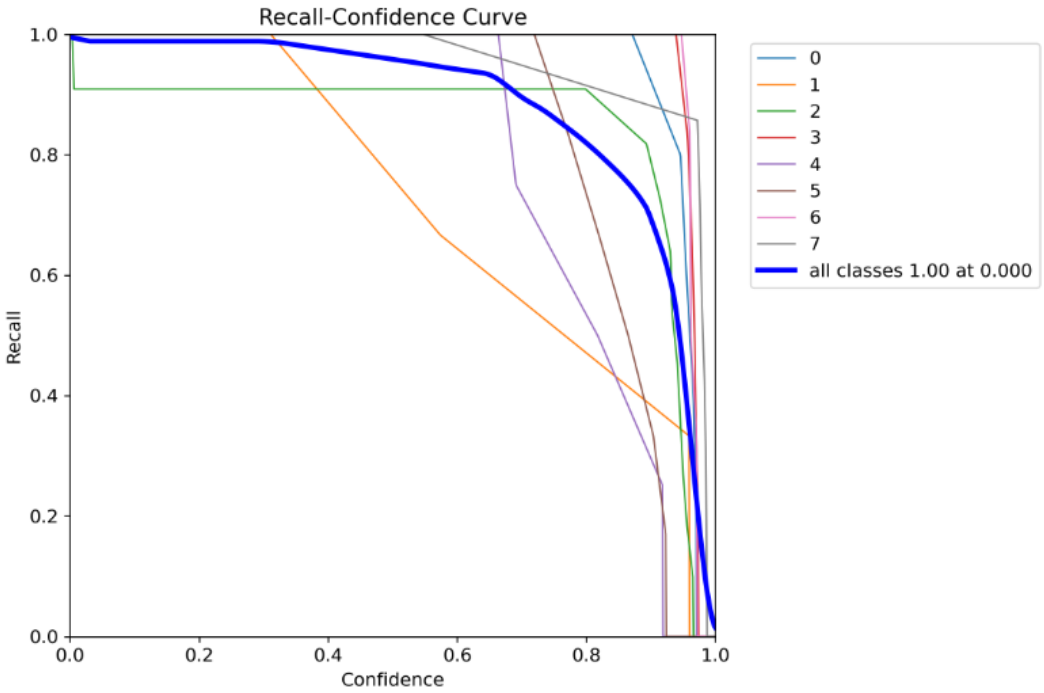
Il grafico seguente è una Precision-Confidence Curve per il modello di riconoscimento delle monete utilizzato.



Il grafico mostra come varia la precisione del modello al variare della soglia di confidenza, quando la confidenza è bassa il modello rileva molti oggetti, includendo molti falsi positivi, quindi la precisione è più bassa. Man mano che la confidenza aumenta, il modello diventa più selettivo, rilevando meno oggetti ma con maggiore precisione.

Le classi: 2e,1e,5c hanno una precisione molto alta su diversi valori di confidenza, questo indica che il modello ha una buona capacità di rilevare queste monete.

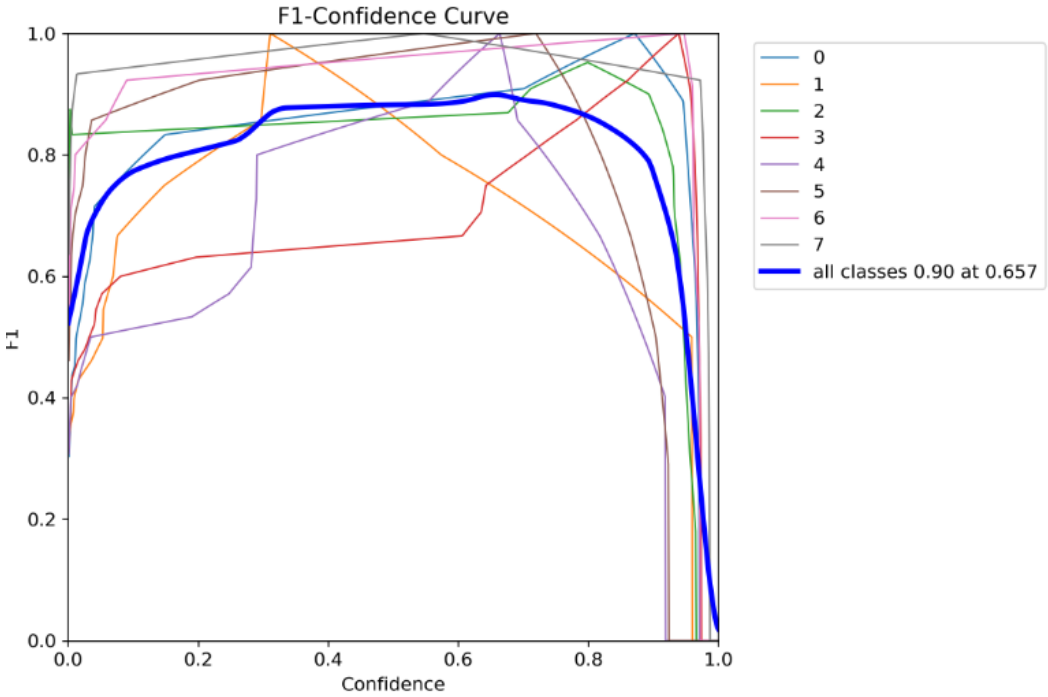
A seguire il grafico Recall-Confidence Curve del modello utilizzato.



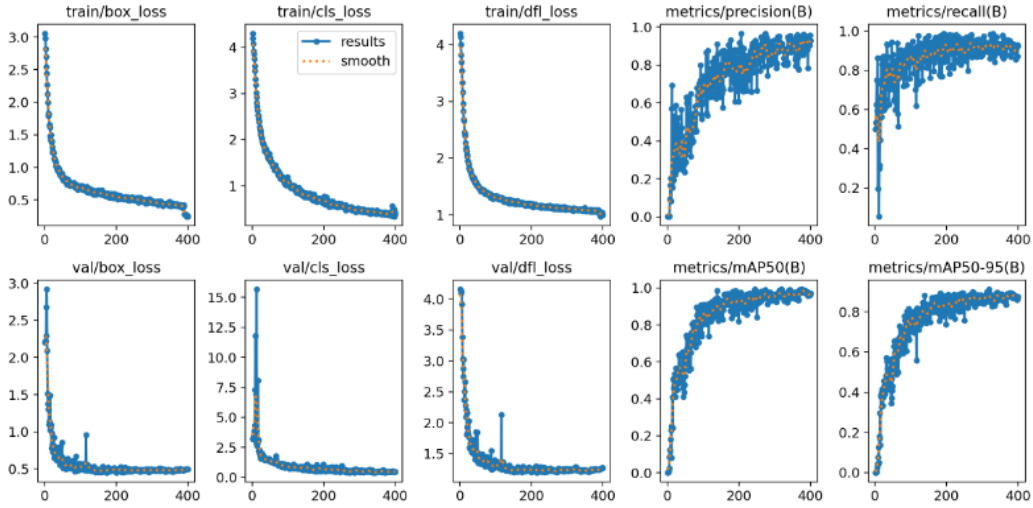
Nell'asse x vi è la soglia di confidenza, mentre nell'asse y il valore di Richiamo corrispondente alla soglia di confidenza x.

Si può notare che la Recall diminuisce all'aumentare della soglia di confidenza, aumentando la soglia di confidenza il modello diventa più selettivo, riducendo il numero di falsi positivi ma aumentando i falsi negativi.

Si mostra il grafico F1-Confidence Curve.

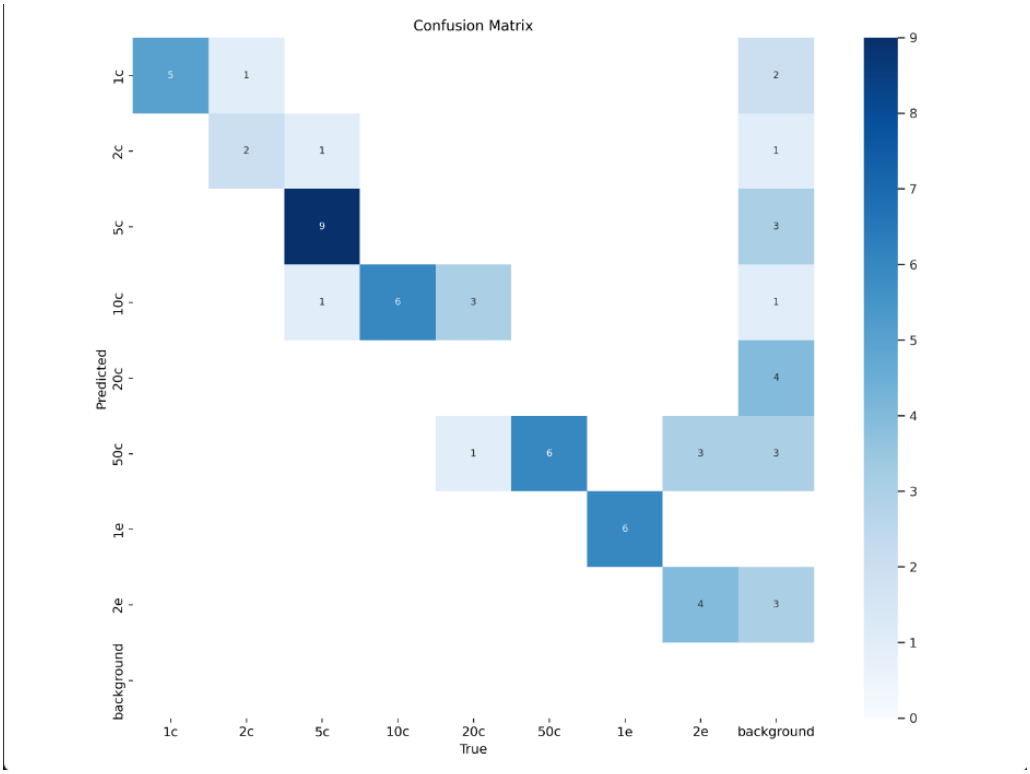


I grafici seguenti mostrano nella prima riga le rappresentazioni dei valori relativi al train mentre nella seconda riga le rappresentazioni relative al validation e alle metriche di performance.



- train/box_loss: perdita della localizzazione dei box durante il training. La perdita diminuisce rapidamente all'inizio per poi diminuire ad un ritmo più lento. Questo indica che il modello sta imparando a localizzare i box in maniera più accurata.
- train/cls_loss: perdita di classificazione durante l'addestramento. La perdita diminuisce rapidamente inizialmente, ciò significa che il modello sta migliorando nella classificazione degli oggetti rilevati.
- train/dfl_loss: perdita di distribuzione durante il train. La perdita diminuisce rapidamente e poi si stabilizza, questo indica che il modello sta migliorando nella previsione della distribuzione degli oggetti.
- metrics/precision(B): precisione del modello durante il training. La precisione aumenta man mano che l'addestramento procede, il che significa che il modello sta diventando più preciso nei suoi rilevamenti.
- metrics/recall(B): richiamo del modello durante il training. Il richiamo aumenta rapidamente all'inizio e poi si stabilizza, indicando che il modello sta migliorando nella rilevazione degli gli oggetti presenti.
- val/box_loss: perdita della localizzazione dei box durante il validation. La perdita diminuisce rapidamente all'inizio per poi stabilizzarsi, rispetto al training i valori possono essere più alti a causa della differenza nei dati di validazione.
- val/cls_loss: perdita di classificazione durante la validazione. Andamento simile alla perdita del training.
- val/dfl_loss: perdita di distribuzione durante il validation. Come nel training diminuisce rapidamente per poi stabilizzarsi.
- metrics/mAP50(B): Media della precisione (mean Average Precision) al 50% di IoU. Aumenta con il procedere del training, mostrando che il modello sta migliorando la rilevazione corretta degli oggetti.
- metrics/mAP50-95(B): Media della precisione (mean Average Precision) su un intervallo di soglie di IoU (dal 50% al 95%). Andamento simile a metrics/mAP50(B).

Le perdite di training e validation diminuiscono nel tempo, questo indica che il modello sta imparando. Le metriche di precisione, richiamo e mAP aumentano, comporta che il modello sta migliorando le sue previsioni.



Il grafico ci mostra come le predizioni del modello si confrontano con i dati reali.
L'asse delle ascisse rappresenta le classi reali mentre l'asse delle ordinate rappresenta le classi predette.
Le celle sulla diagonale principale rappresentano le classi in cui il modello ha fatto una previsione corretta, le celle fuori dalla diagonale principale rappresentano gli errori di classificazione.
Se il modello predice "background" per una determinata immagine, significa che non ha rilevato alcuna delle classi target (10c, 1c, 1e, 20c, 2c, 2e, 50c, 5c) in quella immagine.

- 1c: Ci sono 5 previsioni corrette e 1 classificazione errata come 2c.
- 2c: Ci sono 2 previsioni corrette e 1 classificazione errata come 1c.
- 5c: Ci sono 9 previsioni corrette e nessuna classificazione errata.
- 10c: Ci sono 6 previsioni corrette, 1 classificazione errata come 5c e 3 classificazioni errate come background.
- 20c: C'è 1 previsione corretta e diverse classificazioni errate in altre classi.

- 50c: Ci sono 6 previsioni corrette e classificazioni errate in altre classi.

- 1e: Ci sono 6 previsioni corrette e classificazioni errate in altre classi.

- 2e: Ci sono 3 previsioni corrette e classificazioni errate in altre classi.

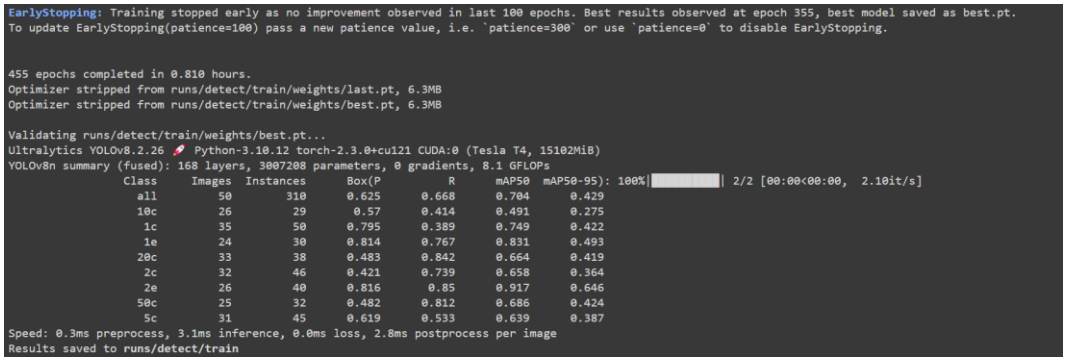
- background: Ci sono 4 previsioni corrette e diverse classificazioni errate in altre classi.

4.2 Dataset a moneta multipla

Si mostrano i risultati ottenuti effettuando il training col dataset in cui sono raffigurate più monete in un'immagine.

4.2.1 Modello YOLOv8n

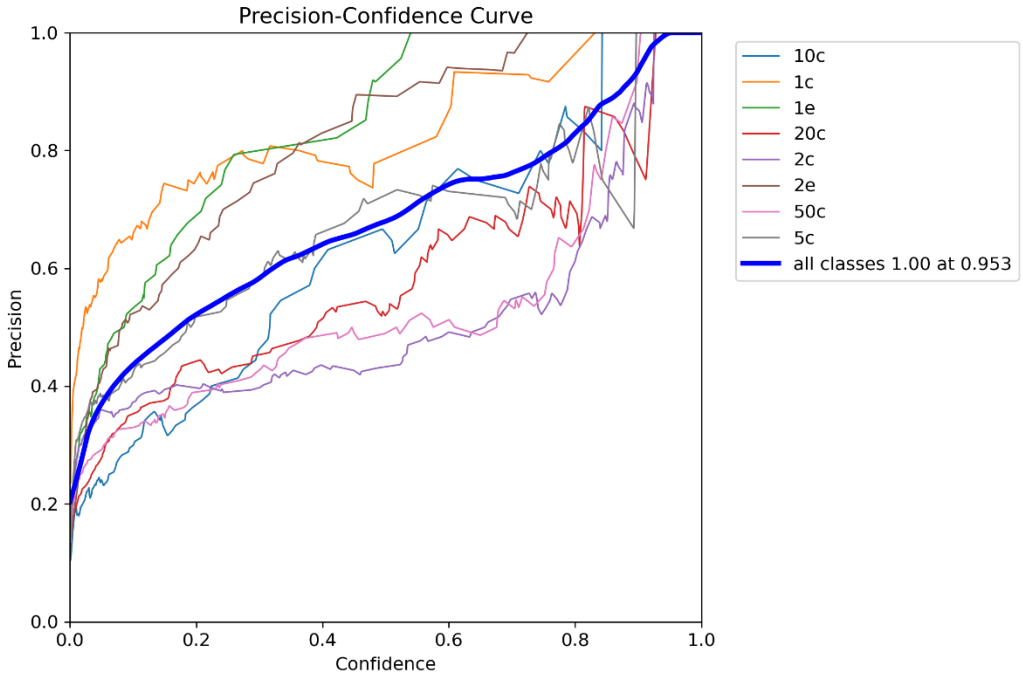
Il training effettuato utilizzando il modello YOLOv8n, ha dato dei discreti risultati, abbiamo usato 500 epoche ma il training si è fermato a 455 perché dall'epoca 355 non ci sono stati risultati migliori.



Il modello è stato valutato tramite le metriche: Precision, Recall, Accuracy ed F1 Score, di seguito i valori ottenuti:

- Precision per classe: [0.61646 0.78132 0.81726 0.51336 0.43565 0.82926 0.48609 0.66105]
- Recall per classe: [0.41379 0.3574 0.76667 0.84211 0.7384 0.85 0.8125 0.51111]
- F1 score per classe: [0.49519 0.49045 0.79116 0.63787 0.54799 0.8395 0.60827 0.57649]
- Precision media: 0.6426
- Recall media: 0.6615
- F1 Score media: 0.6234
- Accuracy: 0.3259

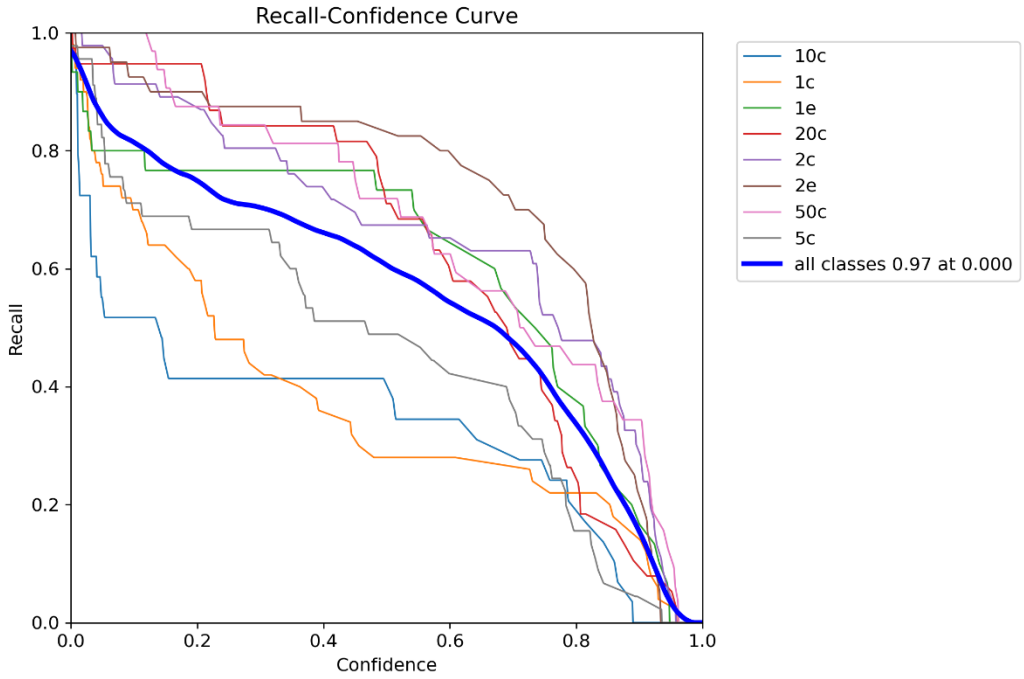
Il grafico seguente è una Precision-Confidence Curve per il modello di riconoscimento delle monete utilizzato.



Il grafico mostra come varia la precisione del modello al variare della soglia di confidenza, quando la confidenza è bassa il modello rileva molti oggetti, includendo molti falsi positivi, quindi la precisione è più bassa. Man mano che la confidenza aumenta, il modello diventa più selettivo, rilevando meno oggetti ma con maggiore precisione.

Le classi: 1e, 10c. 2e hanno una precisione molto alta su diversi valori di confidenza, questo indica che il modello ha una buona capacità di rilevare queste monete.

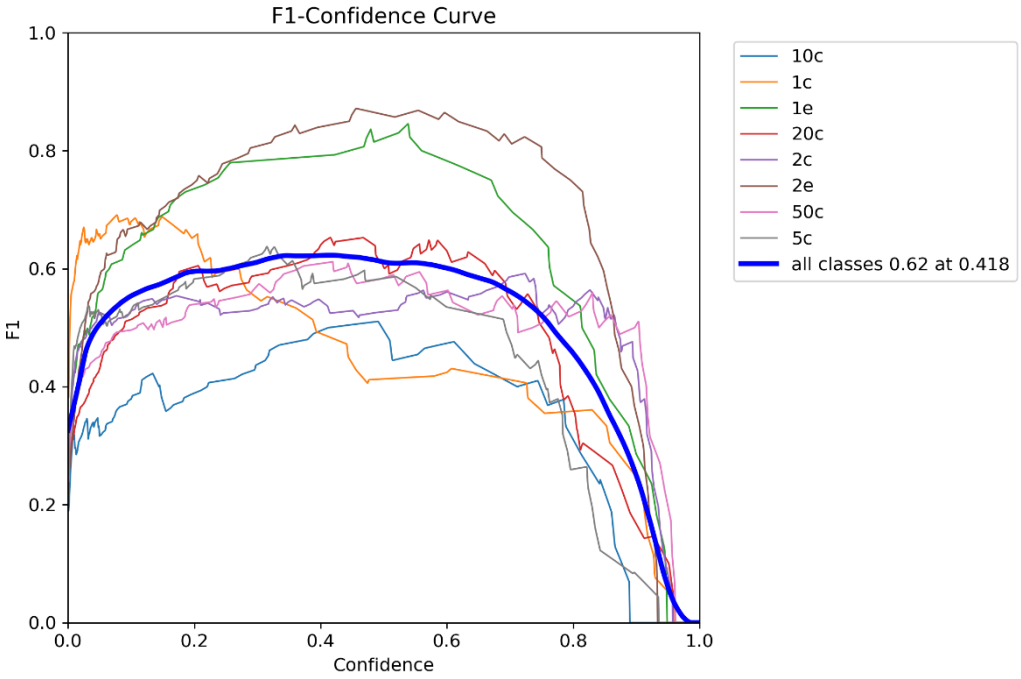
A seguire il grafico Recall-Confidence Curve del modello utilizzato.



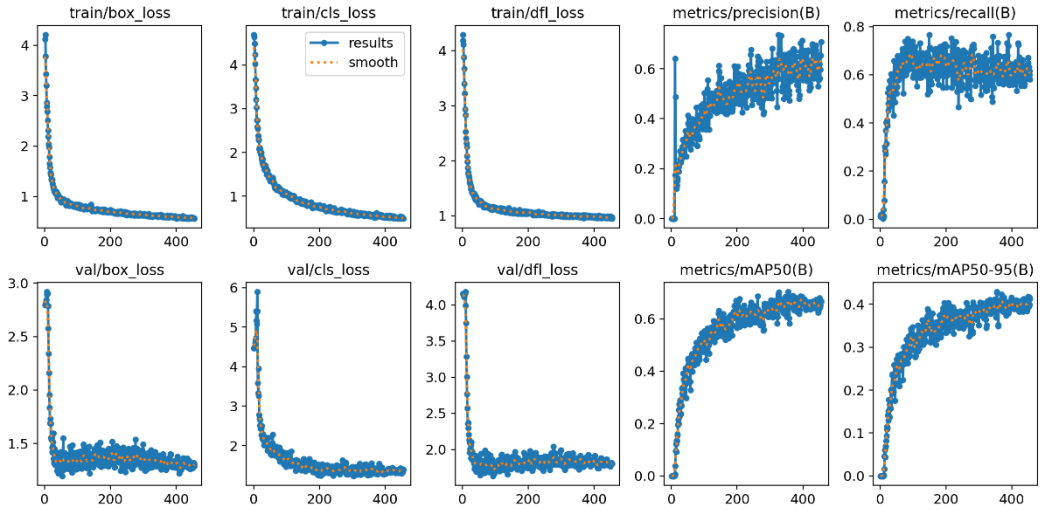
Nell'asse x vi è la soglia di confidenza, mentre nell'asse y il valore di Richiamo corrispondente alla soglia di confidenza x.

Si può notare che la Recall diminuisce all'aumentare della soglia di confidenza, aumentando la soglia di confidenza il modello diventa più selettivo, riducendo il numero di falsi positivi ma aumentando i falsi negativi.

Si mostra il grafico F1-Confidence Curve.



I grafici seguenti mostrano nella prima riga le rappresentazioni dei valori relativi al train mentre nella seconda riga le rappresentazioni relative al validation e alle metriche di performance.

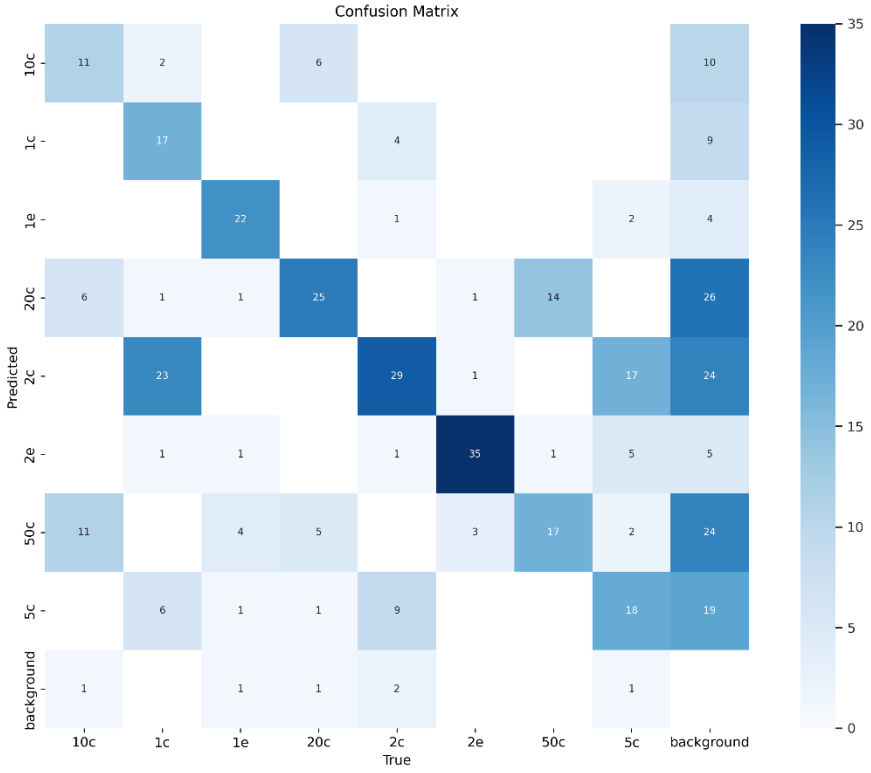


- train/box_loss: perdita della localizzazione dei box durante il training. La perdita diminuisce rapidamente all'inizio per poi diminuire ad un ritmo più lento. Questo indica che il modello sta imparando a localizzare i box in maniera più accurata.
- train/cls_loss: perdita di classificazione durante l'addestramento. La perdita diminuisce rapidamente inizialmente, ciò significa che il modello sta migliorando nella classificazione degli oggetti rilevati.
- train/dfl_loss: perdita di distribuzione durante il train. La perdita diminuisce rapidamente e poi si stabilizza, questo indica che il modello sta migliorando nella previsione della distribuzione degli oggetti.
- metrics/precision(B): precisione del modello durante il training. La precisione aumenta man mano che l'addestramento procede, il che significa che il modello sta diventando più preciso nei suoi rilevamenti.
- metrics/recall(B): richiamo del modello durante il training. Il richiamo aumenta rapidamente all'inizio e poi si stabilizza, indicando che il modello sta migliorando nella rilevazione degli gli oggetti presenti.
- val/box_loss: perdita della localizzazione dei box durante il validation. La perdita diminuisce rapidamente all'inizio per poi stabilizzarsi, rispetto al training i valori possono essere più alti a causa della differenza nei dati di validazione.
- val/cls_loss: perdita di classificazione durante la validazione. Andamento simile alla perdita del training.
- val/dfl_loss: perdita di distribuzione durante il validation. Come nel training diminuisce rapidamente per poi stabilizzarsi.
- metrics/mAP50(B): Media della precisione (mean Average Precision) al 50% di IoU. Aumenta con il procedere del training, mostrando che il modello sta migliorando la rilevazione corretta degli oggetti.
- metrics/mAP50-95(B): Media della precisione (mean Average Precision) su un intervallo di soglie di IoU (dal 50% al 95%). Andamento simile a metrics/mAP50(B).

Le perdite di training e validation diminuiscono nel tempo, questo indica che il modello sta imparando.

Le metriche di precisione, richiamo e mAP aumentano, comporta che il modello sta migliorando le sue previsioni. A fine training, le perdite e le metriche tendono a stabilizzarsi, indicando che il modello raggiunge un punto di convergenza.

Si mostra la Confusion Matrix



Se il modello predice "background" per una determinata immagine, significa che non ha rilevato alcuna delle classi target (10c, 1c, 1e, 20c, 2c, 2e, 50c, 5c) in quella immagine.

- 10c: il modello ha predetto correttamente “10c” per 11 immagini che sono realmente “10c”.
- 1c: il modello ha predetto correttamente “1c” per 17 immagini che sono realmente “1c”.
- 1e: il modello ha predetto correttamente “1e” per 22 immagini che sono realmente “1e”.
- 20c: il modello ha predetto correttamente “20c” per 25 immagini che sono realmente “20c”.
- 2c: il modello ha predetto correttamente “2c” per 29 immagini che sono realmente “2c”.

- 2e: il modello ha predetto correttamente “2e” per 35 immagini che sono realmente “2e”.
- 50c: il modello ha predetto correttamente “50c” per 17 immagini che sono realmente “50c”.
- 5c: il modello ha predetto correttamente “5c” per 18 immagini che sono realmente “5c”.

4.2.2 Modello YOLOv8s

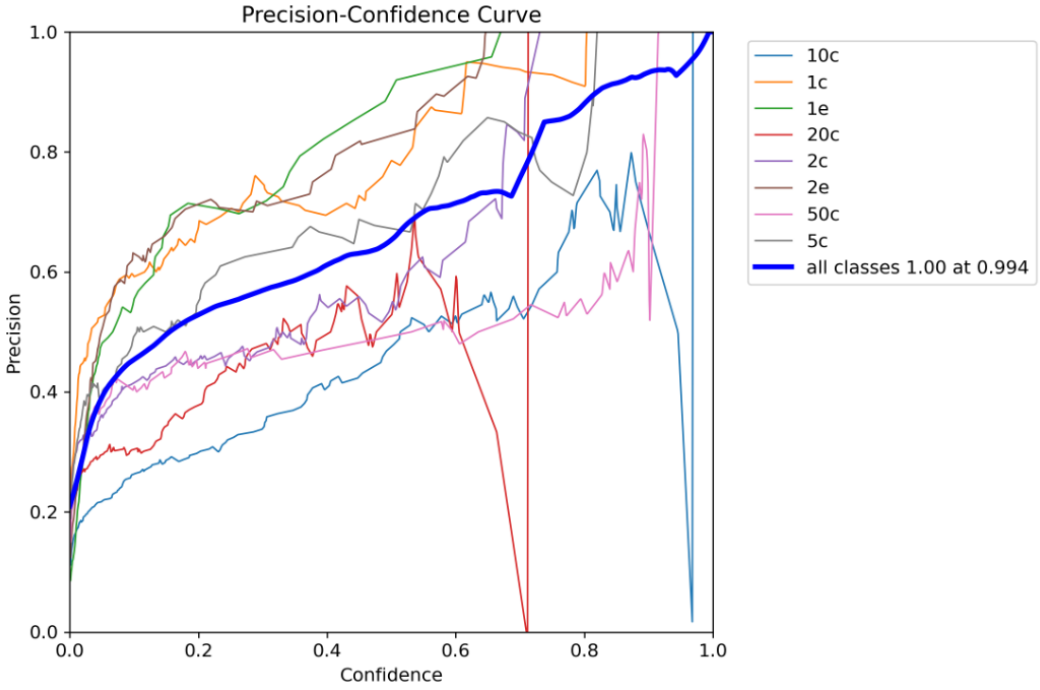
Il training effettuato utilizzando il modello YOLOv8s, ha dato dei buoni risultati, abbiamo usato 500 epoche ma il training si è fermato a 310 perché dall'epoca 210 non ci sono stati risultati migliori.

```
EarlyStopping: Training stopped early as no improvement observed in last 100 epochs. Best results observed at epoch 210, best model saved as best.pt.
To update EarlyStopping(patience=100) pass a new patience value, i.e. `patience=300` or use `patience=0` to disable EarlyStopping.


310 epochs completed in 0.663 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 22.6MB
Optimizer stripped from runs/detect/train/weights/best.pt, 22.5MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.2.26 🚀 Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8s summary (fused): 168 layers, 11128680 parameters, 0 gradients, 28.5 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:01<00:00,  1.15it/s]
                   all         50        310      0.551      0.687      0.672      0.417
                   10c         26         29      0.323      0.828      0.582      0.338
                    1c         35         50      0.706       0.72      0.788      0.452
                    1e         24         30        0.7      0.778      0.844      0.544
                   20c         33         38      0.438      0.697        0.5      0.331
                    2c         32         46       0.46      0.674      0.603      0.308
                    2e         26         40      0.705      0.716       0.86      0.631
                   50c         25         32      0.462      0.531      0.542      0.319
                    5c         31         45      0.617      0.556      0.653      0.409
Speed: 0.4ms preprocess, 5.9ms inference, 0.0ms loss, 4.2ms postprocess per image
Results saved to runs/detect/train
```

Il modello è stato valutato tramite le metriche: Precision, Recall, Accuracy ed F1 Score, di seguito i valori ottenuti:

- Precision per classe: [0.82596 1 0.83224 0.62238 1 0.98501 0.95289 1]
- Recall per classe: [1 0.58637 0.90909 1 0.99487 1 1 0.95999]
- F1 score per classe: [0.90469 0.73926 0.86897 0.76724 0.99743 0.99245 0.97588 0.97959]
- Mean Precision: 0.9023
- Mean Recall: 0.9313
- Mean F1 Score: 0.9032
- Accuracy: 0.4583

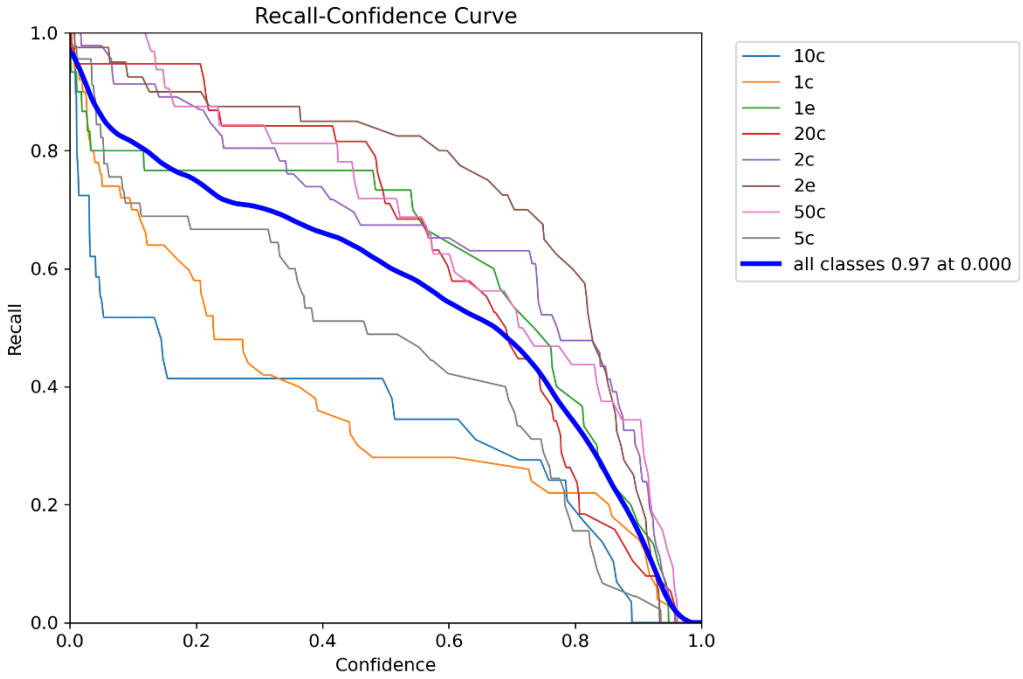
Il grafico seguente è una Precision-Confidence Curve per il modello di riconoscimento delle monete utilizzato.



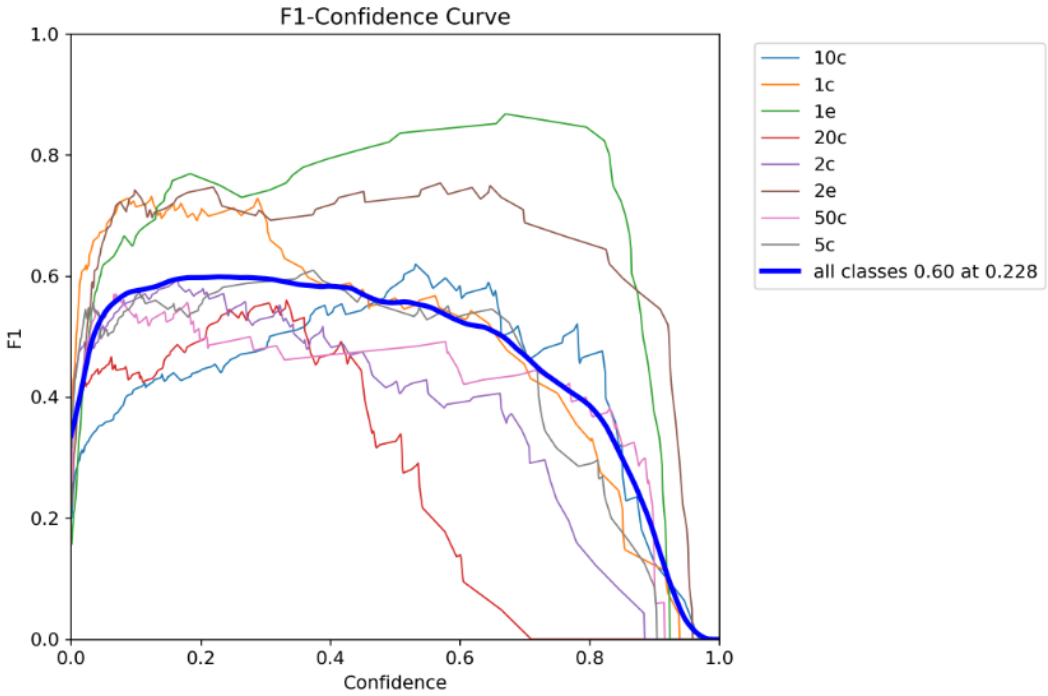
Il grafico mostra come varia la precisione del modello al variare della soglia di confidenza, quando la confidenza è bassa il modello rileva molti oggetti, includendo molti falsi positivi, quindi la precisione è più bassa. Man mano che la confidenza aumenta, il modello diventa più selettivo, rilevando meno oggetti ma con maggiore precisione.

Le classi: 1e, 1c. 20c hanno una precisione molto alta su diversi valori di confidenza, questo indica che il modello ha una buona capacità di rilevare queste monete.

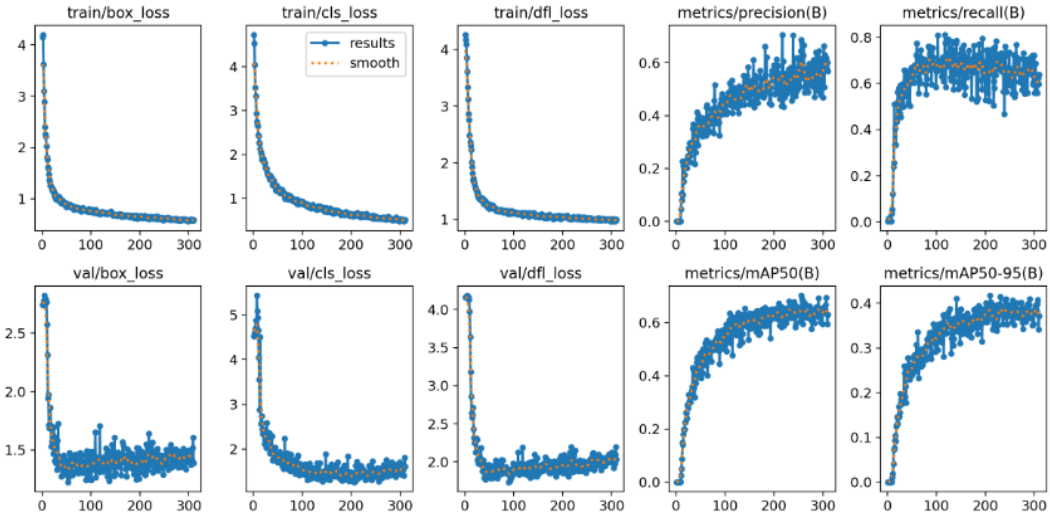
A seguire il grafico Recall-Confidence Curve del modello utilizzato.



Si può notare che la Recall diminuisce all'aumentare della soglia di confidenza, aumentando la soglia di confidenza il modello diventa più selettivo, riducendo il numero di falsi positivi ma aumentando i falsi negativi.



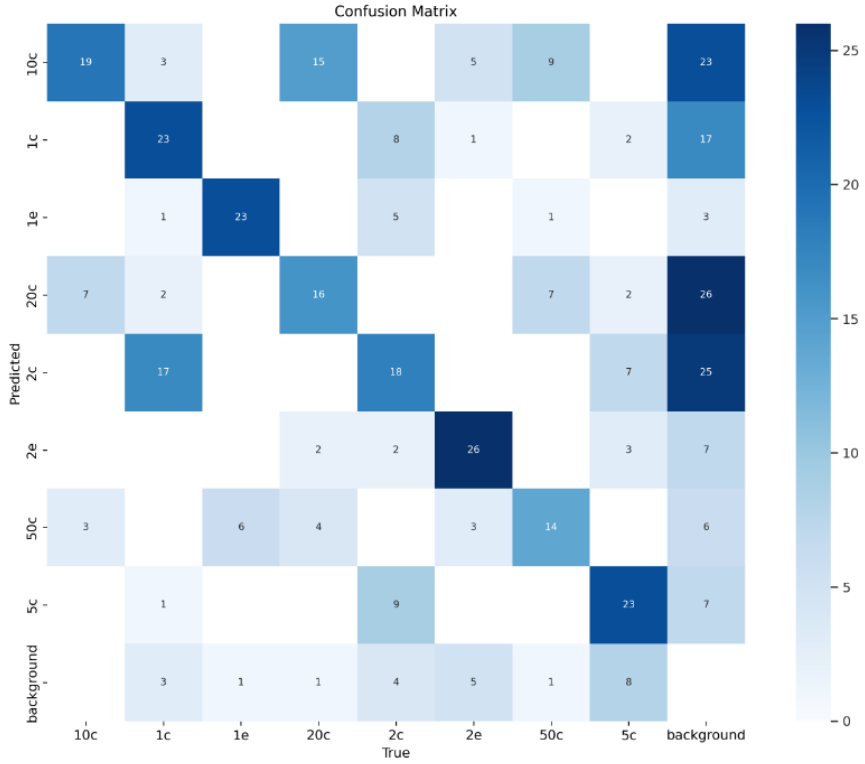
I grafici seguenti mostrano nella prima riga le rappresentazioni dei valori relativi al train mentre nella seconda riga le rappresentazioni relative al validation e alle metriche di performance.



- train/box_loss: perdita della localizzazione dei box durante il training. La perdita diminuisce rapidamente all'inizio per poi diminuire ad un ritmo più lento. Questo indica che il modello sta imparando a localizzare i box in maniera più accurata.

- train/cls_loss: perdita di classificazione durante l'addestramento. La perdita diminuisce rapidamente inizialmente, ciò significa che il modello sta migliorando nella classificazione degli oggetti rilevati.
- train/dfl_loss: perdita di distribuzione durante il train. La perdita diminuisce rapidamente e poi si stabilizza, questo indica che il modello sta migliorando nella previsione della distribuzione degli oggetti.
- metrics/precision(B): precisione del modello durante il training. La precisione aumenta man mano che l'addestramento procede, il che significa che il modello sta diventando più preciso nei suoi rilevamenti.
- metrics/recall(B): richiamo del modello durante il training. Il richiamo aumenta rapidamente all'inizio e poi si stabilizza, indicando che il modello sta migliorando nella rilevazione degli gli oggetti presenti.
- val/box_loss: perdita della localizzazione dei box durante il validation. La perdita diminuisce rapidamente all'inizio per poi stabilizzarsi, rispetto al training i valori possono essere più alti a causa della differenza nei dati di validazione.
- val/cls_loss: perdita di classificazione durante la validazione. Andamento simile alla perdita del training.
- val/dfl_loss: perdita di distribuzione durante il validation. Come nel training diminuisce rapidamente per poi stabilizzarsi.
- metrics/mAP50(B): Media della precisione (mean Average Precision) al 50% di IoU. Aumenta con il procedere del training, mostrando che il modello sta migliorando la rilevazione corretta degli oggetti.
- metrics/mAP50-95(B): Media della precisione (mean Average Precision) su un intervallo di soglie di IoU (dal 50% al 95%). Andamento simile a metrics/mAP50(B).

Le perdite di training e validation diminuiscono nel tempo, questo indica che il modello sta imparando. Le metriche di precisione, richiamo e mAP aumentano, comporta che il modello sta migliorando le sue previsioni. A fine training, le perdite e le metriche tendono a stabilizzarsi, indicando che il modello raggiunge un punto di convergenza.



L'asse x rappresenta le etichette reali (etichette True).
L'asse y rappresenta le etichette previste (etichette Predicted).
La diagonale dall'angolo in alto a sinistra all'angolo in basso a destra mostra il numero di previsioni corrette per ogni classe.
Gli elementi fuori dalla diagonale rappresentano le classificazioni errate.
Se il modello predice "background" per una determinata immagine, significa che non ha rilevato alcuna delle classi target (10c, 1c, 1e, 20c, 2c, 2e, 50c, 5c) in quella immagine.

- 10c: Ci sono 19 previsioni corrette e 3 classificazioni errate come 1c, 15 come 1e, 5 come 20c, 9 come 2e, e 23 come background.

- 1c: Ci sono 23 previsioni corrette, 8 classificazioni errate come 20c, 1 come 2c, 2 come 50c, e 17 come background.

- 1e: Ci sono 23 previsioni corrette, 1 classificazione errata come 1c, 5 come 20c, 1 come 2c, e 3 come background.

- 20c: Ci sono 16 previsioni corrette, 7 classificazioni errate come 10c, 2 come 2e, 7 come 50c, 2 come 5c, e 26 come background.

- 2c: Ci sono 18 previsioni corrette, 17 classificazioni errate come 10c, 7 come 20c, 7 come 2e, e 25 come background.

- 2e: Ci sono 26 previsioni corrette, 2 classificazioni errate come 20c, 2 come 2c, 3 come 50c, e 7 come background.

- 50c: Ci sono 14 previsioni corrette, 6 classificazioni errate come 20c, 3 come 2c, e 6 come background.

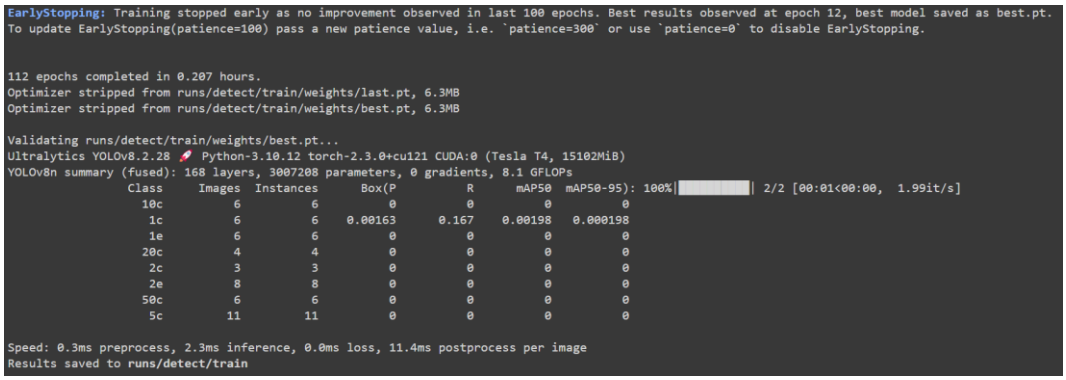
- 5c: Ci sono 23 previsioni corrette, 1 classificazione errata come 2c, 9 come 5c, e 7 come background.

- Per background: Ci sono 8 previsioni corrette, 3 classificazioni errate come 10c, 1 come 1c, 1 come 1e, 4 come 20c, e 5 come 2e.

4.3 Training su Dataset misto

4.3.1 Training moneta multipla, val e test a singola moneta

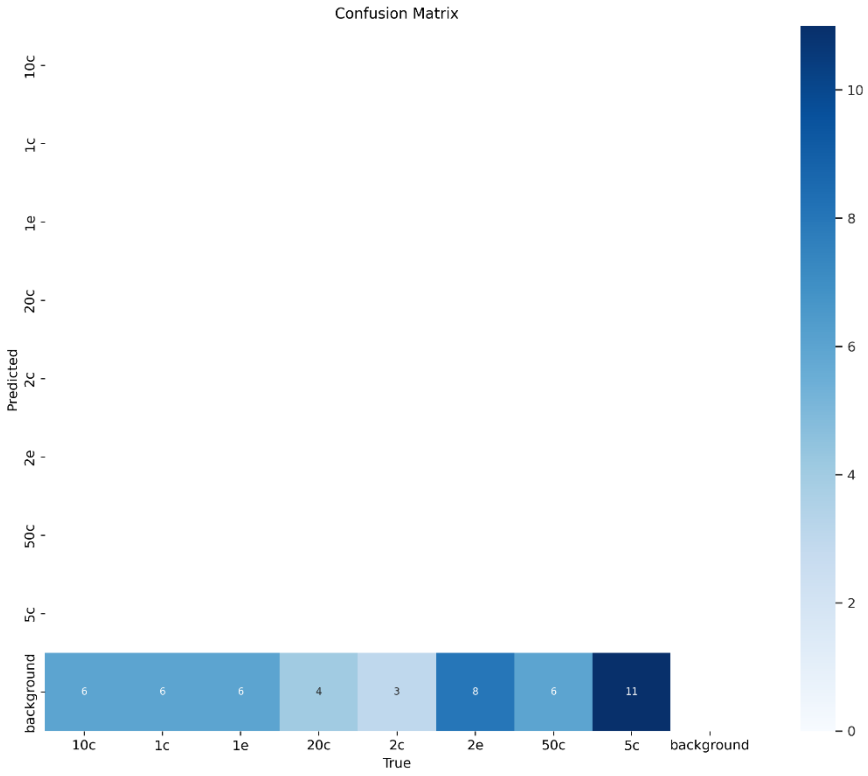
Il training effettuato utilizzando il modello YOLOv8s, ha dato dei pessimi risultati, sono state usate 500 epoche ma il training si è fermato a 112 perché dall'epoca 12 non ci sono stati risultati migliori.



Il modello è stato valutato tramite le metriche: Precision, Recall, Accuracy ed F1 Score, di seguito i valori ottenuti:

- Precision per classe: [0 0.00163 0 0 0 0 0 0]
- Recall per classe: [0 0.167 0 0 0 0 0 0]
- F1 score per classe: [0 0.00198 0 0 0 0 0 0]
- Precision media: 0
- Recall media: 0
- F1 Score media: 0
- Accuracy: 0

Il grafico seguente è una Confusion Matrix



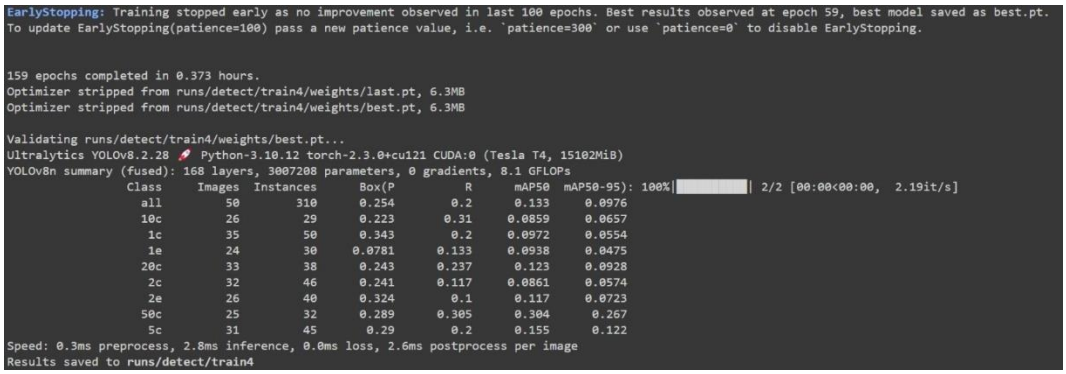
Se il modello predice "background" per una determinata immagine, significa che non ha rilevato alcuna delle classi target (10c, 1c, 1e, 20c, 2c, 2e, 50c, 5c) in quella immagine.

- 10c: il modello ha predetto "background" per 6 immagini che erano realmente "10c".
- 1c: il modello ha predetto "background" per 6 immagini che erano realmente "1c".
- 1e: il modello ha predetto "background" per 6 immagini che erano realmente "1e".
- 20c: il modello ha predetto "background" per 4 immagini che erano realmente "20c".
- 2c: il modello ha predetto "background" per 3 immagini che erano realmente "2c".
- 2e: il modello ha predetto "background" per 8 immagini che erano realmente "2e".
- 50c: il modello ha predetto "background" per 6 immagini che erano realmente "50c".
- 5c: il modello ha predetto "background" per 11 immagini che erano realmente "5c".

Gli esperimenti condotti indicano che il modello non produce risultati accettabili. Pertanto, possiamo concludere che è preferibile evitare l'uso di un dataset di training con immagini raffiguranti più oggetti da identificare e utilizzare invece dataset di validation e testing con immagini contenenti un solo oggetto.

4.3.2 Training moneta singola, val e test a moneta multipla

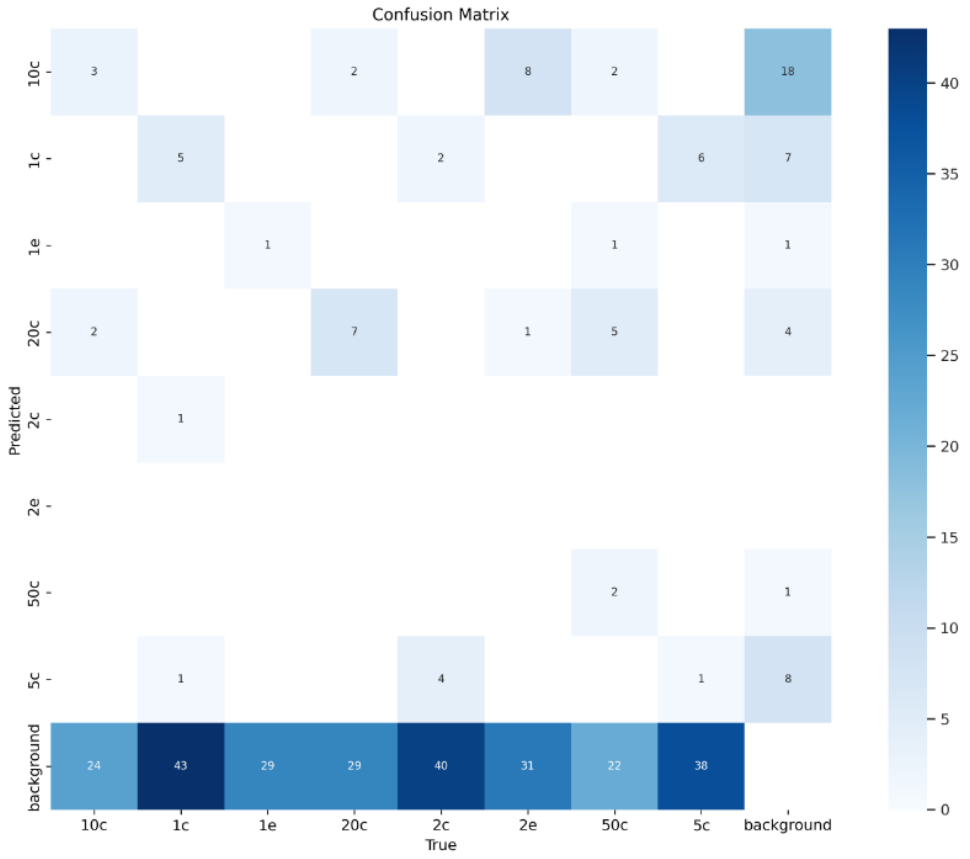
Il training effettuato utilizzando il modello YOLOv8s, sono state usate 500 epoche ma il training si è fermato a 159 perché dall'epoca 59 non ci sono stati risultati migliori.



Il modello è stato valutato tramite le metriche: Precision, Recall, Accuracy ed F1 Score, di seguito i valori ottenuti:

- Precision per classe: [0.22339 0.34336 0.078154 0.24299 0.23977 0.32216 0.28743 0.29002]
- Recall per classe: [0.31034 0.2 0.13333 0.23684 0.11678 0.1 0.30273 0.2]
- F1 score per classe: [0.25979 0.25277 0.098545 0.23988 0.15706 0.15262 0.29489 0.23674]
- Mean Precision: 0.2534
- Mean Recall: 0.2000
- Mean F1 Score: 0.2115
- Accuracy: 0.1118

Il grafico seguente è una Confusion Matrix



La matrice di confusione mostrata nell'immagine rappresenta le prestazioni di un modello di classificazione
Se il modello predice "background" per una determinata immagine, significa che non ha rilevato alcuna delle classi target (10c, 1c, 1e, 20c, 2c, 2e, 50c, 5c) in quella immagine.

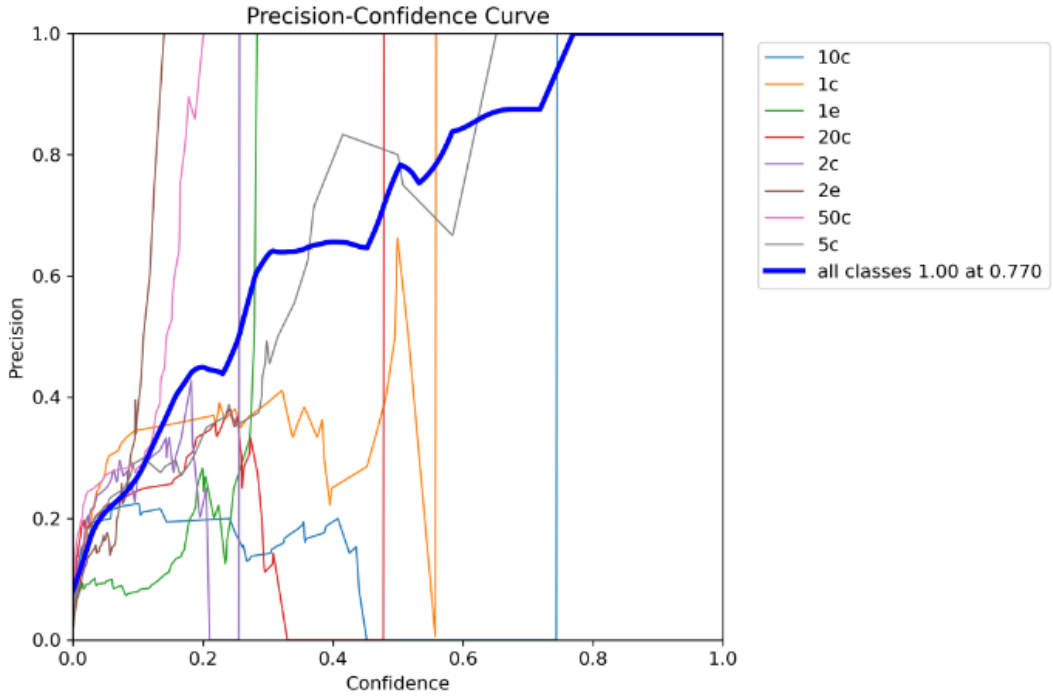
- 10c: Classi vere di 10c sono state classificate come 10c 3 volte, come 1c 2 volte, come 20c 2 volte, come 2c 8 volte, come 50c 2 volte e come background 18 volte.
- 1c: Classi vere di 1c sono state classificate come 1c 5 volte, come 20c 1 volta, come 2c 2 volte, come 5c 6 volte e come background 7 volte.
- 1e: Classe vera di 1e è stata classificata come 1e 1 volta, come 5c 1 volta e come background 1 volta.
- 20c: Classi vere di 20c sono state classificate come 20c 7 volte, come 2c 1 volta, come 5c 5 volte e come background 4 volte.
- 2c: Classe vera di 2c è stata classificata come 2c 1 volta.

- 2e: Classe vera di 2e è stata classificata come 2e 2 volte e come background 1 volta.
- 50c: Classe vera di 50c è stata classificata come 5c 1 volta e come background 8 volte.
- 5c: Classe vera di 5c è stata classificata come 2c 4 volte e come background 1 volta.

Gli esperimenti condotti indicano che il modello non produce risultati accettabili. Pertanto, possiamo concludere che è preferibile evitare l'uso di un dataset di training con immagini raffiguranti più oggetti da identificare e utilizzare invece dataset di validation e testing con immagini contenenti un solo oggetto.

Il background è stato classificato come varie classi, con numeri significativi nelle colonne 1c (43), 2c (40), 5c (38), 10c (24), 20c (29), 50c (22) e 2e (31).

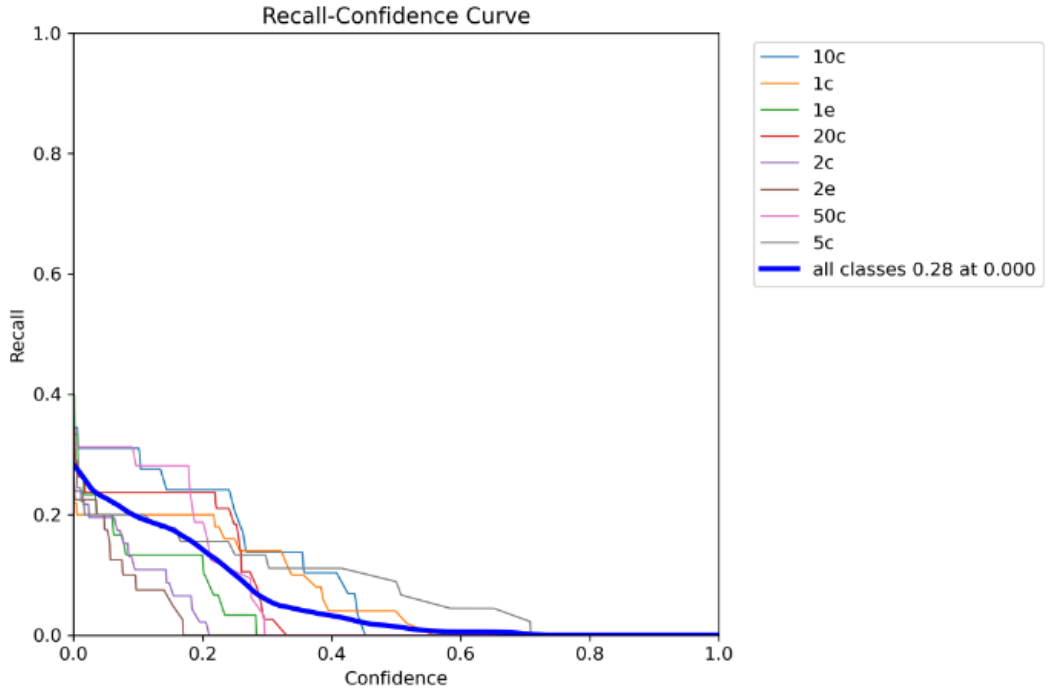
Il grafico seguente è una Precision-Confidence Curve per il modello di riconoscimento delle monete utilizzato.



Il grafico mostra come varia la precisione del modello al variare della soglia di confidenza, quando la confidenza è bassa il modello rileva molti oggetti, includendo molti falsi positivi, quindi la precisione è più bassa. Man mano che la confidenza aumenta, il modello diventa più selettivo, rilevando meno oggetti ma con maggiore precisione.

Le classi: 50c, 20 e 10c hanno una precisione molto alta su diversi valori di confidenza, questo indica che il modello ha una buona capacità di rilevare queste monete.

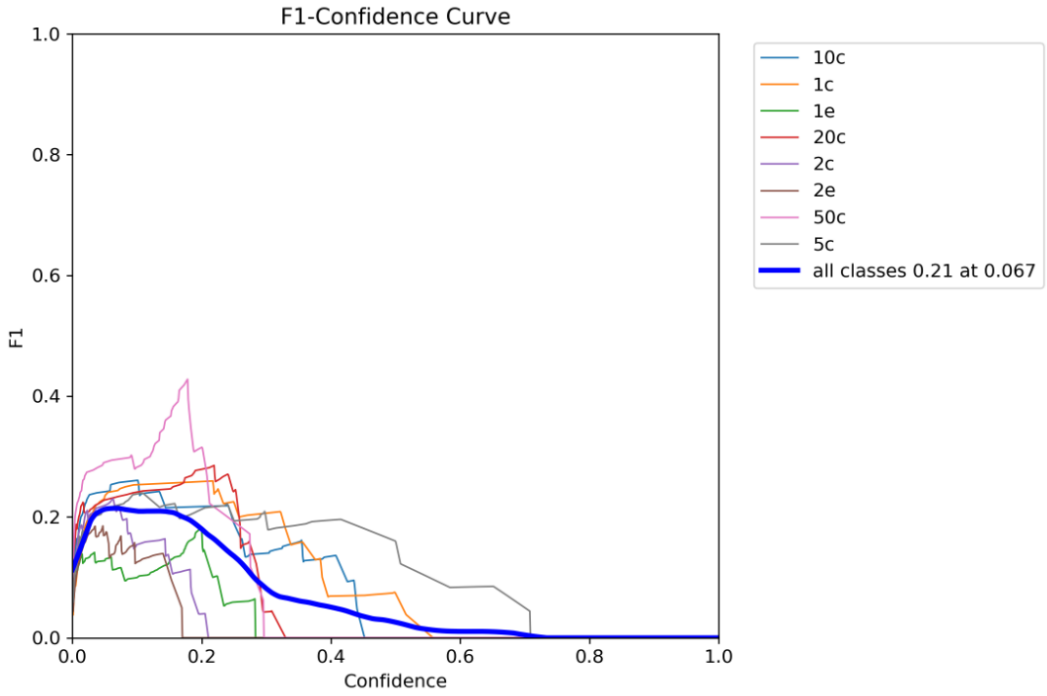
A seguire il grafico Recall-Confidence Curve del modello utilizzato.



Si può notare che la Recall diminuisce all'aumentare della soglia di confidenza, aumentando la soglia di confidenza il modello diventa più selettivo, riducendo il numero di falsi positivi ma aumentando i falsi negativi. Le tre migliori classi in termini di richiamo, basandoci sulle osservazioni delle curve, sono:

- 50c : Mostra un richiamo alto nei primi livelli di confidenza.
- 1e : Dimostra un buon richiamo iniziale;
- 1c : Mantiene un buon richiamo nei primi livelli di confidenza.

Queste tre classi sembrano avere le performance migliori in termini di richiamo rispetto alle altre.

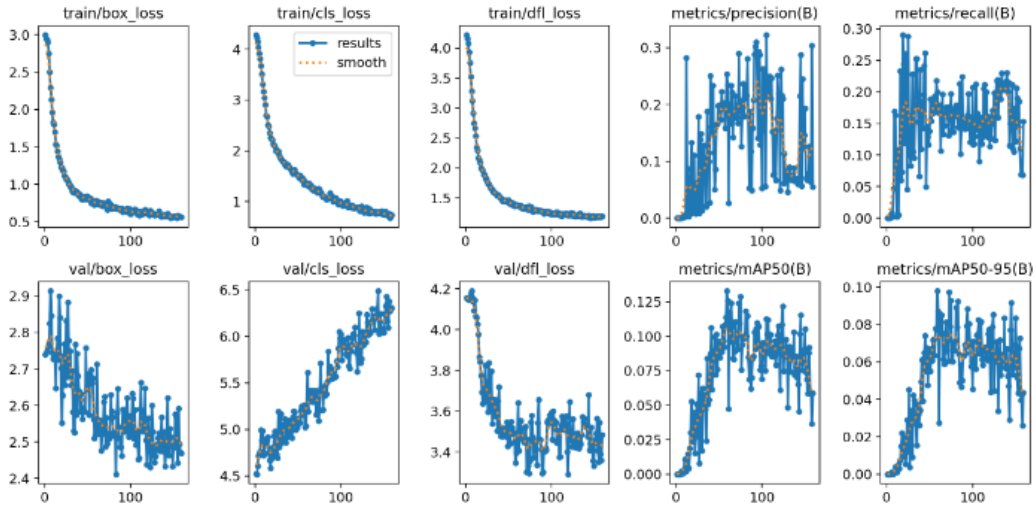


Le tre migliori classi in termini di F1-score, basandoci sulle osservazioni delle curve, sono:

- 5c: Mostra un F1-score alto nei primi livelli di confidenza.
- 20c : Dimostra un buon F1-score iniziale.
- 1c : Mantiene un buon F1-score nei primi livelli di confidenza.

Queste tre classi sembrano avere le performance migliori in termini di F1-score rispetto alle altre.

I grafici seguenti mostrano nella prima riga le rappresentazioni dei valori relativi al train mentre nella seconda riga le rappresentazioni relative al validation e alle metriche di performance.



- train/box_loss: perdita della localizzazione dei box durante il training. La perdita diminuisce rapidamente all'inizio per poi diminuire ad un ritmo più lento. Questo indica che il modello sta imparando a localizzare i box in maniera più accurata.
- train/cls_loss: perdita di classificazione durante l'addestramento. La perdita diminuisce rapidamente inizialmente, ciò significa che il modello sta migliorando nella classificazione degli oggetti rilevati.
- train/dfl_loss: perdita di distribuzione durante il train. La perdita diminuisce rapidamente e poi si stabilizza, questo indica che il modello sta migliorando nella previsione della distribuzione degli oggetti.
- metrics/precision(B): precisione del modello durante il training. La precisione aumenta man mano che l'addestramento procede, il che significa che il modello sta diventando più preciso nei suoi rilevamenti.
- metrics/recall(B): richiamo del modello durante il training. Il richiamo aumenta rapidamente all'inizio e poi si stabilizza, indicando che il modello sta migliorando nella rilevazione degli gli oggetti presenti.
- val/box_loss: perdita della localizzazione dei box durante il validation. La perdita diminuisce rapidamente all'inizio per poi stabilizzarsi, rispetto al training i valori possono essere più alti a causa della differenza nei dati di validazione.

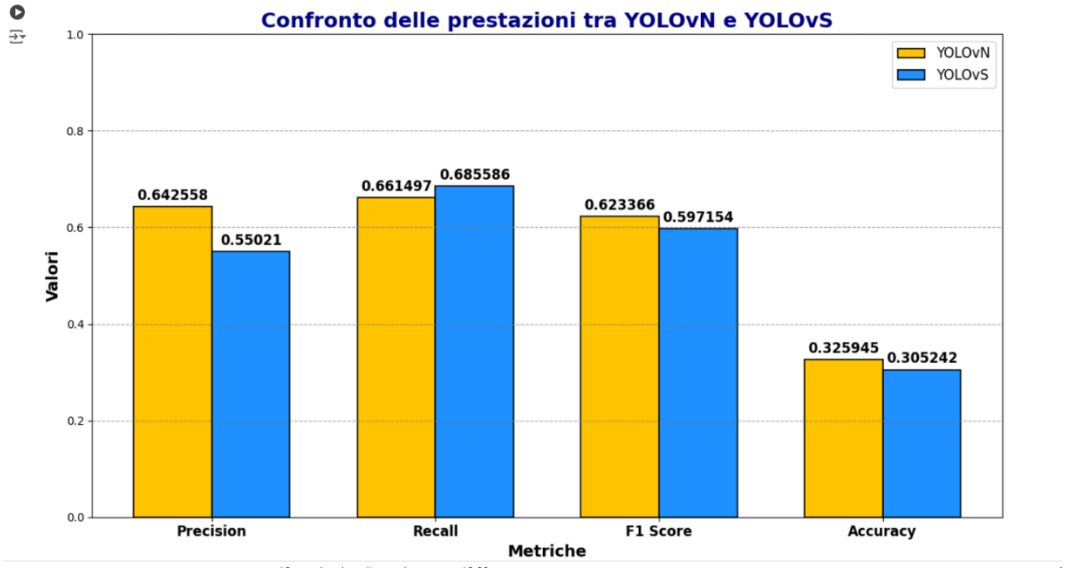
- val/cls_loss: perdita di classificazione durante la validazione. Andamento simile alla perdita del training.
- val/dfl_loss: perdita di distribuzione durante il validation. Come nel training diminuisce rapidamente per poi stabilizzarsi.
- metrics/mAP50(B): Media della precisione (mean Average Precision) al 50% di IoU. Aumenta con il procedere del training, mostrando che il modello sta migliorando la rilevazione corretta degli oggetti.
- metrics/mAP50-95(B): Media della precisione (mean Average Precision) su un intervallo di soglie di IoU (dal 50% al 95%). Andamento simile a metrics/mAP50(B).

Le perdite di training e validation diminuiscono nel tempo, questo indica che il modello sta imparando. Le metriche di precisione, richiamo e mAP aumentano, comporta che il modello sta migliorando le sue previsioni. A fine training, le perdite e le metriche tendono a stabilizzarsi, indicando che il modello raggiunge un punto di convergenza.

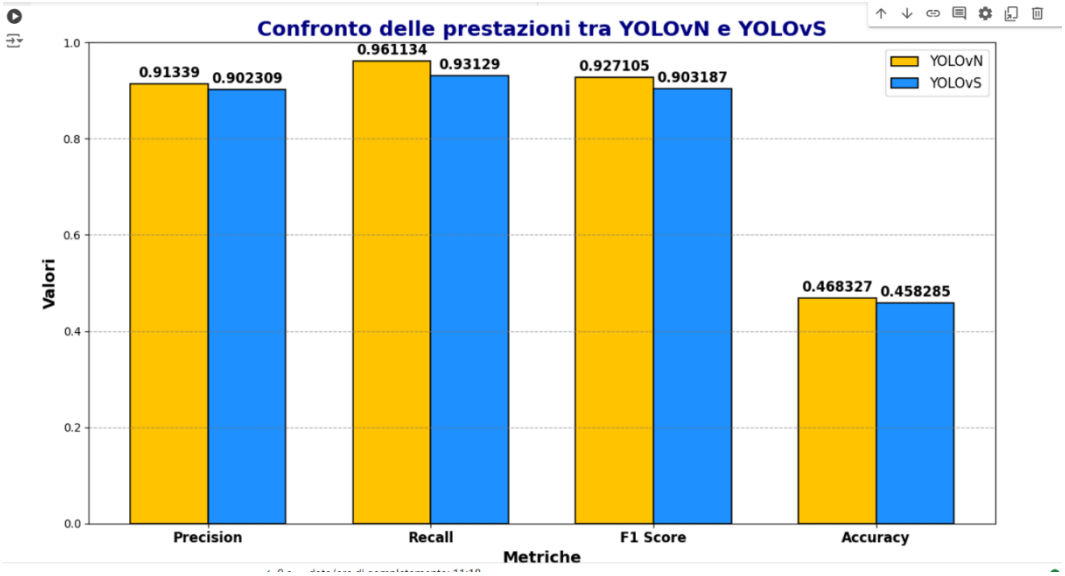
5 Conclusione

L’obiettivo è stato raggiunto con successo, dagli esperimenti condotti è possibile affermare che è possibile sviluppare una soluzione al problema del detection delle monete.

Tra i vari esperimenti condotti, quello che ha dato le migliori prestazioni è stato l’uso del modello YOLOv8n rispetto a YOLOv8s. Si mostra un confronto grafico dei due modelli impiegati nel dataset composto da immagini raffiguranti più monete:



Nel caso del dataset composto da immagini raffiguranti una sola moneta, yolov8n ha avuto prestazioni superiori a yolov8s.



È importante osservare che le prestazioni del modello sono significativamente superiori quando viene utilizzato un dataset contenente immagini di una singola moneta. Questo miglioramento può essere attribuito a diversi fattori inerenti alla semplicità del compito rispetto a quello di rilevare e classificare più monete in una singola immagine.