Sistema di detection di rifiuti plastici su specchi d'acqua

De Ramundo Marco - Perani Xavier

Agosto 2024

Abstract

Obiettivo: ottenere un modello che possa individuare e classificare oggetti sulla superficie di uno specchio d'acqua come un fiume o un lago per poter riconoscere e individuare rifiuti plastici come bottiglie e buste di plastica.

Introduzione

L'inquinamento ambientale dovuto ai rifiuti plastici è un problema che influisce tutti noi, nella fattispecie per quanto riguarda la salvaguardia degli oceani. Le due principali fonti di inquinamento di plastica negli oceani deriva dalla pesca e dai rifiuti gettati nei fiumi. Per poter contrastare la seconda causa è necessario poter effettuare operazioni di monitoraggio per prevenire che rifiuti dannosi possano raggiungere il mare mentre per la prima occorrono strumenti che rendano il recupero dei rifiuti più semplice e meno dispendioso. Anche solo riconoscere, identificare e individuare i rifiuti darebbe una grossa mano allo scopo. Per questo motivo può aver senso adoperare soluzioni quali modelli convoluzionali per effettuare detection degli oggetti in questione per poi agire di conseguenza con la raccolta e la rimozione dall'acqua della plastica.

1 Dataset scelto

Il dataset scelto per il nostro scopo è kili-technology/plastic_in_river che raccoglie immagini di laghetti e bacini d'acqua con oggetti e/o rifiuti in superficie. Il dataset è stato creato per la Kili's Community Challenge, sfida nata per invogliare la community di sviluppatori a sfruttare i modelli di deep learning per

contrastare il problema dell'inquinamento degli oceani. La sfida si è svolta nel febbraio 2022 e il dataset è rimasto accessibile pubblicamente anche dopo la fine dell'evento[1].

Il dataset è composto da un totale di 4259 immagini, già divisi in tre sottoinsiemi per training, validazione e test. Nella tabella 1 è possibile vedere la ripartizione delle istanze per sottoinsieme. Ogni immagine ha anche un file di testo con la lista degli oggetti presenti all'interno, classe dell'oggetto e posizione spaziale rispettando il data format usato da YOLO.

Subset	**Numero istanze**
Train	3407
Val	425
Test	427
Totale	4259

Tabella 1: Distribuzione delle immagini negli insiemi di train, val e test

Le categorie a disposizione per poter classificare gli oggetti identificati sono 4 e sono:

- 0 : `PLASTIC_BAG`
- 1 : `PLASTIC_BOTTLE`
- 2 : `OTHER_PLASTIC_WASTE`
- 3 : `NOT_PLASTIC_WASTE`

La distribuzione delle classi tra le istanze non è uniforme ed è uno dei possibili problemi da affrontare durante l'addestramento del modello. Il rischio è quello di avere difficoltà nel riconoscimento delle categorie che sono meno rappresentate all'interno del dataset. Nella tabella 2 è possibile vedere il numero di istanze per classe all'interno dei vari subsets.

Le dimensioni delle immagini sono varie ma generalemente di qualità alta: la dimensione in larghezza supera i 1000 pixel e gli 800 pixel in altezza.

Nella immagine 1 è possibile in ordine vedere la distribuzione delle istanze nel dataset, la grandezza dei box per istanza all'interno dell'immagine, la distribuzione delle coordinate x e y del box nell'immagine, infine la grandezza del box per istanza.

[1]Attualmente (luglio-agosto 2024) si riscontrano problemi nell'acquisizione del dataset tramite API. Probabilmente il proprietario del dataset ha spostato o tolto il dataset dal proprio server. Per il momento non si sa se il dataset tornerà a disposizione senza problemi. *Edit*: in data 28 Agosto 2024 il proprietario ha annuciato la rimozione del dataset. Attualmente il dataset non è più disponibile

Classe	Train	Val	Test
PLASTIC_BAG	1250	167	85
PLASTIC_BOTTLE	6276	785	854
OTHER_PLASTIC_WASTE	3345	296	122
NOT_PLASTIC_WASTE	1414	212	111

Tabella 2: Distribuzione delle istanze all'interno del dataset

Da notare come la maggior parte delle istanze hanno una dimensione della box rispetto all'immagine ridotta, in genere inferiore al 20% della larghezza e/o altezza dell'immagine. In genere la maggior parte delle istanze, essendo rifiuti sulla superficie dell'acqua, sono concentrati orizzontalmente nella parte centrale dove è plausibile trovare l'orizzonte dello specchio idrico.

La dimensione ridotta del box potrebbe essere utile da tenere in considerazione per quanto riguarda l'iperparametro `imgsz`, image size, ovvero le dimensioni delle immagini in ingresso passate al modello.

2 Modello per Object Detection

L'obiettivo del lavoro è quello di effettuare detection sui rifiuti plastici su immagini e per questo si è optato su una architettura basata su rete convoluzionale che allo stato dell'arte sia efficacie e abbia anche facilità nell'utilizzo. La scelta è ricaduta su YOLO, You Only Look Once, in quanto risulta una delle architetture più utilizzate ed performanti. A maggior ragione, il dataset a nostra disposizione era già predisposto per essere compatibile con YOLO.

Ci siamo basati sulla libreria YOLO di ultralytics in quanto possiede molti metodi e strumenti per lavorare al meglio con il modello. In particolare molto utile la gestione interna per fare augmentation direttamente durante la fase di training.

All'inizio del progetto era a disposizione la versione 8 di YOLO che è stata poi la versione da noi utilizzata sebbene in questi ultimi mesi sono state presentate le versioni 9 e 10. Abbiamo preferito mantenere per continuità con il lavoro la versione 8, che di recente ha visto anche l'aggiornamento alla versione 8.2, una versione che mantiene la stessa struttura per quanto riguarda l'architettura ma ha performance migliori.

Architettura di YOLOv8

L'architettura di YOLOv8 si basa su un approccio end-to-end che suddivide l'immagine in una griglia e applica convoluzioni multiple per predire le boun-

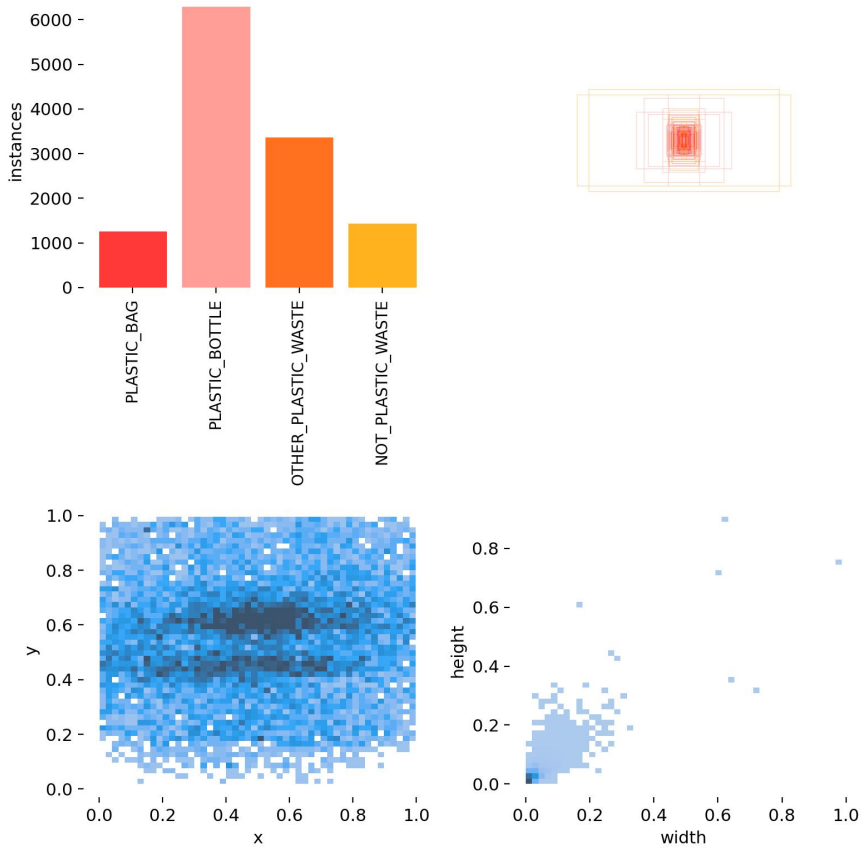

Figura 1: Caratteristiche istanze nel dataset

ding box e le classi degli oggetti all'interno di ciascuna cella della griglia. YOLOv8 utilizza tecniche avanzate come la *Path Aggregation Network* (PANet) per migliorare l'integrazione delle informazioni a diversi livelli di profondità della rete, il che è fondamentale per rilevare oggetti di varie dimensioni e scale. Nella figura 2 è riportato un diagramma semplificato dell'architettura.

Il modello YOLOv8 incorpora anche una *Neck* e una *Head* ottimizzati per massimizzare la capacità di rilevamento attraverso una migliore aggregazione delle caratteristiche e una maggiore flessibilità nelle predizioni finali. L'uso di meccanismi come le *Depthwise Separable Convolutions* contribuisce a ridurre il numero di operazioni computazionali, mantenendo un'elevata capacità espressiva del modello.

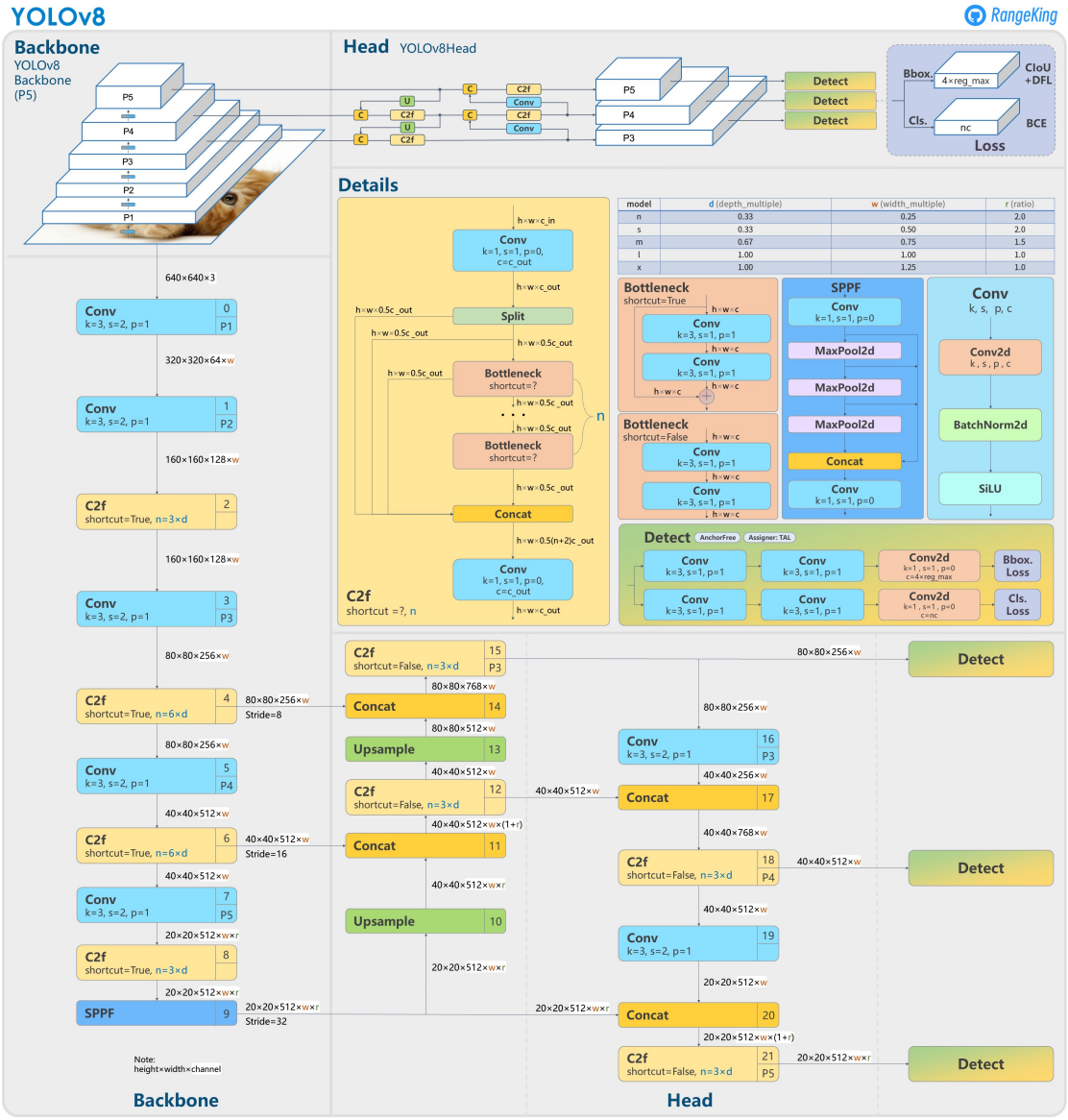

Figura 2: Schema dell'architettura di YOLOv8

Dimensioni delle architetture usate

Nell'ambito del progetto, si è scelto di utilizzare le varianti **small**, **medium** e **nano** di YOLOv8 per il task di object detection, in quanto queste architetture offrono un buon compromesso tra capacità computazionale e performance, risultando particolarmente adatte per l'esecuzione su hardware con risorse limitate quali quelle a noi disponibili.

Vantaggi di YOLOv8 per il Riconoscimento dei Rifiuti Plastici

La scelta di YOLOv8 per il riconoscimento dei rifiuti plastici nei fiumi è giustificata da diversi fattori:

- **Rilevamento in Tempo Reale**: YOLOv8 è noto per la sua capacità di eseguire object detection in tempo reale, il che è cruciale per applicazioni di monitoraggio continuo nei fiumi. La possibilità di rilevare e classificare i rifiuti plastici in tempo reale consente interventi immediati, riducendo il rischio che i rifiuti possano propagarsi ulteriormente.
- **Efficienza Computazionale**: Le versioni di YOLO sono ottimizzate per girare su hardware con risorse limitate, come droni o sistemi embedded. Questo le rende particolarmente adatte per applicazioni sul campo, dove la potenza di calcolo potrebbe essere un vincolo.
- **Accuratezza Elevata**: Nonostante la sua velocità, YOLOv8 mantiene un'accuratezza competitiva grazie all'uso di tecniche avanzate come l'addestramento con *label smoothing* e l'adozione di una griglia più fine per la previsione delle bounding box. Questo è essenziale per distinguere efficacemente i rifiuti plastici da altri detriti naturali presenti nei fiumi, come è possibile vedere in un esempio alla figura 3.

3 Metriche per valutazione performance

Nell'ambito del riconoscimento automatico dei rifiuti plastici nei fiumi utilizzando la rete convoluzionale YOLO (You Only Look Once), l'accurata valutazione delle performance del modello è cruciale. Il contesto operativo, caratterizzato da variabilità ambientali e la presenza di oggetti confondenti, rende fondamentale l'impiego di metriche di valutazione che forniscano una visione completa dell'efficacia del modello. In questa sezione, discuteremo in dettaglio le metriche principali utilizzate per valutare il modello, giustificando la loro rilevanza rispetto all'obiettivo specifico di riconoscimento dei rifiuti plastici.

Precision

La *Precision* è una metrica fondamentale nel contesto del riconoscimento dei rifiuti plastici. Questa misura indica la percentuale di rifiuti classificati correttamente tra tutti quelli identificati come rifiuti dal modello. In altre parole, la

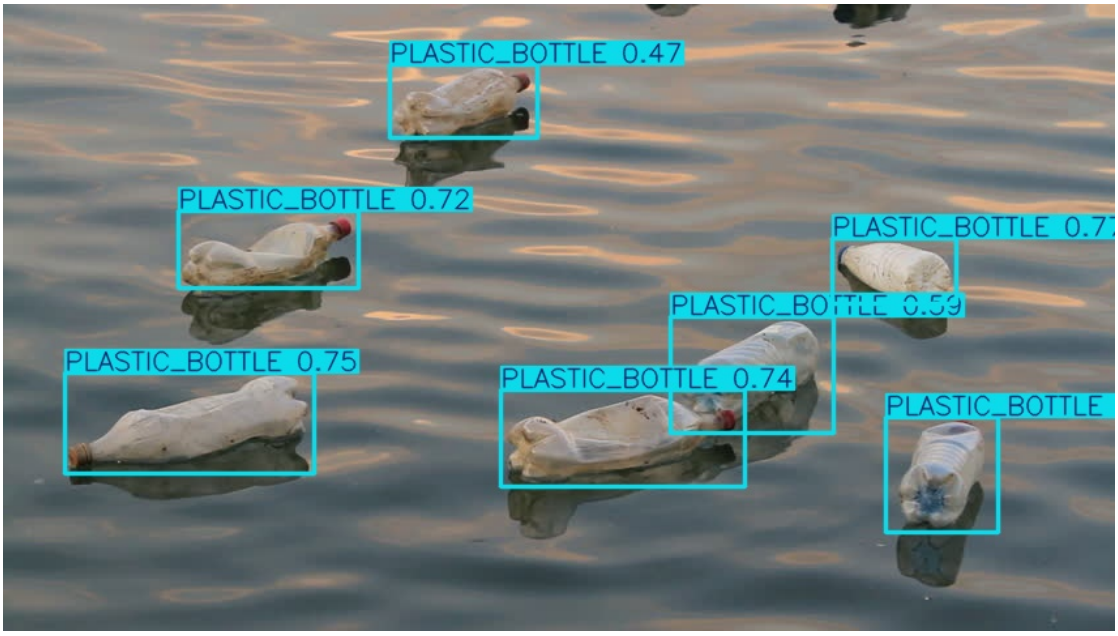

Figura 3: Esempio di output di YOLOv8 per il rilevamento di rifiuti plastici nei fiumi.

precision ci dice quanto possiamo fidarci delle predizioni positive effettuate da YOLO. La formula per calcolarla è la seguente:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

dove:

- *TP* (True Positive): il numero di rifiuti plastici correttamente rilevati dal modello;
- *FP* (False Positive): il numero di oggetti non plastici erroneamente identificati come rifiuti plastici.

Nel contesto dei fiumi, dove possono essere presenti detriti naturali o altre forme di inquinamento, una precision elevata è essenziale per ridurre i falsi allarmi. Un'elevata precision significa che il modello è in grado di distinguere efficacemente i rifiuti plastici da altri oggetti, riducendo il rischio di identificare erroneamente materiali innocui come inquinanti, il che è cruciale per evitare sprechi di risorse in operazioni di pulizia.

Recall

La *Recall* misura la capacità del modello di identificare correttamente tutti i rifiuti plastici presenti. La recall è particolarmente importante in scenari dove la priorità è minimizzare il numero di rifiuti plastici non rilevati (falsi negativi), che possono avere un impatto ambientale significativo se lasciati nei corsi d'acqua. La formula per calcolare la recall è la seguente:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

Nel contesto del riconoscimento dei rifiuti plastici, un alto valore di recall significa che il modello è efficace nel rilevare la maggior parte dei rifiuti, riducendo al minimo il rischio che rifiuti plastici possano sfuggire all'attenzione. Tuttavia, un aumento della recall potrebbe portare a una diminuzione della precision, poiché il modello potrebbe iniziare a classificare erroneamente più oggetti come plastica per evitare di perdere i rifiuti effettivi. Pertanto, è fondamentale trovare un equilibrio tra precision e recall.

F1-Score

L'*F1-Score* è la media armonica tra precision e recall, offrendo una misura bilanciata che tiene conto sia dei falsi positivi che dei falsi negativi. Questa metrica è particolarmente utile in scenari come il riconoscimento dei rifiuti plastici, dove il dataset può presentare classi sbilanciate e dove è importante non sovraottimizzare il modello per una sola metrica a scapito dell'altra. La formula per l'F1-Score è:

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

Nel contesto del riconoscimento dei rifiuti plastici, l'F1-Score è **particolarmente indicato**, poiché consente di valutare l'efficacia complessiva del modello, bilanciando l'accuratezza con la capacità di rilevamento. Un F1-Score elevato indica che il modello YOLO è in grado di mantenere un buon equilibrio tra identificare correttamente i rifiuti plastici e minimizzare gli errori.

mean Average Precision (mAP)

Il *mean Average Precision* (mAP) è una delle metriche più utilizzate e riconosciute nella valutazione delle performance di modelli di *object detection*. Il mAP rappresenta la media delle *Average Precision* (AP) calcolate su tutte le classi considerate nel modello. La *Average Precision*, a sua volta, è l'area sotto la curva

Precision-Recall per ciascuna classe. Il mAP è particolarmente rilevante per il nostro caso di studio, poiché:

- Riflette le prestazioni complessive del modello su più classi di rifiuti plastici (se categorizzati in diverse tipologie);
- Tiene conto della capacità del modello di rilevare i rifiuti plastici in condizioni variabili, come differenti condizioni di illuminazione, presenza di acqua torbida, e variazioni nella forma e dimensione dei rifiuti;
- Considera le variazioni nella soglia di rilevamento del modello, fornendo un'indicazione della sua robustezza e capacità di generalizzazione.

Nel caso del riconoscimento dei rifiuti plastici, un mAP elevato indica che il modello è in grado di rilevare correttamente i rifiuti in diverse situazioni operative, mantenendo un buon equilibrio tra precisione e richiamo per ciascuna classe considerata. Questo è cruciale per garantire che il sistema di monitoraggio automatico sia affidabile in ambienti reali e possa contribuire efficacemente alla riduzione dell'inquinamento nei corsi d'acqua.

4 Esperimenti effettuati

In questa sezione vengono esposti i vari esperimenti di addestramento svolti con annessi risultati e considerazioni.

Modello 1: v8m12 e v8m12bis

I primi esperimenti sono serviti per raccogliere informazioni utili per avere un fase di addestramento che fosse il più efficacie possibile. Innanzitutto si è cercato di capire quanto tempo avrebbe ogni iterazione, ogni epoca e in genere l'addestramento richiesto e capire in particolare quante risorse computazionali erano richieste e quante epoche consentire al modello durante la fase di training.

Questi test, così come molti dei seguenti esperimenti, sono stati eseguiti su un notebook jupiter eseguito su server colab, sfruttando l'accelerazione GPU, in genere una scheda T4, fornita giornalmente per un massimo di 3 ore da parte di Google.

Una delle prime prove svolte riguarda due esecuzioni di training denominate durante il lavoro `v8m12` e `v8m12bis`. La prima è stata un'esecuzione di sole 30 epoche per vedere quanto velocemente le funzioni di loss scendessero di valore e stimare quante epoche concedere successivamente. Il modello utilizzato è quello medio preaddestrato con il dataset COCO, come disponibile dalla

libreria di ultralytics (YOLOv8m.pt). La seconda esecuzione ha visto sempre un numero di epoche pari a 30 ma utilizzando come modello di partenza il miglior modello dell'addestramento precedente in quanto dai risultati come è possibile vedere nella figura 4 c'era ancora margine di miglioramento.

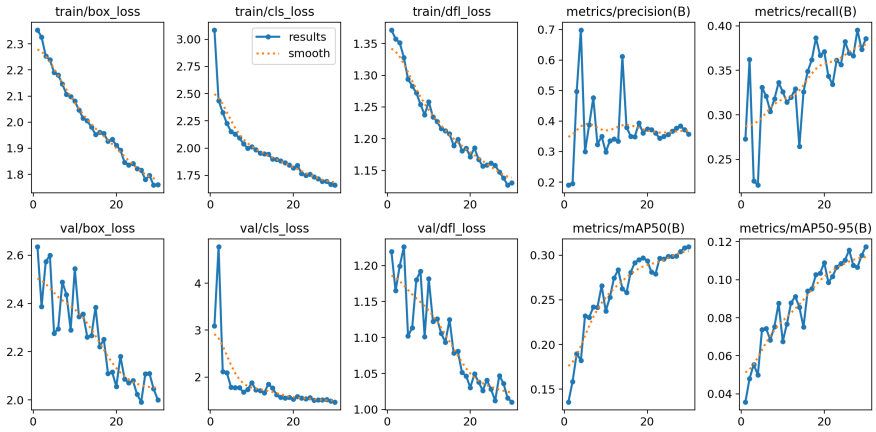

Figura 4: Andamento funzioni di loss e metriche durante l'esecuzione di v8m12

Molti degli iperparametri, se non tutti, sono stati tenuti quelli di default della libreria di ultralytics. L'unico iperparametro impostato per curiosità è `freeze`: in breve abbiamo tenuto 7 layer del modello di YOLO congelati in modo da ridurre il tempo necessario per l'addestramento. Per la bontà dell'esperimento si è dimostrato di poca utilità e gli addestramenti successivi tale iperparametro è stato ignorato, ovvero la modifica dei pesi avveniva su tutto il modello.

In ogni caso a prescindere dai risultati ottenuti, che per la natura dell'esperimento non sono fondamentali, si è visto come il numero di epoche necessario per l'addestramento dovrebbe essere molto più alto per poter raggiungere quanto meno un'efficacia del modello accettabile, così come è stato fatto poi per gli esperimenti successivi.

Modello 2 - small-1202 e small-1203

Avendo sperimentato con i vari modelli, abbiamo iniziato a effettuare alcuni addestramenti aumentando il numero di epoche. Si è deciso quindi di procedere con il training su tutti i parametri della rete e non congelare alcuni layer con l'opzione `freeze` ma bilanciando l'operazione optando per la dimensione del modello più piccola quale la versione small YOLOv8s.

Il numero di epoche per esecuzione è stato impostato a 120 e per tale motivo i vari modelli sono stati nominati come `small-120x`. Il primo dei due modelli, `small-1202`, ha visto una esecuzione comunque non ottimale visto che alcuni

dei valori di loss rispetto alla validazione non sono stati salvati per alcuni problemi durante l'esecuzione su colab. Ciononostante si è visto, come in figura 5, che i valori delle funzioni di loss continuavano a decrescere mentre le metriche o rimanevano costanti, come precision e recall, o andavano a migliorare, come mAP50.

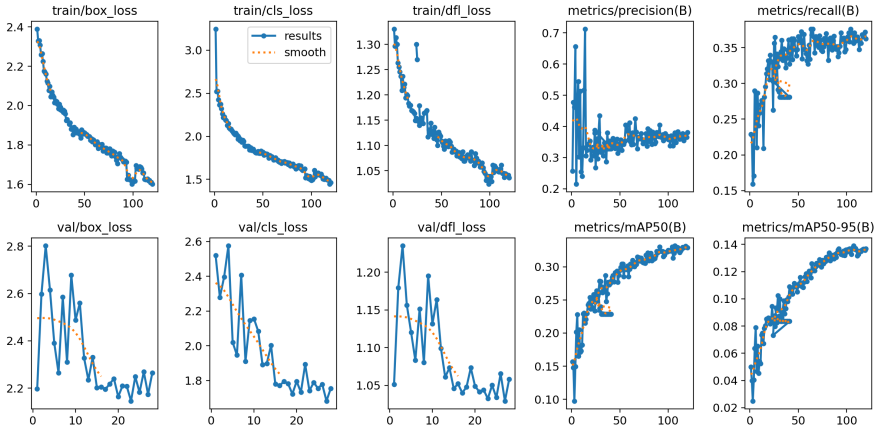

Figura 5: Andamento funzioni di loss e metriche durante l'esecuzione di small-1202

La fase di training è stata pertanto ripetuta con l'esecuzione `small-1203` che rispetto al nome assegnatogli ha avuto 300 epoche a disposizione per l'addestramento mantenendo gli stessi iperparametri del tentativo precedente.

Di seguito sono riportati tutti gli iperparametri più importanti che caratterizzano questi addestramenti: -> aggiunto dropout

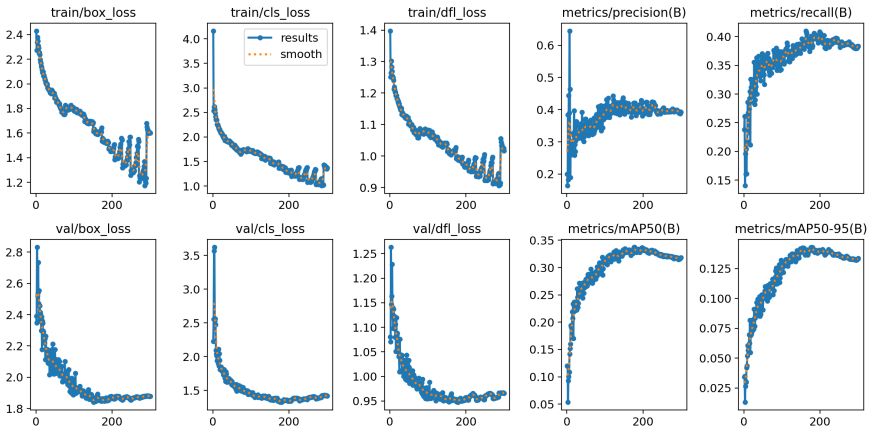

Figura 6: Andamento funzioni di loss e metriche durante l'esecuzione di v8m12

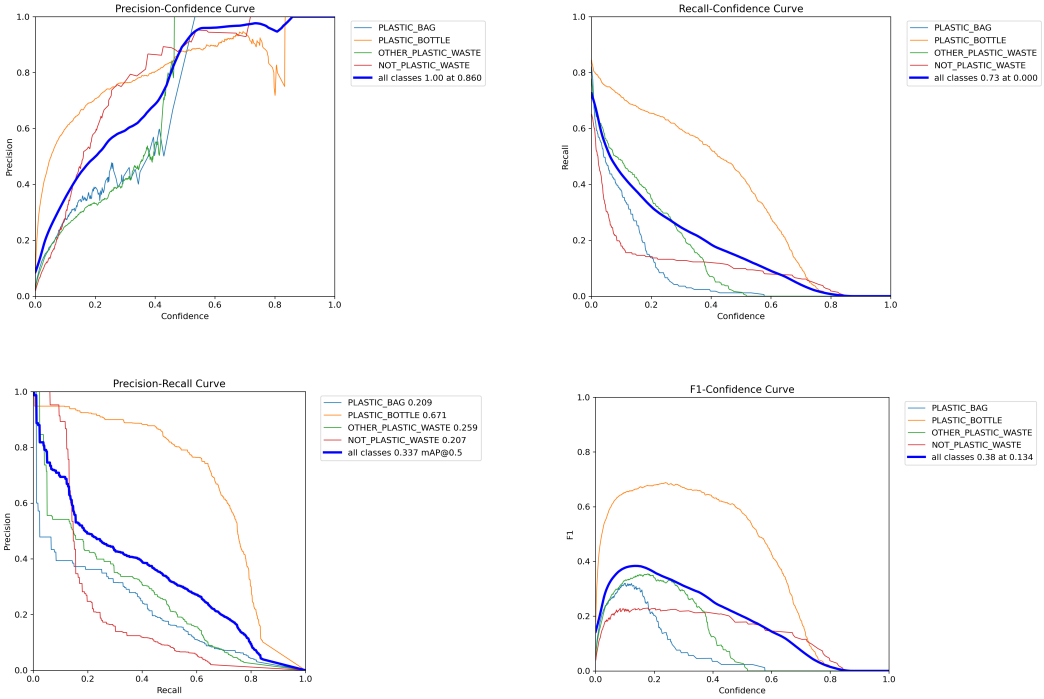

Figura 7: Andamento funzioni di loss e metriche durante l'esecuzione di v8m12

Quello che si è potuto vedere dai risultati e dalle performance di questi modelli è che il modello tende ad avere una discreta precisione e richiamo in generale ma non troppo soddisfacente. In particolare se si

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
ALL	427	1172	0.452	0.450	0.374	0.172
PLASTIC_BAG	34	85	0.402	0.388	0.344	0.120
PLASTIC_BOTTLE	312	854	0.684	0.727	0.724	0.344
OTHER_PLASTIC_WASTE	25	122	0.140	0.377	0.106	0.0367
NOT_PLASTIC_WASTE	49	111	0.582	0.306	0.320	0.187

Tabella 3: Risultati per *small-1203*

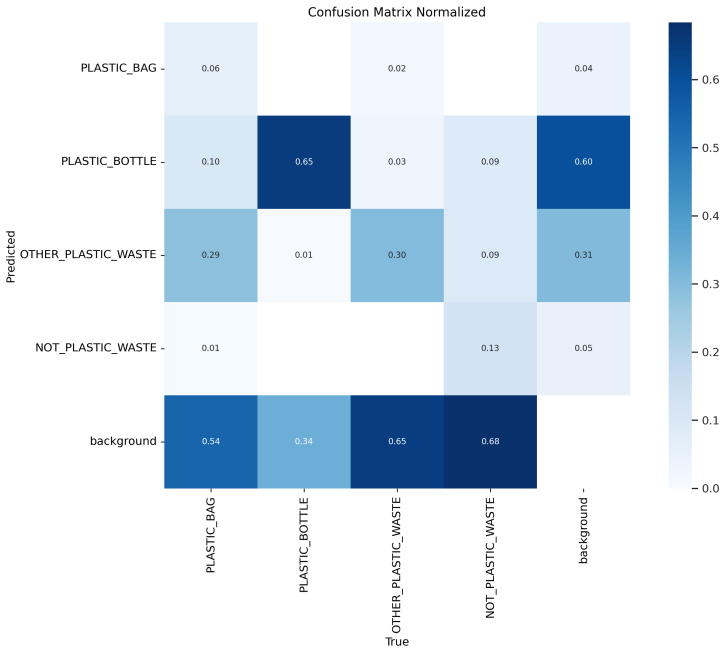

Figura 8: Matrice di confusione normalizzata data dal modello small-1203

Modello 3 - medium-200-0

Proviamo a utilizzare un modello con più parametri, più grande per vedere se riesce ad apprendere meglio le classi

primo modello

Modello 4 - small-tune-03 e small-tune-04

modelli ucraini con paper ma eseguiti male su colab

primo modello

Modelli 5 - *size*-ukra-0

Modelli con rifermento al paper trovato

Modelli addestrati su server serina

primo modello

Modelli 6 - tuning iperparametri

primo modello

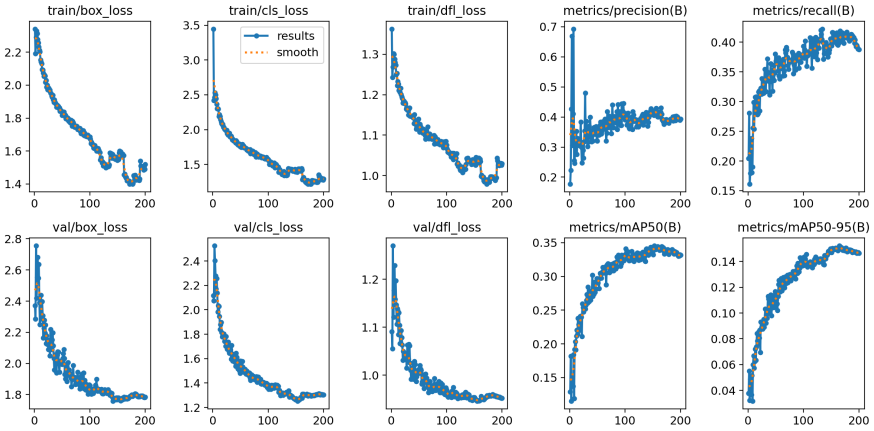

Figura 9: Andamento funzioni di loss e metriche durante l'esecuzione di v8m12

5 Commento risultati e conclusioni

Ultima sezione da scrivere

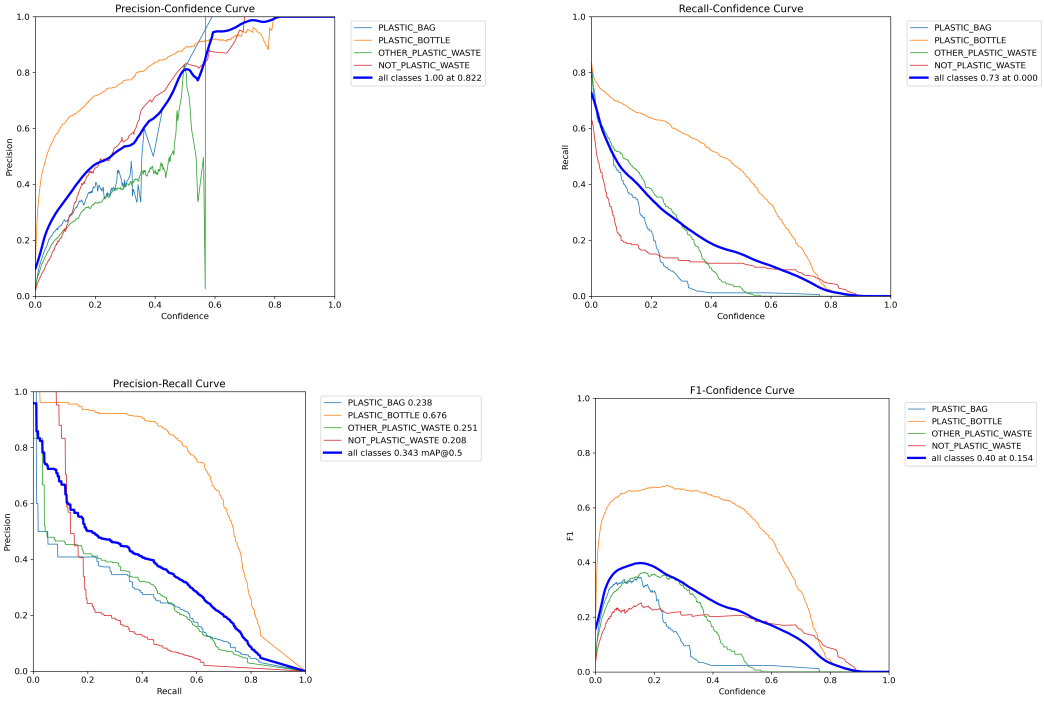

Figura 10: Andamento funzioni di loss e metriche durante l'esecuzione di v8m12

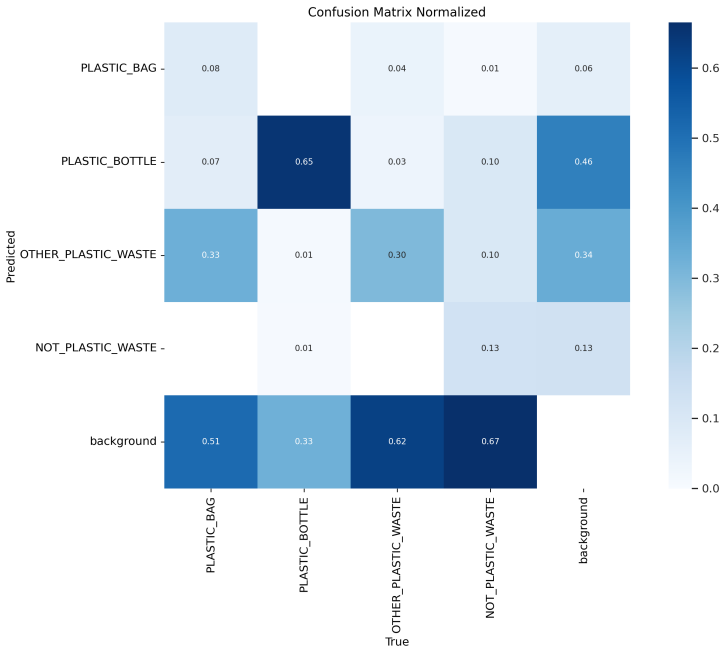

Figura 11: Matrice di confusione normalizzata data dal modello medium-200-0