Benchmarking di architetture YOLOv8 e YOLOv10 in un approccio a due step per la detection di persone e la classificazione di pescatori

Riccardo De Ritis and Emanuele Biccheri

Università Politecnica delle Marche.

Contributing authors: s1115914@studenti.univpm.it;
s116727@studenti.univpm.it;

1 Introduzione

Lo sviluppo dell'Intelligenza Artificiale (IA) ha generato enormi benefici in numerosi campi. Un'area in cui questa tecnologia ha dimostrato un impatto particolarmente significativo è quella del monitoraggio e della prevenzione, sia in ambito legale che medico, grazie all'analisi di immagini attraverso tecniche di Deep Learning. Questo approccio consente di analizzare grandi quantità di dati in tempo reale, individuando modelli di comportamento sospetti e potenziali minacce con una precisione senza precedenti. La capacità dell'IA di apprendere e adattarsi continuamente a nuovi dati migliora considerevolmente l'efficacia delle strategie di prevenzione e di intervento in diversi settori.
Lo scopo del lavoro qui discusso, che si inserisce in un progetto più grande sviluppato in collaborazione con il CNR IRBIM (Istituto per le Risorse Biologiche e le Biotecnologie Marine del Consiglio Nazionale delle Ricerche), è quello di monitorare attività sospette nel settore della pesca ricreativa. Nello specifico, il progetto ha l'obiettivo di analizzare e confrontare diverse tecniche di Deep Learning al fine di creare una rete neurale in grado di monitorare la pesca ricreativa, attraverso delle immagini acquisite da drone, e riconoscere se stanno svolgendo attività di pesca.
La carenza di un dataset appropriato per il progetto, dovuta al fatto che è ancora in fase di acquisizione, ha portato a una fase iniziale di ricerca completamente focalizzata sull'identificazione di possibili dataset alternativi. Questa fase è stata cruciale per garantire che il progetto potesse avanzare nonostante la mancanza temporanea del dataset principale. Di conseguenza, sono stati esplorati vari dataset, analizzando attentamente la loro compatibilità con gli obiettivi del progetto. Successivamente, è

stata condotta un'analisi per valutare il comportamento di una rete a due step, seguita da diverse sessioni di addestramento per esaminare vari modelli e dataset.
Di seguito saranno presentate tutte le fasi di sviluppo, i risultati ottenuti e le considerazioni finali.

2 Stato dell'arte

Nell'articolo *"The Fishnet Open Images Database: A Dataset for Fish Detection and Fine-Grained Categorization in Fisheries"* [1] viene evidenziato il fatto che, negli ultimi anni, più di mille pescherecci commerciali sono stati equipaggiati con sistemi di monitoraggio elettronico (EM) basati su telecamere, per raccogliere dati essenziali per la gestione e la regolamentazione della pesca rivoluzionando il modo in cui vengono affrontate le pratiche di pesca illegale. La Computer Vision emerge come un potenziale e valido alleato, potendo drammaticamente accelerare il processo di analisi dei video grazie alla capacità di supportare il monitoraggio, rilevando e identificando automaticamente comportamenti sospetti e violazioni delle normative di pesca. Questo potenziale è di particolare importanza nel contrasto alla pesca illegale, non dichiarata e non regolamentata (INN), nonché nel garantire il rispetto delle normative e la sostenibilità delle risorse ittiche globali.
Dopo uno studio accurato dello stato dell'arte sui vari modelli di detection, si è deciso di esplorare e confrontare diversi approcci basati su architetture della famiglia YOLO. Questa architettura è stata scelto per la sua efficacia nell'identificazione degli oggetti in tempo reale grazie al suo approccio single-stage, che integra localizzazione e classificazione in un'unica rete neurale. Rispetto ad altri modelli come Faster R-CNN o SSD, YOLO si distingue per la sua velocità di esecuzione, essendo progettato per operare direttamente sull'immagine intera anziché su regioni proposte. Questo lo rende ideale per certi contesti dove la tempestività nella risposta è critica. In particolare, le versioni YOLOv8 [2] e YOLOv10 [3] sono state scelte per la loro comprovata capacità di bilanciare efficienza e accuratezza. Nel contesto della presente ricerca, non è stato trovato nessun articolo specifico riguardante il monitoraggio della pesca ricreativa né è stato possibile accedere a dataset pertinenti su cui basare lo studio. Pertanto, è stato utilizzato anche l'articolo *"A deep-learning framework running on edge devices for handgun and knife detection from indoor video-surveillance cameras"* [4] come punto di partenza per sviluppare una rete neurale a due step. Inizialmente, è stato eseguito uno studio dello stato dell'arte, esaminando approfonditamente le metodologie e le tecniche descritte nel suddetto articolo. Da questo studio è emerso un approccio basato su una rete neurale articolata in due fasi: la prima necessaria per identificare le persone all'interno della scena ripresa dalla telecamera, e la seconda che consente l'identificazione degli oggetti pericolosi all'interno dell'area delimitata (bounding box) in cui si trova la persona.
L'approccio proposto si basa sull'osservazione che l'arma deve necessariamente essere portata da un soggetto umano per essere pericolosa. Ogni passo si basa su un'architettura di deep learning specifica con l'obiettivo di massimizzare il compromesso tra velocità e accuratezza.

L'articolo rappresenta un punto di partenza fondamentale per la ricerca, in quanto fornisce un contesto teorico e metodologico solido per identificare e classificare i pescatori coinvolti nell'attività di pesca ricreativa. Tuttavia, non essendo direttamente centrato sul compito specifico, è stato adattato questo approccio alle esigenze specifiche.

3 Materiali e Metodi

Basandosi sulla struttura descritta nell'articolo [4], è stato deciso di utilizzare un'architettura a due step, impiegando reti della famiglia YOLO sia per la detection che per la classificazione.

Durante lo sviluppo, diverse versioni di YOLOv8 e YOLOv10 sono state implementate per identificare le varianti che offrissero le migliori prestazioni. Questa fase di confronto ha permesso di ottimizzare ulteriormente l'accuratezza e l'efficienza del nostro modello, scegliendo le versioni che si adattavano meglio alle specifiche esigenze.

3.1 YOLO

La famiglia delle YOLO (You Only Look Once) segue un'architettura CNN (Convolutional Neural Network) modulare che può essere suddivisa in tre componenti principali:

- **Backbone**: Estrae le caratteristiche principali dall'immagine di input.
- **Neck**: Raffina e combina le caratteristiche estratte per migliorare la rappresentazione dei dati.
- **Head**: Produce le previsioni finali, come i bounding box e le classi degli oggetti.

Per la detection degli oggetti, YOLOv8 utilizza tutta la struttura sopra descritta. I bounding box e le classi sono predette simultaneamente per ogni regione dell'immagine, permettendo alla rete di identificare e classificare gli oggetti in tempo reale.
Per la classificazione, invece, YOLOv8 utilizza principalmente la backbone e una head modificata per produrre le classi delle immagini. La struttura è simile a una rete di classificazione tradizionale, dove l'output finale è una o più etichette di classe per ogni immagine.
YOLOv10 introduce un nuovo approccio per la rilevazione in tempo reale degli oggetti, affrontando le carenze nella post-elaborazione e nell'architettura del modello riscontrate nelle versioni precedenti. Eliminando la soppressione non massimale (NMS) e ottimizzando vari componenti del modello, YOLOv10 raggiunge prestazioni all'avanguardia con un significativo ridimensionamento computazionale.

L'architettura di YOLOv10 si basa sui punti di forza dei modelli YOLO precedenti, introducendo al contempo diverse innovazioni chiave. Nello specifico, introduce una doppia Head: One-to-Many Head che genera più predizioni per ogni oggetto durante l'addestramento per fornire segnali di supervisione ricchi e migliorare la precisione dell'apprendimento e One-to-One Head che genera una singola migliore predizione per ogni oggetto durante l'inferenza per eliminare la necessità di NMS, riducendo così la latenza e migliorando l'efficienza.

Sia YOLOv8 che YOLOv10 sono disponibili in diverse dimensioni di modelli preaddestrati. In particolare, abbiamo utilizzato e confrontato le varianti *n (nano)*, *s (small)* e *m (medium)*.

3.2 Approccio a doppio step

L'approccio proposto si basa sull'osservazione che la persona deve necessariamente essere in possesso di una canna da pesca per essere identificata come pescatore. Pertanto, è proposta una rilevazione a doppio step, con una prima fase di individuazione delle persone all'interno di un'area e successivamente una rilevazione della presenza di una canna da pesca all'interno dell'area intorno al bounding box di ogni persona.
Per quanto riguarda il primo step, sono state testate diverse versioni di YOLO per selezionare quella con il miglior rapporto tra prestazioni (tempo di inferenza) e accuratezza per la detection delle persone.
Successivamente, è stata eseguita una fase di elaborazione intermedia su ciascun bounding box rilevato all'interno dell'immagine, prima della fase di classificazione. In particolare, per preservare il rapporto d'aspetto degli oggetti, è stato calcolato un ritaglio quadrato dall'immagine originale, posizionando al centro la persona e con i lati pari al doppio dell'altezza, al fine di includere, se presente, anche la canna da pesca, la quale naturalmente occupava un'area più estesa rispetto al singolo bounding box. È stato inoltre verificato che il ritaglio non superasse le dimensioni dell'immagine originale; in caso contrario, il ritaglio è stato adattato di conseguenza. Ogni ritaglio è stato quindi sottoposto al passaggio successivo, dove, sono stati oscurati tutti i bounding box che si trovavano all'interno dell'area di ritaglio della persona di riferimento creando una maschera nera, ponendo attenzione a non oscurare le zone di intersezione tra il bounding box di riferimento e quelli da oscurare.
Una volta acquisite le informazioni sulla posizione di ogni persona all'interno dell'area, il secondo step consiste nella classificazione per determinare se la persona fosse un pescatore o meno. A tale scopo, è stata utilizzata una rete YOLOv8 appositamente addestrata per la classificazione. Non è stato possibile confrontare YOLOv8 con YOLOv10 in questo contesto, poiché YOLOv10, essendo un modello più recente, non disponeva di un modello specifico per la classificazione al momento dello studio. L'idea di questo secondo passaggio era di garantire che la rete fosse adeguatamente addestrata per riconoscere se una persona stesse portando con sé una canna da pesca.
Infine, una volta terminata la classificazione, sono stati raccolti i risultati della fase di detection e quella della classificazione e l'immagine finale di output è stata costruita inserendo nell'immagine originale i bounding box trovati dal primo step etichettati seguendo i risultati ottenuti dal secondo step: in blu le persone e in verde i pescatori.
La scelta di questa proposta a doppio step è stata dettata sia dal fatto di essere riusciti a reperire un dataset sui pescatori con immagini molto ravvicinate, sia dal fatto che così facendo (anche in ottica di una futura implementazione) la rete possa concentrarsi maggiormente sull'apprendere le caratteristiche più specifiche della zona interessata.

3.3 Dataset

Come anticipato nel Capitolo 1 la prima fase del progetto è stata quella inerente alla ricerca di un dataset adeguato per addestrare la rete. In particolare, la ricerca si è concentrata su un primo dataset per il riconoscimento delle persone dall'alto e un secondo per la distinzione tra pescatori e persone.
In realtà, all'inizio è stata eseguita una fase preliminare che coinvolgeva un dataset contenente immagini di canne da pesca, insieme alle relative etichette per l'identificazione. Le numerose ricerche, tuttavia, non hanno generato alcun esito positivo.
É stato trovato il dataset *Fishnet* [5] che consiste in immagini di pescatori prese da delle telecamere installate direttamente sulle imbarcazioni. In questo, però, i pescatori non effettuano pesca tramite canna da pesca, dunque le etichette riguardano solamente la persona, oltre che i pesci (identificati secondo la propria specie). Il più simile alle caratteristiche richieste è stato identificato invece nel dataset *Fisherman* [6], contenente persone con canna da pesca etichettate appunto come *fisherman*. In questo modo, è stato leggermente modificato il nostro target nel secondo step della rete: non più individuare nell'area della persona una canna da pesca, bensì vedere se la persona viene o meno classificata come *fisherman*, cioè una persona con la canna da pesca in mano. Una criticità riscontrata in questo dataset è l'inquadratura eccessivamente ravvicinata in gran parte delle immagini, mentre per il nostro task le immagini dovrebbero avere origine dall'alto, dunque più lontane.
Per far riconoscere le persone in attività generiche e permettere alla rete di distinguerle dai *fisherman*, è stato identificato un secondo dataset *Human Detection Dataset* [7] contenente persone generiche, riprese da un'inquadratura molto simile a quella del dataset *Fisherman*.
Per quanto riguarda i dataset per la detection di persone, in primo luogo è stato identificato il dataset *Aerial Person Detection Computer Vision Project* [8] (di seguito spesso citato con *Aerial* per semplicità), che contiene immagini di persone acquisite dall'alto per mezzo di un drone, con con l'annotazione su ogni individuo presente. Data la vicinanza delle immagini dei pescatori, però, si è ritenuto più opportuno integrare questo dataset di persone dall'alto con *Fishnet* (mantenendo solamente le etichette delle persone), così da rendere la rete più versatile e capace di individuare le persone da più distanze e angolazioni.

3.3.1 Preprocessing

Ogni dataset è stato sottoposto ad un pre-processamento. Più nello specifico, il dataset *Aerial Person Detection Computer Vision Project*, oltre alla classe persona, conteneva anche le classi per furgoni, macchine e molte altre classi che non erano di nostro interesse. Inoltre, le persone erano divise in due classi: *people*, per le persone che non erano in movimento, e *pedestrian*, per le persone che stavano camminando. Per questi motivi, un processamento ha riguardato l'unione delle due classi in una unica, nominata *person*, e il filtraggio di tutte le altre classi che non erano di interesse, con annessa eliminazione di tutte le immagini che non contenevano la classe *person*.
Il dataset *Fishnet* era accompagnato da etichette nel formato CSV, pertanto è stato necessario convertirle per renderle compatibili con la struttura richiesta per YOLO,

che prevede l'organizzazione delle etichette nella cartella *labels*. Questa contiene un file di testo per ogni immagine, in cui è definito per ogni riga un bounding box con classe e dimensioni corrispondenti.

3.3.2 Unione dataset

Al fine di eseguire vari test per individuare il modello più efficace, sono state effettuate ulteriori operazioni che hanno riguardato l'unione di alcuni dataset e la successiva divisione in *training*, *validation* e *test*. Come anticipato, si è pensato di unire i dataset *Aerial Person Detection Computer Vision Project* e *Fishnet*: selezionate 1300 immagini da ciascuno in modo da avere un bilanciamento corretto fra le classi, poi è stato eseguito uno split nelle proporzioni 70-20-10 (rispettivamente per *training*, *validation* e *test*) separatamente in entrambi i dataset. Infine, sono state unite le relative cartelle, ottenendo circa 1800 immagini per il *training*, 500 per il *validation* e 250 per il *test* set. Nelle discussioni successive del presente elaborato, il suddetto dataset verrà chiamato *Mix Detection*.
É stato anche mantenuto singolarmente il dataset *Aerial Person Detection Computer Vision Project*, per valutazioni aggiuntive, ma ridotto per limiti computazionali a circa 1600 immagini complessive, anch'esse poi suddivise nella proporzione 70-20-10.
Per quanto riguarda la fase di classificazione, sono stati uniti i dataset *Fisherman* e *Human Detection Dataset*, per un totale di 1300 immagini, equamente divise tra i due dataset. Successivamente ci si riferirà a quest'ultimo dataset come *Mix Classification*.

3.4 Detection

Al fine di individuare il modello migliore per il task richiesto, sono stati effettuati molteplici addestramenti con diversi dataset e modelli preaddestrati. Da test preliminari e da quanto appreso da [1], sono stati selezionati i seguenti parametri per effettuare il confronto tra le diverse configurazioni:

Table 1 Configurazione dell'addestramento per la detection

optimizer	epochs	imgsz	lr0	momentum	dfl
SGD	100	512	0.0025	0.9	2.0

Come già discusso nella sezione 3.1, YOLOv8 e YOLOv10 presentano diverse scale di modelli preaddestrati. Per eseguire i suddetti confronti, entrambe le architetture sono state addestrate nei tre modelli *n*, *s* ed *m*. Questi addestramenti sono stati effettuati sul dataset *Mix Detection*, al fine di individuare la taglia che meglio garantiva una buona efficacia in relazione al tempo di inferenza.
Utilizzando gli stessi parametri sono stati effettuati due addestramenti utilizzando il modello preaddestrato *yolov8m* sui due dataset di partenza, in modo da verificare che l'unione avesse portato effettivamente miglioramenti, e non che uno dei due preso singolarmente fosse ugualmente sufficiente.
Entrambi i risultati verranno presentati e analizzati nel dettaglio nel Capitolo 4.

3.5 Classificazione

Per quanto riguarda la fase di classificazione, sono state eseguite valutazioni unicamente per YOLOv8, in quanto, come già accennato, YOLOv10 è disponibile solo per il task di detection. É stata effettuata una *5-fold Cross-Validation* testando tre configurazioni differenti di iperparametri sul modello *yolov8m*, sul dataset *Mix Classification*. Di seguito tre configurazioni utilizzate, i cui risultati verranno presentati nel dettaglio nella sezione 4.

Table 2 Configurazioni di addestramento per la detection

epochs	imgsz	batch	optimizer	lr0	momentum
50	512	32	SGD	0.0012	0.90
50	512	32	Adam	0.001	0.90
50	512	32	SGD	0.001	0.97

3.6 Explainability

É stata, infine, implementata anche una fase di *explainability* attraverso EigenCam per YOLOv8 e GradCam per YOLOv10. Entrambe prevedono che vengano selezionati dei layers "target" all'interno del modello che saranno analizzati per restituire una mappa di colori, evidenziando le zone dell'immagine in cui la rete si è maggiormente concentrata per prendere la sua decisione. Questo permette di comprendere meglio il "ragionamento" effettuato dal modello per ottenere i risultati.
Per YOLOv8 è stata utilizzata la repository *YOLO-V8-CAM* [9], importata nella cartella di progetto e utilizzata tramite la funzione *EigenCam* nel secondo step della rete, per aiutare nella comprensione della classificazione.
Per YOLOv10, invece, non erano presenti online risorse che implementassero GradCam, EigenCam o simili. É stata trovata, però, la libreria Python YOLOv8-Explainer [10], che implementa diversi tipi di Cam per la detection di YOLOv8, tra le quali GradCam. Dopo una prima fase di studio del codice della libreria, sono state apportate delle specifiche modifiche per l'utilizzo delle CAM in YOLOV10. La libreria modificata YOLOv10_Explainer è stata aggiunta alla cartella di progetto ed è stata inserita GradCam per la detection tramite YOLOv10.

4 Risultati e Discussioni

Di seguito sono riportati e analizzati i risultati ottenuti da tutti gli addestramenti descritti nella sezione precedente, con un focus sulle soluzioni adottate.
Nella Tabella 3 sono riassunti i risultati espressi in termini di Precision, Recall, mAP50, mAP50-95 e tempi di inferenza, nel confronto tra i modelli n, s e m di detection. Come mostrato nella tabella, la variante m di YOLOv8 dimostra le prestazioni migliori su tutte le metriche considerate. Tuttavia, è importante notare che YOLOv10 nella stessa variante mostra una differenza minima.
Si nota inoltre che le varianti s hanno ottenuto dei risultati leggermente inferiori

rispetto alle relative versioni *m*, registrando però un tempo di inferenza significativamente minore. Con le versioni *n*, invece, la differenza risulta più marcata con risultati che si discostano in modo decisamente non trascurabile.
Un fattore comune, confrontando le due famiglie, è la rilevazione del tempo di inferenza: questo appare sempre minore in YOLOv8, a parità di dimensione del modello.

Table 3 Confronto tra i modelli YOLOv10 e YOLOv8 su GPU NVIDIA Tesla T4 utilizzando il dataset *Mix Detection*

	YOLOv8			YOLOv10		
	n	s	m	n	s	m
Precision	0.762	0.798	**0.815**	0.721	0.8	**0.806**
Recall	0.471	0.525	**0.575**	0.488	0.542	**0.564**
mAP50	0.551	0.605	**0.655**	0.559	0.624	**0.644**
mAP50-95	0.317	0.354	**0.389**	0.313	0.36	**0.384**
Inference (ms)	**3.4**	5.4	9.7	**4.4**	7.2	10.1

I risultati ottenuti dalla detection sono stati eccellenti per tutte le varianti dei modelli YOLO. Si è deciso di utilizzare YOLOv10m invece della variante *m* di YOLOv8, nonostante quest'ultima sembri offrire prestazioni leggermente superiori. Questa scelta deriva dal fatto che, dopo aver condotto test e valutato le diverse metriche ottenute durante l'addestramento di entrambi i modelli, è emerso che YOLOv8 e YOLOv10 utilizzano diverse funzioni di perdita (loss). Entrambi prevedono funzioni specifiche per la predizione dei bounding box, per la classificazione delle classi degli oggetti e per la rilevazione di oggetti piccoli o difficili: rispettivamente, *box_loss*, *cls_loss* e *dfl_loss*. Tali funzioni di perdita, ovviamente, guidano il modello verso una previsione più accurata degli oggetti nelle immagini. Tuttavia YOLOv10 introduce due nuove varianti, "om (Overlap Mismatch)" e "oo (Overlap Outside)", che migliorano la robustezza del modello nel rilevare oggetti di varie forme e nel classificare oggetti con probabilità basse, aspetto in cui YOLOv8 ha mostrato una minore efficacia. É sembrato quindi opportuno dare importanza alla maggior sensibilità di YOLOv10 al task, considerate le prestazioni comunque paragonabili in termini di metriche. Inoltre, l'accuratezza aggiunta da tali funzioni di perdita è stata effettivamente riscontrata con semplici test su un campione di immagini.

Nella Tabella 4 sono riportate le metriche di valutazione delle prestazioni dei modelli, confrontando i tre dataset differenti utilizzati durante lo sviluppo di questo progetto: Aerial, Fishnet e Mix Detection. Come si può osservare, la combinazione dei due dataset ha permesso una detection notevolmente più accurata, passando da un valore di mAP50 di 0.276 per Aerial e 0.043 per Fishnet a uno di 0.644 per Mix Detection. Questo riflette la capacità del modello di effettuare detection in modo più generalizzato, sfruttando sia le immagini da drone di Aerial, che sono scattate da lontano, sia le immagini ravvicinate di Fishnet. Questi risultati confermano l'ipotesi iniziale per cui i due dataset, presi separatamente, non sarebbero risultati adatti ai

fini dell'obiettivo finale, ed evidenziano un significativo miglioramento ottenuto dal processing con l'unione dei due dataset.

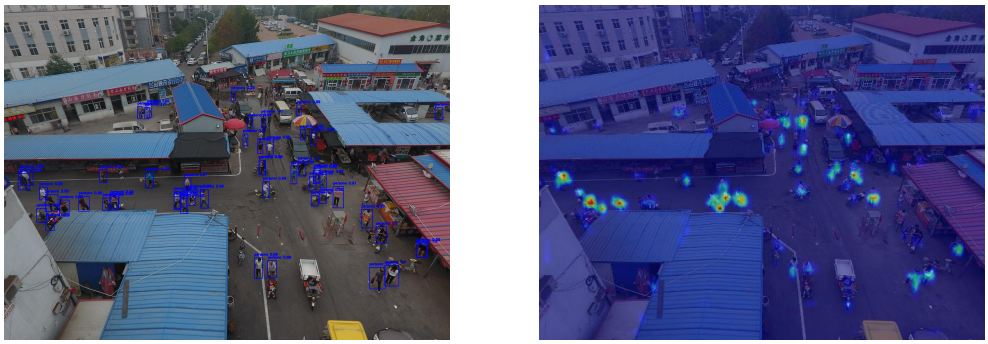
Table 4 Confronto dei dataset su YOLOv10M

Dataset	Precision	Recall	mAP50	mAP50-95
Aerial	0.565	0.265	0.276	0.118
Fishnet	0.28	0.061	0.043	0.02
Mix Detection	**0.806**	**0.564**	**0.644**	**0.384**

L'addestramento della rete YOLOv8 per la classificazione è stato effettuato utilizzando la tecnica di K-Fold Cross Validation, con K pari a 5. Questa tecnica permette di selezionare il miglior modello tra varie configurazioni, nonché quello che ha ottenuto le migliori performance in base ai fold con cui è stato addestrato e validato. Tuttavia, i risultati hanno rivelato che il compito era troppo semplice per un approccio di cross validation, poiché tutti i modelli hanno raggiunto un'accuratezza di circa il 90%, non suggerendo quindi il modello significativamente migliore. La semplicità del task è dettata dal fatto che l'unico dataset reperito (Fisherman) contiene immagini ravvicinate e completamente focalizzate sul pescatore, posto in evidenza. Questo fa si che la rete riesca ad imparare velocemente e sia in grado di valutare in modo molto preciso immagini della stessa tipologia. Ciò non toglie, come discusso di seguito, che questo non ha inficiato sull'efficacia nel secondo step della rete sulla classificazione delle immagini ritagliate.

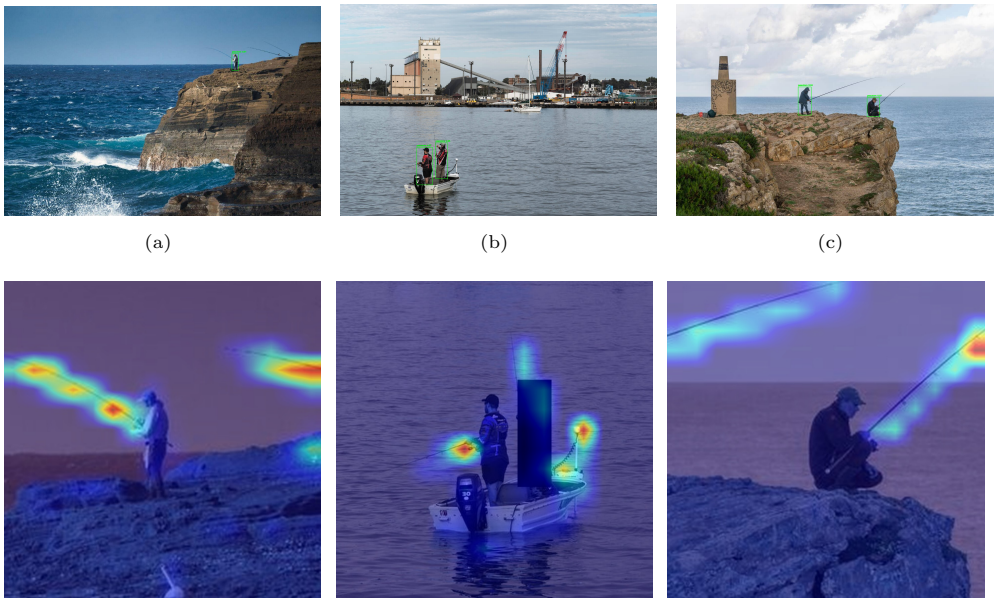
Di seguito sono riportate delle immagini per mostrare l'effettivo funzionamento della rete. Nella Figura 1(a) è riportato il risultato ottenuto in output della rete, corredata di tutti i bounding box rilevati dalla detection e poi classificati nel secondo step. In questa immagine tutte le persone sono state identificate correttamente in *person* (in blu). Nella Figura 1(b), invece, è riportata l'immagine ottenuta utilizzando GradCam alla conclusione dello step di detection. La mappa di colori, dunque, mostra in quali zone il modello di detection si è concentrato per individuare i bounding box delle persone, che sono poi stati soggetti al ritaglio e mandati in input alla seconda rete per la classificazione, fino ad ottenere l'immagine 1(a). Si può notare come la rete si sia concentrata precisamente sulle figure delle persone, riuscendo ad individuare in modo ottimo anche quelle più lontane.
Nella Figura 2 sono riportati ulteriori risultati su immagini contenenti pescatori e quelli della EigenCam, applicata alla fase di classificazione. Nelle immagini 2(a), 2(b) e2(c) sono proposti gli output della rete che ha individuato in modo molto preciso tutti i pescatori, anche da una distanza molto elevata, come in 2(a). Le Figure 2(d), 2(e) e2(f) corrispondono alle EigenCam applicate in fase di classificazione delle tre imamgini sopra. Si può notare come la rete si sia concentrata molto sulla presenza delle canne da pesca all'interno delle immagini ritagliate, per distinguere una persona da un pescatore. Inoltre, la Figura 2(e) mostra come nel passaggio dal primo al secondo step della rete, per ogni bounding box, venga effettuato un ritaglio, andando ad oscurare tramite una maschera tutti quelli diversi dal riferimento. In questo modo la

rete effettuerà la classificazione solamente sulla persona di interesse. Ovviamente, il tutto viene eseguito per tutti i bounding box dell'immagine, mentre qui ne è stato riportato soltanto uno come esempio.



(a) (b)

Fig. 1 Detection e classificazione di pescatori o persone con GradCam per YOLOv10



(a) (b) (c)

(d) (e) (f)

Fig. 2 Detection e classificazione di pescatori o persone con EigenCAM per YOLOv8

5 Conclusioni e Sviluppi Futuri

In tutti gli addestramenti delle varianti di YOLOv8 e YOLOv10 è emerso che YOLOv10 generalmente supera il suo predecessore in termini di accuratezza media, fatta eccezione per la variante *m* che mostra prestazioni leggermente inferiori. L'unione dei dataset *Aerial* e *Fishnet* si è rivelata cruciale per migliorare la capacità dei modelli di rilevare persone sia da lontano che da vicino. Questo ha permesso di ottenere ottimi risultati nella rilevazione di persone da varie angolazioni, prospettive e con immagini di diverse scale di grandezza e risoluzione.
Di conseguenza, abbiamo ritenuto vantaggioso incorporare un secondo step per la classificazione dei pescatori. Poiché al momento dello studio non esisteva un dataset specifico per i pescatori osservati da una distanza appropriata, il ridimensionamento dell'area in cui si trovava ogni singola persona si è dimostrato significativamente utile per il task. Questo approccio ha migliorato la precisione della rilevazione e la classificazione dei pescatori nelle immagini, rispondendo efficacemente agli obiettivi principali dello studio.
Per quanto riguarda gli sviluppi futuri, si potrebbe considerare, in prima istanza, l'applicazione di queste implementazioni su un dataset creato ad hoc per il progetto, valutando la capacità di generalizzazione e le performance della rete nel task specifico. In aggiunta, si potrebbe prevedere l'utilizzo di modelli ottimizzati per dispositivi a bassa potenza. Infatti, avendo addestrato e confrontato vari modelli YOLO (nelle versioni nano, small e medium), è possibile procedere all'installazione dei modelli su droni o dispositivi con limitazioni computazionali. Questo permetterebbe il rilevamento in tempo reale di attività di pesca illegale, anche in condizioni di risorse limitate.
Si potrebbe poi considerare l'uso di tecniche di *super resolution*, al fine di migliorare la qualità delle immagini ritagliate, come quelle catturate da lunghe distanze o parzialmente occluse.
Infine, sarebbe interessante implementare meccanismi di attenzione nei modelli di rilevamento per concentrarsi su aree specifiche delle immagini, come le mani, al fine di determinare se una persona sta impugnando una canna da pesca. Inoltre, per il caso della pesca con galleggiante, sarebbe utile verificare la presenza di una canna a terra vicino al pescatore con un'asta. Questo approccio permetterebbe di rilevare con maggiore precisione i pescatori, distinguendoli da altre persone che potrebbero essere presenti nella scena senza attrezzi specifici per attività di pesca.

References

[1] Kay, J., Merrifield, M.: The fishnet open images database: A dataset for fish detection and fine-grained categorization in fisheries. arXiv preprint arXiv:2106.09178 (2021)

[2] https://docs.ultralytics.com/

[3] https://github.com/THU-MIG/yolov10/tree/main

[4] Berardini, D., Migliorelli, L., Galdelli, A., Frontoni, E., Mancini, A., Moccia, S.: A deep-learning framework running on edge devices for handgun and knife detection from indoor video-surveillance cameras. Multimedia Tools and Applications **83**(7), 19109–19127 (2024)

[5] https://www.fishnet.ai/description

[6] https://universe.roboflow.com/cio-a-d4yjw/01-3js62

[7] https://www.kaggle.com/datasets/constantinwerner/human-detection-dataset

[8] https://universe.roboflow.com/aerial-person-detection/aerial-person-detection

[9] https://github.com/rigvedrs/YOLO-V8-CAM

[10] https://pypi.org/project/YOLOv8-Explainer/