

UNIVERSITÀ POLITECNICA DELLE MARCHE



Progetto Computer Vision & Deep Learning

Sviluppo di un sistema AI per la detection di imbarcazioni della piccola pesca

AUTORI

Davide Ticchiarelli - S1121687
Giampaolo Marino - S1121678
Niccolò Ciotti - S1121676

Anno accademico 2024/2025

Indice

1	Introduzione	1
2	Stato dell'arte	2
3	Metodi e materiali	3
3.1	Dataset	3
3.2	Configurazione	4
4	Confronto tra architetture YOLO	5
4.1	YOLOv10	5
4.2	YOLOv12	7
4.3	Confronto performance tra YOLOv10 e YOLOv12	8
5	Ottimizzazione dei modelli	10
5.1	Modifiche e Integrazioni Architetturali	10
5.1.1	SPDConv - Space-to-Depth Convolution	10
5.1.2	SPANet - Small Path Aggregation Network e sblocco Testa P2	11
5.1.3	Modulo ECSPP	12
5.1.4	Modulo DLC2f	13
5.2	Scelta degli iperparametri	14
5.3	Risultati	15
5.3.1	Analisi delle Performance	15
5.3.2	Analisi delle Emissioni di CO ₂ - CodeCarbon	17
5.3.3	Explainability - EigenCAM	18
6	Conclusioni	20
6.1	Possibili sviluppi futuri	21

1 Introduzione

Il nostro progetto nasce dall'esigenza di sviluppare un modello di AI capace di individuare imbarcazioni di piccole dimensioni in immagini satellitari caratterizzate da alta variabilità di scala, qualità e provenienza. L'obiettivo è quello di affrontare un contesto operativo reale, in cui le immagini rendono particolarmente complesso il compito di rilevamento automatico. Il lavoro si concentra sull'analisi di immagini SAR (Synthetic Aperture Radar), una tecnologia radar in cui, sfruttando lo spostamento della piattaforma portante (satellite, aereo o drone), si combinano gli echi ricevuti in posizioni diverse per creare un'apertura "sintetica" molto estesa, generando così immagini ad altissima risoluzione. A differenza dei sensori ottici tradizionali, il SAR consente di ottenere dati utili anche in presenza di nuvole, scarsa visibilità o durante la notte, risultando fondamentale per il monitoraggio continuo delle aree marittime. Questa caratteristica rende il SAR una delle tecnologie più utilizzate per la sorveglianza dei mari e la gestione della sicurezza marittima.

Il progetto si inserisce in un'iniziativa più ampia promossa dal Consiglio Nazionale delle Ricerche (CNR), volta al monitoraggio e alla tutela delle risorse marine, con particolare attenzione alla rilevazione di attività sospette come la pesca illegale, non dichiarata e non regolamentata (IUU).

A livello tecnico, sono state analizzate le architetture più recenti della famiglia YOLO, in particolare le versioni YOLOv10 (s e m), alle quali sono state applicate modifiche strutturali e di configurazione basate su pubblicazioni scientifiche degli ultimi anni, con l'intento di migliorarne le capacità di rilevamento su target di dimensioni ridotte. Successivamente, queste modifiche sono state implementate anche nella versione YOLOv12 (s e m), al fine di verificare l'operabilità di questo modello e la valutazione delle sue performance. Nei capitoli successivi verranno illustrati nel dettaglio gli approcci adottati, le scelte tecniche e i riferimenti agli studi che hanno guidato l'intero lavoro.

2 Stato dell'arte

Nell'ambito della detection di oggetti, nel corso dello sviluppo scientifico delle architetture e dei modelli, sono state definite due grandi famiglie di architetture volte alla risoluzione di questo task: i modelli *Two Stage Detector* e i modelli *One Stage Detector*.

I modelli *Two Stage Detector* svolgono un lavoro di detection suddividendo il task in due fasi principali: la prima fase, detta proposal generation, consiste nella generazione di proposte di porzioni d'immagine (ROI, Region of Interest) nelle quali potrebbero essere presenti oggetti. Successivamente, nella fase di regression and classification, le proposte vengono affinate attraverso la regressione dei bounding box e la classificazione degli oggetti al loro interno, in base alle categorie definite per il compito di detection. I modelli Two Stage offrono generalmente una maggiore precisione e una migliore capacità di localizzazione degli oggetti, soprattutto in scenari complessi o con oggetti molto piccoli. Tuttavia, questo si traduce in un costo computazionale elevato e in tempi di inferenza più lunghi, che ne limitano l'impiego in applicazioni real-time. Esempi di rete appartenenti a questa famiglia sono RCNN, Fast RCNN e Faster RCNN.

L'altra famiglia di modelli per la detection di oggetti è *One Stage Detector* che, invece di fare prima una generazione delle ROI e poi classificare e definire i bounding box, suddivide l'intera immagine in una griglia e per ogni cella il modello predice direttamente le classi, le coordinate dei bounding box e la relativa confidenza. In questo modo, la rilevazione avviene in un singolo passaggio, senza la fase preliminare di selezione delle regioni di interesse. Al contrario della famiglia *Two Stage Detector*, i modelli *One Stage Detector* sono progettati per garantire velocità e semplicità di utilizzo, sacrificando in parte l'accuratezza, soprattutto quando si tratta di oggetti di piccole dimensioni o immersi in contesti rumorosi. La loro architettura compatta li rende però ideali per applicazioni che richiedono rapidità di risposta. Esempi di reti appartenenti a questa branca sono quelli della famiglia *YOLO* (You Only Look Once) e *SSD* (Single Shot MultiBox Detector).

In particolare, questo progetto ha preso in considerazione i modelli *YOLOv10* e *YOLOv12* che presentano alcune rilevanti innovazioni rispetto ai modelli precedenti. *YOLOv10* è stata progettata al fine di migliorare l'efficienza del rilevamento in scenari real-time. Si distingue per l'introduzione del NMS-free training, che consente al modello di gestire le sovrapposizioni tra oggetti già durante l'addestramento, riducendo i tempi di inferenza e semplificando la pipeline. Inoltre, integra il Partial Self-Attention, un meccanismo che permette di migliorare la capacità del modello di cogliere informazioni globali mantenendo basso il costo computazionale. A differenza del precedente modello, *YOLOv12* è progettata per migliorare la capacità di rilevamento in scenari complessi senza sacrificare la velocità. Introdotto nel 2025, adotta l'Area Attention, che consente una gestione più efficace di oggetti piccoli o sovrapposti, e un backbone ottimizzato per garantire un buon equilibrio tra precisione e leggerezza. Con queste innovazioni, YOLOv12 rafforza la capacità di generalizzazione mantenendo alte prestazioni in tempo reale.

3 Metodi e materiali

3.1 Dataset

Il dataset impiegato per il progetto è un SAR Dataset di rilevamento di navi per il deep learning in condizioni di sfondo complesse, realizzato da esperti SAR. Si è scelto questo dataset perché, a differenza di altri, è open source e, dai risultati sperimentali condotti dagli stessi esperti, risulta che i rilevatori di oggetti raggiungono una precisione media (“mean average precision”) più elevata e mostrano una maggiore capacità di generalizzazione.

Il dataset originale SAR comprendeva diversi sottogruppi; tra questi sono stati esclusi quelli basati su immagini *Sentinel-1*, la cui risoluzione spaziale relativamente grossolana ne compromette l'affidabilità nel rilevamento di imbarcazioni di piccole dimensioni. Di conseguenza, sono stati mantenuti unicamente i sottodataset *S2_Detection*, *S2_FC* e *SDAI*, tutti basati su immagini ad alta risoluzione (Figura 1) e in grado di garantire prestazioni superiori nel riconoscimento di navi in scenari con sfondo complesso.

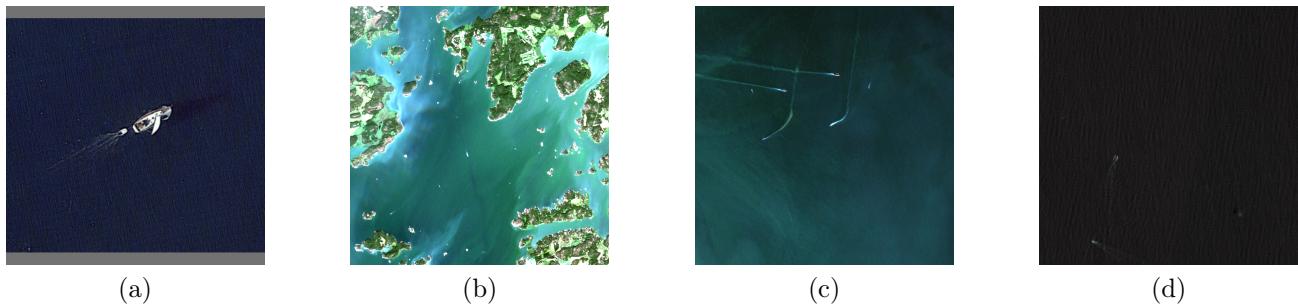


Figura 1: Esempi di immagini nel dataset

Inoltre, nel sottodataset *SDAI*, in cui alcuni campioni derivano da riprese effettuate con drone, è stato effettuato un preprocessing aggiuntivo: sono state eliminate le immagini in cui le imbarcazioni risultavano troppo grandi, in quanto avrebbero introdotto un'eccessiva varianza di scala e avrebbero potuto sbilanciare l'addestramento dei modelli, compromettendone l'efficacia. Alcuni esempi di immagini eliminate sono riportati in Figura 2 .

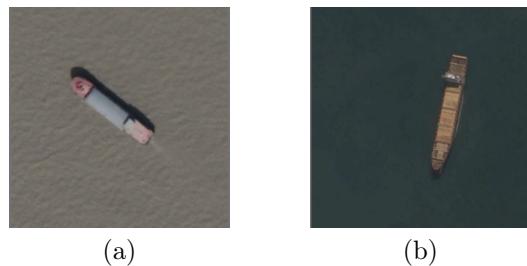


Figura 2: Esempi di immagini eliminate



Per incrementare ulteriormente la varietà e la dimensione del dataset, è stata introdotta una fase di data augmentation strutturata in più passaggi, il cui risultato è presente in *Figura 3*. In primo luogo, tutte le immagini sono state uniformate a una risoluzione di 640×640 pixel, in modo da garantire omogeneità in input ai modelli. Successivamente è stata applicata una prima rotazione casuale in senso orario compresa tra 10° e 35° , al fine di riprodurre differenti orientamenti delle imbarcazioni. Al dataset così ottenuto è stata quindi applicata una seconda rotazione nell'intervallo 35° - 95° , per estendere ulteriormente la copertura angolare. Infine, l'intero insieme di immagini modificato è stato specchiato, in modo da raddoppiare il numero di esempi e migliorare la capacità del modello di generalizzare a contesti di visuale differenti.

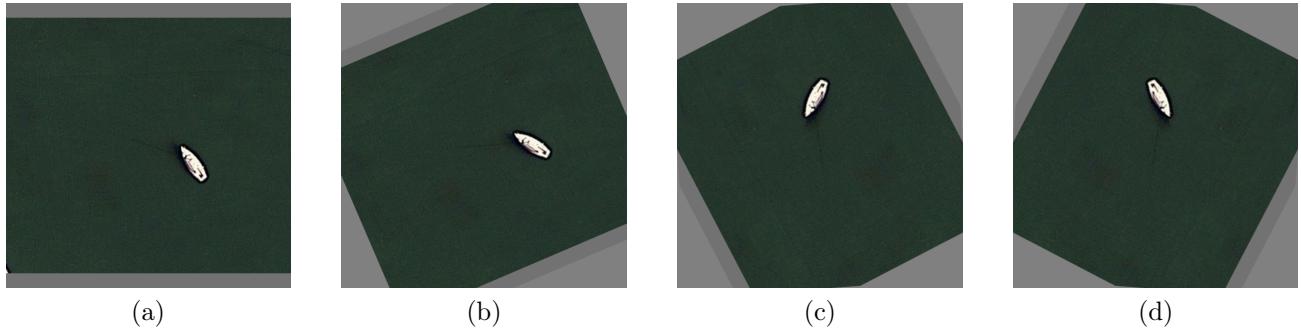


Figura 3: Data augmentation applicata al dataset

3.2 Configurazione

Nel nostro studio, il primo passo è stato creare un ambiente sperimentale solido configurando un container Docker. Questa containerizzazione ha fornito una piattaforma consistente e isolata per condurre i nostri esperimenti includendo l'utilizzo delle capacità computazionali di una CPU Intel(R) Xeon(R) Gold 6254 @ 3.10 GHz e di una GPU NVIDIA RTX A6000 per i nostri addestramenti. Successivamente, abbiamo preparato il nostro dataset per la fase di training suddividendo le immagini e le relative etichette in set di addestramento (70%), validazione (15%) e test (15%), garantendo un equilibrio e una corrispondenza appropriati tra immagini ed etichette (Tabella 1).

Dataset	Train	Val	Test	Totale	% Complessivo
S2_DETECTION	3500	750	750	5000	35,15%
S2_FC	3500	750	750	5000	35,15%
SDAI	2950	642	634	4226	29,71%
Totale immagini	9950	2142	2134	14226	100%

Tabella 1: Ripartizione del dataset in train, val e test

4 Confronto tra architetture YOLO

In questo capitolo verranno indagate nello specifico le architetture YOLO che sono state utilizzate nel progetto andando a porre enfasi sulle strutture generali, il funzionamento dei blocchi, le varie configurazioni e le prestazioni ottenute in seguito ai test svolti.

4.1 YOLOv10

Il punto di partenza del nostro lavoro è stato YOLOv10, un modello che rappresenta una delle evoluzioni più significative all'interno della famiglia YOLO. Come già accennato, questa architettura ha introdotto importanti innovazioni sia a livello di design che di efficienza computazionale. Nel nostro progetto abbiamo scelto di utilizzare le versioni s e m di YOLOv10, privilegiando queste varianti leggere per la loro rapidità di esecuzione, a discapito di una minima perdita in accuratezza.

Abbiamo inoltre definito due configurazioni specifiche per ciascuna di queste versioni, denominate rispettivamente Light e Medium, differenziate in base ai livelli di feature map utilizzati dalla testa di detection: nella configurazione *Light*, la testa riceve in input esclusivamente le feature map provenienti dal livello P3, più adatto alla rilevazione di oggetti piccoli. Nella configurazione *Medium*, oltre a P3, viene attivato anche il livello P4, consentendo di gestire oggetti di dimensioni intermedie. Di seguito possiamo analizzare l'intera architettura del modello:

- *Backbone*: Nel backbone dell'architettura si possono osservare 3 blocchi convoluzionali, 2 blocchi C2f, 2 blocchi SCDown e 1 blocco C2fCIB. I *blocchi convoluzionali* permettono di svolgere le operazioni di convoluzione e di estrazione delle feature a diverse scale con stride pari a 2 dimezzando la dimensionalità dell'input. Il blocco *C2f* prende un input, lo divide in più gruppi di canali, applica convoluzioni su alcuni di essi e poi concatena tutto in uscita. Il blocco *SCDown (Spatial Convolution Downsampling)* combina una convoluzione standard con stride 2 con un'operazione di pooling e integra la concatenazione delle due uscite (dalla convoluzione e dal pooling) migliorando la trasmissione di informazioni al blocco successivo. Infine, il blocco *C2fCIB* è una combinazione del blocco C2f e *CIB (Conditional Information Bottleneck)* che applica un meccanismo che adatta dinamicamente la quantità di informazione trasmessa, regolando il passaggio delle feature in base al contenuto e funge quindi, da filtro per le feature che potrebbero essere più rilevanti per la detection.
- *Neck*: nel neck dell'architettura si osserva la presenza di un blocco *C2f*, tre blocchi *C2fCIB*, due blocchi di upsampling e quattro blocchi di concatenazione, oltre a un blocco *PSA* e un blocco *SPPF*. Quest'ultimo, lo *SPPF (Spatial Pyramid Pooling - Fast)*, è ereditato dalla versione 8 di YOLO (SPP) e realizza un'operazione di pooling con kernel di diverse dimensioni, concatenando le mappe estratte. Questo consente al modello di ampliare la capacità di catturare informazioni su scale differenti, migliorando la rilevazione di oggetti

di dimensioni medio-grandi. Il modulo *PSA* (*Partial Self-Attention*) posto successivamente, divide i canali della feature map in due gruppi: uno viene elaborato da convoluzioni standard, mentre l'altro attraversa un blocco di Self-Attention. I due output vengono poi riuniti tramite concatenazione e fusi con una convoluzione puntuale. In questo modo, il *PSA* permette al modello di catturare informazioni globali mantenendo basso il costo computazionale, risultando particolarmente efficace nei layer profondi a bassa risoluzione. Completano la struttura i blocchi di upsampling e concatenazione, che ripristinano la dimensionalità delle feature map fondendole con i livelli inferiori (P3 e P4). Si nota infine la presenza di un blocco convoluzionale, due blocchi *C2fCIB* e di uno *SCDown* nel percorso *bottom-up*, responsabili di ridurre la risoluzione spaziale preservando informazioni critiche grazie alla combinazione di convoluzione e pooling.

- *Head*: la head prevede la predizione delle mappe di caratteristiche su tre diverse scale, con l'obiettivo di rilevare oggetti che possono variare per dimensione e scala. Questo approccio garantisce una rilevazione accurata e sensibile alle differenti scale presenti nelle immagini.

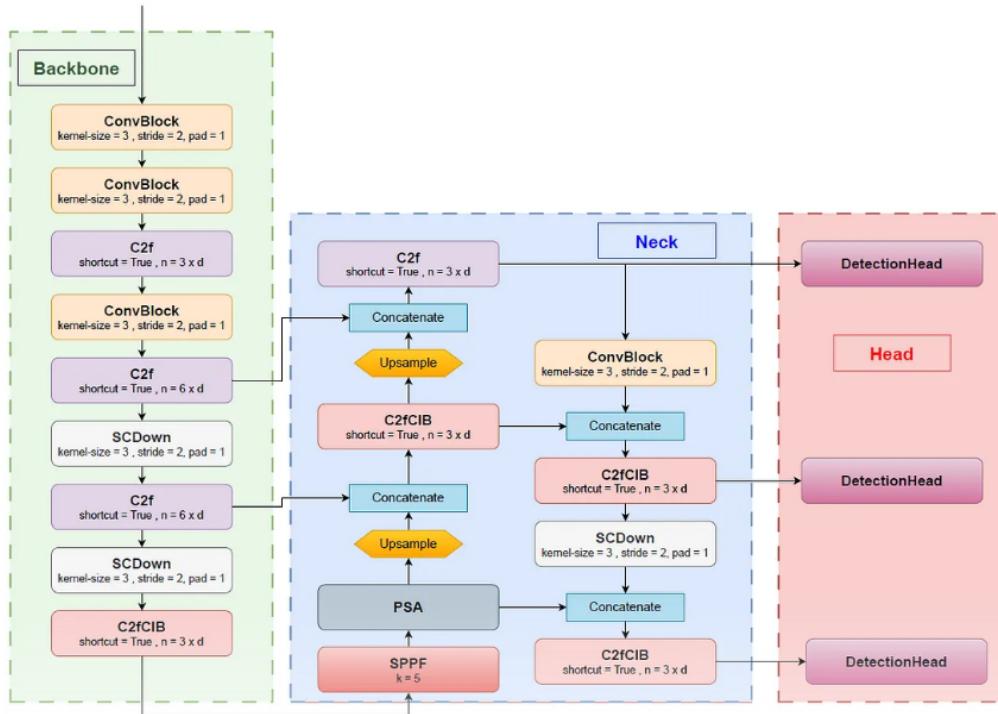


Figura 4: Architettura di YOLOv10

4.2 YOLOv12

Procedendo con il nostro lavoro, abbiamo scelto di adottare anche YOLOv12, nelle varianti s e m, realizzando per entrambe due configurazioni distinte: *Light* e *Medium*. Le due configurazioni si differenziano principalmente per l'impiego della testa di rilevamento, con Light basata sulla sola testa P3 e Medium sull'utilizzo in aggiunta della testa P4, seguendo lo stesso approccio già applicato ai modelli precedenti.

- *Il modulo R-ELAN (Residual Efficient Layer Aggregation Network)*: è evoluzione del classico ELAN, è stato progettato per migliorare la stabilità del training nei modelli di grandi dimensioni e in quelli che fanno largo uso di meccanismi di attenzione. Integra connessioni residue scalate, che agevolano la propagazione dei gradienti e favoriscono la convergenza. La struttura di aggregazione delle feature è stata ripensata attorno a un collo di bottiglia, con l'obiettivo di ridurre la ridondanza informativa e aumentare l'efficienza. Inoltre, in fase di inferenza, il modulo può essere riparametrizzato, consolidando la sua complessità in un'unica convoluzione equivalente, così da mantenere elevate le prestazioni senza introdurre costi computazionali aggiuntivi.
- *Ottimizzazione dei blocchi di attenzione*: Per garantire la compatibilità con le esigenze real-time, YOLOv12 ha rivisto il design dei moduli di attenzione, integrando FlashAttention, un meccanismo che accelera i calcoli riducendo l'uso della memoria grazie a una gestione ottimizzata delle matrici di attenzione, calcolate a blocchi senza necessità di memorizzazione completa. Contestualmente, sono state eliminate le codifiche posizionali, semplificando la rete e migliorandone la rapidità, senza compromettere la capacità di catturare le relazioni spaziali tra le feature.
- *Blocchi C3K2 e A2C2f*: In questa nuova versione sono stati aggiunti dei blocchi che lavorano idfferentemente rispetto alle versioni precedenti evidenziando un aumento delle performance. Il blocco *C3K2* rappresenta un'evoluzione del classico modulo C3, con cui condivide la struttura a percorsi multipli e la concatenazione finale. La principale differenza risiede nell'uso di convoluzioni 3×3 modificate, pensate per ampliare il campo recettivo mantenendo contenuto il costo computazionale. Questo blocco conserva l'efficienza del design originale ma garantisce una migliore trasmissione del gradiente e un miglior equilibrio tra profondità e leggerezza. Il blocco *A2C2f* integra invece il meccanismo di Area Attention unita alla struttura del C2f. In questo caso la feature map viene suddivisa in sotto-regioni (tipicamente 4×4) sulle quali viene applicata un'attenzione locale sfruttando l'implementazione di FlashAttention per ridurre il consumo di memoria e accelerare il calcolo. La parte convoluzionale segue la logica del C2f, con percorsi paralleli e concatenazione delle feature permettendo al blocco di combinare attenzione spaziale e convoluzioni tradizionali. Questo lo rende particolarmente efficace nel catturare informazioni contestuali senza incidere significativamente sulle prestazioni del modello.

4.3 Confronto performance tra YOLOv10 e YOLOv12

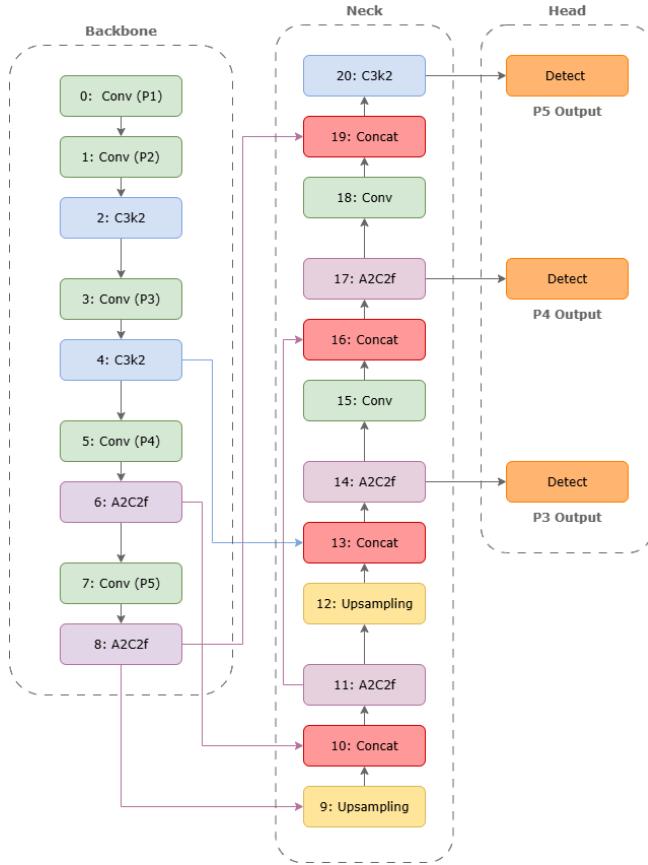


Figura 5: Architettura di YOLOv12

4.3 Confronto performance tra YOLOv10 e YOLOv12

Di seguito verranno esposti i risultati dei training svolti per confrontare le prestazioni dei modelli YOLOv10 (s e m nelle configurazioni Light e Medium) e YOLOv12 (s e m nelle configurazioni Light e Medium).

Modello	Precision	Recall	mAP50	mAP50-95	Parameters	GFLOPs
YOLOv10sLight	0,688	0,553	0,62	0,271	4.930.866	19,4
YOLOv10sMedium	0,689	0,563	0,634	0,277	6.057.140	23
YOLOv10mLight	0,704	0,578	0,642	0,281	11.437.218	54,9
YOLOv12sLight	0,691	0,589	0,643	0,273	4.514.321	15,3
YOLOv12sMedium	0,719	0,593	0,652	0,281	5.201.490	17,5
YOLOv12mLight	0,732	0,587	0,649	0,281	13.276.817	56,7

Tabella 2: Confronto tra i modelli utilizzati di YOLOv10 e YOLOv12

4.3 Confronto performance tra YOLOv10 e YOLOv12

Nella Tabella 2 si possono osservare i risultati ottenuti dai test sui training effettuati su i suddetti modelli (con relative configurazioni) di YOLOv10 e YOLOv12. L’analisi comparativa mostra che YOLOv12 raggiunge sistematicamente valori di precisione e recall superiori rispetto ai corrispondenti modelli YOLOv10. In particolare, *YOLOv12mLight* registra la precisione massima (0,732), mentre il valore più alto di recall (0,593) spetta a *YOLOv12sMedium*. Sul mAP50, YOLOv12 ottiene valori leggermente migliori in tutte le configurazioni, confermando una maggiore affidabilità nella corretta individuazione degli oggetti al superamento della soglia minima di IoU. Per quanto riguarda la capacità di generalizzazione misurata tramite mAP50-95, le due famiglie si equivalgono nelle versioni di fascia media (0,281 sia per YOLOv10mLight sia per YOLOv12sMedium e YOLOv12mLight), evidenziando che le differenze si riducono quando si richiede un’accuratezza più fine sulle predizioni. Sul piano della complessità computazionale, YOLOv12 si distingue per una maggiore efficienza: a parità di configurazione presenta un numero di parametri inferiore rispetto a YOLOv10, fatta eccezione per la versione *mLight*, dove i parametri e i GFLOPs risultano leggermente superiori in YOLOv12.

Questi risultati sono frutto delle innovazioni precedentemente descritte che sono state introdotte nella nuova versione 12 di YOLO come l’ottimizzazione dei blocchi C2f, l’introduzione della Flash Attention o il minor numero di layer presenti nella rete.

5 Ottimizzazione dei modelli

Dopo aver confrontato le prestazioni dei modelli *YOLOv10* e *YOLOv12*, la fase successiva del nostro lavoro si è concentrata sull'esplorazione di possibili interventi architetturali, con l'obiettivo di migliorare le performance delle rispettive baseline.

5.1 Modifiche e Integrazioni Architetturali

Abbiamo adottato come base la versione *mLight*, applicando interventi mirati su *YOLOv10* e successivamente su *YOLOv12*, ispirati a soluzioni presenti in letteratura e focalizzati sull'ottimizzazione del rilevamento di oggetti di piccole dimensioni. Tutte le modifiche sono state condivise tra le due architetture, ad eccezione del modulo *ECSPP*, impiegato solo su *YOLOv10*. Le principali integrazioni introdotte sono descritte nei paragrafi seguenti.

5.1.1 SPDConv - Space-to-Depth Convolution

Il modulo *SPDConv* è stato introdotto per potenziare l'estrazione di caratteristiche da oggetti di piccole dimensioni, come quelli presenti nelle immagini satellitari. A differenza delle tradizionali convoluzioni con stride o pooling, che riducono la risoluzione spaziale ma causano perdita di dettagli, *SPDConv* applica una trasformazione *space-to-depth* seguita da una convoluzione standard. In questo modo, la riduzione spaziale avviene trasferendo le informazioni nella dimensione dei canali, preservando i dettagli fini e migliorando la capacità di rilevamento su target di piccola scala.

Nel nostro approccio, abbiamo sostituito con *SPDConv* tutti i blocchi CBS di downsampling del backbone ad eccezione del primo, mantenuto invariato per garantire una corretta elaborazione delle feature iniziali. Questa modifica ha permesso di ridurre la perdita di informazioni nelle fasi precoci della rete, migliorando la sensibilità verso oggetti piccoli senza compromettere la profondità del modello.

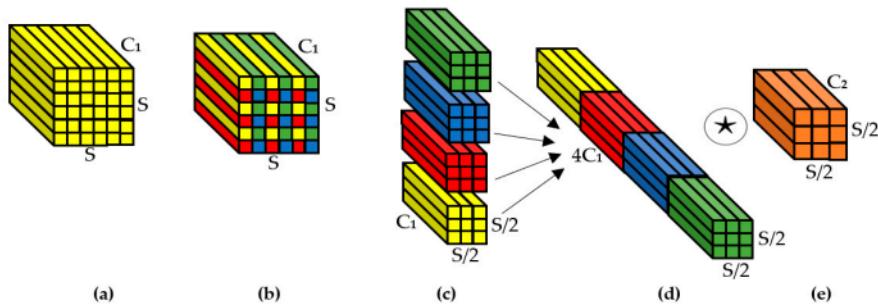


Figura 6: Schema del funzionamento del modulo *SPDConv*

5.1.2 SPANet - Small Path Aggregation Network e sblocco Testa P2

Nel design classico di *YOLO*, il *Path Aggregation Network* (PANet) costituisce il *neck* della rete, responsabile della fusione delle feature map provenienti dai diversi livelli del *backbone*. Tuttavia, questa fusione risulta spesso insufficiente nel rilevamento di oggetti di piccole dimensioni. Per questo motivo, abbiamo modificato la struttura del *neck* adottando una variante denominata *SPANet* (*Small Path Aggregation Network*), che estende il PANet tradizionale integrando anche *P2*, la feature map di livello più basso generata dal *backbone*.

Il livello *P2*, ricco di dettagli spaziali fini, viene inserito nel percorso di aggregazione insieme ai livelli *P3*, *P4* e *P5*, consentendo di combinare informazioni semantiche profonde e dettagli ad alta risoluzione. In particolare, le feature map *P2'*, *P3'* e *P3* vengono concatenate per ottenere *P3''*, che rappresenta una sintesi delle informazioni spaziali e semantiche.

Inizialmente, la detection veniva effettuata solo su *P3''*. Successivamente, per migliorare ulteriormente la sensibilità verso oggetti di piccola scala, abbiamo incluso anche *P2'*, permettendo al modello di sfruttare direttamente i dettagli più fini.

Inoltre, nella sezione bottom-up del *SPANet*, abbiamo sostituito la convoluzioni standard con moduli *SPDConv*, migliorando la precisione del rilevamento e riducendo il numero complessivo di parametri.

Questa modifica ha aumentato la robustezza del modello e la sua efficacia nel rilevamento di oggetti di piccole dimensioni.

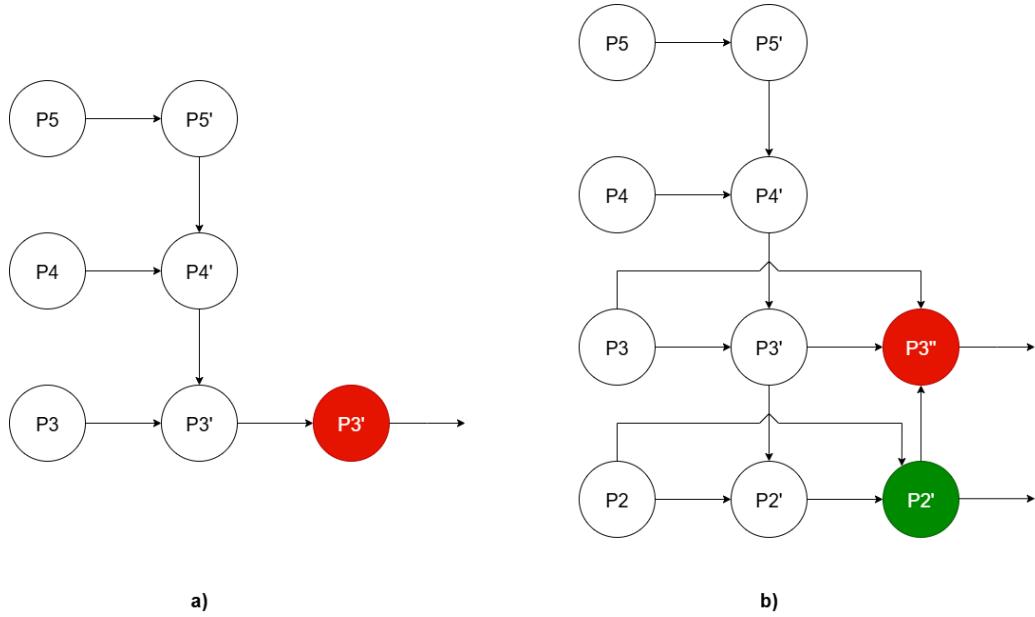


Figura 7: Path aggregation network architecture; (a) PANet; (b) SPANet.

5.1.3 Modulo ECSPP

Il modulo *ECSPP*, presente in *Figura 8*, è un modulo innovativo che combina:

- *ECA-Net* (Efficient Channel Attention), un sofisticato meccanismo di attenzione leggera che migliora la discriminazione delle feature ricalibrando i singoli canali nelle feature map sulla base del contesto spaziale locale;
- *SPPCSPC* (Spatial Pyramid Pooling Cross Stage Partial Conv), ereditato da YOLOv7, per l'estrazione di feature a diverse scale.

Questa fusione potenzia la capacità del backbone di catturare informazioni semantiche multi-scala, migliorando la localizzazione e l'identificazione di oggetti, soprattutto di piccole dimensioni.

Dal punto di vista operativo, il modulo *ECSPP* applica il meccanismo di attenzione *ECA-Net* direttamente sulla feature map in ingresso, al fine di potenziare l'efficacia e la precisione dell'elaborazione dell'immagine. Successivamente, il flusso dei dati viene suddiviso in due percorsi distinti: uno per le operazioni convoluzionali e l'altro dedicato alla struttura SPP. Le due componenti vengono poi riunite tramite concatenazione, ulteriormente raffinate da blocchi convoluzionali aggiuntivi e infine restituite come output del modulo.

Nel nostro progetto, abbiamo sperimentato l'integrazione del modulo *ECSPP* al posto del modulo *SPPF* esclusivamente alla fine del backbone di *YOLOv10*, con l'obiettivo di valutarne l'impatto sulle performance di detection.

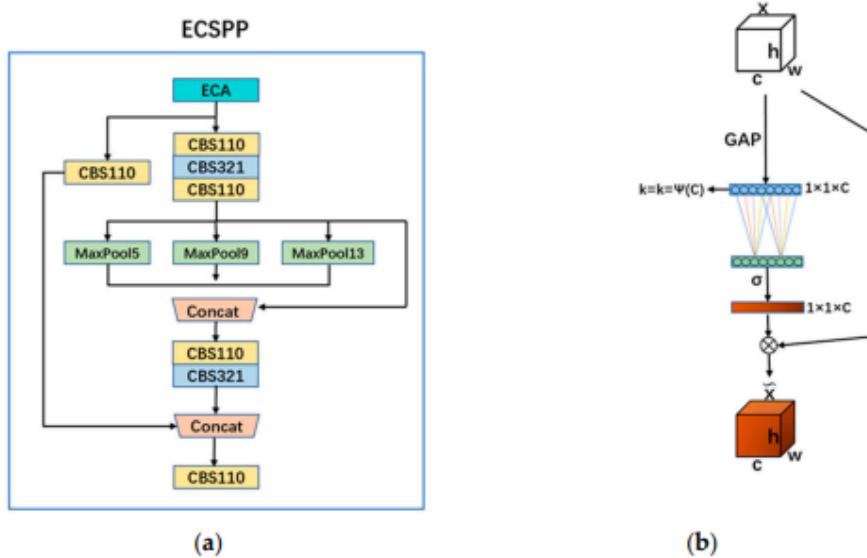


Figura 8: (a) ECSPP; (b) ECA-Net.

5.1.4 Modulo *DLC2f*

Il modulo *DLC2f*, presente in *Figura 9*, rappresenta una versione evoluta del classico blocco *C2f*, progettata per rafforzare la capacità della rete di estrarre informazioni utili da oggetti di piccole dimensioni. Questa evoluzione si ottiene sostituendo il secondo blocco di CBS del bottleneck, all'interno del percorso principale, con un *DL-Block* (Deformable Large Kernel Block), che combina strategie avanzate di estrazione delle feature.

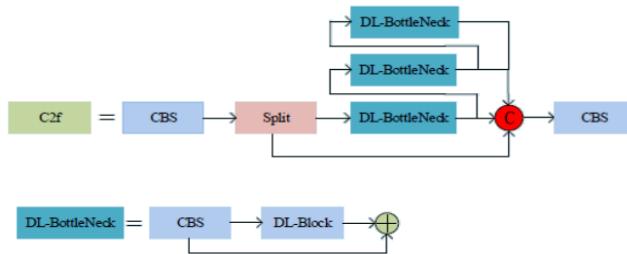
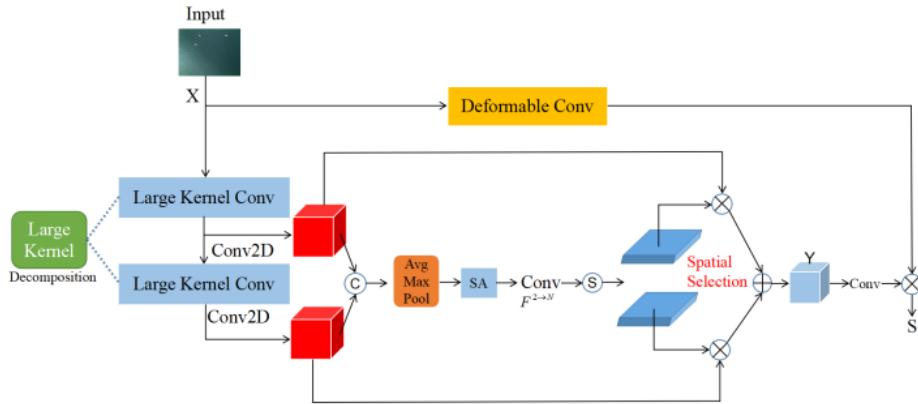


Figura 9: Struttura del modulo *DLC2f*.

Nel *DL-Block*, illustrato in *Figura 10*, le principali innovazioni consistono nell'utilizzo congiunto di:

- *Large Kernel Convolution*: impiegate per ampliare il campo recettivo, favorendo la raccolta di informazioni spaziali e contestuali utili al riconoscimento di oggetti di piccole dimensioni.
- *Deformable Convolution*: utilizzate per adattare dinamicamente il processo di campionamento alle forme e alle geometrie presenti nelle immagini, superando i limiti delle convoluzioni standard a kernel fisso.

La struttura del *DL-Block* prevede una prima fase di estrazione delle feature tramite una *Deformable Convolution*, che consente al kernel di adattarsi dinamicamente al contenuto locale dell'immagine. In parallelo, sono presenti due percorsi distinti basati su *Large Kernel Convolution*, che permettono di ampliare il campo recettivo mantenendo basso il numero di parametri. Le feature prodotte da questi percorsi vengono proiettate tramite convoluzioni 1×1 e concatenate. Su questa concatenazione viene applicato un meccanismo di attenzione spaziale, realizzato tramite pooling, convoluzione e funzione sigmoide, che genera due maschere di attenzione, una per ciascun percorso. Le feature di ciascun ramo large kernel vengono moltiplicate per la rispettiva maschera di attenzione e successivamente sommate. Questa somma viene ulteriormente raffinata tramite una convoluzione aggiuntiva 1×1 . Infine, il risultato viene moltiplicato elemento per elemento con l'output della deformable convolution, generando una rappresentazione focalizzata e adattiva delle regioni rilevanti dell'immagine.

Figura 10: Struttura del *DL-Block*.

A livello di backbone, il *DLC2f* è stato integrato selettivamente nei livelli *P2* e *P3*, tipicamente i più sensibili alla presenza di piccoli oggetti. In questo modo, la rete dovrebbe migliorare sia la raccolta di dettagli spaziali sia la precisione nella localizzazione dei piccoli target.

5.2 Scelta degli iperparametri

Gli iperparametri adottati per l’addestramento dei modelli, Tabella 3, non sono stati selezionati in maniera casuale, ma derivano da un tuning accurato effettuato in un progetto precedente. Questa combinazione, già dimostrata ottimale in termini di compromesso tra performance e velocità, è stata mantenuta inalterata per la presente implementazione a causa delle tempistiche ristrette e delle risorse limitate.

Parametro	Valore
epochs	100
batch	64
optimizer	SGD
lr ₀	0.01
lrf	0.0001
momentum	0.9
weight_decay	0.0005
warmup_epochs	3

Tabella 3: Configurazione degli iperparametri

In uno scenario ideale, avremmo lanciato il comando `yolo tune` di Ultralytics YOLO per avviare automaticamente un processo di hyperparameter evolution basato su algoritmi genetici. Partendo da un set iniziale di valori - scelti tra i default di Ultralytics o derivati da esperienze precedenti - il framework applica ripetute mutazioni casuali agli iperparametri principali (learning rate, momentum, regolarizzazione, dimensione del batch, numero di epoch) mediante il

metodo interno `_mutate`. Questa operazione, gestita automaticamente dalla classe `Tuner`, genera nuovi insiemi di iperparametri da sottoporre all’addestramento. Dopo ogni training, le prestazioni del modello vengono valutate in termini di *AP50* o *F1-score* e i relativi risultati, insieme alla configurazione utilizzata, vengono registrati in un file *CSV*.

Al completamento del processo di ottimizzazione, il sistema restituisce una serie di diversi file e directory che racchiudono i risultati della regolazione, tra cui il file *best_hyperparameters.yaml* che contiene la combinazione di iperparametri più performanti trovati durante il processo di sintetizzazione. Tale file può quindi essere utilizzato per inizializzare i futuri addestramenti con le impostazioni ottimizzate, garantendo così la migliore configurazione possibile sulla base delle sperimentazioni svolte.

5.3 Risultati

5.3.1 Analisi delle Performance

A seguito dei diversi test condotti sui modelli e delle varie implementazioni sviluppate, sono state raccolte tutte le metriche di performance ottenute, al fine di presentarle in modo chiaro e consentire un confronto diretto tra i risultati in modo tale da poter trarre conclusioni oggettive.

Nella Tabella 4 si possono osservare i risultati ottenuti dai modelli di YOLOv10.

Modello	Precision	Recall	mAP50	mAP50-95	Parameters	GFLOPs
YOLOv10mLight	0,704	0,578	0,642	0,281	11.437.218	19,4
YOLOv10mLight-SPD-Conv	0,711	0,575	0,642	0,281	22.124.817	77,5
YOLOv10mLight-SPD-SPANet	0,725	0,581	0,652	0,284	22.529.730	93,2
YOLOv10mLight-SPD-SPANet-ECSPP	0,717	0,587	0,650	0,285	30.664.581	99,7
YOLOv10mLight-SPD-SPANet-P2	0,746	0,704	0,728	0,314	22.669.700	104,1
YOLOv10mLight-SPD-SPANet-P2-DLC2f	0,723	0,694	0,721	0,320	23.089.748	105

Tabella 4: Risultati dei modelli YOLOv10

Partendo dalla *YOLOv10mLight*, che abbiamo scelto come modello di riferimento, possiamo osservare come ogni successiva estensione influisca sulle quattro metriche chiave: Precision, Recall, mAP50 e mAP50-95.

La prima evoluzione con *SPD-Conv* migliora lievemente la precision passando da 0,704 a 0,711, un incremento di 0,994%, ma comporta un leggero calo del recall, che scende da 0,578 a 0,575 (-0,519%). I valori di mAP50 e mAP50-95 restano invece invariati (rispettivamente 0,642 e 0,281), mostrando che il semplice inserimento della convoluzione SPD contribuisce a rinforzare unicamente la capacità del modello di non commettere falsi positivi, a scapito della sua sensibilità complessiva.

Con l’aggiunta di *SPANet*, la precision raggiunge 0,725 (+2,983%) mentre il recall risale a 0,581 (+0,519%), indicando un bilanciamento più favorevole fra accuratezza e copertura. Parallelamente, mAP50 cresce a 0,652 (+1,558%) e mAP50-95 a 0,284 (+1,068%), evidenziando un miglioramento generalizzato nella qualità delle previsioni.



L'introduzione di *ECSPP* all'interno della pipeline *SPD-SPANet* comporta un aumento moderato di tutte le metriche rispetto alla baseline: precision a 0,717 (+1,847%), recall a 0,587 (+1,557%), mAP50 a 0,650 (+1,246%) e mAP50-95 a 0,285 (+1,423%). Rispetto alla sola *SPANet*, *ECSPP* tende a enfatizzare leggermente il recall e, soprattutto, la robustezza sulle soglie più severe (mAP50-95), ma nel complesso non raggiunge i livelli di performance ottenuti dal modello senza *ECSPP*. In particolare, la crescita di precision e mAP50 risulta inferiore rispetto agli incrementi originali di *SPD-SPANet*, mentre l'aumento di recall, seppur positivo, rimane marginale. Questo indica che l'inserimento di *ECSPP* introduce un leggero sovraccarico computazionale o cambiamenti architetturali che, pur contribuendo a bilanciare sensibilità e robustezza, non compensano appieno il calo di efficacia osservato nelle metriche complessive rispetto al passo precedente.

Il salto più significativo si registra attivando la detection su *P2*, che porta la precision a 0,746 (+5,966%), ma soprattutto incrementa il recall fino a 0,704 (+21,799%). Anche i valori di mAP sono soggetti a una notevole impennata: mAP50 raggiunge 0,728 (+13,396%) e mAP50-95 arriva a 0,314 (+11,744%). Questo insieme di miglioramenti mostra chiaramente come *P2* ampli in modo consistente sia la capacità di individuare oggetti che di confermare le predizioni con maggior sicurezza.

Infine, l'aggiunta di *DLC2f* su *SPD-SPANet-P2* conserva un recall molto elevato (0,694, +20,069%) e spinge ulteriormente la mAP50-95 a 0,320 (+13,879%), sebbene la precision scenda leggermente a 0,723 (+2,699%) e mAP50 a 0,721 (+12,305%). Tuttavia, rispetto al modello precedente *SPD-SPANet-P2*, si osserva un calo sia di precision (da 0,746 a 0,723) sia di mAP50 (da 0,728 a 0,721). Questo peggioramento indica che, nonostante l'incremento della confidenza sulle predizioni di fascia alta, l'introduzione di *DLC2F* comporta un lieve degrado dell'accuratezza di base e della capacità complessiva di discriminazione degli oggetti.

In sintesi, l'analisi percentuale rivela che mentre *SPD-Conv* da sola produce effetti trascurabili, *SPANet* fornisce un miglioramento sensibile e *P2* rappresenta il vero punto di svolta in termini di recall e qualità complessiva delle predizioni. L'ulteriore calibrazione con *DLC2F* mantiene alta la robustezza sulle soglie più stringenti, pur sacrificando una parte di precision.

Nella Tabella 5, invece, si possono osservare i risultati ottenuti dai modelli di *YOLOv12*.

Modello	Precision	Recall	mAP50	mAP50-95	Parameters	GFLOPs
YOLOv12mLight	0,732	0,587	0,649	0,281	13.276.817	56,7
YOLOv12mLight-SPD-Conv	0,719	0,6	0,659	0,29	29.424.65	118,9
YOLOv12mLight-SPD-SPANet	0,744	0,606	0,668	0,292	32.145.553	157,5
YOLOv12mLight-SPD-SPANet-P2	0,77	0,736	0,751	0,325	32.212.818	164,3
YOLOv12mLight-SPD-SPANet-P2-DLC2f	0,76	0,72	0,741	0,318	31.000.490	147,5

Tabella 5: Performance dei modelli YOLOv12 modificati

Dalla tabella 5 si evidenzia che le implementazioni e le modifiche apportate al modello YOLOv12mLight hanno apportato dei notevoli incrementi delle metriche chiave per la valutazione delle performance.

Il modello *YOLOv12mLight* rappresenta la baseline del nostro esperimento, caratterizzato da una buona precisione (0,732) ma con un recall piuttosto basso (0,587) e un mAP50-95 modesto (0,281). Questo indica un modello abbastanza selettivo che però fatica a cogliere tutti i target presenti.

L'integrazione del modulo *SPD-Conv* ha portato a un primo incremento del recall (0,600, + 2,215%), del mAP50 (0,659, + 1,541%) del mAP50-95 (0,290, + 3,203%), ma a scapito di una leggera riduzione della precisione (0,719, - 1,776%). Questo suggerisce che il modello diventa più sensibile nel rilevare oggetti (meno falsi negativi) ma più incline a generare falsi positivi. Nonostante ciò, questo risultato si accompagna a un notevole aumento della complessità computazionale, con i parametri e i GFLOPs più che raddoppiati.

Aggiungendo successivamente *SPANet*, si osserva un miglioramento generale: la precisione sale a 0,744 (+ 1,639%) e il recall cresce ulteriormente a 0,606 (+ 3,237%). Il mAP50-95 raggiunge 0,292 (+ 3,915%). Questi dati indicano che l'aggiunta di SPANet contribuisce a bilanciare la tendenza del solo SPD-Conv ad abbassare la precisione, mantenendo però un miglioramento del richiamo.

Il salto più evidente si registra però con l'introduzione del modulo *P2*. Qui, il modello raggiunge un recall di 0,736 (+ 25,383%) e una precisione di 0,770 (+ 5,191%), con un mAP50-95 che si alza sensibilmente a 0,325 (+ 15,658%). Questo significa che il modello riesce a identificare un maggior numero di oggetti senza compromettere la qualità della previsione. Il *P2* risulta quindi determinante nel potenziare la capacità di rilevamento del modello, soprattutto per oggetti di piccole dimensioni.

Infine, l'inserimento del modulo *DLC2f* sembra portare a un leggero calo generale delle metriche con una precisione che è pari a 0,760 (+ 3,825%), un recall a 0,720 (+ 2,658%), un mAP50 a 0,741 (+ 14,176%) e un mAP50-95 a 0,318 (+ 13,167%). Si registra quindi un incremento rispetto al modello di partenza ma un calo delle performance rispetto all'architettura descritta precedentemente a questa. Nonostante ciò, la configurazione con *DLC2f* è caratterizzata da una riduzione del numero di parametri e dei GFLOPs, rendendola quindi una scelta più efficiente in termini computazionali ma con un piccolo calo sulle performance.

In sintesi, l'integrazione progressiva dei moduli ha portato benefici evidenti, in particolare grazie al contributo di *SPANet* e soprattutto di *P2*. L'adozione del *DLC2f* può essere giustificata laddove si voglia privilegiare un migliore rapporto tra costo computazionale e prestazioni.

5.3.2 Analisi delle Emissioni di CO₂ - CodeCarbon

Per completare la valutazione dei modelli, abbiamo considerato anche le emissioni di CO₂ generate durante l'addestramento delle reti neurali, al fine di valutarne l'impatto ambientale. A tal scopo, abbiamo utilizzato *CodeCarbon*, una libreria che stima le emissioni monitorando il consumo energetico dell'hardware, come CPU e GPU.

Il monitoraggio delle emissioni risulta cruciale per diversi motivi:



- Contribuire alla riduzione dell'impatto ambientale dei modelli e promuovere pratiche di sviluppo più sostenibili.
- Migliorare l'efficienza economica: il consumo energetico incide non solo sull'ambiente, ma anche sui costi operativi. Monitorare i consumi consente di adottare strategie efficaci per contenere sia le emissioni sia le spese.

Nella Tabella 6 sono riportate le emissioni di CO₂ (in kg) generate durante le sessioni di addestramento delle versioni baseline di *YOLOv10m* e *YOLOv12m*, confrontate con le rispettive versioni migliorate, che integrano *SPDConv* e *SPANet* con la testa *P2* abilitata.

Modello	CO ₂ (kg)
YOLOv10mLight	0.631
YOLOv10mLight-SPD-SPANet-P2	0.729
YOLOv12mLight	0.687
YOLOv12mLight-SPD-SPANet-P2	0.971

Tabella 6: Emissioni di CO₂ (kg) generate durante l'addestramento

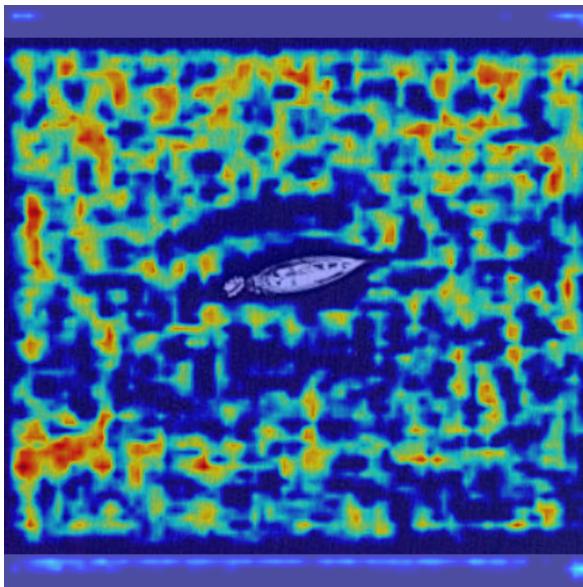
Le emissioni di CO₂ riportate in Tabella 6 mostrano un incremento al crescere della complessità architettonica e delle funzionalità integrate. In particolare, l'introduzione dei moduli *SPDConv*, *SPANet* e della testa *P2* determina un aumento medio delle emissioni di circa il 15% per YOLOv10m e il 40% per YOLOv12m rispetto alle rispettive versioni baseline. Sebbene i valori assoluti possano apparire contenuti, l'impatto complessivo diventa significativo considerando la necessità di ripetere più sessioni di addestramento, soprattutto in ambito di ricerca o su scala industriale.

5.3.3 Explainability - EigenCAM

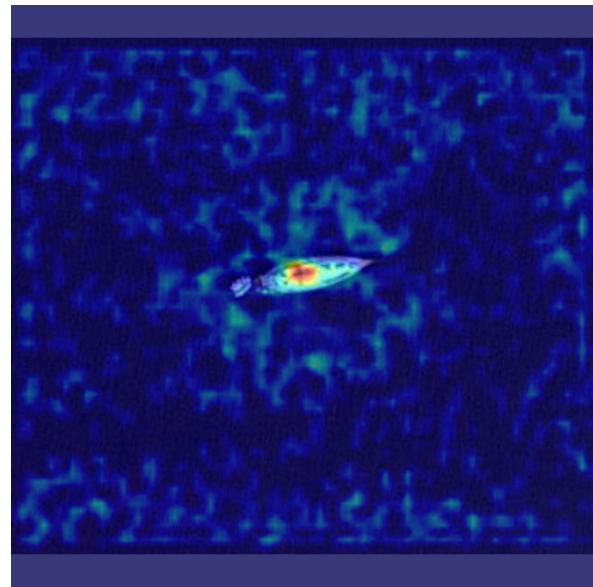
Per analizzare il comportamento dei modelli e visualizzare le aree dell'immagine più rilevanti durante l'inferenza, abbiamo utilizzato la *EigenCAM*.

Il *Class Activation Mapping* (CAM) permette di generare mappe che evidenziano le regioni dell'immagine che influenzano maggiormente la previsione del modello. *EigenCAM* ne rappresenta una variante avanzata, che migliora la stabilità e l'interpretabilità delle mappe rispetto ai metodi tradizionali.

Nella *Figura 11* sono mostrati gli output generati da *EigenCAM* su un'immagine di input. Come si osserva, le aree blu rappresentano le regioni meno rilevanti per la decisione del modello, mentre quelle rosse evidenziano le zone di maggiore influenza.



(a) YOLOv10mLight



(b) Versione modificata

Figura 11: Output di *EigenCAM*

Analizzando i due output ottenuti, si può osservare che mentre *YOLOv10mLight* fatica a focalizzarsi sulle aree di interesse (a), la nostra versione modificata è in grado di isolare con maggiore precisione, migliorando così la capacità di distinguere gli oggetti presenti nelle immagini (b).

6 Conclusioni

Giunti al termine del lavoro, dopo aver analizzato con cura le risorse a disposizione e aver implementato nuovi moduli nelle architetture proposte, combinando tra loro differenti tecniche e strategie operative, è finalmente possibile analizzare i risultati ottenuti e trarre alcune conclusioni sul lavoro svolto.

Come emerso dai risultati, i modelli più performanti sono quelli arricchiti dai moduli SPD-SPANet-P2, che hanno fatto registrare i valori più elevati nelle metriche di valutazione. Al contrario, l'aggiunta del modulo DLC2f non ha portato ad un ulteriore miglioramento delle prestazioni, bensì ha determinato un peggioramento sia nella versione 10 che nella versione 12 di YOLO. Ciò potrebbe essere dovuto all'ulteriore complessità introdotta dal DLC2f che, nonostante sia pensato per potenziare la rilevazione di piccoli oggetti, potrebbe aver generato un'estrazione ridondante o dispersiva delle feature, compromettendo la capacità di generalizzazione. L'impiego combinato di deformable e large kernel convolution potrebbe inoltre aver aumentato il rischio di overfitting, soprattutto su dati complessi come le immagini SAR. La sua integrazione infine, potrebbe aver alterato l'equilibrio già efficace ottenuto dai moduli SPD-SPANet-P2, riducendo le prestazioni complessive. Nella Figura 12 si possono osservare i due modelli più performanti.

Nel complesso l'architettura che si è dimostrata la più efficace è risultata essere YOLOv12-SPD-SPANet-P2, che ha fatto registrare le migliori prestazioni in termini di metriche di valutazione. Si tratta di una rete in grado di individuare correttamente oggetti di piccole dimensioni e mantenere una buona precisione anche su immagini caratterizzate da elevato rumore o background particolarmente complessi.

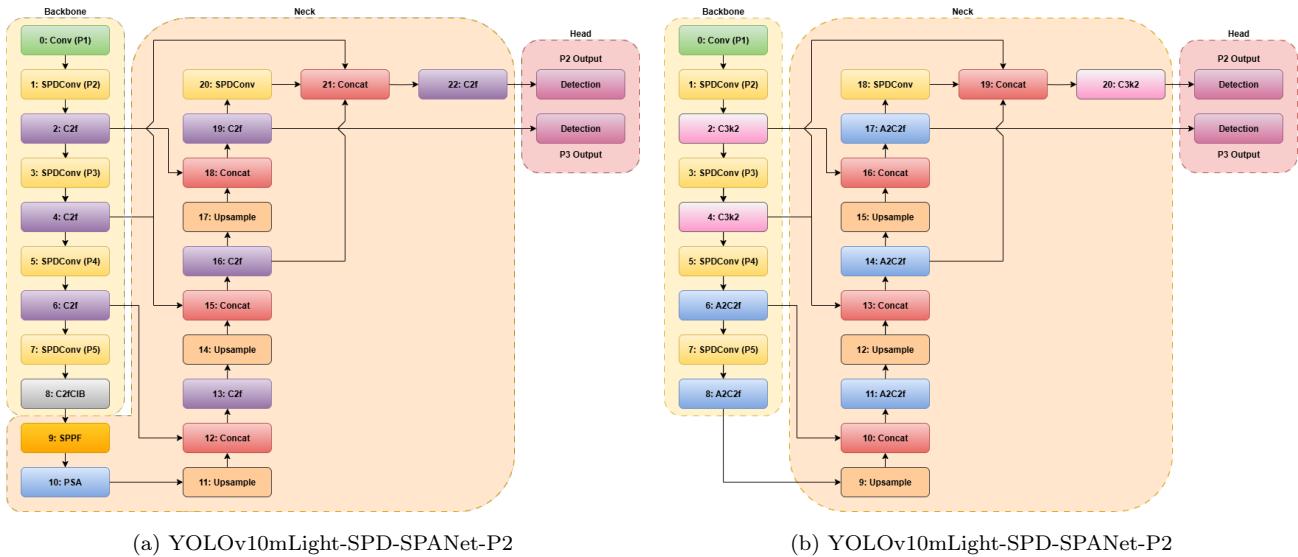


Figura 12: Architetture dei modelli con SPD-SPANet-P2

6.1 Possibili sviluppi futuri

Per gli sviluppi futuri potremmo innanzitutto ampliare il dataset raccogliendo immagini da fonti differenti, in modo da arricchire la varietà e la rappresentatività dei dati di addestramento e migliorare la capacità di generalizzazione in scenari reali.

Un'altra strada promettente potrebbe essere l'integrazione di una branch di Super-Resolution, che durante l'addestramento guida il backbone ad apprendere feature più dettagliate e utili alla rilevazione di oggetti di piccole dimensioni, senza aumentare la complessità in inferenza.

Parallelamente, potrebbe rivelarsi vantaggioso sperimentare funzioni di loss studiate specificamente per il riconoscimento di target minimi, riducendo la perdita di informazioni spaziali critiche. In questo contesto, al posto della tradizionale loss, si può adottare la Wise IoU (WIoU) che introduce un coefficiente di “focalizzazione” non-monotono che assegna pesi maggiori alle anchor di bassa e media qualità, tipicamente quelle coinvolte in piccoli oggetti, e pesi minori alle anchor di alta qualità, rinforzando così l'apprendimento proprio nelle situazioni di maggiore incertezza.

Non va poi trascurata l'opportunità di effettuare un tuning degli iperparametri, ad esempio utilizzando, come citato in precedenza, `yolo tune`, per ottimizzare in modo sistematico learning rate, batch size, numero di epoche e regolarizzazione, con l'obiettivo di massimizzare sia la precisione sia l'efficienza in fase di inferenza.

Inoltre, l'approccio può essere esteso non solo al task di detection ma anche a quello di classificazione: aggiungendo un head dedicato ai tipi di imbarcazione, il modello fornirebbe insieme alle coordinate anche l'etichetta semantica dell'oggetto rilevato.

Infine, l'adozione di moduli all'avanguardia presenti in letteratura, come il Multi-Scale Channel Attention Module (SCAM) o altri meccanismi di attention, potrebbe offrire ulteriori miglioramenti nella fusione e nel filtro delle feature, spingendo ancora più in là i limiti di accuratezza e robustezza del modello.