HUMAN AND OBJECT DETECTION FOR SECURITY SYSTEMS

Giulio Arecco, Erika Astegiano, Alberto Cagnazzo

1. Introduzione

Questo progetto è studiato per imitare il funzionamento di un sistema antifurto. Il sistema è pensato per applicazioni di sicurezza in contesti come la sorveglianza domestica, dove è fondamentale rilevare la presenza di intrusi in modo rapido e accurato.

Il codice sviluppato riceve un flusso video o un feed live da una telecamera e lo analizza fotogramma per fotogramma, utilizzando la libreria OpenCV per l'elaborazione delle immagini. Ogni fotogramma viene migliorato tramite diverse tecniche di image processing. Dopo il miglioramento dell'immagine, un modello di rilevamento basato su YOLOv5, implementato con PyTorch, analizza gli oggetti presenti nel fotogramma.

Gli oggetti riconosciuti sono classificati in tre categorie principali: esseri umani, animali (gatti e cani) e automobili. Quando viene rilevata la presenza di esseri umani, il sistema emette un segnale acustico per avvisare dell'intrusione. Oltre a fornire un metodo di rilevamento in tempo reale, il progetto raccoglie anche statistiche sui rilevamenti, consentendo di valutare le prestazioni del modello su immagini originali, migliorate e peggiorate. Parte del nostro studio si concentra infatti anche sul valutare quanto la presenza di image enhancement nell'immagine vada a influenzare (e eventualmente migliorare) il risultato delle rilevazioni. Ha senso quindi utilizzare dei semplici strumenti di statistica per valutare le prestazioni dello stesso enhancement, quando agisce su immagini qualitativamente peggiorate (tramite operazioni di image processing) e poi successivamente ri-migliorate.

2. Tecnologie utilizzate

2.1 Playsound [1]

Il modulo "playsound" contiene la funzione *playsound*, il cui primo argomento è il *path* del file che si vuole riprodurre come argomento mentre il secondo argomento (opzionale) *block*, è True di default. Se settato a False rende la funzione asincrona.

2.2 Torch [2]

PyTorch è un pacchetto Python che fornisce due funzionalità di alto livello:

- Calcolo con tensori (simile a NumPy) con una forte accelerazione GPU
- Reti neurali profonde costruite su un sistema tape-based autograd system

E’ nota anche come GPU-Ready Tensor Library, poiché fornisce tensori che possono essere eseguiti sia sulla CPU che sulla GPU, accelerando i calcoli in modo significativo. Oltre a questo, PyTorch è utilizzato soprattuto per la costruzione di reti neurali (funzionalità chiave usata per il nostro progetto), che si basano sul concetto di riprodurre le reti come un registratore a nastro. Vi sono poi una serie di ottimizzazioni nel pacchetto riguardanti la flessibilità della rete creata (tecnica di reverse-mode auto-differentiation, che rende possibile cambiare i parametri della rete in tempo reale e senza ritardi) e la memoria utilizzata, con allocatori specializzati, per aumentare la grandezza delle reti addestrabili.

2.3 CV2 [3]

OpenCV (Open Source Computer Vision Library: http://opencv.org) è una libreria open-source che include diverse centinaia di algoritmi di visione artificiale. OpenCV ha una struttura modulare, il che significa che il pacchetto include diverse librerie condivise o statiche. I seguenti moduli sono disponibili:

- **Funzionalità di base (core)**: un modulo compatto che definisce le strutture di dati di base, inclusa la matrice densa multi-dimensionale Mat e le funzioni di base utilizzate da tutti gli altri moduli.
- **Elaborazione delle Immagini (imgproc)**: un modulo di elaborazione delle immagini che include il filtraggio lineare e non lineare delle immagini, le trasformazioni geometriche delle immagini (ridimensionamento, deformazione affina e prospettica, rimappatura generica basata su tabelle), conversione degli spazi di colore, istogrammi, e così via.
- **Analisi Video (video)**: un modulo di analisi video che include algoritmi di stima del movimento, sottrazione dello sfondo e tracciamento degli oggetti.
- **Calibrazione della Fotocamera e Ricostruzione 3D (calib3d)**: algoritmi di base per la geometria multi-visione, calibrazione della fotocamera singola e stereo, stima della posa degli oggetti, algoritmi di corrispondenza stereo e elementi di ricostruzione 3D.
- **Framework di Caratteristiche 2D (features2d)**: rilevatori di caratteristiche salienti, descrittori e confronti di descrittori.
- **Rilevamento di Oggetti (objdetect)**: rilevamento di oggetti e istanze di classi predefinite (ad esempio, volti, occhi, tazze, persone, auto, ecc.).
- **GUI di alto livello (highgui)**: un'interfaccia facile da usare per funzionalità UI semplici.
- **Video I/O (videoio)**: un'interfaccia facile da usare per la cattura di video e codec video.

2.4 YOLOv5 [4]

Ultralytics YOLOv5 è rinomato per le sue capacità di rilevamento degli oggetti ad alta velocità e alta precisione. Costruito su PyTorch, è versatile e facile da usare, rendendolo adatto a vari progetti di visione artificiale. Le caratteristiche principali includono inferenza in tempo reale, supporto per diverse tecniche di allenamento come il Test-Time Augmentation (TTA) e

l'Ensemble di Modelli, e compatibilità con formati di esportazione come TFLite, ONNX, CoreML e TensorRT.

Caratteristiche Chiave di YOLOv5:

- **Architettura Efficiente:** Progettata per bilanciare velocità e accuratezza, YOLOv5 può rilevare oggetti in tempo reale su dispositivi con risorse limitate, come smartphone e droni.
- **Facilità di Addestramento:** Grazie all'integrazione con PyTorch, YOLOv5 consente di addestrare modelli personalizzati su nuovi dataset con relativa facilità.
- **Supporto per Tecniche Avanzate:** Oltre a TTA e Model Ensembling, YOLOv5 supporta altre tecniche moderne come il Mosaic Augmentation e l'Hyperparameter Evolution, che migliorano ulteriormente le prestazioni del modello.
- **Compatibilità e Flessibilità:** I modelli addestrati con YOLOv5 possono essere esportati in vari formati per l'inferenza su diverse piattaforme, inclusi dispositivi mobili e edge devices.

YOLOv5 è spesso utilizzato in applicazioni che richiedono il rilevamento rapido e accurato degli oggetti, come la sorveglianza, la robotica, i veicoli autonomi e la realtà aumentata. In generale, è preferibile rispetto ad altri modelli per casi di rilevazioni in tempo reale e che favoriscano la velocità a discapito della precisione.

2.5 PANDAS [5]

Pandas è un pacchetto Python che fornisce strutture dati veloci, flessibili ed espressive, progettate per rendere il lavoro con dati "relazionali" o "etichettati" facile e intuitivo.

Le due principali strutture dati di pandas, Series (monodimensionale) e DataFrame (bidimensionale), gestiscono la stragrande maggioranza dei casi d'uso tipici in finanza, statistica, scienze sociali e in molte aree dell'ingegneria. Pandas è costruito sopra NumPy ed è progettato per integrarsi bene in un ambiente di calcolo scientifico con molte altre librerie di terze parti. E' lo strumento ideale per gestire una serie di task, tra le quali trasformazione e pulizia dei dati, analisi/modellazione, e infine organizzazione dei risultati dell'analisi in una forma adatta per la visualizzazione grafica o tabellare.

3. Stato dell'arte

Nel contesto dell'analisi di immagini e, soprattutto, nella rilevazione di oggetti, le più importanti reti neurali profonde fanno da padrone, tra le quali possiamo trovare ***MobileNet[8]***, ***YOLO[6]***, ***CenterNet[9]*** e ***OpenPose[10]***. Le prime due sono molto "generiche" nel tipo di applicazione che propongono, concentrandosi sulla classificazione di elementi in un'immagine: in particolare, MobileNet è pensata per essere leggera e adatta all'esecuzione anche su sistemi a bassa potenza di calcolo, mentre YOLO è focalizzato di più sulla velocità . Le ultime due reti citate sono invece più specializzate: CenterNet permette di identificare il centro degli oggetti e estrarre in modo

approssimativo la loro dimensione, mentre OpenPose si focalizza sull'identificazione di pose e schemi corporei degli individui presenti in un'immagine o in un video. Il nostro sistema si appoggia appunto su una di queste, YOLO (nella variante "v5"), che è una delle più performanti in termini di velocità di rilevazione senza sacrificare eccessivamente le prestazioni. YOLO e, in generale, quasi tutte le reti neurali profonde, hanno già all'interno del loro framework degli strumenti di image agumentation, che permettono, in genere, di migliorare la qualità della rilevazione. Nel nostro caso di studio abbiamo voluto combinare questo sistema con un image enhancement, che, nella sua forma "classica", si rifà alle tecniche fondamentali di miglioramento come l'equalizzazione e l'incremento del contrasto, estese poi dall'Adaptive Histogram Equalization (AHE e CLAHE), che applicano questi stessi principi su in regioni più piccole dell'immagine [7]. Vi sono nella letteratura anche una serie di tecniche aggiuntive, come le Wavelet transform techniques, per citarne una, che sono più adatte a casi specifici in cui si conoscono molto bene le condizioni di acquisizione dell'immagine da migliorare. Bisogna ricordare comunque che, per il principio stesso di manipolazione delle immagini, la modifica può anche peggiorare la qualità dell'immagine finale, eventualmente influenzando in modo negativo la successiva analisi.

4. Sviluppo

In questo capitolo verranno descritte le principali scelte progettuali e le soluzioni implementative utilizzate per realizzare il sistema antifurto.

4.1 Selezione e configurazione del modello YOLOv5

Il modello è stato configurato per rilevare solo le classi di interesse: persone (*class 0*) e automobili *(class 2*), oltre ad alcune classi di animali (*class 15* per i gatti e *class 16* per i cani). Inoltre, sono stati impostati alcuni parametri critici:

- **Soglia di confidenza (model.conf = 0.30)**: i rilevamenti vengono considerati validi solo se il livello di confidenza è superiore a questa soglia. Un valore moderato è stato scelto per bilanciare falsi positivi e negativi, assicurando buone prestazioni sia in termini di affidabilità nelle rilevazioni che di tempo di elaborazione.
- **Intersection over Union (IoU, model.iou = 0.50)**: questo parametro gestisce la sovrapposizione tra le rilevazioni, eliminando quelle troppo vicine tra loro. Con un valore di 0.50, garantiamo che rilevamenti di oggetti molto vicini non vengano erroneamente uniti.
- **Risoluzione immagine (model.imgsz = 640)**: le immagini vengono ridimensionate a 640x640 pixel per migliorare il rilevamento senza sacrificare troppo la velocità di elaborazione.
- **Numero massimo di rilevamenti (model.max_det = 20)**: limita il numero massimo di oggetti rilevati per ogni frame a 20, prevenendo sovraccarichi computazionali in scene complesse (dal momento che l'antifurto è programmato per dare l'allarme anche per una singola rilevazione sufficientemente duratura).
- **NMS agnostico alla classe (model.agnostic = True)**: questa impostazione fa sì che il processo di eliminazione dei rilevamenti sovrapposti tratti tutte le classi allo stesso modo, riducendo i falsi positivi tra oggetti simili.

- **Augmentation durante l'inferenza (model.augment = True)**: viene applicata l'augmentation delle immagini in tempo reale per migliorare la robustezza delle previsioni in condizioni variabili, come angolazioni o illuminazioni diverse.

4.2 Miglioramento delle immagini

Nel contesto dello sviluppo di questo prototipo di applicazione di rilevamento, una porzione importante è composta dalla presenza di alcune operazioni di image enhancement che hanno l'obiettivo di migliorare le immagini sottoposte al modello neurale, in modo appunto da ottenere dei risultati migliori nelle rilevazioni. Come vedremo in seguito, analizzando i risultati, non è necessario (anzi è controproducente) utilizzare insieme tutte queste tecniche, per cui nella funzione di enhancement è possibile aggiungere poi solo quelle effettivamente usate, tra tutte le funzioni predisposte. Le possibili tecniche si rifanno quindi ai classici metodi di enhancement per l'image processing:

- **Equalizzazione dell'istogramma**, che si basa sull'appiattimento della componente di value dell'istogramma di un immagine, per risaltare dettagli scuri e diminuire l'intensità di aree molto luminose. Usata con l'obiettivo di mettere in risalto determinati dettagli normalmente poco visibili. Nel nostro caso è stata implementata come equalizzazione del canale Y dell'immagine convertita in spazio di colore YUV, in modo da mantenere l'informazione di colore nell'immagine equalizzata; applicata la funzione di cv2.equalizeHist() sulla Y, otteniamo l'immagine finale come merge degli altri due canali estratti. Riconvertiamo poi successivamente l'immagine in spazio di colore BGR
- Lo **sharpening** è inserito con l'obiettivo di mettere in risalto i bordi degli oggetti nell'immagine, usato necessariamente insieme ad un **filtro Gaussiano** per andare a ridurre la presenza di rumore nell'immagine, che sarà aumentato con l'applicazione del filtro di nitidezza. Nello specifico lo sharpening filter utilizza il seguente kernel di convoluzione: SHARPEN_KERNEL = [[0,-1,0],[-1,5,-1],[0,-1,0]], mentre il kernel gaussiano è G_KERNEL_SIZE = (3,3), ma è possibile cambiare entrambi all'occorrenza.
- **CLAHE, Contrast Limited Adaptive Histogram Equalization,** tecnica usata per migliorare il contrasto locale in modo da rendere gli oggetti più visibili in contesti di luminosità variabile: è un processo che utilizza lo spazio di colori LAB, spazio colore-opponente con L rappresentante la luminosità e A e B per le dimensioni colore-opponente. Il CLAHE viene quindi applicato sulla dimensione L e poi l'immagine finale ricostruita come un merge dei tre canali.
- **Brightness and Contrast adjustment**, tramite una funzione che prende in ingresso due valori di incremento (da 0 a 100) e lo applica alle proprietà del frame dopo averle normalizzate su un intervallo [-255,255] per la brightness e su un intervallo [-127,127] per il contrasto. Se il valore di correzione è diverso da 0, vengono settati shadow e highlight per calcolare alpha_b e gamma_b, usati per una somma pesata dell'immagine con sé stessa (cv2.addWeighted(image,alpha_b,image,0,gamma_b). Un'operazione simile è fatta con il contrasto, tramite alpha_c e gamma_c.

Tutte le operazioni necessarie sono inserite all'interno della funzione "enhance_image" che, acquisito il frame, ritorna l'immagine modificata pronta per essere data in input alla rete neurale.

4.3 Rilevamento con sliding window e deque

Uno degli aspetti cruciali del progetto è il sistema di rilevamento stabile per l'attivazione dell'allarme. Per evitare falsi positivi dovuti a rilevazioni temporanee o errori sporadici, abbiamo implementato una **sliding window** tramite una struttura dati *deque* di lunghezza fissa (WINDOW_SIZE). Ogni volta che viene rilevata una persona, nella deque viene inserito un valore booleano corrispondente alla rilevazione. Se la somma dei valori veri (ovvero rilevazioni) supera una determinata soglia (DETECTIONS_TO_ALARM), viene attivato l'allarme.

Questo approccio garantisce che l'allarme venga attivato solo se una persona è rilevata per un certo numero di frame all'interno della sliding window, riducendo il rischio di attivazioni accidentali.

4.4 Threading e gestione dell'allarme

Per la riproduzione dell'allarme, è stato creato un thread separato che utilizza la libreria playsound per eseguire un suono di avvertimento continuo. La gestione del thread avviene tramite un **evento di terminazione** (stop_event) che permette di interrompere il suono una volta che il programma principale viene chiuso. Questa soluzione garantisce che il sistema possa monitorare la scena senza bloccare il flusso principale di esecuzione, e allo stesso tempo permette una gestione efficiente della riproduzione sonora.

4.5 Menu and statistics

La struttura di test utilizzata si basa su una semplice classe **Statistics** che, laddove istanziata, mantiene una lista di valori e permette l'ottenimento, tramite metodi appositi, della media dei valori in lista. Dato che ci si occupa di rilevare tre classi specifiche, creiamo per ognuna delle classi un oggetto statistics e per ognuna di essa procediamo a stampare la media dei valori di confidence per gli oggetti identificati, che può essere mostrata alla fine della rilevazione complessiva. Per facilitare l'utilizzo è stato sviluppato un semplice menu di interazione, che all'inizio permette di scegliere se analizzare l'immagine da un input webcam o scegliere un file video da sottoporre al sistema.

4.6 Peggioramento della qualità delle immagini

Per gestire la parte di valutazione del modello applicato su immagini di bassa qualità, utilizziamo lo stesso approccio di sviluppo usato per l'image enhancement: definiamo una serie di funzioni che vanno a "distruggere" l'immagine, tra le quali:

- **Downsampling**: sotto-campioniamo l'immagine, cioè riduciamo la risoluzione tramite un'operazione di resize (ad un quarto delle dimensioni di partenza) e poi successivamente ri-scaliamo sulle dimensioni originali

- **Added noise**: aggiungiamo rumore all'immagine generando una maschera di rumore con distribuzione normale (tramite numPy.random.normal) e la sommiamo appunto con la funzione di somma di OpenCV (cv2.add())

Combinando queste due operazioni si abbatte di molto la qualità dell'immagine, permettendoci poi di applicare il modello e di fare le dovute misurazioni.

5. Test e Conclusioni

Sui video di prova proposti, abbiamo ottenuto i seguenti risultati come media delle confidence, per ognuna delle casistiche proposte:

5.1 Humans

- Enhanced: 0.62825
- Original: 0.62783
- Low Quality: 0.50867
- Enhanced Low Quality: 0.48958

5.2 Animals

- Enhanced: 0.52370
- Original: 0.50650
- Low Quality: 0.39228
- Enhanced Low Quality: 0.40799

5.3 Cars

- Enhanced: 0.59300
- Original: 0.59911
- Low Quality: 0.42097
- Enhanced Low Quality: 0.43464

Quello che possiamo osservare confrontando i risultati ottenuti, nelle diverse combinazioni che sono state provate, è che complessivamente l'andamento è altalenante e l'enhancement ha degli effetti positivi (a partire dall'immagine originale di buona qualità) solo in casi particolari in cui i soggetti nelle scene corrispondano al caso ideale per la rilevazione, laddove invece per inquadrature più complesse il miglioramento dell'immagine non provoca nulla nei risultati della rete e anzi in alcuni casi peggiora la confidence degli oggetti rilevati. Una considerazione importante che si può fare riguarda la variazione dei risultati in relazione al cambio di ordine delle componenti dell'image enhancement: scambiare cioè l'ordine di applicazione delle diverse operazioni di processing provoca una variazione nei risultati finali e quindi è importante riuscire ad identificare la sequenza corretta per ottenere i migliori risultati. Per questo motivo è stato necessario sperimentare nel pratico tante combinazioni differenti e decidere in alcuni casi di non utilizzare affatto alcune delle funzioni predisposte, dato che andavano a peggiorare il risultato. La sequenza finale di operazioni scelta è: clahe, sharpen e gaussian blur. Per quanto riguarda invece l'enhancement applicato sull'immagine "distrutta", i risultati non sembrano essere migliori e anzi nel rilevamento di umani il risultato è stato peggiore. In questo

caso l'operazione di miglioramento andava ad eliminare determinati dettagli dai frame o, in generale, a distorcere punti dell'immagine, peggiorando in alcuni casi la rilevazione. Bisogna ricordare però, riguardo a questo, che i video utilizzati non hanno un risultato ottimo di riferimento, cioè non possiamo considerare la varianza della confidence misurata rispetto all'ottimo, ma solo il suo valore assoluto, che potrebbe in molti casi essere fuorviante a causa, ad esempio, di una massiccia presenza di falsi positivi.

6. Appendice

[1]: https://pypi.org/project/playsound/

[2]: https://pypi.org/project/torch/

[3]: https://docs.opencv.org/4.x/d1/dfb/intro.html

[4]: https://docs.ultralytics.com/yolov5/#what-environments-are-supported-for-running-yolov5-applications

[5]: https://pandas.pydata.org/docs/getting_started/overview.html

[6]: https://arxiv.org/abs/1506.02640

[7]:https://www.researchgate.net/publication/340419040_Image_Enhancement_Techniques_An_Exhaustive_Review

[8]: https://arxiv.org/abs/1704.04861

[9]: https://arxiv.org/abs/1904.08189

[10]: https://github.com/CMU-Perceptual-Computing-Lab/openpose