

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI MATEMATICA

CORSO DI LAUREA IN INFORMATICA

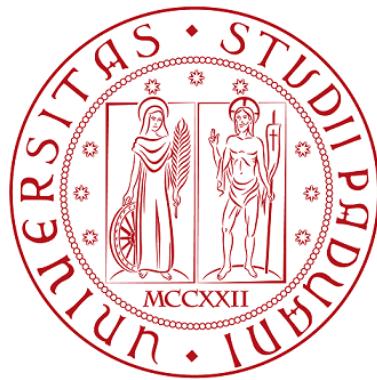
Riconoscimento e tracciamento di elementi su video ad alta risoluzione

Laureando:

Davide LIU

Relatore:
Prof. Lamberto BALLAN

Tutor aziendale:
Leonardo DAL ZOVO



Anno Accademico 2018/2019

Indice

1 Introduzione	5
1.1 Note esplicative	5
2 Ambiente Aziendale	6
3 Analisi dei problemi	7
3.1 Tecniche di computer vision	7
3.2 Computer vision applicata su immagini ad alta risoluzione	8
3.3 Frammentazione dell'immagine	9
3.4 Tecniche di object tracking	11
3.5 Object tracking con continue object detections	12
4 Progettazione	14
4.1 Progettazione algoritmo per riconoscimento di elementi in un'immagine frammentata	14
4.1.1 Scomposizione del frame originale in regioni	14
4.1.2 Rimozione degli elementi individuati più volte all'interno delle aree di sovrapposizione	15
4.1.3 Creazione di raggruppamenti di labels correlate	15
4.1.4 Miglioramento: raggruppamenti di labels utilizzati come region proposal	18
4.2 Progettazione algoritmo per tracciamento di elementi	19
4.2.1 Filtro di Kalman	19
4.2.2 Assegnazione detection-tracker	20
4.2.3 Gestione delle detections e dei trackers non assegnati	21
4.2.4 Miglioramento: utilizzo dell'hash dell'immagine come metrica di supporto	22
5 Tecnologie	23
5.1 Python	23
5.2 Pycharm	23
5.3 Tensorflow	24
5.4 OpenCV	24
5.5 Numpy	24
5.6 Matplotlib	25
5.7 Pytest	25

6 Sviluppo	26
6.1 Sviluppo algoritmo per riconoscimento di elementi in un'immagine frammentata	26
6.1.1 Implementazione	26
6.1.2 Test di integrazione	28
7 Risultati ottenuti	29
7.1 Metriche utilizzate per singole immagini	29
7.1.1 Precision	29
7.1.2 Recall	29
7.1.3 Intersection over union	30
7.1.4 Average Precision	30
7.1.5 Mean Average Precision	31
8 Glossario	32
9 Bibliografia	34
10 Appendice	36

Elenco delle tabelle

Elenco delle figure

1	Logo di Studiomapp	6
2	Esempio di un'immagine con box, categoria e probabilità per ogni elemento riconosciuto in essa	7
3	Esempio di object detection in un frame di un video in 4K	9
4	Esempio di un'immagine in alta risoluzione suddivisa in regioni senza sovrapposizioni	10
5	Frames di un video nei quali viene tracciata un' auto (ordinati da sinistra a destra e dall'alto al basso)	12
6	Esempi di labels erroneamente individuate a causa della frammentazione: nel primo caso la ricostruzione avviene correttamente, nel secondo caso è presente un errore.	16
7	Esempio di assegnazione detection-tracker: i rettangoli rossi raffigurano le bounding boxes dei trackers mentre quelle verdi rappresentano le bounding boxes individuate dalla detection. I trackers 1 e 2 vengono assegnati regolarmente, l'elemento tracciato dal tracker 3 è sparito dal frame ed è comparso un nuovo elemento che verrà tracciato da un nuovo tracker.	21
8	Logo di Python	23
9	Logo di Tensorflow	24
10	Esempio di intersezione e di unione	30

1 Introduzione

La computer vision è un ambito dell'intelligenza artificiale il cui scopo è quello di insegnare alle macchine non solo a vedere un' immagine, ma anche a riconoscere gli elementi che la compongono in modo da poter interpretare il suo contenuto come farebbe il cervello di un qualsiasi essere umano. Nonostante le attuali tecniche di deep learning rendano possibile questo compito, è comunque necessaria una grande quantità di immagini e di tempo per poter allenare una rete neurale a sufficienza in modo da riuscire a riconoscere correttamente degli oggetti in un' immagine non incontrata durante il processo di allenamento.

Lo scopo del progetto di stage è stato quello di progettare e realizzare un sistema di riconoscimento e tracciamento di specifici elementi all' interno di un video ad alta risoluzione. Questo progetto ha comportato sfide e complessità aggiuntive rispetto all' analisi degli elementi presenti in una singola foto sia per il fatto che un video è composto da una sequenza di frames anzichè da una singola immagine, sia per il fatto che i frames trattati erano in formato Ultra High Definition (4K) e quindi elaborare l'intero frame in una sola volta sarebbe stato troppo oneroso dal punto di vista computazionale.

Riassumendo i tre problemi principali che sono stati affrontati, in ordine sequenziale, sono i seguenti:

- Frammentazione dell'immagine;
- Tracciamento degli elementi;
- Stabilizzazione delle labels degli elementi tracciati.

Ognuno dei sotto-problemi è stato analizzato a fondo, ne è stata trovata un' adeguata soluzione ed è stata applicata ai fini di realizzare un prodotto il più performante possibile.

1.1 Note esplicative

Allo scopo di evitare ambiguità a lettori esterni al gruppo, si specifica che all'interno del documento verranno inseriti dei termini con un carattere 'G' come pedice, questo significa che il significato inteso in quella situazione è stato inserito nel Glossario

2 Ambiente Aziendale



Figura 1: Logo di Studiomapp

STUDIOMAPP, fondata a fine 2015, è una startup innovativa con sedi a Ravenna e Roma, in Italia. Sviluppa algoritmi di intelligenza artificiale specifici per Geo-calcolo e dati geo-spaziali in modo da fornire soluzioni innovative per smart cities, mobilità, trasporti e logistica, turismo e beni culturali, immobiliare e real estate, agricoltura, territorio e gestione delle risorse naturali, adattamento ai cambiamenti climatici.

E' la prima startup dell'Emilia Romagna selezionata dall'ESA BIC Lazio, l'incubatore dell'Agenzia Spaziale Europea (ESA) ed è supportata da importanti istituzioni, acceleratori e grandi attori. Membro fondatore dal 2016 della rete Copernicus Academy, si impegna a diffondere i benefici dell'utilizzo dei dati di Osservazione della Terra per la qualità della vita dei cittadini e la competitività delle PMI.

Ha acquisito una solida esperienza nella promozione di Copernicus, il programma europeo di osservazione della Terra, nell'ecosistema Startup condividendo conoscenza e formazione. Ad oggi la società ha organizzato più di 40 eventi che hanno raggiunto più di 1000 persone che vanno dal pubblico generale, agli studenti, alle startup, ai funzionari pubblici, alle autorità locali, alle PMI.

3 Analisi dei problemi

3.1 Tecniche di computer vision

La computer vision ha come obiettivo quello di riconoscere e classificare alcuni specifici elementi presenti in un'immagine. Identificare un oggetto significa localizzarne la posizione esatta all'interno dell'immagine. Una volta trovata la sua locazione, l'oggetto viene messo in evidenza disegnando un rettangolo attorno ad esso, detto anche bounding box, in modo che lo racchiuda con la maggiore precisione possibile. La classificazione ha invece come scopo quello individuare la categoria di appartenenza di un oggetto e la probabilità che essa sia realmente quella corretta. Per classificare un singolo elemento in un'immagine viene tipicamente utilizzata una Convolutional Neural Networks (CNN) allenata con grandi quantità di immagini che possono essere tranquillamente reperite in rete già raggruppate in datasets come ad esempio ImageNet. La vera sfida salta fuori non appena ci troviamo a dover identificare e classificare nella stessa immagine diversi oggetti appartenenti a categorie diverse, di differenti dimensioni e posizioni e talvolta anche sovrapposti. Questa situazione è molto comune quando ci troviamo ad osservare qualsiasi foto rappresentante il mondo reale. Il risultato ottimale sarebbe quindi di avere una bounding box di dimensioni corrette per ogni oggetto identificato mostrando anche la categoria di appartenenza dell'elemento e la sua probabilità.

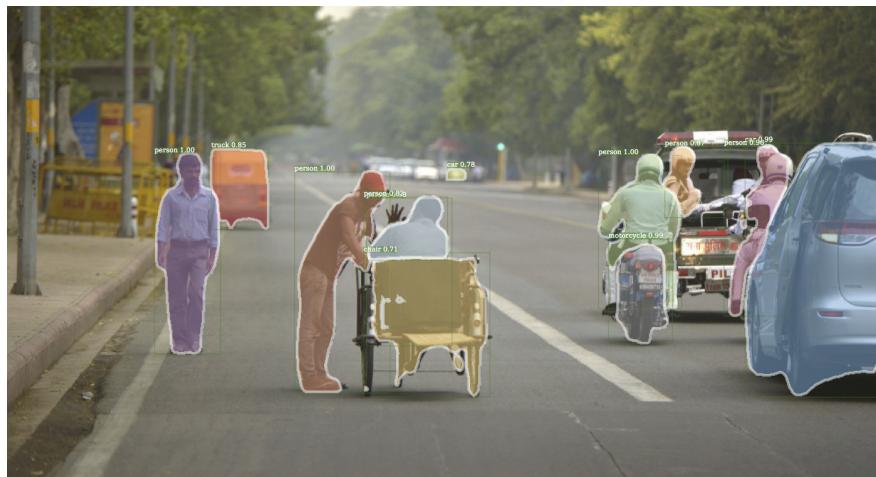


Figura 2: Esempio di un'immagine con box, categoria e probabilità per ogni elemento riconosciuto in essa

Il modo più semplice per andare incontro a questo problema è quello di utilizzare una

R-CNN (Regional Convolutional Neural Network) la quale tramite un algoritmo greedy chiamato Selective Search estrae delle regioni di interesse nelle quali è probabile che vi sia presente un oggetto. Queste regioni vengono poi date singolarmente in input ad una normale CNN la quale ha il compito di estrarne le caratteristiche principali per poi utilizzare una Support Vector Machine (SVM) presente nell'ultimo strato della CNN per rilevare la presenza di un oggetto ed eventualmente classificarlo. Questo tipo di soluzione presenta il difetto di richiedere molto tempo durante l'operazione di ricerca delle regioni di interesse.

La versione più avanzata della R-CNN ed attualmente in uso nei sistemi di computer vision attuali è chiamata Faster R-CNN e risolve il bottleneck della sua antecedente sostituendo la Selective Research con una Region Proposal Network (RPN). Essa prende come input un'immagine di qualsiasi dimensione e restituisce come output un insieme di rettangoli associati ad una probabilità che essi contengano un oggetto o meno. Tramite una CNN viene prima costruita la mappa delle caratteristiche più significative dell'immagine, in seguito viene utilizzata una sliding window per scorrere la mappa delle caratteristiche e darle in input a due fully-connected layers dei quali uno serve per individuare le coordinate del box dell'oggetto¹ mentre l'altro serve per ritornare la probabilità che nel box vi sia effettivamente un oggetto². Infine queste regioni vengono passate ad una R-CNN che avrà come al solito il compito di riconoscere e classificare l'oggetto.

3.2 Computer vision applicata su immagini ad alta risoluzione

Sebbene sul web sia abbastanza facile reperire grandi quantità di immagini con cui allenare i propri modelli, tuttavia la maggior parte di questi datasets contiene solamente immagini a bassa risoluzione ed è difficile trovare in rete grandi datasets di immagini o video in 4K. Inoltre, la maggior parte dei modelli sono stati progettati per lavorare su immagini a bassa risoluzione (tra i 200 e i 600 pixels) sia per il fatto che una bassa risoluzione è comunque sufficiente per riconoscere e classificare un elemento, sia perchè è più efficiente lavorare su immagini di bassa qualità che su immagini in con una risoluzione molto alta.

Lo svantaggio che questo comporta è che nelle immagini in bassa risoluzione si perdono molti dei dettagli che invece potrebbero essere catturati da un'immagine o da un video ad alta risoluzione. In aggiunta, i video in 4K o addirittura 8K sono al giorno d'oggi sempre più diffusi perciò anche gli attuali modelli dovranno prima o poi adattarsi per trattare efficacemente immagini di tali risoluzioni. Nella figura sottostante si può vedere un esempio di un frame in alta risoluzione nel quale, diminuendone le dimensioni e perdendo quindi qualità, non sarebbe stato possibile riconoscere alcune delle persone individuate nel frame.

¹Box-regression layer

²Box-classification layer



Figura 3: Esempio di object detection in un frame di un video in 4K

3.3 Frammentazione dell'immagine

Un'immagine in 4K ha una risoluzione di 3840 x 2160 pixels per un totale di 8294400 di pixels. Una rete neurale in grado di ricevere un input così corposo dovrebbe allenare un numero molto elevato di parametri risultando così in un processo molto lungo e dispendioso tanto che molti dei modelli attuali effettuano un ridimensionamento dell'immagine per adattarla al meglio al loro input. L'idea per aggirare il problema è quella di frammentare l'immagine in diverse sotto-immagini di dimensioni minori e quindi gestibili efficacemente da una singola rete neurale. Questa particolare strategia è chiamata frammentazione dell'immagine e risulta essere molto utile in quanto permette di analizzare un'immagine ad alta risoluzione scomponendola in frammenti di dimensioni minori anziché gestirla nella sua interezza.

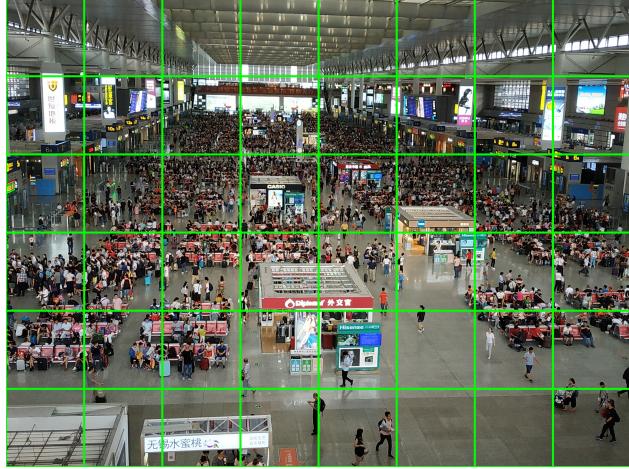


Figura 4: Esempio di un’immagine in alta risoluzione suddivisa in regioni senza sovrapposizioni

Tuttavia questo procedimento non è esente da difficoltà. Nel caso in cui un oggetto dovesse trovarsi su più regioni diverse, esso potrebbe venire identificato più volte o addirittura essere riconosciuto ogni volta come se fosse un oggetto appartenente ad una categoria diversa.

Un primo approccio per risolvere questo problema è quello di porre maggiore attenzione alle labels degli oggetti posizionati in prossimità dei confini delle regioni in quanto con molta probabilità è possibile che l’oggetto continui invece nella regione adiacente piuttosto che essere interamente contenuto nella regione esaminata. Se è quindi presente una label anche in una regione adiacente si passa allora alla verifica che le due labels possano effettivamente appartenere allo stesso elemento. Per fare ciò bisogna assicurarsi che le due labels siano compatibili sia in termini di categoria che di posizione entro una certa soglia ed in caso affermativo fonderle in una sola label contenente l’oggetto intero.

Un’altra soluzione esaminata è quella di suddividere l’immagine intera in regioni con sovrapposizione il cui scopo è quello di generare intersezioni tra labels in prossimità dei confini. In questo modo un elemento contenuto all’interno di un’area di sovrapposizione tra più regioni genererebbe due o più labels quasi completamente sovrapposte. Il problema può essere facilmente gestito con un algoritmo di Non-Maximum Suppression, il quale, per ogni insieme di labels parzialmente sovrapposte e con stessa categoria, tende ad eliminare le labels con probabilità minore tenendo valida solo quella con probabilità massima. Tuttavia il caso più insidioso ed allo stesso tempo più frequente avviene quando due o più labels non solo hanno un’intersezione dentro un’area di sovrapposizione ma la loro box si estende anche al di fuori

di esse. E' proprio per far fronte a questo problema che è stato ideato ed implementato un algoritmo il quale verrà descritto nella *sezione 4.1*.

3.4 Tecniche di object tracking

Il tracciamento di oggetti in un video è uno di quei problemi presenti nell'ambito dell'informatica che non sono ancora stati risolti ottenendo risultati soddisfacenti. Il tracciamento consiste non solo nel localizzare l'oggetto tracciato in una sequenza di frames ma anche nel riconoscere che l'oggetto è sempre lo stesso.

Il problema non è per niente banale se pensiamo che in un video uno stesso oggetto può spostarsi, nascondersi dietro qualche altro elemento, deformarsi, cambiare le sue dimensioni, la sua illuminazione, la sua velocità ed il tipo di background. Nel caso ancora peggiore un oggetto tracciato potrebbe addirittura scomparire in un frame per poi ripresentarsi solamente dopo che sono trascorsi un numero casuale di frames.

Un algoritmo di tracking ottimale dovrebbe essere in grado di far fronte a tutti questi problemi riconoscendo quindi l'oggetto da esso tracciato tramite per esempio l'assegnazione di un ID univoco.

allo stato dell'arte, questo problema viene affrontando eseguendo inizialmente una normale object detection sul primo frame del video in modo tale da individuarne tutti gli elementi presenti e le rispettive labels. In seguito, ognuno di questi elementi viene assegnato ad un tracker che avrà il compito di tracciare l'elemento nei frames successivi.

Tracciare un elemento è un' operazione meno onerosa rispetto alla sua individuazione in quanto il tracker conosce già alcune informazioni relative all'oggetto tracciato acquisite nei frames precedenti potendo così tenere in memoria uno storico del suo stato, come per esempio, le sue ultime locazioni. Per aumentare la sua precisione, un tracker non tiene conto solamente della locazione dell'elemento osservato ma può anche conservare altre utili informazioni aggiuntive come la sua direzione, una previsione delle locazioni future analizzandone la sua traiettoria oppure può memorizzare un hash³ dei pixels presenti all'interno del bounding box dell'oggetto per poi confrontarlo con l'hash dello stesso oggetto calcolato nel frame successivo. I trackers più performanti come CSK, MOSSE e GOTURN utlizzano alcune delle informazioni sopra descritte per costruire un filtro di correlazione in modo da localizzare l'oggetto nel frame successivo e migliorare la precisione del filtro con le successive individuazioni, lo scopo del filtro è quello di minimizzare la differenza tra l'output ricostruito e quello originale.

Nonostante tutti questi accorgimenti è però inevitabile che col trascorrere dei frames i trac-

³Al contrario degli hash usati in crittografia, un hash applicato alle immagini è progettato in modo tale che piccole variazioni dei pixels dell'immagine non risultino in un hash molto diverso da quello originale

kers cominceranno ad essere sempre più imprecisi nell'individuare il loro oggetto tracciato. In particolare, questa situazione può accadere molto velocemente in quei video dove avvengono molti spostamenti e sovrapposizioni tra elementi. E' quindi buona norma aggiornare i trackers con le locazioni corrette effettuando una nuova detection ogni fissato numero di frames.

E' qui che si presenta il problema di maggiore rilevanza: una nuova detection effettuata su un nuovo frame non tiene conto delle informazioni acquisite in precedenza in quanto queste con molta probabilità sarebbero errate o imprecise. Il problema è quindi quello di riassegnare a ciascun oggetto individuato lo stesso ID che possedeva in precedenza ed assegnare un nuovo ID ad ogni oggetto comparso nel video per la prima volta.



Figura 5: Frames di un video nei quali viene tracciata un'auto (ordinati da sinistra a destra e dall'alto al basso)

3.5 Object tracking con continue object detections

Considerata la scarsa affidabilità degli attuali algoritmi di tracciamento, nella soluzione individuata gli oggetti vengono identificati effettuando una nuova detection per ogni frame del video in modo da assicurarsi di avere sempre una buona accuratezza ed allo stesso tempo migliorare l'efficacia dei trackers migliorando gradualmente la precisione dei loro filtri di predizione. Questo metodo sacrifica l'efficienza del processo per migliorarne l'efficacia. A meno che non venga utilizzato un algoritmo di detection molto veloce come SSD (Single Shot Detector) non è possibile applicare il tracking sui video in tempo reale in quanto il frame rate che ne risulterebbe sarebbe troppo basso. A seguito di una nuova detection, per effettuare una corretta riassegnazione degli ID viene fatto un confronto tra le labels presenti nel frame precedente con quello corrente con lo scopo di trovare una corrispondenza biunivoca tra due

labels e capire quando entrambe si riferiscono allo stesso elemento in modo da garantire un corretto trasferimento dell'ID. Per rendere tutto ciò possibile viene utilizzato un filtro in grado di predire le locazioni future di un oggetto tenendo traccia dei suoi bounding boxes e poi correggersi tramite misurazioni successive. Anche per realizzare questa soluzione è stato ideato ed implementato un algoritmo descritto nella *sezione 4.2*.

4 Progettazione

4.1 Progettazione algoritmo per riconoscimento di elementi in un'immagine frammentata

Per quanto riguarda il problema della frammentazione dei frames in 4K è stato individuato un apposito algoritmo in grado di effettuare il riconoscimento degli oggetti nei frames presenti in un video senza doverli ridimensionare ma utilizzando la tecnica della frammentazione dell'immagine.

4.1.1 Scomposizione del frame originale in regioni

Un frame in 4K viene quindi scomposto in una matrice di $R \times C$ sotto-immagini chiamate regione_G in modo tale che ogni regione sia efficacemente analizzabile da un modello come Faster R-CNN. Per facilitare l'operazione di riconoscimento degli elementi da parte della rete, ogni regione si sovrappone leggermente con le sue regioni adiacenti. Per definire la quantità di pixels da coinvolgere nella sovrapposizione viene definito uno stride_G che indica quanti pixels della regione tralasciare, sia in verticale che in orizzontale, prima che cominci quella successiva, ovviamente lo stride deve essere minore della larghezza di una regione. Una Faster R-CNN dopo aver elaborato singolarmente ogni regione come se fosse una singola immagine darà in output una lista di label_G con le seguenti caratteristiche:

- (**max-x, max-y**): coordinate del vertice in alto a sinistra del rettangolo rappresentante il bounding box dell'oggetto riconosciuto;
- (**min-x, min-y**): coordinate del vertice in basso a destra del rettangolo rappresentante il bounding box dell'oggetto riconosciuto;
- **Categoria_G** : è un numero naturale che indica la categoria di appartenenza dell'elemento individuato;
- **Score_G** : rappresenta la misura di probabilità che la classificazione ottenuta sia effettivamente quella corretta.

Successivamente viene aggiustata la posizione delle labels individuate in modo da trasstrarle nella loro posizione corretta all'interno dell'immagine originale non frammentata. Questo viene fatto aggiungendo un adeguato offset alle coordinate dei vertici dei bounding boxes delle labels sulla base della loro regione di appartenenza.

4.1.2 Rimozione degli elementi individuati più volte all'interno delle aree di sovrapposizione

A causa della presenza delle aree di sovrapposizione dovute alla struttura delle regioni, gli elementi giacenti in queste particolari zone del frame verranno individuati tante volte quante sono le regioni che si sovrappongono in quella determinata area. Per eliminare le copie duplicate e tenerne solo una viene utilizzato un algoritmo chiamato Average Non-Max Suppression (ANMS) che è una variante del Non-Max Suppression tipicamente utilizzato dai modelli di visione artificiale. Per ogni gruppo di labels sovrapposte, invece che tenere la label con lo score maggiore ed eliminare tutte le altre, il nuovo bounding box viene calcolato come la media dei bouding boxes box di tutte labels e lo score viene calcolato come la media tra tutti gli scores. Questo metodo è fondato sul ragionamento che non bisognerebbe buttare via delle informazioni già possedute ma piuttosto riutilizzarle per scoprire qualcosa di nuovo. Per esempio ad uno stesso elemento visualizzato dentro due sezioni differenti di un' immagine potrebbero venirgli assegnati due score diversi. Mentre NMS conserverebbe solo il valore più alto tra i due, ANMS li utilizzerebbe entrambi per ottenere un valore ancora più affidabile aumentando quindi la veridicità della classificazione.

4.1.3 Creazione di raggruppamenti di labels correlate

A questo punto tutti gli elementi sono stati individuati e classificati ma rimane comunque il problema che, a causa della precedente scomposizione, gli oggetti situati all'interno delle aree di sovrapposizione risulterebbero individuati due o più volte. Come si può vedere in figura 6, questo numero varia in base al numero di regioni sulle quali giace l'oggetto. Il secondo problema è che due elementi *vicini*⁴, anche se classificati nella stessa categoria, non è detto che necessariamente debbano rappresentare lo stesso elemento. Un esempio di questo caso lo si può sempre notare in figura 6. Un caso ancora peggiore lo si ha quando non solo l'elemento è situato su più regioni differenti ma sussiste anche il problema che ogni parte dell'elemento verrebbe classificata in modo diverso a causa della loro ambiguità. Infine, è anche possibile che un oggetto si distribuisca su più regioni adiacenti ed ogni sua label presenti dimensioni diverse per ogni regione. La soluzione individuata consiste nel raggruppare labels correlate tra loro in insiemi di labels dette raggruppamenti_G per poi racchiuderli in una label che identifica l'elemento rappresentato dal raggruppamento. Due labels sono in correlazione tra di loro se soddisfano una condizione di correlazione sotto riportata.

Condizione di correlazione: Per effettuare un corretto raggruppamento delle labels viene anche tenuta in considerazione la categoria a loro associata tramite la classificazione insieme

⁴Due labels sono considerate vicine se sono intersecate tra di loro ed intersecano lo stesso confine di regione

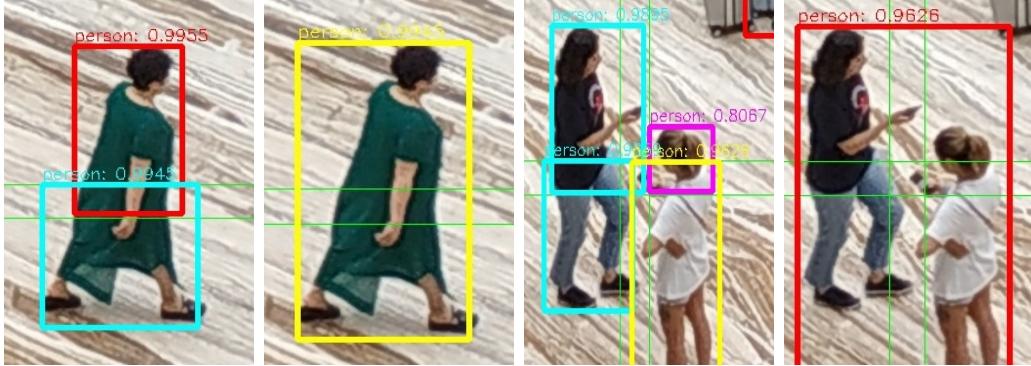


Figura 6: Esempi di labels erroneamente individuate a causa della frammentazione: nel primo caso la ricostruzione avviene correttamente, nel secondo caso è presente un errore.

allo score assegnato. Per definire il risultato della condizione è inoltre necessario stabilire una **soglia di affidabilità** che indica lo score minimo che una label deve possedere per poter considerare la sua classificazione come affidabile o meno. Di seguito vengono riportati i vari casi per decidere se la condizione è vera o falsa.

- **True:** Le due labels hanno la stessa categoria ed entrambe con score uguale o maggiore della soglia;
- **True:** Le due labels hanno la stessa categoria ma almeno una delle due ha score minore della soglia;
- **True:** Le due labels hanno categoria diversa ma almeno una delle due ha score minore della soglia;
- **False:** Le due labels hanno categoria diversa ed entrambe con score uguale o maggiore della soglia;

Prima di cominciare con il raggruppamento, vengono inizialmente individuate tutte le labels che intersecano i confini di regioni causati dalle aree di sovrapposizione o che non distino più di un fissato numero di pixels, detto overlap_G , da esse. L'overlap viene definito in quanto anche se nell'immagine reale un oggetto interseca un confine di regione è possibile che a causa di errori di imprecisione del modello, il box della label risulti leggermente distaccato dalla linea che rappresenta il confine. E' quindi possibile ovviare a questo problema aumentando lo spessore del confine di tanti pixels quanti indicati dall'overlap. Per lo stesso motivo precedente, ai fini di controllare se una label interseca un'altra entità o meno viene anche tenuta

in considerazione una tolleranza G che indica di quanto il bounding box di una label può essere distante da un'entità affinché questa venga comunque considerata come intersecata. Per creare i raggruppamenti di labels è stato ideato il seguente algoritmo:

1. Vengono tenute solo le labels che intersecano almeno un confine di regione e vengono inizializzate come *non controllate* e *non raggruppate*;
2. Viene selezionata una label qualsiasi *non controllata* e la si imposta come *controllata*;
3. Per ogni label *controllata* ma non ancora *raggruppata* controlla se ci sono altre labels *non controllate* che rispettino ognuna delle seguenti condizioni:
 - Devono essere *vicine* o distanti entro la tolleranza fissata;
 - Devono rispettare la *condizione di correlazione*;
 - La loro box non deve intersecare una regione al di fuori delle aree di sovrapposizione che sia già intersecata da una qualsiasi altra label *controllata*⁵;
 - Deve essere rispettata una **soglia di matching**: La posizione delle loro aree deve essere compatibile entro una certa soglia ovvero che, i lati lungo il quale le due labels vengono unite siano tali che la differenza tra quello maggiore e quello minore sia inferiore ad una certa soglia che può essere sia definita come una proporzione rispetto ad uno dei due lati oppure un valore in pixels.
4. Le labels così trovate diventano a loro volta *controllate*;
5. Si ripetono i punti 3 e 4 fino a che non sia più possibile trovare ulteriori labels;
6. Tutte le label *controllate* vengono ora classificate come *raggruppate* e viene assegnato un numero progressivo ad ogni label *raggruppata* in modo da identificarne il gruppo di appartenenza;
7. Si ripetono i punti da 2 a 6 fino a che tutte le labels non vengano raggruppate. L'algoritmo in questo modo termina sempre ed è possibile che un raggruppamento comprenda una sola label.

E' da notare che labels intersecanti ma interamente comprese in una sola regione non sono motivo di interesse in quanto l'algoritmo di Non-Maximum Suppression utilizzato dalla rete

⁵Questo perché se due labels sono state individuate come elementi distinti all'interno di una regione allora è probabile che lo siano anche nell'immagine intera in quanto viene supposto che un modello non commetta errori di riconoscimento

in fase di post-processing ci assicura che labels intersecanti nella stessa regione individuino elementi diversi. In seguito bisogna trasformare ogni raggruppamento in una nuova label che racchiuda tutte le labels che lo compongono. Per fare questo vengono esaminate le coordinate di ogni vertice di tutte le bounding boxes di un raggruppamento in modo tale da trovare quattro nuovi vertici di un rettangolo che soddisfi i requisiti sopra discussi. La nuova label così creata andrà a sostituire le labels del rispettivo raggruppamento e per deciderne la categoria e lo score viene applicata una **regola di classificazione**: categoria e score assegnati saranno pari alla categoria e allo score posseduti dalla label appartenente al raggruppamento con score maggiore.

A questo punto l'algoritmo può dirsi concluso ed è in grado di riconoscere gli elementi in un'immagine in 4K con un'accuratezza accettabile e buona velocità. Tuttavia in casi particolari come quello mostrato in figura 6 l'algoritmo commetterebbe un errore di classificazione in quanto individuerebbe due elementi distinti con una sola label comune.

4.1.4 Miglioramento: raggruppamenti di labels utilizzati come region proposal

Un ulteriore miglioramento dell'algoritmo potrebbe essere ottenuto utilizzando le labels ottenute dal procedimento descritto in precedenza come nuove regioni sulle quali applicare nuovamente una detection per identificare gli elementi contenuti nella regione ma con maggiore precisione in quanto questa volta l'area non verrà affetta da problemi di frammentazione dando quindi la possibilità alla rete di esaminare l'area per intero. La regione viene prima inizializzata rimuovendo la sua label e poi ripopolata con le nuove labels identificate dalla rete. Il primo problema che salta fuori è che durante questo procedimento la rete identificherà nuovamente anche quegli elementi che casualmente si trovavano dentro la regione coinvolta ma che erano già stati trovati anche in precedenza. Tuttavia questo problema viene tranquillamente risolto applicando un algoritmo di Average Non-Max Suppression, utilizzato già in precedenza, per eliminare oggetti quasi completamente sovrapposti. Il secondo problema riguarda ancora gli oggetti che stanno a cavallo tra la regione interessata e l'immagine originale, questa volta però, avendoli già individuati nella loro interezza durante la prima fase è quindi solamente necessario unire la nuova label con quella già trovata in precedenza in modo da ottenerne una nuova con precisione maggiore. Un caso particolare lo si ha quando la regione in esame risulti essere così estesa da vanificare i vantaggi ottenuti dalla frammentazione. Per far fronte a questo problema basta ridimensionare l'area coinvolta fino a portarla ad avere dimensioni gestibili da una rete. In questo caso la perdita di risoluzione e quindi di dettagli non comporterebbe un grave problema in quanto gli elementi visibili solo grazie all'alta definizione sarebbero già stati individuati nella fase precedente. Nel caso in cui questi oggetti dovessero venire nuovamente identificati verrebbero gestiti dall'ANMS per ottenerne una migliore approssimazione. Questa funzionalità permette di migliorare

l'accuratezza quando si vogliono identificare oggetti che si estendono su due o più regioni o per migliorare il riconoscimento di gruppi di elementi sovrapposti e molto vicini tra loro in prossimità di un confine.

4.2 Progettazione algoritmo per tracciamento di elementi

Per affrontare il problema del tracciamento degli elementi, viene utilizzato un algoritmo di tracking supportato da una detection applicata ad ogni frame del video al fine di garantirne una migliore accuratezza. Essendo i video in qualità 4K, per effettuare la detection viene sempre utilizzato l'algoritmo descritto in sezione 4.1 in modo da non ridurre la qualità dei frames.

4.2.1 Filtro di Kalman

Lo scopo di un tracker è quello di predire la posizione di un oggetto in un frame a partire dallo storico delle sue locazioni passate per mezzo di un filtro. I trackers implementati utilizzano un filtro di Kalman in quanto essi si rivelano molto efficaci nell'effettuare predizioni anche in sistemi soggetti a continui cambiamenti come lo è per esempio un video. Il secondo vantaggio è quello di garantire una buona resistenza contro i rumori causati da detections imprecise, le quali possono per esempio avere luogo in presenza di oggetti parzialmente occultati o deformati a causa di qualche spostamento. Infine, questa tipologia di filtri sono anche computazionalmente veloci in quanto, una volta implementati, la loro esecuzione si traduce in semplici moltiplicazioni tra matrici. L'applicazione del filtro di Kalman consiste in due fasi distinte: predizione ed aggiornamento. La prima fase ha come scopo quello di usare la locazione precedente per predire quella attuale effettuando anche una piccola correzione per far fronte alle variazioni introdotte da possibili fonti esterne (rumore). In seguito sono riportate le formule relative alla fase di predizione:

$$x_k = F_k x_{k-1} + B_k u_k$$

$$P_k = F_k P_{k-1} F_k^T + Q_k$$

Dove x_k è la predizione della posizione dell'oggetto x nel frame k , F_k è il modello di transizione di stato applicato alla posizione x_{k-1} , B_k è una matrice di controllo alla quale viene applicato il vettore di controllo u_k e rappresentano le variazioni subite da x_k causate da una fonte esterna, in questo rappresentano movimenti irregolari dell'oggetto rispetto alla sua traiettoria. P_k è la predizione della covarianza di x_k mentre Q_k è la covarianza del processo che genera rumore. Nella seconda fase, quella di aggiornamento, viene usata la misurazione corrente, che in questo caso sarà il bounding box dell'oggetto tracciato individuato dalla nuova detection,

per rifinire ulteriormente la sua locazione esatta. In seguito sono riportate le formule relative alla fase di aggiornamento:

$$\begin{aligned}x_k' &= x_k + K(z_k - H_k x_k) \\P_k' &= P_k - K H_k P_k \\K &= P_k H_k^T (H_k P_k H_k^T + R_k)^{-1}\end{aligned}$$

Dove x_k' è la nuova stima della posizione ottenuta dopo aver effettuato l'aggiornamento, K è una matrice detta anche guadagno di Kalman, z_k è il valore misurato, in questo caso corrisponderà alla label individuata con la detection effettuata sul frame k. H_k è una matrice che serve per scalare z_k in modo tale da renderlo compatibile con lo stato dell'oggetto tracciato ed infine R_k è la covarianza di z_k . Riassumendo, lo scopo del filtro di Kalman è quello di individuare la posizione reale di un oggetto a partire da una sua predizione rispetto alla sua posizione precedente e da una misurazione reale che però può essere soggetta ad imprecisioni.

4.2.2 Assegnazione detection-tracker

Il passo successivo è quello di abbinare ognuno dei bounding box stimati dai filtri dei trackers con le bounding boxes individuate da una detection. Questo risultato viene ottenuto tramite un algoritmo di assegnazione conosciuto anche come algoritmo di Kuhn-Munkres. La metrica utilizzata dall'algoritmo è l'IoU tra i bounding boxes stimati dai trackers e quelli individuati dalla detection, tale metrica viene spiegata in dettaglio nella sezione 7.1.3. L'obiettivo dell'algoritmo è quello di assegnare le detections ai trackers massimizzando la somma dell'IoU delle bounding boxes associate. Questa soluzione si basa sul ragionamento che più due bounding boxes sono sovrapposte, più è probabile che esse rappresentino lo stesso oggetto. Inoltre, perchè due bounding boxes possano essere associate, il loro IoU deve anche essere maggiore o uguale di una certa soglia, detta **soglia di IoU**, in quanto tra due boxes poco sovrapposte è probabile che non vi sia alcuna correlazione. Il primo passo dell'algoritmo consiste nel creare una matrice di dimensioni $n \times m$ dove n è il numero di oggetti individuati dalla detection e m è il numero di trackers. In ogni cella della matrice viene calcolato il valore dell'IoU tra la box predetta dal tracker i ed il box j individuato dalla detection, con $0 < i < m$ e $0 < j < n$. Se le colonne sono minori delle righe allora bisogna ruotare la matrice in modo tale che le colonne siano tante almeno quante sono le righe. In seguito vengono ripetuti i seguenti passi:

1. Per ogni riga, sottrarre il valore minimo della riga a tutti gli elementi della stessa riga. In questo modo, in ogni riga sarà presente almeno una cella con valore pari a 0.

-
2. Per ogni colonna, sottrarre il valore minimo della colonna a tutti gli elementi della stessa colonna. In questo modo, in ogni colonna sarà presente almeno una cella con valore pari a 0.
 3. Tracciare delle linee attraverso tutte le righe e le colonne che contengano almeno un elemento pari a 0 in modo tale da tracciare il minor numero di linee possibile.
 4. Se sono state tracciate esattamente $k=\min(n,m)$ allora l'algoritmo termina qui, altrimenti si procederà con il passo 5.
 5. Trova la cella minore che non sia tracciata da alcuna linea e poi sottrarre quel valore a tutte le righe non tracciate, sommare poi quel valore ad ogni colonna tracciata. Infine tornare al passo 3.

Al termine dell'algoritmo si selezionano $k=\min(n,m)$ zeri dalle celle della matrice tali che ogni zero appartenga ad una sola riga e ad una sola colonna. Le coordinate di queste celle corrisponderanno agli assegnamenti detection-tracker ottimali da attuare nella matrice originale.

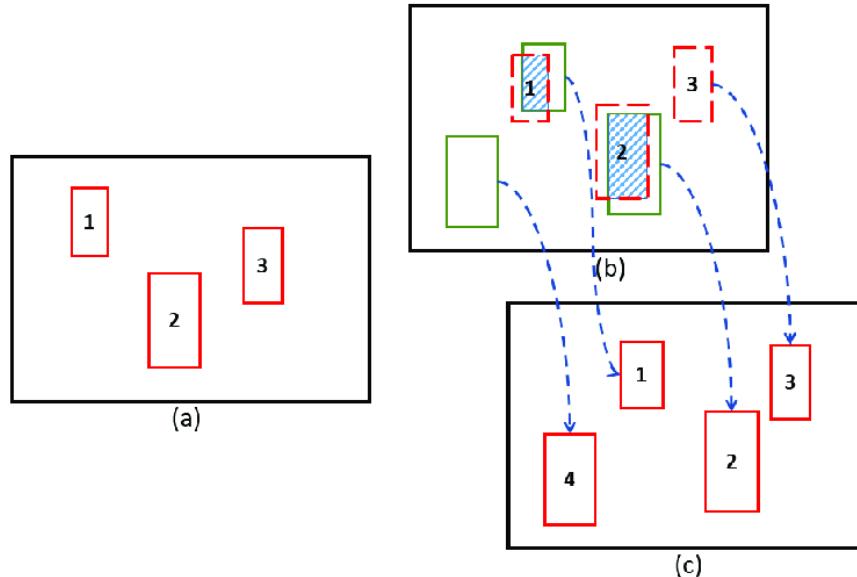


Figura 7: Esempio di assegnazione detection-tracker

Come si può vedere in figura 7 i rettangoli rossi raffigurano le bounding boxes dei trackers mentre quelle verdi rappresentano le bounding boxes individuate dalla detection. I trackers

1 e 2 vengono assegnati regolarmente, l'elemento tracciato dal tracker 3 è sparito dal frame ed è comparso un nuovo elemento che verrà tracciato da un nuovo tracker.

4.2.3 Gestione delle detections e dei trackers non assegnati

Nel caso in cui nella matrice precedente si abbia avuto che n è diverso da m , significa che sono presenti dei trackers o delle detections che non sono state assegnate. Inoltre, le coppie tracker-detection associate con successo dall'algoritmo il quale IoU sia però inferiore alla soglia stabilita verranno scartate ed aggiunte nelle rispettive liste di trackers e detections non assegnate. La mancata assegnazione di un tracker al suo elemento potrebbe avere due cause scatenanti: la prima è che l'oggetto tracciato sia momentaneamente sparito dal video a causa di una detection non andata a buon fine. Questo evento può essere per esempio dovuto ad una momentanea perdita di qualità di un frame, un occultamento di un oggetto o da una sfocatura causata da un movimento. La seconda causa è che l'oggetto sia effettivamente assente in un frame del video a causa di un suo spostamento oppure da un cambio di inquadratura. In questo caso è poco probabile che l'oggetto scomparso si ripresenti nel frame successivo ma è più ragionevole pensare che esso non si ripresenti più, almeno nel breve periodo. In quanto un tracker non ha modo di sapere quale delle due cause sia quella che abbia realmente portato alla sua mancata assegnazione, allora in presenza di questo evento viene solamente incrementato un contatore interno al tracker che tiene conto dei frames consecutivi trascorsi senza che il tracker venga associato all'oggetto da esso tracciato. Non appena questo contatore supera una certa soglia, detta **soglia di cancellazione**, il relativo tracker viene cancellato e l'elemento da esso tracciato viene considerato come perduto. Se un elemento perduto dovesse successivamente ricomparire in un frame esso verrà considerato come un nuovo oggetto venendo quindi tracciato da un nuovo tracker. Un discorso analogo vale anche per quelle labels che sono individuate dalla detection ma che non vengono assegnate a nessuno dei trackers già esistenti. Con molta probabilità si tratta di una nuova oggetto ed è quindi opportuno creare un nuovo tracker per iniziare a tracciarlo. Tuttavia, prima che il legame tracker-detection diventi effettivo non basta che l'assegnazione avvenga una sola volta. Potrebbe infatti accadere che a causa di una detection sbagliata un oggetto di una determinata categoria compaia erroneamente in un solo frame per poi non ripresentarsi più. In questo caso sfortunato si verrebbe a creare un tracker la cui utilità sarebbe poco significativa consumando un Id per associarlo ad un oggetto individuato per errore. Per risolvere questo problema ogni tracker implementa un altro contatore per tenere traccia di quante volte al tracker stesso viene assegnato un oggetto, questo contatore rappresenta quindi la durata in frames del tracciamento. Solamente dopo che questo contatore abbia superato una certa soglia, detta **soglia di assegnazione**, verrà assegnato un ID al tracker e la sua corrispondente bounding box verrà mostrata nel video.

4.2.4 Miglioramento: utilizzo dell'hash dell'immagine come metrica di supporto

5 Tecnologie

5.1 Python

Il linguaggio di programmazione utilizzato per perseguire gli obiettivi del progetto è stato Python v3.7, si tratta di un linguaggio di programmazione di alto livello il cui obiettivo è quello di facilitare la leggibilità del codice ed adattarsi a diversi paradigmi di programmazione come quello procedurale, ad oggetti e funzionale. La motivazione per la scelta dell'utilizzo di questo linguaggio, oltre alla sua semplicità, è per il suo supporto di numeri frameworks e moduli relativi al deep learning e alla computer vision tra i quali Tensorflow e OpenCV. Qualsiasi modulo aggiuntiva può essere semplicemente installata eseguendo il comando:

```
pip install nome_modulo
```

Per lanciare un programma viene utilizzato il comando:

```
python nome_file.py
```

Altri moduli che sono stati utilizzati comprendono Numpy, Matplotlib e Pytest.



Figura 8: Logo di Python

5.2 Pycharm

Pycharm è un IDE per programmare in Python sviluppato da JetBrains. Le sue caratteristiche più importanti includono:

- Un sistema intelligente di completamento automatico del codice;
- Analisi statica del codice eseguita a tempo di esecuzione;
- Individuazione e risoluzione veloce degli errori tramite proposte di correzione;
- Possibilità di lavorare in un ambiente di sviluppo virtuale dove per ogni progetto vengono installate solamente le proprie dipendenze e i propri moduli.

5.3 Tensorflow

Tensorflow è un framework gratuito e open-source per lo sviluppo e l'allenamento di modelli di machine learning come le reti neurali. Ha la particolarità che i dati vengono gestiti attraverso dei grafi computazionali dove i nodi rappresentano delle operazioni matematiche da eseguire e gli archi rappresentano degli array multidimensionali contenenti i dati sui quali svolgere le operazioni (tensori). La sua architettura permette di svolgere le operazioni sia usando la CPUs che le GPUs in modo da eseguire operazioni con alto livello di parallelismo. Il modello di rete neurale utilizzato per effettuare la detection sulle immagini è stato implementato con Tensorflow.



Figura 9: Logo di Tensorflow

5.4 OpenCV

OpenCV è una libreria open-source orientata allo sviluppo di applicazioni di computer vision in tempo reale. E' stata scritta originariamente in C++ ma offre anche il supporto ad altri linguaggi di programmazione come Python. OpenCV è un' ottima libreria quando si ha bisogno di manipolare immagini e video permettendo di compiere operazioni sia di alto livello che di basso livello operando sui singoli pixels. Sono inoltre presenti diversi algoritmi di object detection ed object tracking già implementati permettendo quindi di sperimentarli tutti senza apportare troppe modifiche al codice esistente. La libreria è stata anche utilizzata per scomporre un video nei suoi singoli frames oltre che per disegnare su di essi.

5.5 Numpy

Numpy è un modulo di Python che fornisce il supporto per la gestione di matrici e array multidimensionali di grandi dimensioni. Dispone anche di una vasta collezione funzioni

matematiche per lavorare ad alto livello su di essi. Viene spesso utilizzato per operare su grandi quantità di dati in modo rapido ed efficiente.

5.6 Matplotlib

Matplotlib è una libreria grafica sviluppata per Python impiegata principalmente per disegnare grafici o altre figure con lo scopo di rappresentare dei dati utilizzando il minor numero di righe di codice possibile. In fase di allenamento di una rete neurale viene spesso usato per mostrare l'andamento della variazione dell'errore e dell'accuratezza.

5.7 Pytest

Pytest è un framework per Python che permette di scrivere dei test per testare il codice permettendone la loro esecuzione in modo automatico. Per installarlo viene utilizzato il comando:

```
pip install pytest
```

mentre per lanciarne l'esecuzione viene usato il comando:

```
pytest nome_file
```

Per convenzione il file contenente i test relativi ad un solo modulo è stato chiamato "nome_modulo_test.py" dove "nome_modulo.py" è il modulo ad esso associato. I metodi che eseguono un test devono iniziare con "test_," , in caso contrario non verranno riconosciuti come tali e la loro esecuzione non verrà lanciata. Sono stati implementati dei test di unità per tutte le funzioni più complesse o critiche in modo da verificarne il corretto funzionamento. Sono anche stati sviluppati dei test di integrazione per ogni algoritmo implementato in modo da assicurarsi del loro corretto funzionamento. Per valutare l'esecuzione dell'intero sistema sono invece state utilizzate delle metriche per misurare la bontà del risultato in quanto i test di sistema non sarebbero stati adatti per misurare un risultato non deterministico come il riconoscimento di oggetti.

6 Sviluppo

6.1 Sviluppo algoritmo per riconoscimento di elementi in un'immagine frammentata

Al fine di implementare l'algoritmo è stata creata una libreria contenente numerose funzioni per lavorare con le labels ritornate dal modello come per esempio ottenerne la posizione o controllare le loro intersezioni con altre entità. Una labels è composta da un array formato da quattro numero interi caratterizzanti il vertice in alto a sinistra e quello in basso a destra del bounding box, un intero per rappresentare la categoria dell'elemento ed infine un numero decimale compreso tra 1 e 0 per indicarne lo score. Viene data la possibilità all'utente di ridefinire la propria *condizione di correlazione*, la *regola di classificazione* e la *condizione di matching* passandole come parametro direttamente nella funzione. Altri parametri passabili includono: le labels, le dimensioni delle regioni e dello stride e la quantità di overlap, sia in verticale che in orizzontale, e di tolleranza in pixels. Alla fine l'algoritmo ritornerà una lista di tutte le labels presenti nell'immagine.

Nella seguente sezione ne viene riportata una sua implementazione.

6.1.1 Implementazione

```
1 from modules.faster_nms import faster_nms
2 import numpy as np
3 import modules.labels as lb
4
5
6 def process_image_labels(boxes, region_size=(300, 300), stride_size=(270, 270),
7     , overlap=(0, 0), tol=0, threshold_match=50, correlation_condition=None,
8     find_category=None, find_group_label=None):
9
10     # set user defined conditions
11     if correlation_condition is None:
12         condition = lb.condition
13     if find_category is None:
14         find_category = lb.find_category
15     if find_group_label is None:
16         find_group_label = lb.find_group_label
17
18     w, h = lb.img_dim_from_boxes(boxes)
19     regions = lb.generate_regions(w, h, region_size, stride_size)
20     boxes = faster_nms(boxes, overlapThresh=0.8)
```

```

20     # Keeps only the labels intersecting one or more region edges (ne_boxes =
21     # near edges boxes)
22     ne_boxes, ne_boxes_indexes = lb.get_intersect_edges_labels(boxes, regions,
23     tol, overlap)
24     # All labels are set as not-checked and not-grouped
25     checked = np.zeros((np.ma.size(ne_boxes, 0)))
26     grouped = np.ones((np.ma.size(ne_boxes, 0))) * (-1)
27     # number of label groups
28     n_group = 0
29     for i in range(len(ne_boxes)):
30         checked_boxes = np.zeros((0, 6))
31         # Select a label and sets it as checked
32         if not checked[i]:
33             checked[i] = True
34         checked_boxes = np.vstack([checked_boxes, ne_boxes[i]])
35         while True:
36             found = 0
37             # If a label is checked then it means it is also not-grouped
38             # For each label checked but not-grouped:
39             for j in range(len(ne_boxes)):
40                 if checked[j] and grouped[j] == -1:
41                     for k in range(len(ne_boxes)):
42                         # join conditions
43                         if not checked[k] and lb.intersection(ne_boxes[j, :],
44                             ne_boxes[k, :], tol) and condition(ne_boxes[j, :],
45                             ne_boxes[k, :], regions, tol,
46                             overlap) and lb.matched(ne_boxes[j, :],
47                             ne_boxes[k, :], threshold_match)
48                         and not lb.belong_same_region_strict_group(ne_boxes[k, :],
49                             checked_boxes, regions, tol):
50                             # The new found label is set as checked
51                             checked[k] = True
52                             found += 1
53                             checked_boxes = np.vstack([checked_boxes,
54                             ne_boxes[k]])
55                             # The cycle is repeated until no new labels can be found
56                             if found == 0:
57                                 break
58             # All the checked labels are set as grouped and is them assigned a
59             # number with the
60             # purpose to identify their group ID
61             for j in range(len(ne_boxes)):
62                 if checked[j] and grouped[j] == -1:
63                     grouped[j] = n_group
64             n_group += 1

```

```

56     # Now we have that each label only belong to a single group
57     # Now that we have grouped the labels each group is merged into a label
58     # containing all of them
59     new_boxes = np.zeros((0, 6))
60     for i in range(n_group):
61         # For each group
62         v = np.zeros((0, 6))
63         count = 0
64         for j in range(len(ne_boxes)):
65             if grouped[j] == i:
66                 v = np.vstack((v, ne_boxes[j, :]))
67                 count += 1
68         # Find the coordinates of the labels containing the whole group
69         (xx1, yy1), (xx2, yy2) = find_group_label(v)
70         # find_category can be redefined by the user, it is based on the
71         # category and score of the labels of the group
72         categories, scores = find_category(v)
73         for j in range(len(categories)):
74             new_boxes = np.vstack((new_boxes, np.array([xx1, yy1, xx2, yy2,
75             categories[j], scores[j]])))
76     # new_boxes are the labels resulting from grouping algorithm
77
78
79
80     return final_boxes

```

Listing 1: Python example

6.1.2 Test di integrazione

Al fine di verificarne la correttezza sono stati implementati 80 test di integrazione. Essi hanno il compito di testare l'algoritmo su diverse disposizioni e quantità di labels impostando i parametri con differenti valori in modo tale da riconoscerne il maggior numero di errori possibile. Sono state in tutto testate in totale cinque disposizioni di labels differenti con due diversi valori di overlap, tolleranza, stride e dimensione delle regioni arrivando ad un totale 80 test passati tutti con esito positivo.

7 Risultati ottenuti

7.1 Metriche utilizzate per singole immagini

Nell'ambito della computer vision vengono tipicamente utilizzate due tipi di metriche per misurare due diverse proprietà:

- Corretta determinazione della posizione degli oggetti;
- Rilevamento l'esistenza degli oggetti nell'immagine e la loro corretta classificazione.

Le metriche utilizzate per misurare la bontà dei risultati del progetto sono AP (Average Precision) e mAP (mean Average Precision) le quali misurano la seconda proprietà sopra elencata. Prima di iniziare a descrivere le due metriche è necessario definire tre concetti che saranno poi coinvolti per il calcolo dei valori delle metriche.

7.1.1 Precision

La precision misura l'accuratezza delle classificazioni ed indica la percentuale di classificazioni corrette sulla base di quelle totali e viene calcolata come:

$$precision = \frac{TP}{TP + FP}$$

Dove TP (True Positives) è il numero di classificazioni corrette e FP (False Positives) è il numero di classificazioni errate. E' da sottolineare che se uno stesso elemento viene individuato più di una volta, solo la prima volta conterà come TP mentre il resto delle volte conterà come FP. Questa metrica fornisce un'idea sulla correttezza dei risultati.

7.1.2 Recall

La recall misura quanti oggetti sono stati individuati e classificati correttamente sulla base degli oggetti totali, viene calcolata come:

$$precision = \frac{TP}{TP + FN}$$

Dove TP (True Positives) è sempre il numero di classificazioni corrette mentre FN (False Negatives) è il numero di oggetti che non sono stati individuati ma che se sarebbero stato corretto individuare. Questa metrica fornisce un'idea sulla completezza dei risultati.

7.1.3 Intersection over union

L'intersection over union (IoU) misura il grado di sovrapposizione tra due aree e viene usato per misuare il grado di sovrapposizione tra la box dell'oggetto individuato e la box dell'oggetto reale. Viene calcolata come:

$$IoU = \frac{AI}{AU}$$

Dove AI (Area dell'Intersezione) corrisponde all'area dell'intersezione tra le due box e AU (Area dell'Unione) corrisponde all'area dell'unione tra le due box. Una label per essere considerata come corretta deve avere un valore di IoU maggiore di una soglia stabilità (per esempio 0.5).

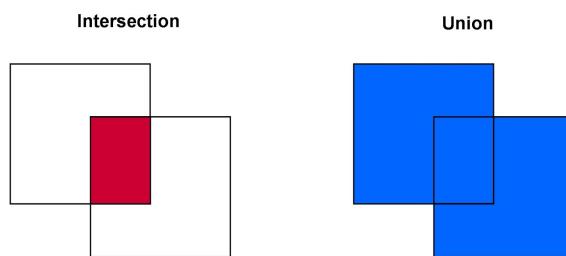


Figura 10: Esempio di intersezione e di unione

7.1.4 Average Precision

Una delle due metriche utilizzate è l'average precision (AP) e per calcolarla si fa la media delle precisioni di undici diversi valori di recall equamente distribuiti. Questa metrica viene applicata ad una sola categoria di elementi e viene calcolata tramite la seguente formula:

$$AP = \frac{1}{11} \sum_{Recall_i} Precision(Recall_i)$$

Dove $i=[0, 0.1, 0.2, \dots, 1.0]$ in quanto $0 < recall < 1$. Inoltre la precisione di uno specifico valore di recall viene calcolata nel seguente modo:

$$Precision(Recall_i) = \max Precision(Recall_j) \quad \text{and} \quad j \geq i$$

L'AP è un valore che riassume la forma della curva precision/recall per una data categoria.

7.1.5 Mean Average Precision

La mean average precision (mAP) è la media delle AP di tutte le categorie calcolata su diverse soglie di IoU:

$$mAP_{IoU=x\%} = \frac{1}{n} \sum_{i=0}^n AP_i$$

Dove i è il numero totale di categorie sulle quali è stata calcolata la propria AP e x rappresenta diverse soglie di IoU.

8 Glossario

B

Box E' un rettangolo che identifica il perimetro entro quale l'oggetto riconosciuto si trova.

C

Categoria L'obiettivo della classificazione è quello di assegnare all'oggetto individuato la sua categoria di appartenenza.

L

Label Etichetta che viene data ad ogni elemento riconosciuto e ne indica la categoria di appartenenza, una bounding box ed uno score.

O

Overlap L'overlap indica lo spessore in pixels dei confini delle regioni. E' usato per sapere se un' entità interseca un confine o meno.

R

Raggruppamento Insieme di labels correlate tra loro tale che da una qualsiasi label del gruppo sia possibile raggiungere qualsiasi altra label dello stesso insieme passando solo per label *vicine*⁶.

Regione Una sezione di un'immagine con un'area di dimensione minore dell'area dell'immagine originale.

S

Score La probabilità che la categoria associata all'elemento riconosciuto sia quella corretta.

⁶Due label sono considerate vicine se sono intersecate tra di loro ed intersecano lo stesso confine di regione

Soglia E' il valore che lo score di una label deve eguagliare o superare per essere considerata affidabile.

Stride E' un attributo da tenere conto in fase di frammentazione di un'immagine ed indica quanti pixels della regione tralasciare, sia in verticale che in orizzontale, prima che cominci quella successiva. Lo stride orizzontale può avere dimensione diversa da quello verticale. La sua applicazione ha come scopo quello di creare delle zone di sovrapposizione tra le varie regioni.

T

Tolleranza La distanza in pixels per la quale una label può discostare da una zona di interesse per cui sia ancora considerata essere intersecata con la zona di interesse.

9 Bibliografia

Riferimenti bibliografici

- [1] Sito web dell'azienda Studiomapp.
<https://www.studiomapp.com/en/>
- [2] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. University of Toronto.
- [3] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. UC Berkeley.
- [4] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Microsoft Research.
- [5] Vit Ruzicka, Franz Franchetti. *Fast and accurate object detection in high resolution 4K and 8K video using GPUs*. Carnegie Mellon University.
- [6] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, Michael Felsberg. *Accurate Scale Estimation for Robust Visual Tracking*. Linköping University.
- [7] David Held, Sebastian Thrun, Silvio Savarese. *Learning to Track at 100 FPS with Deep Regression Networks*. Stanford University.
- [8] Imagenet database.
<http://www.image-net.org/>
- [9] Guida all'utilizzo del framework Tensorflow.
<https://www.tensorflow.org/>
- [10] Guida all'utilizzo del modulo matplotlib.
<https://matplotlib.org/>
- [11] Guida all'utilizzo del modulo numpy.
<https://www.numpy.org/>
- [12] Guida all'utilizzo del modulo OpenCV.
<https://www.pyimagesearch.com/>

-
- [13] Studio delle metriche per l'object detection.
[https://medium.com/@jonathan_hui/
map-mean-average-precision-for-object-detection-45c121a31173](https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173)
 - [14] Tracking e detection
<https://github.com/kcg2015/Vehicle-Detection-and-Tracking>
<https://towardsdatascience.com/computer-vision-for-tracking-8220759eee85>
 - [15] Filtro di Kalman
<https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/>
 - [16] Algoritmo di Kuhn-Munkres
<https://brilliant.org/wiki/hungarian-matching/>

10 Appendice