DSproj

Gruppo di lavoro

• Sebastiano Regini, 677636, s.regini@studenti.uniba.it

Repository

https://github.com/SebastianoReginiUniba/DSproj

AA 2021-2022

Introduzione

Il progetto si basa sulla computer vision per verificare, all'interno di un video, se le persone rispettano il distanziamento sociale. Si occupa dell'identificazione dei soggetti ripresi e calcola la distanza tra questi, evidenziando con un color code verde coloro che sono lontani da altri di almeno un metro e in rosso coloro che violano questo vincolo. Infine effettua una misurazione sull'esistenza di possibili cluster, rappresentanti eventuali assembramenti che si creano all'interno di un determinato frame del video.

Lo studio è stato effettuato per fare in modo di comprendere le potenzialità offerte dalla computer vision unita alla possibilità di effettuare calcoli una volta estratte determinate informazioni da un'immagine (come, appunto, il calcolo della distanza tra due elementi di tipo "persona" individuati in ogni singolo frame del video, in modo da poter verificare se questa supera o meno un determinata soglia pre-definita).

Inoltre, il calcolo dei cluster permette di considerare un insieme di oggetti singoli come un gruppo, andando quindi ad identificare l'esistenza di insiemi di elementi (da 2 a più) che violano questo vincolo. Questo ci permette di comprendere i momenti (frame) in cui si sono venute a creare situazioni di violazione della regola di distanziamento sociale e ad indicare con buona precisione quanti agglomerati di individui erano presenti, in modo da poter effettuare diverse considerazioni (quanto è stato rispettato il distanziamento, quanti raggruppamenti si sono formati ed in quali particolari momenti ecc.)

Rilevamento degli oggetti

Per poter verificare il rispetto delle regole sul distanziamento, occorre prima di tutto identificare le persone all'interno di un video. Si è deciso di fare questo controllo utilizzando una particolare architettura di rete neurale chiamata YOLO. Esistono varie versioni di questa rete, che differiscono dalle dimensioni delle immagini sulle quali sono state addestrate. Si è

optato per l'utilizzo della 320 (ovvero quella addestrata con immagini di dimensioni 320x320).

Per recuperare questo modello addestrato, si utilizzano anche i pesi (file *.weights*) e le configurazioni (file *.cfg*). Inoltre, tale rete neurale è addestrata utilizzando il Coco dataset, che contiene 200.000 immagini labelizzate, con un milione di oggetti al suo interno divisi in 80 categorie. Tali labels sono contenuti all'interno del file *coco.names*.

È stata caricata la rete neurale attraverso la funzione **load_yolo**, nel quale vengono letti i file *.weight* e *.cfg* per poter utilizzare YOLO. Si carica un array che conterrà le classi riconosciute da tale modello. Per poter eseguire una classificazione con una rete neurale caricata con openCV, bisogna avere a disposizione l'oggetto corrispondente all'ultimo strato, perciò sono stati estratti i nodi di questo strato dai layers ed inseriti in una variabile a sé, *output_layers*.

Successivamente è stata creata una nuova funzione, **detect_object**, che utilizza la rete neurale per riconoscere i tipi di oggetto che è possibile effettivamente riconoscere con questo modello. Per poter utilizzare l'immagine che gli viene passata (ovvero un frame del video) essa viene prima di tutto ridimensionata, per portarla alle stesse dimensioni delle immagini con cui è stata addestrata la rete, e poi utilizzata per creare un *blob*, cioè un oggetto con la codifica utilizzata da openCV per darlo in input ad una rete neurale.

Una volta preparato, si eseguirà la *forward propagation*, ovvero il processo di calcolo di un output di una rete neurale. Dall'attributo *shape* dell'output è possibile ottenere un vettore di due elementi: mentre il primo è pari al numero di quadrati in cui viene sezionata l'immagine per effettuare la classificazione, il secondo è la dimensione del vettore restituito dalla classificazione di ognuno di quei quadrati. Il vettore è composto da:

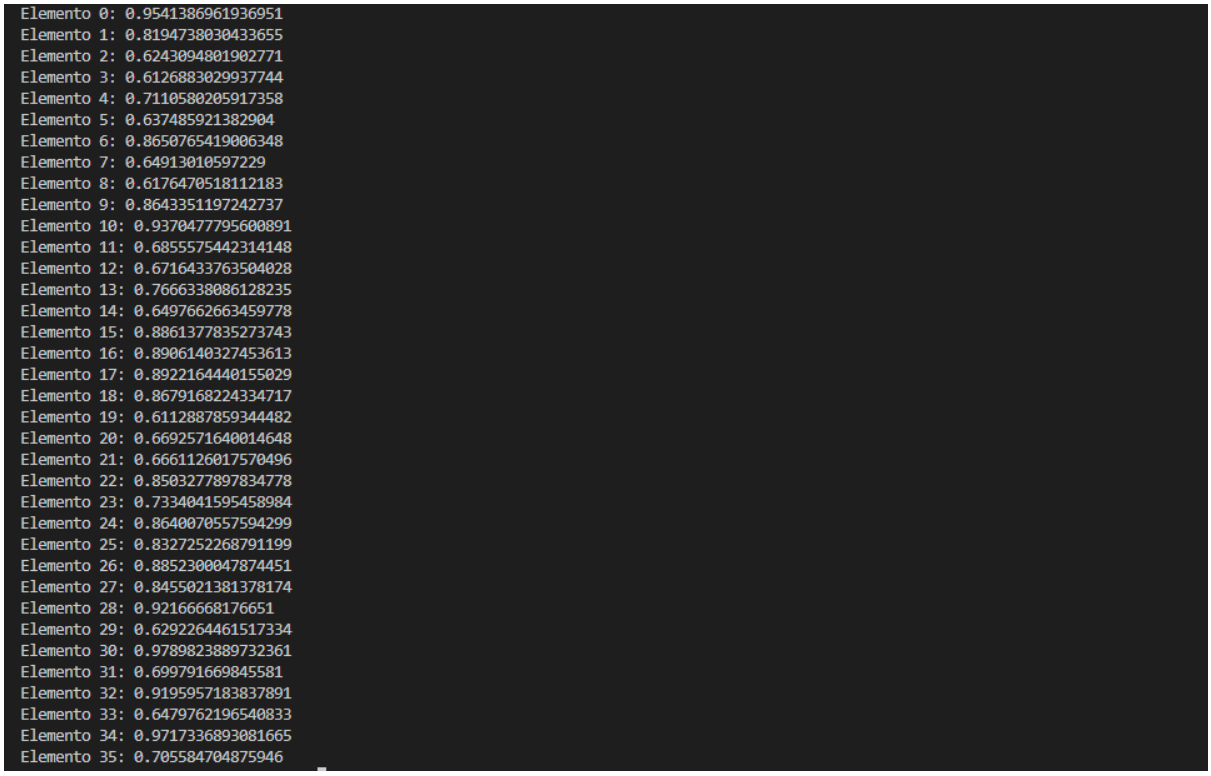
- p_c, cioè la probabilità che l'immagine contenga un oggetto;
- b_x e b_y, ovvero le coordinate del bounding box;
- b_h e b_w, l'altezza e la larghezza del bounding box;
- c_1, c_2, c_3 ecc, le probabilità di appartenenza ad una classe.

Nel caso di YOLO, poiché riesce a riconoscere 80 classi di oggetti, questo valore sarà pari a 85.

Poiché la rete potrebbe generare più box per lo stesso oggetto, quindi riconoscerlo più volte, si è deciso di utilizzare la tecnica nota come *Non Max Suppression*, che andrà a mantenere solo i box con probabilità maggiore. Fortunatamente openCV mette già a disposizione questa tecnica attraverso il metodo cv2.dnn.NMSBoxes().

Valutazione

Come si può constatare, gli elementi individuati dalla rete hanno una percentuale abbastanza alta. Non mancano casi meno accurati, con valori che scendono fino al 61%, ma nella maggior parte dei casi il valore si mantiene sopra l'80%.



Calcolo della distanza

Il problema principale dell'identificazione della distanza tra le persone in un video è che la prospettiva impedisce di avere una distanza uniforme. Essendo il video in 2D mentre ciò che ha catturato è un ambiente tridimensionale e quindi comprensiva di profondità, i soggetti più "lontani" avranno una distanza più piccola da identificare, mentre quelli più vicini dovranno averne una più grande.

Si è optato per stimare una distanza fissa, pari a 30 pixel, salvata in una costante.

La formula presa in considerazione è quella della distanza euclidea. Utilizzeremo una funzione del modulo **distance** della libreria **scipy**, ovvero **cdist**.

Si crea la funzione **compute_distances** che prende in input i centroidi corrispondenti ai soggetti individuati da YOLO e calcola la distanza prendendo un elemento dell'insieme e confrontandolo con tutti quelli dello stesso insieme.

C'è un problema che si viene a creare con questa pratica: il calcolo della distanza di un elemento con sé stesso. Questo problema porta potenzialmente a molti falsi positivi. La risoluzione si ottiene attraverso la riassegnazione di tale distanza, andando a sostituirla con un valore molto alto. Questo viene fatto servendosi della funzione **eye** di **numpy**, poiché le distanze dei centroidi con loro stessi si trovano nella diagonale principale della matrice delle distanze che viene restituita dalla funzione **cdist** citata in precedenza.

Clusterizzazione degli oggetti rilevati

È stata introdotta la clusterizzazione per identificare (e stampare nel frame) la quantità di gruppi di persone che non rispettano il distanziamento sociale.

Si è deciso di utilizzare la libreria scikit-learn per effettuare la clusterizzazione degli oggetti rilevati. L'algoritmo DBSCAN viene utilizzato per effettuare la clusterizzazione, mentre la metrica utilizzata per quest'ultima è la distanza tra i centroidi degli oggetti rilevati.

L'algoritmo DBSCAN è in grado di identificare delle "anomalie" rispetto alla normale distribuzione degli oggetti con un'elevata precisione. I risultati mostrano che viene identificato, per ogni frame, il numero esatto di anomalie (cluster) presenti.

Il progetto utilizza una distanza minima, tra i centroidi degli oggetti, di 30 pixel come soglia per determinare se gli oggetti sono in contatto. Inoltre, la clusterizzazione viene effettuata solo sugli oggetti con una confidenza superiore al livello di confidenza minimo impostato, che nel particolare caso è posto pari a 2 (un cluster, perciò, sarà identificato dalla vicinanza di almeno due persone).

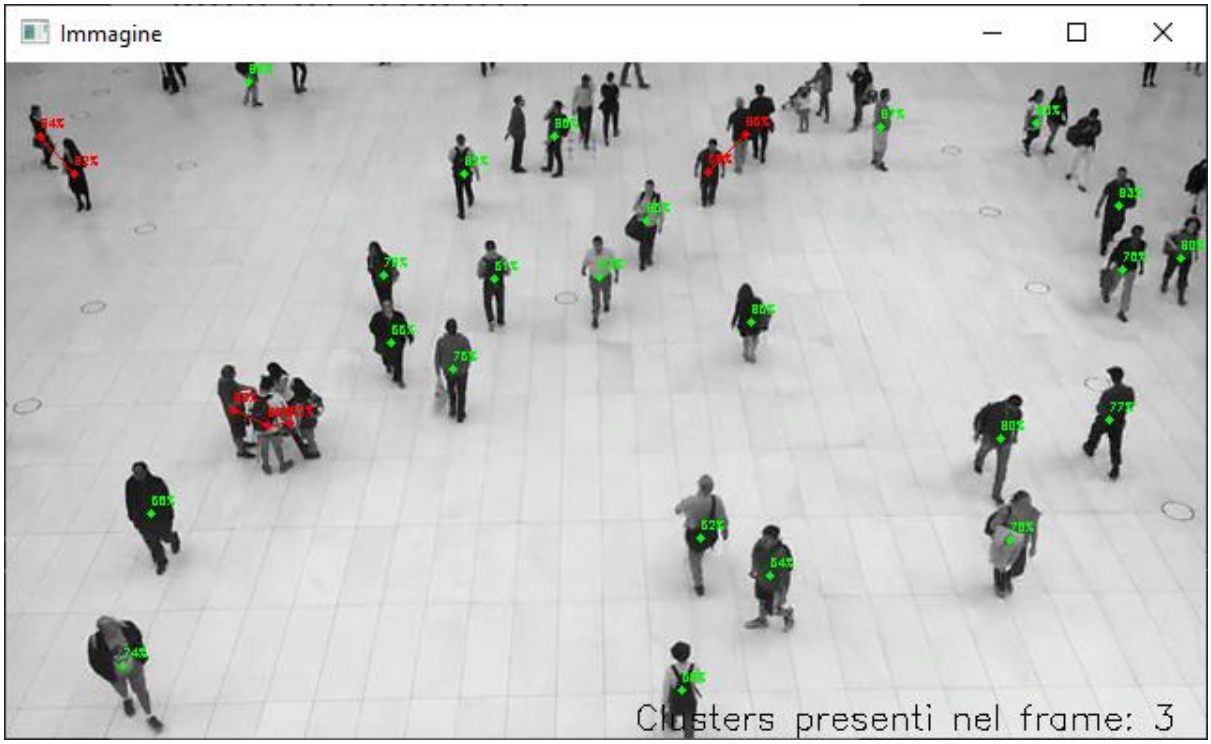
Valutazione

```
Cluster 0: 6.45%, con numero di elementi 2 su un totale di 31 soggetti
Cluster 1: 9.68%, con numero di elementi 3 su un totale di 31 soggetti


Cluster 0: 6.25%, con numero di elementi 2 su un totale di 32 soggetti
Cluster 1: 6.25%, con numero di elementi 2 su un totale di 32 soggetti
Cluster 2: 9.38%, con numero di elementi 3 su un totale di 32 soggetti


Cluster 0: 6.25%, con numero di elementi 2 su un totale di 32 soggetti
Cluster 1: 6.25%, con numero di elementi 2 su un totale di 32 soggetti
Cluster 2: 9.38%, con numero di elementi 3 su un totale di 32 soggetti


Cluster 0: 5.88%, con numero di elementi 2 su un totale di 34 soggetti
Cluster 1: 5.88%, con numero di elementi 2 su un totale di 34 soggetti
Cluster 2: 5.88%, con numero di elementi 2 su un totale di 34 soggetti
Cluster 3: 8.82%, con numero di elementi 3 su un totale di 34 soggetti


Cluster 0: 5.56%, con numero di elementi 2 su un totale di 36 soggetti
Cluster 1: 5.56%, con numero di elementi 2 su un totale di 36 soggetti
Cluster 2: 5.56%, con numero di elementi 2 su un totale di 36 soggetti
Cluster 3: 8.33%, con numero di elementi 3 su un totale di 36 soggetti
Cluster 4: 5.56%, con numero di elementi 2 su un totale di 36 soggetti


Cluster 0: 5.88%, con numero di elementi 2 su un totale di 34 soggetti
Cluster 1: 5.88%, con numero di elementi 2 su un totale di 34 soggetti
Cluster 2: 8.82%, con numero di elementi 3 su un totale di 34 soggetti
Cluster 3: 8.82%, con numero di elementi 3 su un totale di 34 soggetti


Cluster 0: 6.25%, con numero di elementi 2 su un totale di 32 soggetti
Cluster 1: 6.25%, con numero di elementi 2 su un totale di 32 soggetti
Cluster 2: 9.38%, con numero di elementi 3 su un totale di 32 soggetti
Cluster 3: 9.38%, con numero di elementi 3 su un totale di 32 soggetti


Cluster 0: 9.38%, con numero di elementi 3 su un totale di 32 soggetti
Cluster 1: 9.38%, con numero di elementi 3 su un totale di 32 soggetti
```

I cluster vengono correttamente individuati, per ogni frame vengono presi in considerazione tutti quelli presenti e viene calcolata la percentuale di soggetti implicati in essi sul totale di persone individuate nell'intero fotogramma.

Il numero fortunatamente è basso, il che implica un rispetto delle norme del distanziamento sociale nella maggior parte degli individui catturati dal video.

Esempio di output



Si può vedere dal frame come la rete riconosca la maggior parte delle persone, nonostante l'inquadratura abbia un angolo inclinato e la ripresa sia fatta dall'alto (probabilmente da una telecamera).

Su ogni soggetto viene apposto il centroide che lo rappresenta, insieme ad un piccolo numero che identifica la percentuale con la quale è stato riconosciuto come "persona". I centroidi di colore verde sono applicati sui soggetti che rispettano (almeno in un determinato frame) il distanziamento sociale, mentre quelli di colore rosso evidenziano quelli che stanno mantenendo una distanza tra loro inferiore alla soglia minima di sicurezza, in più sono connessi tra loro tramite una linea di colore rosso per evidenziare effettivamente chi è troppo vicino a chi.

Per ognuno dei frame, viene stampato in basso a destra il numero di clusters presenti, che coincide con i raggruppamenti di persone individuate con il colore rosso e con una linea che li collega.

Conclusione

Il progetto ha dimostrato l'efficacia dell'utilizzo dell'algoritmo YOLO per la rilevazione degli oggetti in immagini e dell'algoritmo DBSCAN per la clusterizzazione degli oggetti rilevati. La combinazione di queste tecniche ha permesso di sviluppare un sistema di rilevamento delle persone che rispettano/non rispettano il distanziamento sociale efficace e preciso.