UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in
Ingegneria Informatica e dell'Automazione

Progettazione e sviluppo di un modulo software integrativo per acquisizione, validazione automatica ed invio ad infrastruttura cloud di immagini, nell'ambito della planogram compliance analysis.

Design and development of an integrative software module for acquisition, automatic validation and sending of images to the cloud infrastructure, as part of a planogram compliance analysis.

Relatore:
Chiar.mo Prof. Adriano Mancini

Correlatore:
Ing. Marina Paolanti

Laureando:
Emanuele Incicco

Anno Accademico 2019-2020

Ringraziamenti

Ho il dovere di dedicare questa parte dell'elaborato alle persone che mi hanno supportato nella redazione dello stesso.

Innanzitutto, ringrazio il mio relatore *Adriano Mancini* e il mio correlatore *Marina Paolanti*, sempre pronti a darmi le giuste indicazioni in ogni fase della realizzazione dell'elaborato.

Un ringraziamento speciale inoltre va a *Rocco Pietrini* presso la società Grottini Lab che mi ha aiutato a condurre le ricerche oggetto dell'elaborato.

Indice

Capitolo 1

Introduzione

Il modulo software realizzato è entrato a far parte di una complessa architettura[1] che si occupa della gestione del *refill* degli scaffali di negozi quali, ad esempio, supermercati.

Per gestire il *refill* si ha bisogno di acquisire immagini dei prodotti contenuti negli scaffali. In queste potrebbero però essere presenti delle persone che coprono alcuni dei prodotti.

Questo modulo deve occuparsi quindi di acquisire immagini prive di persone, in cui i prodotti negli scaffali sono perciò ben visibili, e di caricare le stesse in un'architettura *cloud*[2].

Le immagini seriate nel tempo e caricate nel *cloud* verranno poi utilizzate per gestire appunto il *refill* degli scaffali.

1.1 Obiettivo

Il modulo ha l'obiettivo di acquisire l'immagine dei prodotti contenuti negli scaffali, effettuare la *detection*, per verificare che non ci siano persone presenti, ed eventualmente inviare l'immagine acquisita nel *cloud*.

Come già anticipato la gestione del *refill* può essere realizzata solo se nell'immagine non ci sono persone che coprono i prodotti che si trovano negli scaffali, pertanto è necessaria la *detection* prima dell'invio della foto al *cloud*. In questo modo le immagini che verranno inviate alla piattaforma *cloud* mostreranno i soli prodotti contenuti negli scaffali senza eventuali persone presenti. Queste immagini seriate nel tempo verranno poi utilizzate per la gestione del *refill* degli scaffali.

[1] L'architettura nel suo complesso è spiegata dettagliatamente nella sezione 1.2.

[2] L'obiettivo del modulo software realizzato è spiegato più in dettaglio nella sezione 1.1.

Il modulo dovrà essere ottimizzato in modo da poterlo eseguire su una *Raspberry Pi 3A+* alimentata a batteria.

Si cercherà dapprima di accedere alla fotocamera della Raspberry (*Raspberry Pi Camera Module v2*) e scattare una foto. Successivamente si effettua la *detection*, se non vengono identificate persone nell'immagine acquisita si procede al caricamento della stessa nel *cloud* (un *bucket AWS*), altrimenti la si scarterà.

La *detection* è effettuata utilizzando modelli di reti pre-allenati sul *dataset COCO*.

1.2 Contributo della tesi

Il modulo realizzato è entrato a far parte di un'architettura complessa che si occupa principalmente di analizzare l'interazione che i clienti hanno con gli scaffali e, in particolare, con i prodotti in essi presenti.

Da questa osservazione partono poi diversi studi, come ad esempio la re-identificazione dei clienti, da realizzare in modalità *top-view*, così da garantire buoni risultati in caso di affollamento e nel contempo rispettando la *privacy* delle persone.

Altro oggetto di studio è il conteggio delle persone che entrano ed escono da un negozio, automatizzando quindi il processo di enumerazione e di direttive ai clienti (attendere o entrare).

Un ulteriore esempio è la gestione dei prodotti presenti nel negozio, proprio in quest'ultimo tipo di osservazione entra a far parte questo lavoro di tesi. Si riesce ad automatizzare e semplificare il processo di verifica di quali prodotti sono terminati e in quali scaffali o, eventualmente, il conteggio dei prodotti attualmente presenti.

Per realizzare tali funzioni bisogna però possedere immagini prive di eventuali persone che coprono i prodotti. Il compito del modulo software realizzato è proprio quello di garantire che ciò si realizzi, cioè che vengano acquisite e caricate nel *cloud* soltanto immagini prive di persone. Da queste poi si possono effettuare le azioni desiderate per la gestione del *refill*.

Questo ambito di ricerca ha ottenuto una particolare attenzione a causa della crescente domanda di applicazioni che monitorano il comportamento umano, ancor di più in questo ultimo periodo a causa della diffusione del *SARS-CoV-2*.

1.3 Struttura della tesi

Questa introduzione ha l'obiettivo di descrivere gli aspetti peculiari del modulo che si è realizzato.

Nel Capitolo 2 si analizzeranno gli strumenti utilizzati per la realizzazione del progetto. Si descriveranno dapprima i diversi modelli di reti utilizzati, in seguito il *dataset* COCO e il *dataset* utilizzato per i test, fino ad arrivare alla descrizione della libreria OpenCV e di Amazon S3, cioè il servizio di storage offerto da *Amazon Web Services* che utilizzeremo per l'archiviazione delle immagini nel *cloud*.

Nel Capitolo 3 descriveremo più in dettaglio il funzionamento e i passi seguiti per la realizzazione del modulo, ponendo l'enfasi sulle metriche usate per la valutazione dei vari modelli di reti.

Nel Capitolo 4 si riporteranno e confronteranno i risultati ottenuti dai test effettuati. Si analizzeranno infine i tempi richiesti dai diversi modelli e nei diversi sistemi in cui si sono eseguiti i test per effettuare l'operazione di *detection*.

Il capitolo finale esporrà le conclusioni e gli sviluppi futuri che possono essere realizzati a partire da questo lavoro di tesi.

1.4 Scelta dei modelli

Come già anticipato per la realizzazione di questo modulo si sono utilizzati modelli di reti già allenati sul *dataset COCO*. La bontà dei risultati ottenuti ci ha permesso infatti il loro utilizzo senza doverli ri-allenare.

Si è focalizzata l'attenzione su cinque modelli:

- SSD MobileNet v1;
- SSD MobileNet v2;
- SSD MobileNet v3 Small;
- SSD MobileNet v3 Large;
- Faster-RCNN Inception V2.

Sono state scelte le *MobileNets* in quanto sono reti che possiedono un buon rapporto velocità di esecuzione/precisione dei risultati ottenuti, rendendole perciò adatte in molti ambiti applicativi e tali da essere utilizzate anche in situazioni in cui si hanno a disposizione risorse limitate. Le *MobileNets* inoltre sono le reti che attualmente si utilizzano nella complessa architettura

della quale questo modulo è entrato a far parte (l'intero sistema spiegato nella sezione 1.2).

Infine si è scelto il modello di rete *Faster-RCNN Inception V2* così da osservarne il comportamento ed effettuare un confronto con le *MobileNets*. Questo modello di rete offre, in generale, più accuratezza a svantaggio della velocità.

Capitolo 2

Strumenti Utilizzati

In questo capitolo inizialmente si descriveranno e confronteranno i modelli di reti che si sono scelti. In seguito verranno descritti gli strumenti utilizzati per lo sviluppo di tale modulo.

2.1 Reti

In questa prima sezione del capitolo 2 si descrivono più in dettaglio i modelli di reti utilizzati nella realizzazione del modulo.

I modelli sono stati scaricati dal *Tensorflow detection model zoo* che riporta, oltre al link per il *download* dell'archivio ".tar.gz" contenente il modello, altre informazioni, quali:

- Velocità del modello (in ms), tale velocità dipende molto dall'hardware su cui il modello viene utilizzato. I tempi riportati sono stati ottenuti utilizzando una scheda grafica *Nvidia GeForce GTX TITAN X*;
- *mAP (mean Average Precision)* COCO del modello[1], valutato su un sottoinsieme del *dataset* COCO. Il valore riportato è stato arrotondato all'intero più vicino;
- Tipo di output, *boxes* o *masks*.

Nella tabella 2.1 si riportano le informazioni presenti nel *Tensorflow detection model zoo* per alcuni dei modelli che sono stati utilizzati nello sviluppo di tale progetto.

Nella tabella 2.2 troviamo invece le informazioni per i modelli *mobile*, cioè i restanti modelli utilizzati. Rispetto ai precedenti la velocità di questi ultimi

[1]La metrica *mAP (mean Average Precision)* è definita nella sezione 3.3.1.

modelli è indicata come *Pixel 1 Latency* e rappresenta appunto la *performace* ottenuta su tale dispositivo (*Google Pixel 1*).

Tabella 2.1: COCO - Trained models

Model name	**Speed (ms)**	**COCO mAP**	**Outputs**
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes

Tabella 2.2: COCO - Trained mobile models

Model name	**Pixel 1 Latency (ms)**	**COCO mAP**	**Outputs**
ssd_mobilenet_v3_large_coco	119	23	Boxes
ssd_mobilenet_v3_small_coco	43	15	Boxes

2.1.1 Caratteristiche Modelli Utilizzati

Un primo metro di paragone per l'efficienza dei modelli di reti è il *MACs (Multiply-Accumulates)*, o *MADDs (Multiply Adds)*. Questo parametro rappresenta il numero di operazioni di accumulazione multiple di cui si ha bisogno per calcolare un'inferenza in una singola immagine.

Nella figura 2.1 viene riportato il grafico che rappresenta i valori del *MADDs (Multiply Adds)* per diversi livelli di accuratezza e per vari modelli di reti, fra cui quelli utilizzati in questo progetto.

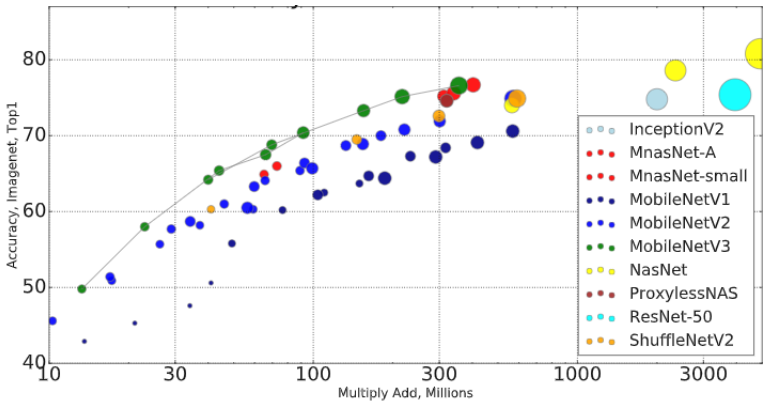

Figura 2.1: Accuratezza vs *MADDs* vs Modelli

Dalla figura precedente possiamo osservare che per le *MobileNets* abbiamo un aumento dell'accuratezza al crescere della versione. Si nota infatti

che per valori corrispondenti del *MADDs (Multiply Adds)* abbiamo una maggiore accuratezza della *MobileNetV2* rispetto la *MobileNetV1* e un ulteriore incremento dell'accuratezza della *MobileNetV3* rispetto la *MobileNetV2*.

2.1.2 MobileNets

Le caratteristiche principali dei modelli *MobileNets* sono: le piccole dimensioni, la bassa latenza (quindi la velocità) e la bassa energia richiesta. Tali proprietà li rendono adatti a soddisfare i vincoli delle risorse in una vastità di casi d'uso. Questi modelli possono anche essere eseguiti efficientemente su dispositivi mobili con l'utilizzo del *framework TensorFlow Lite*.

La figura seguente (Figura 2.2) riporta il confronto fra i tempi computazionali richiesti dai modelli *MobileNetV1* e *MobileNetV2* utilizzando il *framework TensorFlow Lite* nel dispositivo *Google Pixel 1*.

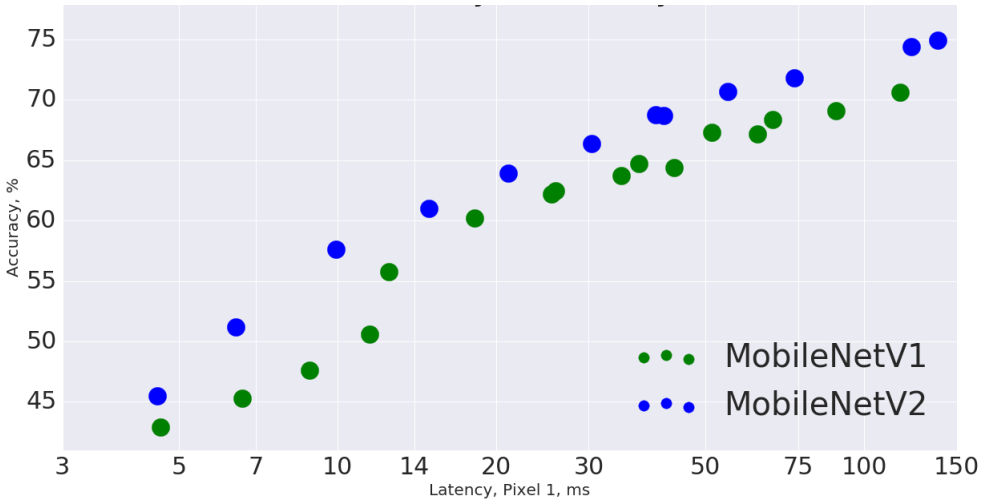

Figura 2.2: Tempi *MobileNetV1 vs MobileNetV2*

Come si deduce dalla figura la MobileNetV2 dà risultati con un'accuratezza maggiore per diversi valori di latenza, aumentano la velocità (cioè si riduce la latenza) e l'accuratezza. Con la *MobileNetV2* riusciamo quindi ad ottenere risultati, al pari del valore della latenza, con una maggiore accuratezza rispetto ai risultati della *MobileNetV1* o risultati, al pari del valore dell'accuratezza, con minore latenza (cioè in minor tempo).

La figura 2.3 riporta il confronto fra i tempi computazionali richiesti dai modelli *MobileNetV2* e *MobileNetV3* utilizzando il *framework TensorFlow Lite* nel dispositivo *Google Pixel 1*. Nel grafico di sinistra abbiamo

il confronto fra i modelli *mobile small* mentre in quello di destra troviamo rappresentato il confronto fra i modelli *mobile large*.

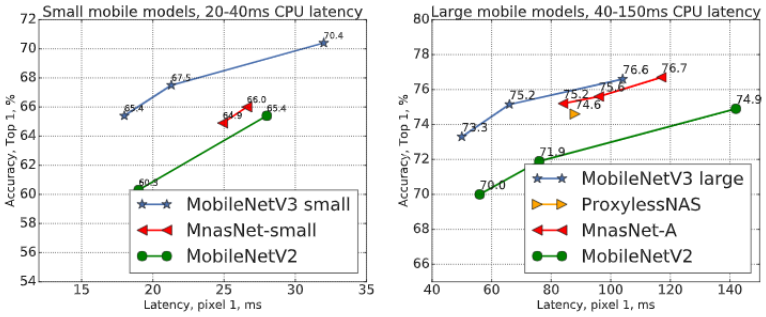

Figura 2.3: Tempi *MobileNetV2 vs MobileNetV3*

Si può osservare innanzitutto che i modelli *mobile small* danno risultati con un'accuratezza minore ma in un intervallo di latenza più ristretto (quindi con tempi più brevi) rispetto ai modelli *mobile large* che richiedono invece tempi maggiori a vantaggio però dell'accuratezza dei risultati.

Altra osservazione può essere fatta fra le *MobileNetV2* e *MobileNetV3*, si ha infatti che la *MobileNetV3* dà risultati migliori in termini di accuratezza, a parità della latenza e quindi del tempo impiegato. Con la *MobileNetV3* si ha quindi un aumento dell'accuratezza e della velocità (una riduzione quindi della latenza).

2.1.3 Faster-RCNN Inception V2

Il modello *Faster-RCNN Inception V2* ci permette di ottenere in genere risultati con una maggiore accuratezza rispetto i modelli *MobileNets*, con un aumento però del tempo di esecuzione. Si è scelto quindi tale modello per confrontarlo con le *MobileNets*, così da osservare i valori dell'accuratezza dei risultati ottenuti tenendo in considerazione i diversi tempi di esecuzione richiesti dai modelli.

2.2 Dataset

2.2.1 Dataset COCO

COCO è un *dataset* a larga scala utilizzato per la *object detection*, la segmentazione e la *captioning*. Tutti i modelli utilizzati per lo sviluppo di tale

lavoro di tesi sono pre-allenati sul *dataset* COCO, in particolare sul *dataset* COCO 2014, infatti TensorFlow utilizza tale versione del *dataset* COCO nella configurazione di default.

I valori COCO *mAP (mean Average Precision)* che sono presenti nelle tabelle 2.1 e 2.2, ottenuti dal *Tensorflow detection model zoo*, sono stati calcolati considerando il *dataset* MSCOCO (COCO Minival Split) 14, cioè considerando solo una parte dell'intero *dataset COCO 14*.

2.2.2 Dataset Utilizzato

Per il calcolo delle metriche (Sezione 3.3) e per i test effettuati (Sezione 3.4) è stato utilizzato un *dataset* di immagini acquisite direttamente dalla fotocamera delle Raspberry. In questo modo si possiede un set di immagini "reali", cioè simili a quelle che verranno poi acquisite dal modulo durante il suo utilizzo.

Le immagini del set sono state numerate progressivamente, cosicché vengono identificate, all'interno del *dataset*, dal loro nome.

Le Raspberry utilizzate per l'acquisizione di queste immagini si trovano in posizioni differenti, si potranno quindi effettuare i test su immagini acquisite da posizioni e angolazioni differenti così da rendere ancor più veritieri i risultati ottenuti. Grazie alle immagini di questo *dataset* è stato possibile quindi osservare i risultati in casi d'uso molto vicini a quelli che effettivamente si presenteranno durante l'utilizzo del modulo.

Il *dataset* considerato ha un totale di 41 immagini RGB con dimensioni pari a 1152x768. Queste sono state tutte acquisite dalla *Raspberry Pi Camera Module v2* di diverse Raspberry e rispecchiano le seguenti casistiche che si possono verificare:

- Immagine priva di persone, è il caso ideale, in cui nell'immagine compaiono gli scaffali con i relativi prodotti senza alcuna persona che copre gli stessi;

- Immagine con persone, è il caso svantaggioso, in cui alcuni dei prodotti sono totalmente o parzialmente coperti da uno o più clienti o anche da eventuali addetti del negozio che stanno eseguendo il *refill* degli scaffali. Se ci troviamo in una di queste situazioni l'immagine viene scartata.

2.3 Servizi

2.3.1 OpenCV

OpenCV (Open Source Computer Vision Library) è una libreria software *open source* che opera nell'ambito della *computer vision* e del *machine learning*.

Per lo sviluppo di tale progetto si sono utilizzate due differenti versioni:

- 4.1.2, versione installata di default nei *notebooks* di Google Colab;
- 4.3.0, rilasciata in Aprile 2020 è l'ultima versione della libreria disponibile[2]. Questa versione è stata installata nella Raspberry e nel sistema Ubuntu, cioè gli ulteriori sistemi (oltre ai *notebooks* di Colab) utilizzati per effettuare i test.

L'installazione della versione 4.3.0 della libreria OpenCV è stata necessaria per l'utilizzo dei modelli *MobileNetV3*. Questi infatti hanno problemi di compatibilità con la versione 4.1.2 di OpenCV, essendo modelli di reti recenti.

Nei test effettuati nei *notebooks* di Google Colab si è deciso quindi di non considerare i modelli *MobileNetV3*, in quanto, come già detto, su tali *notebooks* è installata la versione 4.1.2 di OpenCV.

Nei test effettuati sugli altri sistemi (Ubuntu e Raspberry) verranno invece considerati tutti i modelli di reti.

Tale scelta è stata necessaria in quanto non è ancora disponibile un pacchetto Python per la versione 4.3.0 di OpenCV. L'installazione dell'ultima versione è stata possibile in quei due soli sistemi in quanto la si è dovuta compilare da sorgente.

DNN (Deep Neural Networks) Module

Tale modulo è stato centrale nello sviluppo del progetto, fra le varie funzionalità ci offre infatti la possibilità di caricare modelli di reti da *frameworks* differenti, quali TensorFlow e Caffe. Nello sviluppo di tale progetto si è utilizzata la funzionalità di caricamento di un modello pre-allenato utilizzando il *framework* TensorFlow.

Il modulo DNN di OpenCV necessita di due file per il caricamento di un modello di rete dal *framework* TensorFlow:

- ".pb", sta per *Protocol Buffers*, cioè un linguaggio sviluppato da Google indipendente dalla lingua, indipendente dalla piattaforma ed estensibile per la serializzazione di dati strutturati. Questo file conterrà le

[2]Al momento della scrittura della tesi, Luglio 2020.

informazioni del modello a cui fa riferimento. Si trova all'interno dell'archivio ".tar.gz" del modello di rete scaricato dal *Tensorflow detection model zoo*;

- ".pbtxt", un file di configurazione basato sul file ".pb". Tale file può essere importato dalla repository *OpenCV Extra* o generato a partire dal file ".pb" del modello utilizzato (come verrà spiegato nella sezione 3.4).

Una volta ottenuti entrambe i file effettuare la *detection* utilizzando il modulo *DNN (Deep Neural Networks)* di OpenCV è immediato. Basta infatti caricare il modello di rete dal *framework* desiderato, TensorFlow nel caso specifico, e passare in *input* l'immagine da sottoporre alla *detection*. L'immagine passata in *input* viene prima ridimensionata in accordo con le dimensioni delle immagini che prende in *input* il modello che si sta considerando.

2.3.2 AWS (Amazon Web Services)

Amazon Web Services (AWS) è una piattaforma *cloud* che, come vantaggi principali, offre molteplici servizi e possiede una vasta *community*.

Amazon S3

Amazon Simple Storage Service (Amazon S3) è un servizio di *storage* offerto da AWS che, fra le caratteristiche principali, vanta di un'ottima scalabilità, disponibilità dei dati, sicurezza ed elevate prestazioni offerte. I file in Amazon S3 vengono definiti come *objects*, oggetti appunto.

Per effettuare l'*upload* delle immagini è stato dapprima creato un *bucket S3*[3] e in seguito utilizzato il pacchetto Python *Boto3*. Tale pacchetto è il *Software Development Kit* (SDK) di AWS per Python, esso permette di creare, configurare e gestire i servizi offerti da AWS, come EC2 e S3.

Si sono utilizzati quindi i metodi predefiniti di Boto3 per gestire Amazon S3, in particolare per effettuare l'*upload* delle immagini nel *bucket S3*, dopo aver verificato che non vi siano persone presenti.

Bucket S3

Il *bucket S3* non è altro che una sezione dell'intero spazio di archiviazione che si ha a disposizione con Amazon S3.

Lo spazio di archiviazione, reso disponibile dal servizio di *storage* Amazon S3, può essere visto quindi come un insieme di *bucket* S3.

[3]Spiegato successivamente.

Tali *bucket* possono essere considerati come *directory* fra loro indipendenti e che memorizzano informazioni differenti.

Capitolo 3

Progettazione e Sviluppo

In questo capitolo si descriveranno i passi seguiti per lo sviluppo del modulo e l'utilizzo degli strumenti introdotti nel capitolo precedente (Capitolo 2).

3.1 Workflow del progetto

Come primo passo della progettazione si è definito il *workflow* del modulo (Figura 3.1). Si sono specificate cioè le azioni da eseguire nelle differenti situazioni in cui il modulo si trova durante l'esecuzione.

Come già anticipato nell'introduzione (Sezione 1.1) nello sviluppo del modulo si è tenuto conto della sua esecuzione nella *Raspberry Pi 3A+* ed è quindi stato ottimizzato per essa. Ad esempio, considerando l'alimentazione a batteria, si ha l'esigenza di realizzare un modulo efficace ed efficiente nei tempi di esecuzione.

Per i precedenti motivi si è fatta la scelta di terminare l'esecuzione del modulo in caso di situazioni di errore (errori nell'accesso alla fotocamera) o situazioni svantaggiose (presenza di persone nell'immagine acquisita), così da evitare condizioni di attesa e massimizzare la durata della batteria.

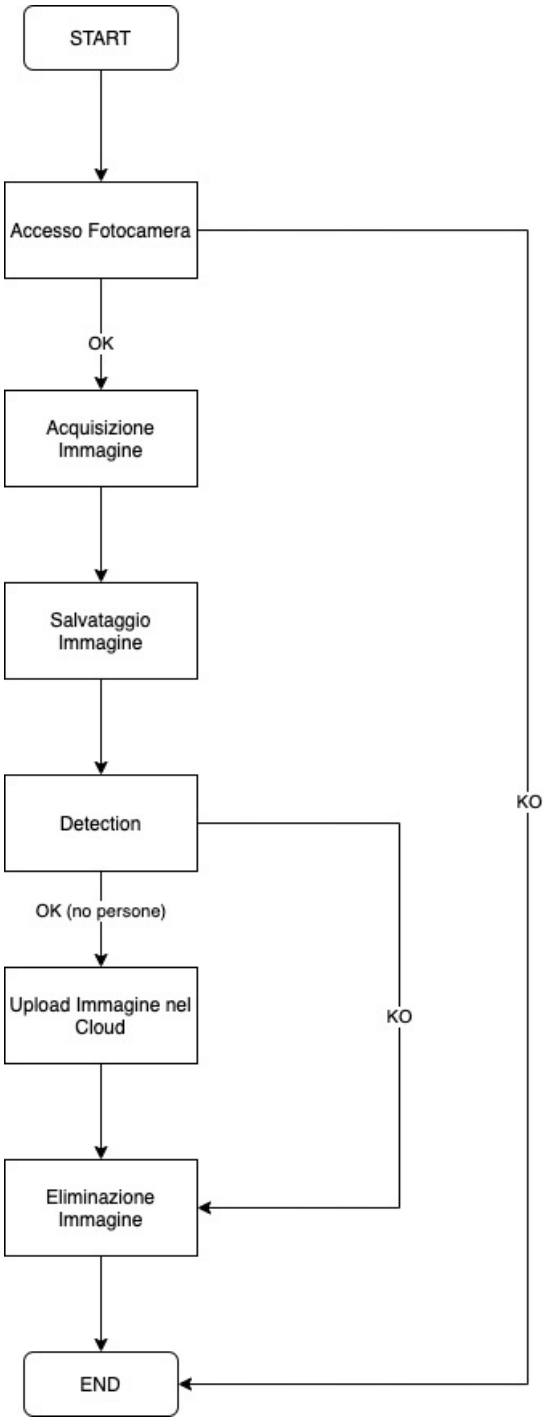

Figura 3.1: Workflow Modulo

3.2 Annotazione immagini

Altro passo preliminare è l'annotazione delle immagini acquisite dalle Raspberry, cioè delle immagini del *dataset* descritto precedentemente (Sezione 2.2.2). L'annotazione delle immagini è necessaria per calcolare le metriche descritte nella sezione successiva (Sezione 3.3).

Per realizzare le annotazioni è stato utilizzato *LabelImg*, un *tool* grafico per annotare le immagini ed etichettare gli oggetti in esse presenti. In questo caso specifico l'unica *label* utilizzata per definire gli oggetti è la *label* "person", ci interessa infatti soltanto la *detection* delle persone eventualmente presenti nelle immagini del *dataset* utilizzato.

Una volta effettuate le annotazioni si ottengono dei file con estensione ".xml", uno per ogni immagine annotata, che riportano i dettagli delle annotazioni tra cui, informazione principale, la posizione degli oggetti annotati.

3.3 Metriche reti

Ottenuti i file ".xml" dalle annotazioni, descritte nella sezione precedente (Sezione 3.2), si sono generati i seguenti file che sono necessari per il calcolo delle metriche dei vari modelli di reti utilizzati:

- *".csv"*, ottenuto dall'esecuzione dello script *generate_csv.py*[1]. Tale script legge i contenuti delle annotazioni delle immagini (i file ".xml") e genera un unico file ".csv" al cui interno vengono riportati i dettagli di ogni oggetto annotato. Fra queste informazioni troviamo: la posizione, la classe di appartenenza (cioè la *label* dell'oggetto), le dimensioni e il nome dell'immagine in cui compare;

- *".pbtxt"*, ottenuto dall'esecuzione dello script *generate_pbtxt.py*[2]. Tale script legge il file ".csv" creato precedentemente e genera la *label map*, cioè una mappa contenente gli ID e i nomi delle classi degli oggetti presenti nelle annotazione. In questo caso specifico conterrà una sola classe ("person") con ID pari ad 1;

- *".record"*, ottenuto dall'esecuzione dello script *generate_tfrecord.py*[3]. Tale script legge i file creati precedentemente (il file ".csv" e la *label map*) e genera un file *TFRecord*.

[1]Lo script utilizzato può essere ottenuto dalla *repository Detection Util Scripts*.
[2]Vedi nota 1.
[3]Vedi nota 1.

I file precedenti possono essere utilizzati anche per l'allenamento di un modello per la *object detection* utilizzando il *framework TensorFlow*. In questo progetto non sono stati ri-allenati modelli di reti in quanto si sono ottenuti dei buoni risultati già con i modelli pre-allenati sul *dataset* COCO. I risultati verranno mostrati nel capito seguente (Capitolo 4).

3.3.1 Alcuni concetti preliminari

Prima di descrivere le metriche calcolate, COCO e Pascal VOC, si definiscono in questa parte alcuni concetti preliminari per la comprensione delle sezioni seguenti.

- *Confidenza*, è la probabilità che un box contenga un particolare oggetto;
- *IoU (Intersection over Union)*, è definito come l'area dell'intersezione divisa l'area dell'unione fra un *bounding box* previsto (B_p) e un *ground-truth box* (B_{gt}):
  $IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$;

Entrambi i parametri definiti precedentemente, confidenza e IoU, vengono utilizzati come criteri per determinare se un rilevamento è *TP (true positive)* o *FP (false positive)*.

- *Precision*, è definito come la divisione fra il numero di *true positives* e la somma di *true positives* e *false positives*:
  $precision = \frac{TP}{TP+FP}$;
- *Recall*, è definito come la divisione fra il numero di *true positives* e la somma di *true positives* e *false negatives*:
  $recall = \frac{TP}{TP+FN}$.

Settando la soglia di confidenza a diversi livelli otteniamo differenti coppie di *precision* e *recall*, rappresentando graficamente tali valori si ottiene la *curva precision-recall* su cui si basa la metrica *AP (Average Precision)* definita in seguito.

Oltre alla curva *precision-recall* si può ottenere la curva *recall-IoU*: impostando la soglia del parametro IoU a diversi livelli si ottengono di conseguenza diversi valori di *recall*. Mappando il parametro IoU nell'asse delle ascisse (per un intervallo che va da 0.5 a 1.0) e i diversi livelli di *recall* nell'asse delle ordinate si ottiene tale curva. La metrica *AR (Average Recall)*, definita di seguito, si basa su questa curva.

- *AP (Average Precision)*, tale metrica è basata sulla *curva precision-recall*, rappresenta il valore medio del parametro *precision* fra tutti i livelli di *recall*[4];

- *mAP (mean Average Precision)*, il calcolo dell'AP prevede un'unica classe ma dato che in un'*object detection* ci sono spesso più classi, la mAP è definita come la media della metrica AP fra le K classi:
  $mAP = \frac{\sum_{i=1}^{K} AP_{\mathrm{i}}}{K}$ dove K>1 è il numero di classi;

- *AR(Average Recall)*, è, come nel caso della AP, una metrica per la valutazione della *performance* della *detection*. Rappresenta il valore medio della *recall* fra tutti i livelli di IoU (che va da 0.5 a 1.0) e può essere calcolata come due volte l'area al di sotto della *curva recall-IoU*;

- *mAR(mean Average Recall)*, è definita come la media della metrica AR fra le K classi:
  $mAR = \frac{\sum_{i=1}^{K} AR_{\mathrm{i}}}{K}$ dove K>1 è il numero di classi.

3.3.2 Metriche PASCAL VOC

Inizialmente si sono calcolate le metriche *Pascal VOC*, queste possono essere considerate come le metriche standard per la valutazione della *performance* di una *object detection*.

Calcolando le metriche Pascal VOC si ottiene la sola metrica *mAP (mean Average Precision)*. Si considera cioè una singola soglia del parametro *IoU (Intersection over Union)*, pari a 0.5.

3.3.3 Metriche COCO

In seguito al calcolo delle metriche *Pascal VOC* si sono calcolate le metriche *COCO*, queste possono essere viste come una variante delle prime. Calcolando le metriche *COCO* si ottengono, al contrario di quanto avveniva con le metriche *Pascal VOC*, diversi valori della metrica *mAP (mean Average Precision)* ed inoltre si ottiene la metrica *mAR (mean Average Recall)*.

[4]Prima di calcolare la metrica AP si effettua l'interpolazione dei valori di *precision* a livelli di *recall* multipli, così da ridurre le oscillazioni della curva.

Una prima suddivisione fra i valori della metrica *mAP (mean Average Precision)* si ha considerando soglie differenti del parametro *IoU (Intersection over Union)*, queste sono:

- $mAP^{IoU=.50:.05:.95}$, rappresenta il valore medio della metrica *mAP* per oltre dieci soglie del parametro *IoU*;
- $mAP^{IoU=.50}$, corrisponde alla metrica *mAP* calcolata con le metriche *Pascal VOC*, quindi per un valore del parametro *IoU* pari a 0.5;
- $mAP^{IoU=.75}$, rappresenta una metrica più stringente, per un valore del parametro *IoU* pari a 0.75.

Oltre alla suddivisione per diversi valori della soglia del parametro *IoU*, c'è un'ulteriore suddivisione della metrica *mAP (mean Average Precision)* che considera diverse scale di oggetti[5]:

- mAP^{small}, rappresenta il valore della metrica *mAP* per gli oggetti di piccole dimensioni, cioè che coprono un'area minore di 32px[6];
- mAP^{medium}, rappresenta il valore della metrica *mAP* per gli oggetti di medie dimensioni, cioè che coprono un'area compresa fra 32px e 96px[7];
- mAP^{large}, rappresenta il valore della metrica *mAP* per gli oggetti di grandi dimensioni, cioè che coprono un'area maggiore di 96px[8].

Le metriche COCO ci forniscono, oltre alla *mAP (mean Average Precision)*, un'ulteriore metrica, la *mAR (mean Average Recall)*. Anch'essa viene utilizzata per la valutazione della *object detection*.

Come avveniva per la *mAP (mean Average Precision)* anche per la *mAR (mean Average Recall)* abbiamo diversi valori. Una prima suddivisione si ha considerando il numero di oggetti individuati per ogni immagine:

- $mAR^{max=1}$, corrisponde al valore della metrica *mAR* per un numero di oggetti individuati pari a 1;
- $mAR^{max=10}$, corrisponde al valore della metrica *mAR* per un numero di oggetti individuati fino a 10;

[5]Tutti i seguenti valori della *mAP (mean Average Precision)* rappresentano i valori medi di questa metrica per oltre dieci soglie del parametro *IoU*.

[6]L'area è misurata come numero di pixel nella *segmentation mask*.

[7]Vedi nota 6.

[8]Vedi nota 6.

- $mAR^{max=100}$, corrisponde al valore della metrica mAR per un numero di oggetti individuati fino a 100.

Un'ulteriore suddivisione avviene considerando, come per la metrica mAP, le diverse scale di oggetti individuati:

- mAR^{small}, rappresenta il valore della metrica mAR per gli oggetti di piccole dimensioni, cioè che coprono un'area minore di 32px[9];
- mAR^{medium}, rappresenta il valore della metrica mAR per gli oggetti di medie dimensioni, cioè che coprono un'area compresa fra 32px e 96px[10];
- mAR^{large}, rappresenta il valore della metrica mAR per gli oggetti di grandi dimensioni, cioè che coprono un'area maggiore di 96px[11].

3.3.4 Metriche ottenute

Descritti i dettagli delle metriche COCO e Pascal VOC si riportano ora i valori ottenuti considerando il *dataset* di immagini acquisite dalle Raspberry ed i file che sono stati descritti nella sezione 3.3.

Nelle tabelle 3.1 e 3.2 si riportano i valori delle *metriche COCO (mAP e mAR)* ottenute dai vari modelli di reti utilizzati.

Tabella 3.1: Metriche COCO - mAP

Modello	$\mathbf{mAP^{IoU=.50:.05:.95}}$	$\mathbf{mAP^{IoU=.50}}$	$\mathbf{mAP^{IoU=.75}}$	$\mathbf{mAP^{large}}$
SSD MobileNet v1	0.670	0.927	0.891	0.670
SSD MobileNet v2	0.709	0.945	0.906	0.714
SSD MobileNet v3 Small	0.658	0.911	0.793	0.661
SSD MobileNet v3 Large	0.664	0.949	0.819	0.664
Faster-RCNN with Inception v2	0.697	1.000	0.750	0.697

Tabella 3.2: Metriche COCO - mAR

Modello	$\mathbf{mAR^{max=1}}$	$\mathbf{mAR^{max=10}}$	$\mathbf{mAR^{max=100}}$	$\mathbf{mAR^{large}}$
SSD MobileNet v1	0.609	0.721	0.721	0.721
SSD MobileNet v2	0.636	0.770	0.770	0.770
SSD MobileNet v3 Small	0.621	0.752	0.752	0.752
SSD MobileNet v3 Large	0.615	0.755	0.755	0.755
Faster-RCNN with Inception v2	0.618	0.755	0.755	0.755

[9]L'area è misurata come numero di pixel nella *segmentation mask*.
[10]Vedi nota 9.
[11]Vedi nota 9.

Si osservi che le metriche mAP e mAR per le scale di oggetti *small* e *medium* non sono definite nelle tabelle precedenti. Questo perché nelle immagini del *dataset* sottoposte alle *detections* sono state annotate le sole persone eventualmente presenti, che ricoprono quindi un'area tale da essere associate alla scala di oggetti *large*.

Le metriche $mAR^{max=10}$ e $mAR^{max=100}$ hanno i medesimi valori in quanto nelle immagini, del *dataset* utilizzato, sono presenti al massimo tre persone contemporaneamente e quindi non ci sono immagini per cui il numero di persone presenti è tale da farle rientrare esclusivamente nel calcolo della metrica $mAR^{max=100}$.

Altra osservazione può essere fatta considerando i valori delle metriche $mAR^{max=1}$ e $mAR^{max=10}$. Per il calcolo della seconda metrica si considerano immagini con un numero massimo di 10 persone, tale insieme racchiude perciò anche le immagini contenenti una sola persona e quindi utilizzate per il calcolo della prima metrica. Considerando ciò e osservando che i valori della $mAR^{max=1}$ sono minori (quindi peggiori) rispetto ai valori della $mAR^{max=10}$, possiamo concludere che si ottengono valori migliori della metrica mAR considerando immagini con più di una persona presente in esse.

Come si osserva dalle tabelle precedenti (Tab. 3.1 e 3.2) i valori ottenuti delle metriche mAP e mAR sono alti, tali da non rendere necessario il riallenamento dei modelli di reti che si utilizzano.

Nella tabella seguente (Tabella 3.3) si riportano i valori delle *metriche Pascal VOC* ottenute dai vari modelli di reti utilizzati.

Tabella 3.3: Metriche Pascal VOC - mAP

Modello	**mAP**
SSD MobileNet v1	0.936
SSD MobileNet v2	0.954
SSD MobileNet v3 Small	0.919
SSD MobileNet v3 Large	0.953
Faster-RCNN with Inception v2	1.000

Le metriche *mAP Pascal VOC* calcolate equivalgono alle metriche COCO $mAP^{IoU=.50}$, cioè ottenute per un valore del parametro *IoU (Intersection over Union)* pari a 0.5. Confrontando infatti le tabelle corrispondenti (Tab. 3.1 e 3.3) si osserva che i valori di tali metriche sono pressoché uguali (trascurando piccole variazioni).

In questo caso specifico i valori delle metriche *AP (Average Precision)* e *AR (Average Recall)* corrispondono ai valori delle rispettive medie *mAP*

(mean Average Precision) e *mAR (mean Average Recall)*. Questo perché, come si osserva dalla definizione di tali metriche (Sezione 3.3.1), le mAP e mAR sono le medie fra tutte le classi degli oggetti identificati ma nel nostro caso specifico questo numero di classi è pari ad 1 (la sola classe "person").

3.4 Test Eseguiti

Dopo aver calcolato le metriche (COCO e Pascal VOC), dei modelli di reti utilizzati, si sono eseguiti dei test per verificare come questi si comportano con le immagini acquisite dalle Raspberry.

Prima di eseguire i test è stato necessario generare il file ".pbtxt" a partire dal file ".pb" per ogni modello di rete utilizzato. Come già anticipato (Sezione 2.3.1), infatti, il modulo *DNN (Deep Neural Networks)* di OpenCV necessita di questo file di configurazione aggiuntivo per il caricamento di un modello dal *framework* TensorFlow.

I file ".pbtxt" sono stati generati utilizzando i seguenti *scripts*:

- *"tf_text_graph_ssd.py"*[12], script utilizzato per generare il file di configurazione per i modelli *MobileNets*;
- *"tf_text_graph_faster_rcnn.py"*[13], script utilizzato per generare il file di configurazione per il modello *Faster-RCNN Inception v2*.

Una volta generati questi file è stato possibile eseguire i test utilizzando i diversi modelli di reti. I test sono stati eseguiti su tre sistemi differenti, i cui dettagli vengono riportati di seguito:

- *Notebooks di Google Colab*, troviamo pre-installate le versioni 4.1.2 della libreria OpenCV e 2.2.0 del *framework* TensorFlow. Si sono esclusi dai test realizzati in questi *notebooks* i modelli MobileNetV3 (Small e Large) in quanto non sono compatibili con la versione 4.1.2 di OpenCV. Per l'esecuzione dei test è stata utilizzata l'accelerazione *hardware GPU* offerta dai *notebooks* di Google Colab;
- *Ubuntu*, in seguito ai test sui *notebooks* di Colab si sono effettuati gli stessi su un sistema Ubuntu 18.04. In tale sistema si sono installate le versioni 4.3.0 di OpenCV (ovvero l'ultima disponibile[14]) e 1.15.0 del *framework* TensorFlow. Si sono eseguiti i test con tutti i modelli di

[12]Script presente nella *repository* della libreria OpenCV.
[13]Vedi nota 12.
[14]Al momento della scrittura della tesi, Luglio 2020.

reti, compresi i modelli MobileNetV3, disponendo dell'ultima versione della libreria OpenCV;

- *Raspberry*, come ultimo passo si sono effettuati i test direttamente sulla Raspberry (*Raspberry Pi 3A+*) così da osservare i risultati ottenuti nel sistema in cui andrà realmente in esecuzione il modulo. Anche nella Raspberry si è installata l'ultima versione della libreria OpenCV (4.3.0).

I test sono stati eseguiti in due diverse configurazioni, così da osservare eventuali cambiamenti nei risultati dei diversi tipi di esecuzioni:

- Configurazione 1; per il primo tipo di test sono state utilizzate le immagini acquisite dalle Raspberry senza apportare ad esse alcuna modifica (ad eccezione del ridimensionamento dell'immagine prima di avviare la *detection*);

- Configurazione 2; il secondo tipo di test è stato ottenuto applicando, alle stesse immagini sottoposte ai test in configurazione 1, un *flip* orizzontale, cioè riflettendo l'immagine orizzontalmente. Si è deciso di effettuare i test in questa seconda configurazione così da osservare se i risultati cambiano ed eventualmente come (migliorando o peggiorando).

Nei test eseguiti in configurazione 2 non si considerano i tempi computazionali in quanto, questi, dipendono in maniera considerevole dal modello utilizzato e dal sistema in cui questo va in esecuzione. Si è deciso quindi di osservare soltanto eventuali variazioni dei valori della confidenza ottenuti dalle varie *detections*.

Per i test effettuati, in entrambe le configurazioni, è stata definita una soglia alla confidenza dei risultati ottenuti pari allo 0.45. Eventuali persone con una confidenza maggiore del 45% verranno quindi considerate *TP (true positive)*, le altre *FP (false positive)*.

Capitolo 4

Risultati e discussioni

In questo capitolo vengono presentati i risultati ottenuti dai test effettuati nei diversi sistemi (*notebooks* Colab, Ubuntu e Raspberry). Si potranno quindi osservare i comportamenti dei diversi modelli di reti e come questi cambiano nei diversi sistemi utilizzati per l'esecuzione dei test.

Si analizzeranno dapprima i risultati ottenuti dalle *detections* nella configurazione 1, cioè senza apportare alcuna modifica alle immagini del *dataset*. In seguito si osserveranno i risultati ottenuti dai test effettuati in configurazione 2. In questa seconda configurazione si riflettono orizzontalmente le immagini da sottoporre alle *detections*, così da osservare eventuali cambiamenti ai valori di confidenza ottenuti.

Si riassumeranno poi i valori ottenuti dalle *detections* eseguite, in entrambe le configurazioni, soffermando l'attenzione sul numero totale di persone disponibili nelle immagini e il numero totale di persone identificate dai diversi modelli, calcolando poi il rapporto fra questi due valori. Grazie a queste due tabelle riassuntive (Tab. 4.3 e 4.4) si potranno confrontare facilmente i risultati ottenuti dalle due differenti configurazioni. In particolare si potrà confrontare il numero di persone identificate da ogni modello e di conseguenza il numero di *FN (false negatives)* che si sono presentati nelle diverse *detections*.

Nell'ultima sezione (Sezione 4.4) si riporteranno i tempi computazionali. Si confronteranno quindi i tempi ottenuti dai diversi modelli di reti utilizzati e si osserverà come questi variano al variare del sistema in cui i modelli vengono eseguiti.

Dalle *detections* eseguite nei diversi sistemi si sono ottenuti valori conformi di confidenza. Tale parametro (confidenza) infatti dipende esclusivamente dal modello di rete utilizzato e non dal sistema nel quale il modello viene utilizzato.

Per agevolare la lettura delle tabelle 4.1 e 4.2 sono stati eseguiti i test soltanto su 30 immagini delle 41 disponibili nel *dataset* utilizzato.

4.1 Risultati Detections in Configurazione 1

Nella tabella 4.1 vengono riportati i risultati ottenuti dai test effettuati in configurazione 1[1]. Ogni immagine del *dataset* viene identificata dal nome (ad es. immagine1, immagine2, ...), tali nomi corrisponderanno a quelli presenti nelle tabelle delle sezioni seguenti (Sez. 4.2 e 4.4).

Nella tabella 4.1 troviamo le seguenti informazioni:

- Nome immagine;
- Nome modello;
- Numero di persone, indica il numero di persone realmente presente nell'immagine sottoposta alla *detection*;
- Confidenza, persona1, persona2, persona3, indica la confidenza per ognuna delle persone identificate nelle immagini (sono presenti soltanto tre colonne in quanto dalle immagini sottoposte alla *detection* vengono al massimo identificate tre persone contemporaneamente).

Il numero di persone e il numero di percentuali di confidenza potrebbero non coincidere, in questo caso siamo in presenza di *FP (false positives)* o *FN (false negatives)*. Dalla tabella seguente quindi si può osservare facilmente se si è in presenza di una delle due situazioni (FP o FN).

[1]Per rispetto della privacy delle persone presenti nelle immagini non si riportano gli screenshot delle *detections* effettuate ma soltanto i valori ottenuti.

Tabella 4.1: Risultati Configurazione 1

Immagine	**Modello**	**NR. Persone**	**Persona1**	**Persona2**	**Persona3**
immagine1	SSD MobileNet v1	1	94.96%	–	–
immagine1	SSD MobileNet v2	1	96.74%	55.20%	–
immagine1	SSD MobileNet v3 Small	1	70.81%	45.61%	–
immagine1	SSD MobileNet v3 Large	1	70.61%	–	–
immagine1	Faster-RCNN Inception v2	1	**98.72%**	–	–
immagine2	SSD MobileNet v1	1	85.88%	–	–
immagine2	SSD MobileNet v2	1	**99.08%**	–	–
immagine2	SSD MobileNet v3 Small	1	56.67%	–	–
immagine2	SSD MobileNet v3 Large	1	48.38%	–	–
immagine2	Faster-RCNN Inception v2	1	99.00%	–	–
immagine3	SSD MobileNet v1	0	–	–	–
immagine3	SSD MobileNet v2	0	–	–	–
immagine3	SSD MobileNet v3 Small	0	–	–	–
immagine3	SSD MobileNet v3 Large	0	–	–	–
immagine3	Faster-RCNN Inception v2	0	–	–	–
immagine4	SSD MobileNet v1	0	–	–	–
immagine4	SSD MobileNet v2	0	–	–	–
immagine4	SSD MobileNet v3 Small	0	–	–	–
immagine4	SSD MobileNet v3 Large	0	–	–	–
immagine4	Faster-RCNN Inception v2	0	–	–	–
immagine5	SSD MobileNet v1	1	94.45%	–	–
immagine5	SSD MobileNet v2	1	**98.44%**	–	–
immagine5	SSD MobileNet v3 Small	1	47.09%	–	–
immagine5	SSD MobileNet v3 Large	1	52.60%	–	–
immagine5	Faster-RCNN Inception v2	1	90.49%	–	–
immagine6	SSD MobileNet v1	1	55.49%	–	–
immagine6	SSD MobileNet v2	1	**91.43%**	–	–
immagine6	SSD MobileNet v3 Small	1	74.04%	–	–
immagine6	SSD MobileNet v3 Large	1	–	–	–
immagine6	Faster-RCNN Inception v2	1	78.32%	–	–
immagine7	SSD MobileNet v1	1	91.76%	–	–
immagine7	SSD MobileNet v2	1	**94.69%**	–	–
immagine7	SSD MobileNet v3 Small	1	–	–	–
immagine7	SSD MobileNet v3 Large	1	53.00%	–	–
immagine7	Faster-RCNN Inception v2	1	87.01%	–	–
immagine8	SSD MobileNet v1	1	95.87%	–	–
immagine8	SSD MobileNet v2	1	94.21%	–	–
immagine8	SSD MobileNet v3 Small	1	60.34%	–	–
immagine8	SSD MobileNet v3 Large	1	70.99%	–	–
immagine8	Faster-RCNN Inception v2	1	**98.56%**	–	–
immagine9	SSD MobileNet v1	1	80.26%	–	–
immagine9	SSD MobileNet v2	1	80.39%	–	–
immagine9	SSD MobileNet v3 Small	1	64.52%	–	–
immagine9	SSD MobileNet v3 Large	1	61.10%	–	–
immagine9	Faster-RCNN Inception v2	1	**98.47%**	–	–
immagine10	SSD MobileNet v1	1	95.23%	–	–
immagine10	SSD MobileNet v2	1	90.81%	–	–
immagine10	SSD MobileNet v3 Small	1	64.36%	–	–
immagine10	SSD MobileNet v3 Large	1	70.28%	–	–
immagine10	Faster-RCNN Inception v2	1	**97.45%**	–	–

Continua nella prossima pagina

Tabella 4.1 – continuo della tabella precedente

Immagine	Modello	NR. Persone	Persona1	Persona2	Persona3
immagine11	SSD MobileNet v1	1	86.74%	–	–
immagine11	SSD MobileNet v2	1	89.01%	–	–
immagine11	SSD MobileNet v3 Small	1	72.94%	–	–
immagine11	SSD MobileNet v3 Large	1	72.96%	–	–
immagine11	Faster-RCNN Inception v2	1	**98.81%**	–	–
immagine12	SSD MobileNet v1	1	89.52%	–	–
immagine12	SSD MobileNet v2	1	89.88%	–	–
immagine12	SSD MobileNet v3 Small	1	73.20%	–	–
immagine12	SSD MobileNet v3 Large	1	65.61%	–	–
immagine12	Faster-RCNN Inception v2	1	**98.79%**	–	–
immagine13	SSD MobileNet v1	3	88.98%	83.92%	–
immagine13	SSD MobileNet v2	3	94.39%	93.06%	59.19%
immagine13	SSD MobileNet v3 Small	3	71.17%	59.52%	54.57%
immagine13	SSD MobileNet v3 Large	3	81.55%	70.92%	61.72%
immagine13	Faster-RCNN Inception v2	3	**98.07%**	**94.83%**	**90.56%**
immagine14	SSD MobileNet v1	0	–	–	–
immagine14	SSD MobileNet v2	0	–	–	–
immagine14	SSD MobileNet v3 Small	0	–	–	–
immagine14	SSD MobileNet v3 Large	0	–	–	–
immagine14	Faster-RCNN Inception v2	0	–	–	–
immagine21	SSD MobileNet v1	1	80.39%	–	–
immagine21	SSD MobileNet v2	1	92.53%	–	–
immagine21	SSD MobileNet v3 Small	1	–	–	–
immagine21	SSD MobileNet v3 Large	1	62.72%	–	–
immagine21	Faster-RCNN Inception v2	1	**96.66%**	–	–
immagine22	SSD MobileNet v1	1	48.03%	–	–
immagine22	SSD MobileNet v2	1	68.68%	–	–
immagine22	SSD MobileNet v3 Small	1	–	–	–
immagine22	SSD MobileNet v3 Large	1	57.49%	–	–
immagine22	Faster-RCNN Inception v2	1	**96.60%**	–	–
immagine23	SSD MobileNet v1	1	–	–	–
immagine23	SSD MobileNet v2	1	47.13%	–	–
immagine23	SSD MobileNet v3 Small	1	61.83%	–	–
immagine23	SSD MobileNet v3 Large	1	57.95%	50.76%	–
immagine23	Faster-RCNN Inception v2	1	**91.37%**	–	–
immagine24	SSD MobileNet v1	0	–	–	–
immagine24	SSD MobileNet v2	0	–	–	–
immagine24	SSD MobileNet v3 Small	0	–	–	–
immagine24	SSD MobileNet v3 Large	0	49.60%	–	–
immagine24	Faster-RCNN Inception v2	0	–	–	–
immagine25	SSD MobileNet v1	1	96.39%	–	–
immagine25	SSD MobileNet v2	1	**99.40%**	–	–
immagine25	SSD MobileNet v3 Small	1	64.47%	52.35%	–
immagine25	SSD MobileNet v3 Large	1	79.91%	53.89%	–
immagine25	Faster-RCNN Inception v2	1	98.70%	–	–
immagine27	SSD MobileNet v1	1	93.26%	–	–
immagine27	SSD MobileNet v2	1	93.68%	–	–
immagine27	SSD MobileNet v3 Small	1	68.50%	48.70%	–
immagine27	SSD MobileNet v3 Large	1	75.87%	–	–
immagine27	Faster-RCNN Inception v2	1	**95.73%**	–	–

Continua nella prossima pagina

Tabella 4.1 – continuo della tabella precedente

Immagine	Modello	NR. Persone	Persona1	Persona2	Persona3
immagine28	SSD MobileNet v1	2	89.12%	84.44%	–
immagine28	SSD MobileNet v2	2	96.94%	93.55%	–
immagine28	SSD MobileNet v3 Small	2	71.06%	70.96%	47.83%
immagine28	SSD MobileNet v3 Large	2	67.95%	64.86%	48.93%
immagine28	Faster-RCNN Inception v2	2	**98.44%**	**97.64%**	–
immagine29	SSD MobileNet v1	2	91.51%	86.57%	–
immagine29	SSD MobileNet v2	2	97.94%	90.17%	–
immagine29	SSD MobileNet v3 Small	2	72.78%	63.13%	53.27%
immagine29	SSD MobileNet v3 Large	2	79.41%	78.79%	–
immagine29	Faster-RCNN Inception v2	2	**98.57%**	**97.96%**	–
immagine31	SSD MobileNet v1	1	83.08%	–	–
immagine31	SSD MobileNet v2	1	76.68%	–	–
immagine31	SSD MobileNet v3 Small	1	–	–	–
immagine31	SSD MobileNet v3 Large	1	65.11%	46.01%	–
immagine31	Faster-RCNN Inception v2	1	**98.36%**	–	–
immagine32	SSD MobileNet v1	1	79.43%	–	–
immagine32	SSD MobileNet v2	1	85.74%	47.11%	45.24%
immagine32	SSD MobileNet v3 Small	1	69.50%	–	–
immagine32	SSD MobileNet v3 Large	1	75.97%	–	–
immagine32	Faster-RCNN Inception v2	1	**98.74%**	–	–
immagine34	SSD MobileNet v1	2	94.37%	51.84%	–
immagine34	SSD MobileNet v2	2	96.68%	**76.52%**	–
immagine34	SSD MobileNet v3 Small	2	66.67%	–	–
immagine34	SSD MobileNet v3 Large	2	68.56%	51.02%	–
immagine34	Faster-RCNN Inception v2	2	**98.76%**	–	–
immagine35	SSD MobileNet v1	1	**90.11%**	–	–
immagine35	SSD MobileNet v2	1	61.13%	46.53%	–
immagine35	SSD MobileNet v3 Small	1	52.95%	–	–
immagine35	SSD MobileNet v3 Large	1	60.19%	–	–
immagine35	Faster-RCNN Inception v2	1	85.44%	–	–
immagine36	SSD MobileNet v1	1	94.10%	–	–
immagine36	SSD MobileNet v2	1	88.43%	–	–
immagine36	SSD MobileNet v3 Small	1	64.80%	–	–
immagine36	SSD MobileNet v3 Large	1	65.05%	–	–
immagine36	Faster-RCNN Inception v2	1	**97.69%**	–	–
immagine37	SSD MobileNet v1	2	66.71%	–	–
immagine37	SSD MobileNet v2	2	84.41%	–	–
immagine37	SSD MobileNet v3 Small	2	–	–	–
immagine37	SSD MobileNet v3 Large	2	–	–	–
immagine37	Faster-RCNN Inception v2	2	**97.99%**	–	–
immagine38	SSD MobileNet v1	2	66.22%	53.71%	–
immagine38	SSD MobileNet v2	2	75.52%	74.20%	–
immagine38	SSD MobileNet v3 Small	2	65.84%	58.38%	–
immagine38	SSD MobileNet v3 Large	2	60.34%	–	–
immagine38	Faster-RCNN Inception v2	2	**97.44%**	**96.13%**	–
immagine39	SSD MobileNet v1	1	76.82%	–	–
immagine39	SSD MobileNet v2	1	80.04%	48.05%	45.47%
immagine39	SSD MobileNet v3 Small	1	–	–	–
immagine39	SSD MobileNet v3 Large	1	54.03%	47.49%	–
immagine39	Faster-RCNN Inception v2	1	**96.03%**	–	–

Dalla tabella precedente possono essere fatte le seguenti osservazioni in merito ai risultati ottenuti dalle *detections* in questa prima configurazione:

- *FP (false positives)*, si osserva che i modelli *SSD MobileNet v1* e *Faster-RCNN Inception v2* sono quelli che danno, generalmente, risultati migliori in termini di FP (cioè un numero minore di FP), rispetto gli altri modelli che generano un numero di FP maggiore;

- *FN (false negatives)*, si sono ottenuti risultati che mostrano come, generalmente, il modello *SSD MobileNet v3 Small* è propenso a generare un numero maggiore di FN. Gli altri modelli generano un numero di FN contenuto, in particolare i modelli *SSD MobileNet v2* e *Faster-RCNN Inception v2*;

- *Confidenza*, in termini di valori ottenuti abbiamo che il modello *Faster-RCNN Inception v2* ci fornisce, generalmente, risultati con valori maggiori di confidenza. Si osservi come anche i modelli *SSD MobileNet v1* e *SSD MobileNet v2* forniscono dei risultati con valori di confidenza elevati. Risultati con valori minori di confidenza si ottengono, generalmente, dal modello *SSD MobileNet v3 Small* e *SSD MobileNet v3 Large*;

- *Persone identificate*, il numero di persone identificate è, ragionevolmente, più alto in quei modelli che offrono dei risultati con numeri minori di *FN (false negatives)*. Abbiamo quindi che i modelli *SSD MobileNet v2* e *Faster-RCNN Inception v2* riescono ad identificare, fra le persone disponibili, un numero maggiore di soggetti.

Pur avendo delle percentuali di confidenza differenti fra i vari modelli si sono ottenuti comunque valori molto elevati. Considerando che si sono utilizzati modelli di reti senza averli ri-allenati possiamo ritenere i risultati molto soddisfacenti.

Si può inoltre osservare che i casi di *FP (false positives)* si sono verificati più spesso in immagini in cui c'è almeno una persona realmente presente. Questa situazione non crea particolari problemi per il modulo in questione in quanto si deve scartare l'immagine se vi è almeno una persona, indipendentemente dal numero di individui presenti.

Si avrebbero invece dei problemi nel caso in cui si verifichino *FP (false positives)* in immagini in cui in realtà non c'è alcuna persona, in questo caso infatti si scarterebbe una foto che in realtà è priva di persone. Come si può vedere dalla tabella, per le immagini sottoposte alle *detections*, tale situazione si verifica una sola volta (immagine24) e per un solo modello (*SSD*

MobileNet v3 Large), possiamo quindi concludere che, questo caso, si verifica molto raramente.

Un discorso analogo vale per i *FN (false negatives)*, la situazione peggiore che si può verificare è quella in cui nessuna delle persone presenti nell'immagine venga individuata (in particolar modo per le immagini in cui c'è esattamente una sola persona questo è più probabile) ma, come si vede dalla tabella precedente, anche tali casi si verificano di rado e con un numero limitato di modelli.

4.2 Risultati Detections in Configurazione 2

In questa sezione vengono riportati i risultati ottenuti dai test effettuati nella seconda configurazione e le osservazioni che da questi possono essere compiute.

I test si sono ripetuti applicando alle immagini, del *dataset* utilizzato, il *flip orizzontale* (cioè riflettendole orizzontalmente), così da osservare se i risultati cambiano.

Si riporta nella tabella seguente (Tabella 4.2) i valori della confidenza ottenuti, così da poterli confrontare con i valori ottenuti dalle *detections* effettuate precedentemente.

Per i test eseguiti in questa seconda configurazione non verranno riportati i tempi computazionali nella sezione 4.4 in quanto, come detto precedentemente, è di interesse osservare soltanto eventuali variazioni dei valori di confidenza.

Nella tabella seguente vengono riportate le stesse informazioni che erano contenute nella tabella per i test eseguiti in configurazione 1, ovvero:

- Nome immagine;
- Nome modello;
- Numero di persone;
- Confidenza, persona1, persona2, persona3.

Tabella 4.2: Risultati Configurazione 2

Immagine	**Modello**	**NR. Persone**	**Persona1**	**Persona2**	**Persona3**
immagine1	SSD MobileNet v1	1	96.02%	–	–
immagine1	SSD MobileNet v2	1	96.56%	47.52%	–
immagine1	SSD MobileNet v3 Small	1	68.11%	48.44%	48.42%
immagine1	SSD MobileNet v3 Large	1	69.50%	–	–
immagine1	Faster-RCNN Inception v2	1	**98.21%**	–	–
immagine2	SSD MobileNet v1	1	89.58%	–	–
immagine2	SSD MobileNet v2	1	98.38%	–	–
immagine2	SSD MobileNet v3 Small	1	54.29%	–	–
immagine2	SSD MobileNet v3 Large	1	51.83%	–	–
immagine2	Faster-RCNN Inception v2	1	**99.08%**	–	–
immagine3	SSD MobileNet v1	0	–	–	–
immagine3	SSD MobileNet v2	0	–	–	–
immagine3	SSD MobileNet v3 Small	0	–	–	–
immagine3	SSD MobileNet v3 Large	0	–	–	–
immagine3	Faster-RCNN Inception v2	0	–	–	–
immagine4	SSD MobileNet v1	0	–	–	–
immagine4	SSD MobileNet v2	0	–	–	–
immagine4	SSD MobileNet v3 Small	0	–	–	–
immagine4	SSD MobileNet v3 Large	0	–	–	–
immagine4	Faster-RCNN Inception v2	0	–	–	–
immagine5	SSD MobileNet v1	1	91.27%	–	–
immagine5	SSD MobileNet v2	1	96.16%	–	–
immagine5	SSD MobileNet v3 Small	1	57.47%	–	–
immagine5	SSD MobileNet v3 Large	1	59.21%	46.13%	–
immagine5	Faster-RCNN Inception v2	1	**98.17%**	–	–
immagine6	SSD MobileNet v1	1	88.56%	–	–
immagine6	SSD MobileNet v2	1	**94.78%**	–	–
immagine6	SSD MobileNet v3 Small	1	74.96%	–	–
immagine6	SSD MobileNet v3 Large	1	64.36%	–	–
immagine6	Faster-RCNN Inception v2	1	92.44%	–	–
immagine7	SSD MobileNet v1	1	94.56%	–	–
immagine7	SSD MobileNet v2	1	**95.93%**	–	–
immagine7	SSD MobileNet v3 Small	1	52.35%	–	–
immagine7	SSD MobileNet v3 Large	1	52.40%	–	–
immagine7	Faster-RCNN Inception v2	1	79.06%	–	–
immagine8	SSD MobileNet v1	1	92.42%	–	–
immagine8	SSD MobileNet v2	1	90.74%	–	–
immagine8	SSD MobileNet v3 Small	1	72.06%	–	–
immagine8	SSD MobileNet v3 Large	1	74.23%	–	–
immagine8	Faster-RCNN Inception v2	1	**98.84%**	–	–
immagine9	SSD MobileNet v1	1	75.55%	–	–
immagine9	SSD MobileNet v2	1	85.52%	–	–
immagine9	SSD MobileNet v3 Small	1	62.83%	–	–
immagine9	SSD MobileNet v3 Large	1	61.90%	–	–
immagine9	Faster-RCNN Inception v2	1	**99.23%**	–	–
immagine10	SSD MobileNet v1	1	95.43%	–	–
immagine10	SSD MobileNet v2	1	92.03%	–	–
immagine10	SSD MobileNet v3 Small	1	64.53%	–	–
immagine10	SSD MobileNet v3 Large	1	67.95%	–	–
immagine10	Faster-RCNN Inception v2	1	**98.76%**	–	–

Continua nella prossima pagina

Tabella 4.2 – continuo della tabella precedente

Immagine	Modello	NR. Persone	Persona1	Persona2	Persona3
immagine11	SSD MobileNet v1	1	88.94%	–	–
immagine11	SSD MobileNet v2	1	80.46%	–	–
immagine11	SSD MobileNet v3 Small	1	77.27%	–	–
immagine11	SSD MobileNet v3 Large	1	68.74%	–	–
immagine11	Faster-RCNN Inception v2	1	**98.78%**	–	–
immagine12	SSD MobileNet v1	1	88.11%	–	–
immagine12	SSD MobileNet v2	1	86.86%	–	–
immagine12	SSD MobileNet v3 Small	1	72.47%	–	–
immagine12	SSD MobileNet v3 Large	1	66.63%	–	–
immagine12	Faster-RCNN Inception v2	1	**97.38%**	–	–
immagine13	SSD MobileNet v1	3	91.74%	90.32%	45.18%
immagine13	SSD MobileNet v2	3	93.28%	90.02%	–
immagine13	SSD MobileNet v3 Small	3	72.18%	69.91%	50.94%
immagine13	SSD MobileNet v3 Large	3	77.88%	70.87%	59.70%
immagine13	Faster-RCNN Inception v2	3	**98.40%**	**98.11%**	**84.59%**
immagine14	SSD MobileNet v1	0	–	–	–
immagine14	SSD MobileNet v2	0	–	–	–
immagine14	SSD MobileNet v3 Small	0	–	–	–
immagine14	SSD MobileNet v3 Large	0	–	–	–
immagine14	Faster-RCNN Inception v2	0	–	–	–
immagine21	SSD MobileNet v1	1	78.14%	–	–
immagine21	SSD MobileNet v2	1	96.59%	–	–
immagine21	SSD MobileNet v3 Small	1	–	–	–
immagine21	SSD MobileNet v3 Large	1	71.35%	–	–
immagine21	Faster-RCNN Inception v2	1	**97.70%**	–	–
immagine22	SSD MobileNet v1	1	59.73%	–	–
immagine22	SSD MobileNet v2	1	59.99%	–	–
immagine22	SSD MobileNet v3 Small	1	46.46%	–	–
immagine22	SSD MobileNet v3 Large	1	55.11%	46.20%	–
immagine22	Faster-RCNN Inception v2	1	**93.18%**	–	–
immagine23	SSD MobileNet v1	1	50.62%	–	–
immagine23	SSD MobileNet v2	1	56.63%	–	–
immagine23	SSD MobileNet v3 Small	1	61.86%	45.11%	–
immagine23	SSD MobileNet v3 Large	1	52.80%	46.72%	–
immagine23	Faster-RCNN Inception v2	1	**90.01%**	–	–
immagine24	SSD MobileNet v1	0	–	–	–
immagine24	SSD MobileNet v2	0	–	–	–
immagine24	SSD MobileNet v3 Small	0	–	–	–
immagine24	SSD MobileNet v3 Large	0	–	–	–
immagine24	Faster-RCNN Inception v2	0	–	–	–
immagine25	SSD MobileNet v1	1	98.42%	–	–
immagine25	SSD MobileNet v2	1	**99.67%**	–	–
immagine25	SSD MobileNet v3 Small	1	62.09%	–	–
immagine25	SSD MobileNet v3 Large	1	84.14%	56.88%	–
immagine25	Faster-RCNN Inception v2	1	97.77%	–	–
immagine27	SSD MobileNet v1	1	87.38%	–	–
immagine27	SSD MobileNet v2	1	95.29%	–	–
immagine27	SSD MobileNet v3 Small	1	65.59%	49.29%	–
immagine27	SSD MobileNet v3 Large	1	71.59%	–	–
immagine27	Faster-RCNN Inception v2	1	**96.75%**	–	–

Continua nella prossima pagina

Tabella 4.2 – continuo della tabella precedente

Immagine	Modello	NR. Persone	Persona1	Persona2	Persona3
immagine28	SSD MobileNet v1	2	93.80%	86.62%	–
immagine28	SSD MobileNet v2	2	98.00%	92.27%	–
immagine28	SSD MobileNet v3 Small	2	69.75%	65.52%	53.80%
immagine28	SSD MobileNet v3 Large	2	71.73%	64.24%	48.98%
immagine28	Faster-RCNN Inception v2	2	**98.11%**	**96.90%**	–
immagine29	SSD MobileNet v1	2	91.44%	81.69%	–
immagine29	SSD MobileNet v2	2	96.80%	95.88%	–
immagine29	SSD MobileNet v3 Small	2	66.50%	64.27%	–
immagine29	SSD MobileNet v3 Large	2	77.68%	77.05%	–
immagine29	Faster-RCNN Inception v2	2	**99.20%**	**96.41%**	–
immagine31	SSD MobileNet v1	1	76.67%	–	–
immagine31	SSD MobileNet v2	1	83.31%	–	–
immagine31	SSD MobileNet v3 Small	1	–	–	–
immagine31	SSD MobileNet v3 Large	1	61.08%	–	–
immagine31	Faster-RCNN Inception v2	1	**97.24%**	–	–
immagine32	SSD MobileNet v1	1	84.69%	–	–
immagine32	SSD MobileNet v2	1	86.01%	–	–
immagine32	SSD MobileNet v3 Small	1	66.55%	–	–
immagine32	SSD MobileNet v3 Large	1	73.08%	–	–
immagine32	Faster-RCNN Inception v2	1	**97.31%**	–	–
immagine34	SSD MobileNet v1	2	91.44%	–	–
immagine34	SSD MobileNet v2	2	96.01%	**84.55%**	–
immagine34	SSD MobileNet v3 Small	2	67.34%	–	–
immagine34	SSD MobileNet v3 Large	2	68.50%	49.81%	–
immagine34	Faster-RCNN Inception v2	2	**98.82%**	–	–
immagine35	SSD MobileNet v1	1	82.48%	–	–
immagine35	SSD MobileNet v2	1	58.05%	–	–
immagine35	SSD MobileNet v3 Small	1	55.38%	–	–
immagine35	SSD MobileNet v3 Large	1	61.24%	–	–
immagine35	Faster-RCNN Inception v2	1	**88.19%**	–	–
immagine36	SSD MobileNet v1	1	92.87%	–	–
immagine36	SSD MobileNet v2	1	95.62%	–	–
immagine36	SSD MobileNet v3 Small	1	69.74%	–	–
immagine36	SSD MobileNet v3 Large	1	66.21%	–	–
immagine36	Faster-RCNN Inception v2	1	**97.84%**	–	–
immagine37	SSD MobileNet v1	2	86.27%	–	–
immagine37	SSD MobileNet v2	2	80.91%	–	–
immagine37	SSD MobileNet v3 Small	2	–	–	–
immagine37	SSD MobileNet v3 Large	2	48.57%	–	–
immagine37	Faster-RCNN Inception v2	2	**98.88%**	**63.31%**	–
immagine38	SSD MobileNet v1	2	78.87%	63.39%	–
immagine38	SSD MobileNet v2	2	68.38%	49.68%	–
immagine38	SSD MobileNet v3 Small	2	65.64%	54.20%	–
immagine38	SSD MobileNet v3 Large	2	61.16%	–	–
immagine38	Faster-RCNN Inception v2	2	**97.25%**	**97.10%**	–
immagine39	SSD MobileNet v1	1	84.66%	–	–
immagine39	SSD MobileNet v2	1	85.96%	–	–
immagine39	SSD MobileNet v3 Small	1	–	–	–
immagine39	SSD MobileNet v3 Large	1	62.16%	–	–
immagine39	Faster-RCNN Inception v2	1	**98.91%**	–	–

Dalla tabella precedente possono essere fatte le seguenti osservazioni in merito ai risultati ottenuti dalle *detections* in questa seconda configurazione:

- *FP (false positives)*, si osserva come, in questa seconda configurazione, si è ottenuto un numero minore di FP rispetto la prima. In particolar modo il modello *SSD MobileNet v2* nella prima configurazione era uno dei modelli che dava il numero più alto di FP mentre ora vi è un numero di occorrenze minimo. Per gli altri modelli si ottengono numeri pressoché uguali alla configurazione precedente. Possiamo quindi dire che, in generale, il numero di casi di FP ottenuti migliora in questa seconda configurazione;

- *FN (false negatives)*, si può osservare che anche il numero di FN è generalmente diminuito rispetto ai test eseguiti nella prima configurazione;

- *Confidenza*, in termini di valori di confidenza ottenuti abbiamo risultati che rispecchiano la prima configurazione, ovvero si ha che il modello *Faster-RCNN Inception v2* ci fornisce, mediamente, risultati con valori maggiori di confidenza. Abbiamo poi anche i modelli *SSD MobileNet v1* e *SSD MobileNet v2* che ci forniscono dei risultati con valori di confidenza elevati, metre risultati con valori minori di confidenza si ottengono, generalmente, dal modello *SSD MobileNet v3 Small* e *SSD MobileNet v3 Large*;

- *Persone identificate*, il numero di persone identificate è maggiore rispetto alla prima configurazione in quanto il numero di *FN (false negatives)*, come abbiamo già detto, è diminuito.

In conclusione, abbiamo ottenuto dei risultati molto buoni e in alcuni casi migliori nonostante il *flip* orizzontale applicato alle immagini che sono state sottoposte alle *detections* utilizzando i vari modelli di reti.

Il fatto che in alcuni casi i risultati migliorino ci potrebbe suggerire di sfruttare tale configurazione nelle *detections* effettuate, cioè si potrebbe pensare di effetture la *detection* alla stessa immagine applicando il *flip orizzontale* nel caso in cui la prima *detection* non abbia rilevato alcuna persona nell'immagine, così da aggiungere un controllo ulteriore.

Tale scelta non si è applicata per questo modulo in quanto si è data più importanza a rendere efficiente la durata della batteria con cui è alimentata la Raspberry.

4.3 Risultati Detections in Breve

In questa sezione vengono riassunti i risultati ottenuti dai test eseguiti nelle due configurazioni.

Si riportano soltanto il numero delle persone disponibili in tutte le immagini validate e il numero delle persone individuate, senza tener conto dei casi di *FP (false positives)*. In seguito si calcola il rapporto fra questi due valori, si può osservare quindi i modelli che generalmente danno dei risultati migliori (cioè individuano un numero di persone, fra quelle disponibili, maggiore).

Nella prima tabella (Tabella 4.3) troviamo riportati i risultati ottenuti dai test eseguiti nella prima configurazione, senza cioè apportare alcuna modifica alle immagini sottoposte alle *detections*.

Nella seconda tabella (Tabella 4.4) troviamo invece i risultati ottenuti dai test eseguiti nella seconda configurazione, cioè riflettendo le immagini orizzontalmente prima di sottoporle alle *detections*.

Nelle tabelle in questione troviamo le seguenti informazioni:

- Nome modello;
- Immagini validate, rappresenta il numero di immagini, appartenenti al *dataset* utilizzato, che sono state sottoposte alle *detections*;
- Persone disponibili, rappresenta il numero totale di persone presenti nelle immagini sottoposte alle varie *detections*;
- Persone trovate, rappresenta il numero totale di persone identificate all'interno delle immagini;
- Rapporto, è il rapporto, espresso in percentuale, fra il numero totale di persone trovate e il numero totale di persone disponibili nelle immagini. Rappresenta quindi la "bontà" con cui ogni modello individua il numero di persone fra quelle disponibili.

Tabella 4.3: Risultati Configurazione 1 - In Breve

Modello	**Immagini validate**	**Persone disponibili**	**Persone trovate**	**Rapporto**
SSD MobileNet v1	30	33	30	90.91%
SSD MobileNet v2	30	33	32	96.97%
SSD MobileNet v3 Small	30	33	25	75.76%
SSD MobileNet v3 Large	30	33	29	87.88%
Faster-RCNN Inception v2	30	33	31	93.94%

Tabella 4.4: Risultati Configurazione 2 - In Breve

Modello	**Immagini validate**	**Persone disponibili**	**Persone trovate**	**Rapporto**
SSD MobileNet v1	30	33	31	93.94%
SSD MobileNet v2	30	33	31	93.94%
SSD MobileNet v3 Small	30	33	27	81.82%
SSD MobileNet v3 Large	30	33	31	93.94%
Faster-RCNN Inception v2	30	33	32	96.97%

Dalle tabelle precedenti possiamo osservare come valori molto buoni, in termini di numero di persone identificate, sono ottenuti dai modelli *SSD MobileNet V1*, *SSD MobileNet V2* e *Faster-RCNN Inception v2* in entrambe le configurazioni.

Per i modelli *SSD MobileNet V3 Small* e *SSD MobileNet V3 Large* abbiamo valori leggermente minori rispetto gli altri, che migliorano nella seconda configurazione.

4.4 Tempi computazionali

Si riportano in tale sezione i tempi computazionali richiesti dai modelli di reti utilizzati nei diversi sistemi. I tempi computazionali dipendono infatti in modo rilevante dall'hardware su cui i modelli vengono utilizzati.

Nelle prossime tabelle vengono riportati i tempi di esecuzione richiesti dai modelli soltanto per alcune delle *detections* effettuate. Si è fatta questa scelta per due motivi, innanzitutto per agevolare la lettura delle tabelle, inoltre i tempi ottenuti dai diversi modelli sono, all'interno dello stesso sistema, pressoché costanti. Si ottengono infatti, per ogni sistema, tempi medi che non discostano di molto dai tempi ottenuti dalle singole *detections* delle immagini, per cui non è stato necessario riportare i tempi di tutte le *detections*.

I tempi presenti nelle tabelle di questa sezione fanno riferimento ai test eseguiti nella prima configurazione.

Nelle prossime tabelle troviamo riportate le seguenti informazioni:

- Nome immagine;
- Nome modello;
- Tempo di esecuzione, in secondi.

Si sono effettuati dapprima i test sui *notebooks* di Google Colab. Nella tabella seguente (Tabella 4.5) vengono riportati i tempi computazionali, richiesti dai diversi modelli, ottenuti dalle *detections* effettuate sulle immagini del *dataset* utilizzato. Si è utilizzata l'accelerazione *hardware GPU* offerta dai *notebooks* di Colab.

Tabella 4.5: Tempi Computazionali Google Colab

Immagine	**Modello**	**Tempo (s)**
immagine1	SSD MobileNet v1	0.16171
immagine1	SSD MobileNet v2	0.24769
immagine1	Faster-RCNN Inception v2	1.26030
immagine2	SSD MobileNet v1	0.16668
immagine2	SSD MobileNet v2	0.24385
immagine2	Faster-RCNN Inception v2	1.25760
immagine3	SSD MobileNet v1	0.16139
immagine3	SSD MobileNet v2	0.22916
immagine3	Faster-RCNN Inception v2	1.24406
immagine5	SSD MobileNet v1	0.16183
immagine5	SSD MobileNet v2	0.23149
immagine5	Faster-RCNN Inception v2	1.30062
immagine8	SSD MobileNet v1	0.15713
immagine8	SSD MobileNet v2	0.23156
immagine8	Faster-RCNN Inception v2	1.24499
immagine13	SSD MobileNet v1	0.16737
immagine13	SSD MobileNet v2	0.23558
immagine13	Faster-RCNN Inception v2	1.24899
immagine15	SSD MobileNet v1	0.15727
immagine15	SSD MobileNet v2	0.22807
immagine15	Faster-RCNN Inception v2	1.32258
immagine25	SSD MobileNet v1	0.16693
immagine25	SSD MobileNet v2	0.23283
immagine25	Faster-RCNN Inception v2	1.26928
immagine28	SSD MobileNet v1	0.15654
immagine28	SSD MobileNet v2	0.23928
immagine28	Faster-RCNN Inception v2	1.28216
immagine32	SSD MobileNet v1	0.17125
immagine32	SSD MobileNet v2	0.23886
immagine32	Faster-RCNN Inception v2	1.30084
immagine34	SSD MobileNet v1	0.15739
immagine34	SSD MobileNet v2	0.22603
immagine34	Faster-RCNN Inception v2	1.28283
immagine35	SSD MobileNet v1	0.15904
immagine35	SSD MobileNet v2	0.23870
immagine35	Faster-RCNN Inception v2	1.26863

Continua nella prossima pagina

Tabella 4.5 – continuo della tabella precedente

Immagine	Modello	Tempo (s)
immagine36	SSD MobileNet v1	0.15486
immagine36	SSD MobileNet v2	0.23753
immagine36	Faster-RCNN Inception v2	1.24752
immagine37	SSD MobileNet v1	0.16111
immagine37	SSD MobileNet v2	0.23124
immagine37	Faster-RCNN Inception v2	1.25036
immagine38	SSD MobileNet v1	0.16448
immagine38	SSD MobileNet v2	0.23374
immagine38	Faster-RCNN Inception v2	1.29439
immagine39	SSD MobileNet v1	0.16733
immagine39	SSD MobileNet v2	0.22791
immagine39	Faster-RCNN Inception v2	1.27227

Nella tabella precedente non troviamo i tempi computazionali richiesti dai modelli MobileNetV3 Small e MobileNetV3 Large in quanto non sono compatibili con la versione 4.1.2 di OpenCV che è installata nei *notebooks* di Google Colab e non è ancora disponibile un pacchetto Python per installare la versione 4.3.0 di tale libreria.

Nella tabella seguente (Tabella 4.6) si riportano i tempi computazionali medi calcolati a partire dalla tabella precedente.

Tabella 4.6: Tempi Computazionali Medi Google Colab

Modello	Tempo Medio (s)
SSD MobileNet v1	0.16202
SSD MobileNet v2	0.23460
Faster-RCNN Inception v2	1.27171

Si può osservare come il modello *SSD MobileNet V1* impiega un tempo minore rispetto gli altri due modelli, di poco superiore è il tempo impiegato dal modello *SSD MobileNet V2*. Si osserva invece che il modello *Faster-RCNN Inception v2* è quello che impiega maggior tempo (circa 8 volte maggiore rispetto al tempo impiegato dal modello *SSD MobileNet V1*).

Si riportano nella tabella seguente (Tabella 4.7) i tempi computazionali ottenuti dai test eseguiti sul sistema Ubuntu. In questo caso troviamo i tempi computazionali di tutti i modelli di reti in quanto, su tale sistema, è stato possibile installare la versione 4.3.0 di OpenCV compilandola da sorgente.

Tabella 4.7: Tempi Computazionali Ubuntu

Immagine	**Modello**	**Tempo (s)**
immagine1	SSD MobileNet v1	0.05205
immagine1	SSD MobileNet v2	0.10381
immagine1	SSD MobileNet v3 Small	0.03587
immagine1	SSD MobileNet v3 Large	0.08054
immagine1	Faster-RCNN Inception v2	0.40502
immagine2	SSD MobileNet v1	0.04953
immagine2	SSD MobileNet v2	0.08277
immagine2	SSD MobileNet v3 Small	0.03351
immagine2	SSD MobileNet v3 Large	0.07074
immagine2	Faster-RCNN Inception v2	0.41371
immagine3	SSD MobileNet v1	0.05183
immagine3	SSD MobileNet v2	0.08113
immagine3	SSD MobileNet v3 Small	0.03467
immagine3	SSD MobileNet v3 Large	0.07067
immagine3	Faster-RCNN Inception v2	0.40065
immagine5	SSD MobileNet v1	0.05047
immagine5	SSD MobileNet v2	0.08315
immagine5	SSD MobileNet v3 Small	0.03331
immagine5	SSD MobileNet v3 Large	0.07163
immagine5	Faster-RCNN Inception v2	0.40227
immagine8	SSD MobileNet v1	0.05121
immagine8	SSD MobileNet v2	0.08147
immagine8	SSD MobileNet v3 Small	0.03375
immagine8	SSD MobileNet v3 Large	0.11834
immagine8	Faster-RCNN Inception v2	0.46417
immagine13	SSD MobileNet v1	0.05016
immagine13	SSD MobileNet v2	0.08263
immagine13	SSD MobileNet v3 Small	0.03519
immagine13	SSD MobileNet v3 Large	0.07364
immagine13	Faster-RCNN Inception v2	0.42685
immagine15	SSD MobileNet v1	0.05064
immagine15	SSD MobileNet v2	0.08174
immagine15	SSD MobileNet v3 Small	0.03401
immagine15	SSD MobileNet v3 Large	0.07168
immagine15	Faster-RCNN Inception v2	0.39106
immagine25	SSD MobileNet v1	0.05161

Continua nella prossima pagina

Tabella 4.7 – continuo della tabella precedente

Immagine	Modello	Tempo (s)
immagine25	SSD MobileNet v2	0.09039
immagine25	SSD MobileNet v3 Small	0.03837
immagine25	SSD MobileNet v3 Large	0.07715
immagine25	Faster-RCNN Inception v2	0.52946
immagine28	SSD MobileNet v1	0.05039
immagine28	SSD MobileNet v2	0.09852
immagine28	SSD MobileNet v3 Small	0.03491
immagine28	SSD MobileNet v3 Large	0.08312
immagine28	Faster-RCNN Inception v2	0.41854
immagine32	SSD MobileNet v1	0.05066
immagine32	SSD MobileNet v2	0.08725
immagine32	SSD MobileNet v3 Small	0.03458
immagine32	SSD MobileNet v3 Large	0.07190
immagine32	Faster-RCNN Inception v2	0.42969
immagine34	SSD MobileNet v1	0.04988
immagine34	SSD MobileNet v2	0.08872
immagine34	SSD MobileNet v3 Small	0.03438
immagine34	SSD MobileNet v3 Large	0.07227
immagine34	Faster-RCNN Inception v2	0.39514
immagine35	SSD MobileNet v1	0.05661
immagine35	SSD MobileNet v2	0.08271
immagine35	SSD MobileNet v3 Small	0.04643
immagine35	SSD MobileNet v3 Large	0.08775
immagine35	Faster-RCNN Inception v2	0.42798
immagine36	SSD MobileNet v1	0.05786
immagine36	SSD MobileNet v2	0.10686
immagine36	SSD MobileNet v3 Small	0.04113
immagine36	SSD MobileNet v3 Large	0.08515
immagine36	Faster-RCNN Inception v2	0.39638
immagine37	SSD MobileNet v1	0.04979
immagine37	SSD MobileNet v2	0.09005
immagine37	SSD MobileNet v3 Small	0.03379
immagine37	SSD MobileNet v3 Large	0.07124
immagine37	Faster-RCNN Inception v2	0.43680
immagine38	SSD MobileNet v1	0.07576
immagine38	SSD MobileNet v2	0.11157
immagine38	SSD MobileNet v3 Small	0.04585

Continua nella prossima pagina

Tabella 4.7 – continuo della tabella precedente

Immagine	Modello	Tempo (s)
immagine38	SSD MobileNet v3 Large	0.09084
immagine38	Faster-RCNN Inception v2	0.40266
immagine39	SSD MobileNet v1	0.05287
immagine39	SSD MobileNet v2	0.11878
immagine39	SSD MobileNet v3 Small	0.03457
immagine39	SSD MobileNet v3 Large	0.08830
immagine39	Faster-RCNN Inception v2	0.43212

Dai tempi computazionali presenti nella tabella precedente sono stati calcolati i tempi computazionali medi contenuti nella tabella riportata di seguito.

Tabella 4.8: Tempi Computazionali Medi Ubuntu

Modello	Tempo Medio (s)
SSD MobileNet v1	0.05321
SSD MobileNet v2	0.09197
SSD MobileNet v3 Small	0.03652
SSD MobileNet v3 Large	0.08031
Faster-RCNN Inception v2	0.42328

Dalla tabella precedente si può osservare che i modelli *SSD MobileNet v2* e *SSD MobileNet v3 Large* richiedono, mediamente, tempi di esecuzione simili. Tempi minori sono impiegati dai modelli *SSD MobileNet v1* e *SSD MobileNet v3 Small*.

Il modello *Faster-RCNN Inception v2* richiede, rispetto agli altri, un tempo di esecuzione molto maggiore (circa 8 volte maggiore rispetto al tempo impiegato dal modello di rete *SSD MobileNet V1*).

Infine si sono eseguiti i test direttamente nella Raspberry, osservando quindi i tempi computazionali ottenuti nel sistema in cui andrà realmente in esecuzione il modulo realizzato. I tempi di esecuzione sono riportati nella tabella seguente (Tabella 4.9), da cui poi verranno calcolati i tempi computazionali medi che troviamo riportati nella tabella 4.10.

Tabella 4.9: Tempi Computazionali Raspberry

Immagine	**Modello**	**Tempo (s)**
immagine1	SSD MobileNet v1	0.33838
immagine1	SSD MobileNet v2	0.57797
immagine1	SSD MobileNet v3 Small	0.15960
immagine1	SSD MobileNet v3 Large	0.34277
immagine1	Faster-RCNN Inception v2	3.24788
immagine2	SSD MobileNet v1	0.31757
immagine2	SSD MobileNet v2	0.57008
immagine2	SSD MobileNet v3 Small	0.15896
immagine2	SSD MobileNet v3 Large	0.33949
immagine2	Faster-RCNN Inception v2	3.37151
immagine3	SSD MobileNet v1	0.33095
immagine3	SSD MobileNet v2	0.57811
immagine3	SSD MobileNet v3 Small	0.15910
immagine3	SSD MobileNet v3 Large	0.34028
immagine3	Faster-RCNN Inception v2	3.20338
immagine5	SSD MobileNet v1	0.32237
immagine5	SSD MobileNet v2	0.57445
immagine5	SSD MobileNet v3 Small	0.15873
immagine5	SSD MobileNet v3 Large	0.34227
immagine5	Faster-RCNN Inception v2	3.24623
immagine8	SSD MobileNet v1	0.32206
immagine8	SSD MobileNet v2	0.57306
immagine8	SSD MobileNet v3 Small	0.15707
immagine8	SSD MobileNet v3 Large	0.33941
immagine8	Faster-RCNN Inception v2	3.15279
immagine13	SSD MobileNet v1	0.32282
immagine13	SSD MobileNet v2	0.57328
immagine13	SSD MobileNet v3 Small	0.16040
immagine13	SSD MobileNet v3 Large	0.34440
immagine13	Faster-RCNN Inception v2	3.19916
immagine15	SSD MobileNet v1	0.31397
immagine15	SSD MobileNet v2	0.57932
immagine15	SSD MobileNet v3 Small	0.15726
immagine15	SSD MobileNet v3 Large	0.34315
immagine15	Faster-RCNN Inception v2	3.23310
immagine25	SSD MobileNet v1	0.31905

Continua nella prossima pagina

Tabella 4.9 – continuo della tabella precedente

Immagine	Modello	Tempo (s)
immagine25	SSD MobileNet v2	0.57590
immagine25	SSD MobileNet v3 Small	0.15716
immagine25	SSD MobileNet v3 Large	0.34173
immagine25	Faster-RCNN Inception v2	3.16929
immagine28	SSD MobileNet v1	0.32533
immagine28	SSD MobileNet v2	0.59179
immagine28	SSD MobileNet v3 Small	0.15771
immagine28	SSD MobileNet v3 Large	0.34700
immagine28	Faster-RCNN Inception v2	3.26719
immagine32	SSD MobileNet v1	0.32774
immagine32	SSD MobileNet v2	0.58659
immagine32	SSD MobileNet v3 Small	0.15843
immagine32	SSD MobileNet v3 Large	0.35117
immagine32	Faster-RCNN Inception v2	3.23733
immagine34	SSD MobileNet v1	0.33119
immagine34	SSD MobileNet v2	0.58911
immagine34	SSD MobileNet v3 Small	0.16580
immagine34	SSD MobileNet v3 Large	0.34729
immagine34	Faster-RCNN Inception v2	3.22348
immagine35	SSD MobileNet v1	0.31619
immagine35	SSD MobileNet v2	0.57653
immagine35	SSD MobileNet v3 Small	0.15799
immagine35	SSD MobileNet v3 Large	0.34322
immagine35	Faster-RCNN Inception v2	3.16587
immagine36	SSD MobileNet v1	0.33206
immagine36	SSD MobileNet v2	0.58547
immagine36	SSD MobileNet v3 Small	0.15887
immagine36	SSD MobileNet v3 Large	0.34775
immagine36	Faster-RCNN Inception v2	3.21890
immagine37	SSD MobileNet v1	0.32968
immagine37	SSD MobileNet v2	0.58094
immagine37	SSD MobileNet v3 Small	0.15777
immagine37	SSD MobileNet v3 Large	0.34272
immagine37	Faster-RCNN Inception v2	3.21254
immagine38	SSD MobileNet v1	0.32672
immagine38	SSD MobileNet v2	0.58648
immagine38	SSD MobileNet v3 Small	0.15857

Continua nella prossima pagina

Tabella 4.9 – continuo della tabella precedente

Immagine	Modello	Tempo (s)
immagine38	SSD MobileNet v3 Large	0.35007
immagine38	Faster-RCNN Inception v2	3.26856
immagine39	SSD MobileNet v1	0.32871
immagine39	SSD MobileNet v2	0.59075
immagine39	SSD MobileNet v3 Small	0.15907
immagine39	SSD MobileNet v3 Large	0.34567
immagine39	Faster-RCNN Inception v2	3.22431

Tabella 4.10: Tempi Computazionali Medi Raspberry

Modello	Tempo Medio (s)
SSD MobileNet v1	0.32530
SSD MobileNet v2	0.58061
SSD MobileNet v3 Small	0.15891
SSD MobileNet v3 Large	0.34427
Faster-RCNN Inception v2	3.22760

Dalla tabella precedente osserviamo come, mediamente, il modello *SSD MobileNet v3 Small* richiede minor tempo rispetto agli altri. Troviamo poi i modelli *SSD MobileNet v1* e *SSD MobileNet v3 Large* con tempi computazionali simili, seguiti dal modello *SSD MobileNet v2*.

Il modello *Faster-RCNN Inception v2* anche in questo sistema richiede un tempo di esecuzione molto maggiore rispetto agli altri (circa 10 volte maggiore rispetto al tempo impiegato dal modello *SSD MobileNet v1*).

Si osservi che, nella Raspberry, si ottengono dei tempi medi maggiori rispetto agli altri sistemi ma comunque sotto al secondo (ad eccezione del modello *Faster-RCNN Inception v2*).

Grazie ai risultati delle *detections*, contenuti nelle tabelle 4.1 e 4.2, e ai tempi computazionali medi ottenuti possiamo avere un primo resoconto sui modelli che offrono un miglior rapporto accuratezza/tempo di esecuzione impiegato.

Infine i risultati ottenuti sulla Raspberry, essendo questo il sistema in cui verrà eseguito il modulo, sono di particolare importanza. È infatti di interesse conoscere i modelli che operano in maniera più efficace ed efficiente in esso.

Capitolo 5

Conclusioni e Sviluppi futuri

In questo capitolo si esporranno le conclusioni, in considerazione dei risultati ottenuti, ed alcuni dei possibili sviluppi futuri che da questo lavoro di tesi possono essere realizzati.

Innanzitutto è da osservare che gli obiettivi prefissati sono stati raggiunti. Il modulo software realizzato compie infatti tutte le funzioni che erano state definite, cioè l'acquisizione dell'immagine dalla fotocamera della Raspberry, la successiva verifica della presenza di eventuali persone (*detection*) ed infine l'invio al *cloud* nel caso in cui l'immagine sia priva di persone che coprono i prodotti contenuti negli scaffali.

Si può anche notare che sono stati ottenuti valori di accuratezza elevati.

Altra osservazione può essere fatta circa i tempi computazionali ottenuti dai test eseguiti sulla Raspberry. Gli stessi sono molto contenuti, ad eccezione del modello *Faster-RCNN Inception v2* che, come ci aspettavamo, richiede più tempo. Questo ci permette di eseguire il modulo realizzato in maniera efficiente, così da ottimizzare la durata della batteria con cui la Raspberry è alimentata.

Nonostante i risultati soddisfacenti si potrebbe pensare ad eventuali sviluppi futuri che aggiungano altre funzioni e migliorino ancor di più l'efficacia del modulo realizzato.

In primo luogo si può osservare come, in alcuni casi, i risultati ottenuti dall'esecuzione dei test in configurazione 2 (cioè applicando il *flip* orizzontale all'immagine da sottoporre alla *detection*) siano migliori rispetto quelli ottenuti dalla prima configurazione. Si potrebbe pensare quindi, nel caso in cui dalla *detection* dell'immagine originale non si individui nessuna persona, di sottoporre l'immagine ad un'ulteriore *detection* riflettendola orizzontalmente. In questo modo si aggiungerebbe un ulteriore controllo e si ridurrebbe quindi il numero di *FN (false negatives)* che si verificano nelle *detections*,

cioè immagini in cui sono presenti una o più persone che vengono inviate al *cloud*.

Un altro possibile sviluppo è la gestione dell'accensione e dello spegnimento della Raspberry. Con l'implementazione attuale del modulo abbiamo che l'accensione e lo spegnimento della Raspberry non vengono gestiti, si verificano cioè in maniera indipendente dal modulo realizzato.

Nel caso in cui l'accensione della Raspberry avvenga in tutti casi svantaggiosi, cioè in situazioni in cui nel raggio d'azione della fotocamera della Raspberry troviamo una o più persone per cui l'immagine deve essere scartata, si deve aspettare necessariamente l'accensione successiva. Ciò potrebbe portare ad attese lunghe se l'accensione predefinita della Raspberry avviene ad esempio 2-3 volte alla settimana. Una soluzione a questa problematica potrebbe essere quella di gestire, tramite il modulo, anche lo spegnimento e l'accensione della Raspberry. In questo modo, nel caso in cui l'immagine viene scartata in quanto contiene delle persone, si potrebbe impostare l'accensione della Raspberry dopo 10 minuti senza dover aspettare l'accensione predefinita successiva.

Gestendo quindi l'accensione e lo spegnimento della Raspberry si riuscirebbe sempre a caricare un'immagine nel *cloud* in tempi ristretti, anche se le prime immagini acquisite vengono scartate in quanto presentano una o più persone.

Altro possibile sviluppo che renderebbe il modulo più efficiente potrebbe consistere nella divisione delle immagini acquisite in settori. Nel caso in cui la Raspberry acquisisca un'immagine in cui è presente una o poche persone per cui risulta essere parzialmente coperta, anziché scartare l'intera immagine si potrebbe pensare di dividerla in settori e inviare al *cloud* soltanto quelli privi di persone e scartare gli altri.

Per avere nel *cloud* l'immagine completa, dell'area in questione, si potrebbe impostare una riaccensione della Raspberry poco dopo. In questo modo si controllano i settori che in un primo momento si erano scartati per vedere se ora sono privi di persone e quindi si possono inviare alla piattaforma *cloud*.

In conclusione è da notare il fatto che sono stati ottenuti ottimi risultati senza aver ri-allenato i modelli di reti ma utilizzando quelli pre-allenati sul *dataset* COCO. Tale bontà dei risultati è stata sicuramente avvantaggiata dalla classe di "oggetti" che vogliamo identificare, cioè la classe "person". Questa risulta essere infatti fra le classi più comuni da sottoporre alla *object detection*.

Utilizzando modelli di reti differenti e sistemi differenti su cui eseguire i test è stato possibile fare un confronto oggettivo fra i risultati ottenuti da ogni modello ed in ogni sistema. In questo modo si può scegliere il modello

che si presta a soddisfare i vincoli in maniera più efficace ed efficiente per il particolare caso applicativo in cui si utilizza il modulo.

Bibliografia

[1] M. Paolanti, L. Romeo, M. Martini, A. Mancini, E. Frontoni, and P. Zingaretti, "Robotic retail surveying by deep learning visual and textual data," *Robotics and Autonomous Systems*, vol. 118, pp. 179–188, 2019.

[2] R. Pietrini, D. Manco, M. Paolanti, V. Placidi, E. Frontoni, and P. Zingaretti, "An iot edge-fog-cloud architecture for vision based planogram integrity," in *Proceedings of the 13th International Conference on Distributed Smart Cameras*, pp. 1–5, 2019.

[3] R. Vaira, R. Pietrini, R. Pierdicca, P. Zingaretti, A. Mancini, and E. Frontoni, "An iot edge-fog-cloud architecture for vision based pallet integrity," in *International Conference on Image Analysis and Processing*, pp. 296–306, Springer, 2019.

[4] E. Frontoni, F. Marinelli, R. Rosetti, and P. Zingaretti, "Shelf space re-allocation for out of stock reduction," *Computers & Industrial Engineering*, vol. 106, pp. 32–40, 2017.

[5] E. Frontoni, A. Mancini, P. Zingaretti, and V. Placidi, "Information management for intelligent retail environment: The shelf detector system," *Information*, vol. 5, no. 2, pp. 255–271, 2014.

[6] A. Mancini, E. Frontoni, P. Zingaretti, and V. Placidi, "Smart vision system for shelf analysis in intelligent retail environments," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 55911, p. V004T08A045, American Society of Mechanical Engineers, 2013.

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, pp. 21–37, 2016.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 91–99, Curran Associates, Inc., 2015.

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[11] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[12] C. Eunhyang Kim, M. Maktab Dar Oghaz, J. Fajtl, V. Argyriou, and P. Remagnino, "A comparison of embedded deep learning methods for person detection," *CoRR*, vol. abs/1812.03451, 2018.

[13] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7310–7311, July 2017.

[14] "https://github.com/opencv/opencv."

[15] "https://github.com/opencv/opencv_extra."

[16] "https://github.com/tensorflow/models."

[17] "https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md."

[18] "https://github.com/godpp/detection_util_scripts."

[19] "https://github.com/tzutalin/labelimg."

[20] "http://host.robots.ox.ac.uk/pascal/voc/."

[21] "https://cocodataset.org/#detection-eval."

[22] "https://www.raspberrypi.org/products/raspberry-pi-3-model-a-plus/."

[23] "https://www.raspberrypi.org/products/camera-module-v2/."

[24] "https://cocodataset.org/."

[25] "https://opencv.org/."

[26] "https://www.tensorflow.org/lite."

[27] "https://aws.amazon.com/."

[28] "https://pypi.org/project/boto3/."

[29] "https://ubuntu.com/."

[30] "https://colab.research.google.com/."

Elenco delle figure

Elenco delle tabelle