# POLITECNICO DI TORINO

Corso di Laurea Magistrale

in Ingegneria Informatica

## Tesi di Laurea Magistrale

Deep Learning-Based Real-Time Multiple-Object Detection on a rover


Relatore
Prof. Paolo Garza
Prof. Aldo Algorry(UNC)

Candidato
Leonardo Forese

Anno Accademico 2020/2021

*Alla mia famiglia,*
*per avermi permesso di essere quello che sono.*

*Ai professori Aldo M. Agorry e Gustavo Wolfmann,*
*per avermi accompagnato durante questo progetto.*

*Al professor Paolo Garza,*
*per avermi sostenuto nonostante la distanza in questi tempi difficili.*

*Ai nostri colleghi e amici,*
*per il loro aiuto e la loro disponibilità.*

*Agli argentini un ringraziamento speciale,*
*per avermi ricevuto con rispetto e calore.*

**Sommario**

Lo studio mira a implementare attraverso una rete neurale un algoritmo di intelligenza artificiale che riconosca e differenzi diversi tipi di oggetti.
L'intelligenza deve essere in grado di classificare un oggetto, parametrizzare lo spazio dell'immagine e permettere il tracciamento di uno di essi per mezzo di una telecamera in tempo reale.
In questo progetto si testeranno diversi algoritmi e diversi framework applicati a due reti neurali (YOLO e SSD), al fine di valutare quale di essi è il più affidabile e il più adatto per l'architettura NVIDIA Jetson-Nano.
I migliori risultati saranno basati sulla mAP (precisione media media) e la sua velocità in FPS. Le applicazioni di questo progetto sono ideali per un sistema autonomo dotato di una GPU e capace di riconoscere e tracciare oggetti in tempo reale.

Infine, sarà implementato un modulo per migliorare le prestazioni del sistema: un modulo di sicurezza, che permette all' automa di proteggersi in certi casi limite.
Si analizzano le diverse opzioni in termini di reti utilizzate, cercando anche di ottimizzare l'uso dell'hardware scegliendo l'algoritmo che meglio rispetta il trade-off tra prestazioni e velocità.

Il seguente lavoro fa parte di un più ampio progetto di ricerca sostenuto dall'Università Nazionale di Cordoba, chiamato "INTEGRAZIONE DELLA RILEVAZIONE DEGLI OGGETTI NELLE IMMAGINI IN TEMPO REALE NELLA GUIDA AUTONOMA DEL VEICOLO"-"INTEGRACIÓN DE LA DETECCIÓN DE OBJETOS EN IMÁGENES EN TIEMPO REAL EN LA CONDUCCIÓN AUTÓNOMA DE VEHÍCULOS", finalizzato al funzionamento autonomo di un veicolo terrestre di tipo rover e al quale partecipano i direttori di questo lavoro.

# Indice

# 1

# Introduzione

## 1.1 Motivzione

Il Machine Learning gioca oggi un ruolo sempre più importante nella vita quotidiana. Le sue applicazioni sono molteplici e comprendono diversi settori, da quello militare, alla medicina, alla sicurezza o alla gestione del traffico.
L'apprendimento profondo è una sottocategoria dell'apprendimento automatico. Si riferisce alla funzione e alla struttura di un cervello che è conosciuto come una rete neurale artificiale. Con l'introduzione delle CNN (Convolutional Neuronal Network) e l'aumento della potenza di calcolo dei processori, c'è stato un aumento esponenziale nell'uso delle intelligenze artificiali con tecniche di Deep Learning e soprattutto nella categoria della Computer Vision e quindi nel riconoscimento e tracciamento di oggetti in immagini o video.
Negli ultimi anni, c'è un crescente bisogno di sistemi autonomi nei settori della ricerca, della domotica, delle città intelligenti e militari. Per i compiti di computer vision è necessario prendere oggetti e video in movimento per scopi di analisi. Una soluzione fattibile a questo problema è un dispositivo capace di trovare autonomamente un percorso verso l'oggetto di interesse desiderato con la capacità di prendere immagini e video, identificare e trasportare gli oggetti desiderati e rispondere a vari comandi dati da un umano. Un tale dispositivo potrebbe essere usato in un ambiente estremo come Marte per raccogliere e restituire campioni per analisi scientifiche, o potrebbe essere

usato nell'oceano per scopi di ricerca. L'obiettivo principale del progetto è quello di costruire un'intelligenza che può essere installata in un sistema autonomo e quindi dotarlo della capacità di percepire l'ambiente e prendere decisioni.

## 1.2 Obiettivi

In questo progetto, l'obiettivo è quello di sviluppare, attraverso una rete neurale artificiale in grado di rilevare gli oggetti, un sistema che possa rendere indipendente un veicolo di tipo rover. Un veicolo autonomo è un sistema in grado di percepire il suo ambiente e muoversi in sicurezza con poco o nessun input umano.

Più specificamente, l'intelligenza deve essere in grado di selezionare ed enumerare le persone nell'immagine, assegnare ad ogni persona una diversa identità e, quindi, mantenerla durante il passaggio da un fotogramma all'altro.
Uno degli obiettivi principali del progetto sarà quello di studiare, evidenziare e realizzare un modulo di sicurezza per gestire vari casi limite. I vari controlli saranno effettuati sia su oggetti in movimento che statici, analizzando le loro interazioni associate ai casi citati.
Inoltre, il suo compito sarà quello di tracciare e seguire fotogramma dopo fotogramma sempre lo stesso oggetto (in questo progetto gli oggetti "persona" sono in primo piano), e quindi cercare, in tempo reale, di mantenere il bersaglio concentrato sull'immagine corrente, dando in uscita informazioni utili al sistema autonomo in modo che possa prendere la giusta decisione sui suoi movimenti futuri. Gli obiettivi da sviluppare nel progetto sono i seguenti:

- Indagare e selezionare una rete sufficientemente veloce e precisa.
- Parametrizzare lo spazio dell'immagine e selezionare l'oggetto simile a una persona più vicino al centro.
- Tracciare staticamente l'oggetto target e calcolate l'angolo di movimento del sistema.

- Implementare casi d'uso per validare il sistema in ambienti reali.

- Valutare i possibili problemi di rilevamento e inseguimento del bersaglio, e poi adattare il comportamento dell'intelligenza a quest'ultimo facendo un modulo Safe.

Chiaramente, la scelta e lo sviluppo della rete dovranno tenere conto dell'hardware limitato, poiché il Jetson Nano non ha una GPU di fascia alta.
Infine, e come obiettivo personale dello studente, si intende approfondire la conoscenza dell'intelligenza artificiale in generale, l'elaborazione delle immagini e il rilevamento che viene eseguito, non solo con questo tipo di reti.

## 1.3 Metodologia del lavoro

Strumenti utilizzati:

- NVIDIA Jetson Nano
- Micro Usb 32Gb
- Telecamera Raspberry pi v2
- Monitor HDMI
- Alimentazione MicroUSB 5V-2A
- Periferiche

La metodologia di lavoro scelta è quella iterativa e incrementale. Questo modello è stato creato in risposta alle debolezze del modello tradizionale a cascata, e raggruppa una serie di compiti in piccole fasi ripetitive (iterazioni). Ciò che si cerca in ogni iterazione è di evolvere il prodotto in modo incrementale, implementando nuove caratteristiche e nuove funzionalità. Questo modello consiste in una serie di passi che si ripetono in ogni incremento, iniziando con un'analisi e finendo con il test e la documentazione della funzionalità implementata.

Alcuni dei vantaggi del modello iterativo e incrementale rispetto ad altri modelli di sviluppo del software sono:

- Gli utenti non devono aspettare la consegna del sistema completo prima di poterlo usare.

- Gli utenti possono usare gli incrementi come prototipi e fare esperienza con i vari requisiti sviluppati durante il progetto.

- Permette di separare la complessità del progetto in problemi più piccoli che vengono risolti in ogni iterazione.

- Il prodotto finale che è stato chiaramente realizzato con fasi incrementali è molto meno probabile che abbia un bug, e nel caso in cui qualcosa non funzioni si può tornare alla fase precedente attraverso i backup di ogni fase.

Ci sono due fasi del progetto che riguardano la progettazione e l'implementazione del software. In queste fasi si procede all'analisi degli strumenti con cui si lavorerà, alla progettazione di prototipi funzionali e all'estensione della funzionalità e ai miglioramenti strutturali di questi prototipi.[1]

## 1.4 Tappe del progetto

Il progetto è diviso in cinque fasi:

- Nella prima fase verrà fatto prima uno studio del problema e l'estrazione dei requisiti, segue uno studio delle basi teoriche, della documentazione e dei progetti correlati e come ultima parte della prima fase viene fatta un'analisi sulla selezione degli strumenti software e hardware per lo sviluppo.

- Nella seconda fase si testano diverse reti neurali tenendo conto dell'hardware e si procede alla progettazione e l'implementazione dell'algoritmo di object detection.

- La terza fase consiste nella progettazione e lo sviluppo delle modalità Most Central Object, Farthest Mode, Select Detection Class.

- Il quarto passo riguarderà la progettazione e l'implementazione del Safe Module che renderà l'intelligenza adatta ai casi reali.

- Nella quinta fase si testeranno i moduli implementati, dando una metrica ai parametri dell'algoritmo per trovare il giusto equilibrio tra prestazioni e velocità.

## 1.5 Glossario e Abbreviature

**YOLO** *You Look Only Once*.

**CNN** *Convolutional Neural Network.*

**R-CNN** *Region Based Convolutional Neural Networks*.

**MBO** *Most Central Object.*
**BBOX** *Bounding Box*.

**RELU** *Rectified Linear Unit*.

**IoU** *Intersection Over Union.*

**SSD** *Single Shot Detector.*

**mAP** *Mean Average Precision*.

**VGG16** *Visual Geometry Group version 2016*. Nombre de la red neural

**MAC** *Multiplier Accumulator*.

**ML** *Machine Learning*

**SOT** *Single Object Tracking*.

**FM** *Farthest Mode*.

**GPIO** *General Purpose Input/Output*. Entrada/Salida de Propósito General.

**CUDA** *Compute Unified Device Architecture*.

**API** *Application Program Interface*.

**CPU** *Central Process Unit*.

**GPU** *Graphic Process Unit*.

**CUDNN** *NVIDIA CUDA Deep Neural Network*

**FPS** *Frame Por Second*.

**AP** *Average Precision*.

**DNN** *Deep Neural Network.*

**BGR** *Blue Green Red pixel layout.*

**RGBA** *Red Green Blue alplha pixel layout.*

**DMA** *Direct Memory Access.*

# 2

# Related Work

Le esigenze di oggi vanno ben oltre la semplice classificazione o localizzazione su immagini statiche, oggi la necessità è quella di avere analisi in tempo reale: nessuno vorrebbe essere a bordo di un'auto autonoma che impiega diversi minuti (o anche solo secondi) per riconoscere le immagini.
YOLO è molto più accurato degli altri algoritmi, e anche se è un po' più lento, è ancora uno degli algoritmi più veloci che esistono.
Per questo motivo in questo progetto, come primo algoritmo da testare si è deciso di utilizzare YOLO e poi SSD Mobilenet v2.

## 2.1 YOLO

Yolo è un algoritmo per la detezione di oggetti. Il suo obiettivo è quello di classificare le immagini e di rilevarne il corretto posizionamento all'interno di essi.
La grande differenza rispetto alle altre reti è che yolo invece di utilizzare una pipeline per realizzare tutto il processo in amniera completamente indipendente.
Il tutto richiede molto tempo, cosa che in un sistema reale, specialmente adattato al contesto di un rover non può essere accettato.
L'intero processo di YOLO è quindi questo, viene presa un'immagine come input, e si ottiene alla fine due oggetti, un vettore di bounding box nel quale sono associate ad ogni cella la previsione della classe fatta sull'oggetto.
Andiamo ora nel dettaglio come è stata pensata l'architettura.
Ogni foto è divisa in una matrice di SxS celle, una cella è responsabile dell'oggetto se esso cade nel centro della cella stessa.

La previsione del bounding box ha 5 componenti: (x, y, w, h, confidence). Le coordinate (x, y) rappresentano il centro del bounding box, rispetto alla posizione della cella della griglia. Queste coordinate sono normalizzate per essere tra 0 e 1. Le dimensioni della box (w, h) sono anch'esse normalizzate a [0, 1], relativamente alle dimensioni dell'immagine

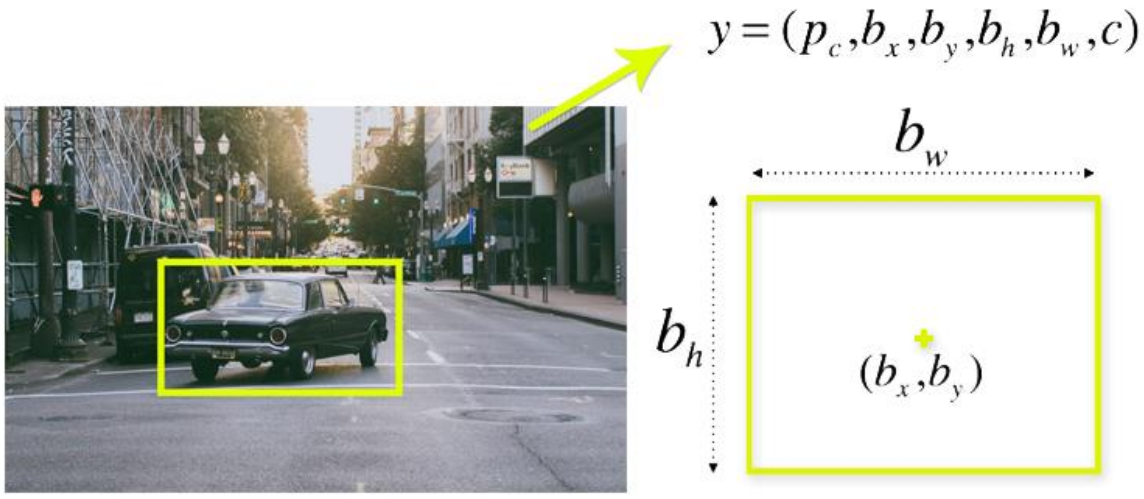


**Figura 2.1:** *Spiegazione delle bbox di YOLO*

In totale ci sono quindiS x S x B * 5 uscite relative alle previsioni delle bounding box.

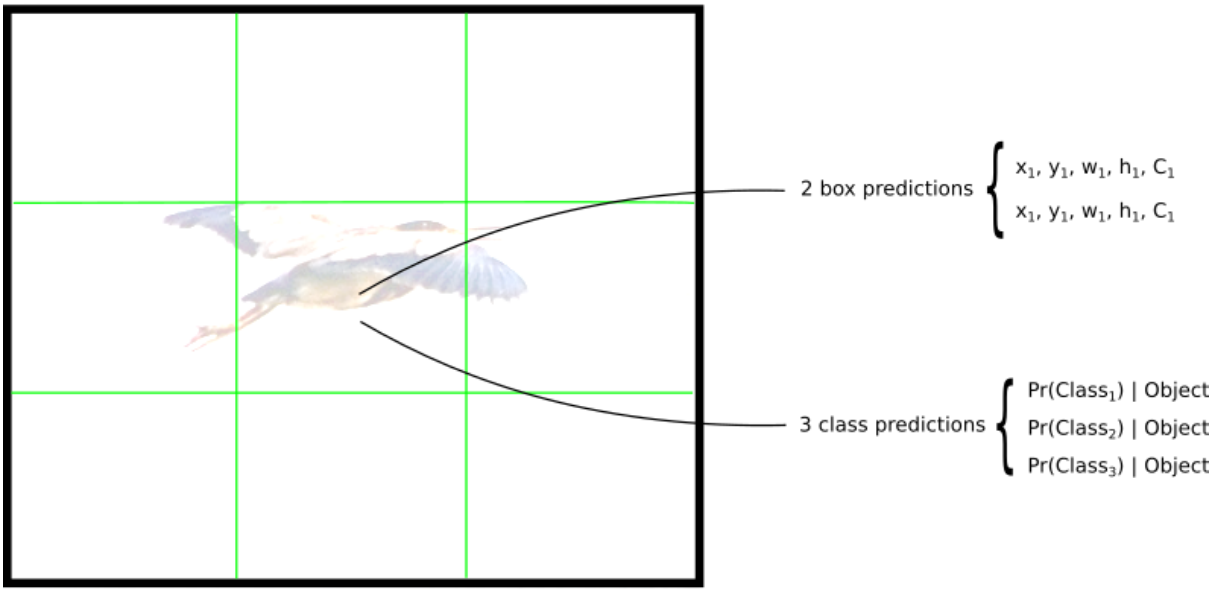


**Figura 2.2:** *Esempio di output di YOLO*

## 2.1.1 La Rete

L’architettura della rete ricorda quella di una CNN:

- convolutional layers e max pooling
- da 2 layers fully connected nel finale

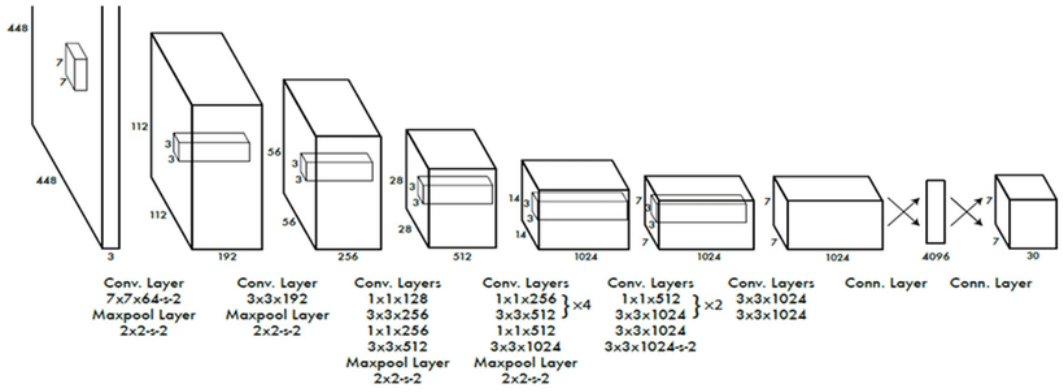


**Figura 2.3:** *Architettura di YOLO*

Alcuni commenti sull'architettura:

- Si noti che l'architettura è stata creata per l'uso nel set di dati Pascal VOC, dove gli autori hanno usato S=7, B=2 e C=20. Questo spiega perché le features map finali sono 7x7, e spiega anche la dimensione dell'output (7x7x(2*5+20)).

- Le sequenze di strati di riduzione 1x1 e di strati di convoluzione 3x3 sono state ispirate dal modello GoogLeNet (Inception).

- Lo strato finale usa una funzione di attivazione lineare. Tutti gli altri strati usano una RELU. (Φ(x) = x, se x>0; 0,1x altrimenti).

## 2.1.2 Il training

Descriviamo il training come segue:

- All’inizio sono stati allenati i primi 20 strati convoluzionali utilizzando il set di dati della competizione ImageNet classe 1000, utilizzando una dimensione di ingresso in input dell’immagine di 224x224.
- Si è aumentata quindi la dimensione a 448x448.
- Si è allenata l’intera rete per 135 epoche, batch 64,momentum 0.9.
- Learning Rate: per le prime epoche, il Learning Rate è aumentato lentamente da 0,001 a 0,01. Allenando per circa 75 epoche e poi inizia a diminuire.
- Uso del Data Augmentation con scale e traslazioni casuali e regolazione dell' esposizione e saturazione in modo casuale[3].

### 2.1.3 Darknet (Neural Network Framework)

YOLO è sviluppato su Darknet che è un framework open source per reti neurali scritto in C e CUDA. È veloce e supporta il calcolo su CPU e GPU. Darknet si installa con solo due dipendenze opzionali: OpenCV se gli utenti vogliono una più ampia varietà di classi di immagini supportate e CUDA, se vogliono il GPU Computing. È un framework veloce e altamente accurato (l'accuratezza di un modello di addestramento personalizzato dipende dai dati di addestramento, dalle epoche, dalla dimensione del batch e da alcuni altri fattori) per il rilevamento di oggetti in tempo reale (può essere utilizzato anche per le immagini).

### 2.1.4 CUDA

CUDA (Compute Unified Device Architecture) è una piattaforma di calcolo parallelo e un modello di interfaccia di programmazione delle applicazioni (API) creato da Nvidia. Permette agli sviluppatori di software e agli ingegneri di utilizzare un'unità di elaborazione grafica (GPU) CUDA-compatibile per l'elaborazione generale.

I core CUDA sono processori paralleli, così come la CPU può essere dual- o quad-core, le GPU NVIDIA ospitano diverse centinaia o migliaia di core. I core sono responsabili dell'elaborazione di tutti i dati in entrata e in uscita dalla GPU.[9]

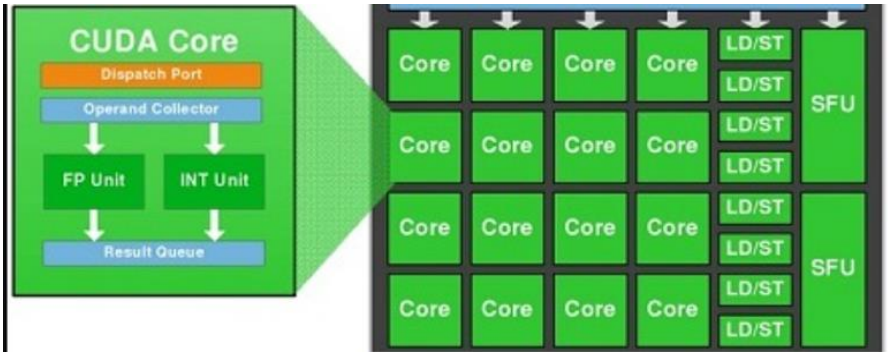


**Figura 2.4:** *Architettura CUDA*

## 2.2 SSD Network

Come seconda opzione, si è deciso di testare l'SSD, per vedere se si adatta all'hardware meglio o peggio di YOLO.
L'SSD è progettato per il rilevamento di oggetti in tempo reale. La più veloce R-CNN utilizza una rete di proposte di regioni per creare scatole di delimitazione e utilizza tali scatole per classificare gli oggetti. L'SSD accelera il processo eliminando la necessità della rete di proposte della regione. Per recuperare il calo di precisione, l'SSD applica alcuni miglioramenti, tra cui caratteristiche multi-scala e caselle predefinite. Sulla base del seguente confronto, corrisponde alla velocità di elaborazione in tempo reale e supera persino l'accuratezza della R-CNN più veloce (l'accuratezza è misurata come la media dell'accuratezza media di mAP: la precisione delle previsioni).

| System | VOC2007 test *mAP* | FPS (Titan X) | Number of Boxes | Input resolution |
|---|---|---|---|---|
| Faster R-CNN (VGG16) | 73.2 | 7 | ~6000 | ~1000 x 600 |
| YOLO (customized) | 63.4 | 45 | 98 | 448 x 448 |
| SSD300* (VGG16) | 77.2 | 46 | 8732 | 300 x 300 |
| SSD512* (VGG16) | **79.8** | 19 | 24564 | 512 x 512 |

**Figura 2.5:** *Comparazione tra YOLO, SSD, FASTER R-CNN*

### 2.2.1 La rete

Il rilevamento degli oggetti SSD consiste in 2 parti:

- Estrarre le features map.
- Applicare i filtri di convoluzione per rilevare gli oggetti.

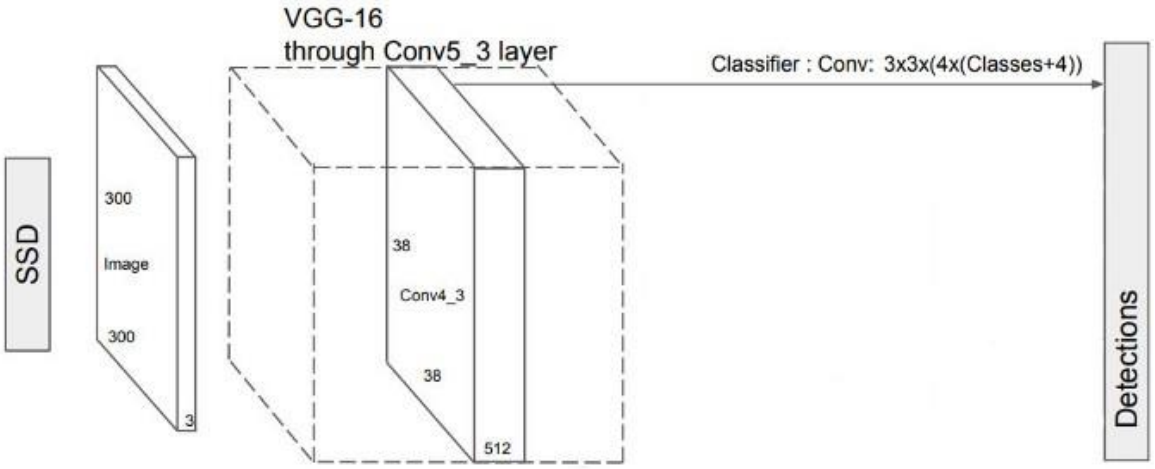


**Figura 2.6:** *Prima parte dell'architettura di SSD*

L'SSD utilizza VGG16 per estrarre le features maps.Si mostra nella figura 2.7 il layer Conv4_3 come una matrice 8x8 nel quale ogni cella fa 4 previsioni.


**Figura 2.7:** *Spiegazione bbox di SSD*

Queste previsioni restituiscono una bounding box e 21 punteggi per ogni classe (una classe extra per nessun oggetto), si sceglie infine il punteggio più alto come classe per l'oggetto.Conv4_3 fa quindi un totale di 38 × 38 × 4 previsioni. Alcune di queste non rilevano quindi nessun oggetto, per questo motivo l'SSD riserva una classe "0" per indicare che non ha un oggetto.

L'SSD quindi utilizza piu strati per rilevare gli oggetti, utilizzando mappe di caratteristiche multi-scala, essendo inoltre una CNN queste features map vanno di volta in volta ad essere diminuite spazialmente. SSD inoltre utilizza strati di risoluzione più bassa per rilevare oggetti su larga scala. Per esempio, le mappe di caratteristiche 4× 4 sono utilizzate per gli oggetti in grande scala. Aggiunge altri 6 strati di convoluzione ausiliari dopo VGG16. Cinque di loro saranno aggiunti per il rilevamento degli oggetti. In tre di questi strati, vengono fatte 6 previsioni invece di 4. In totale, l'SSD fa 8732 previsioni usando 6 strati.

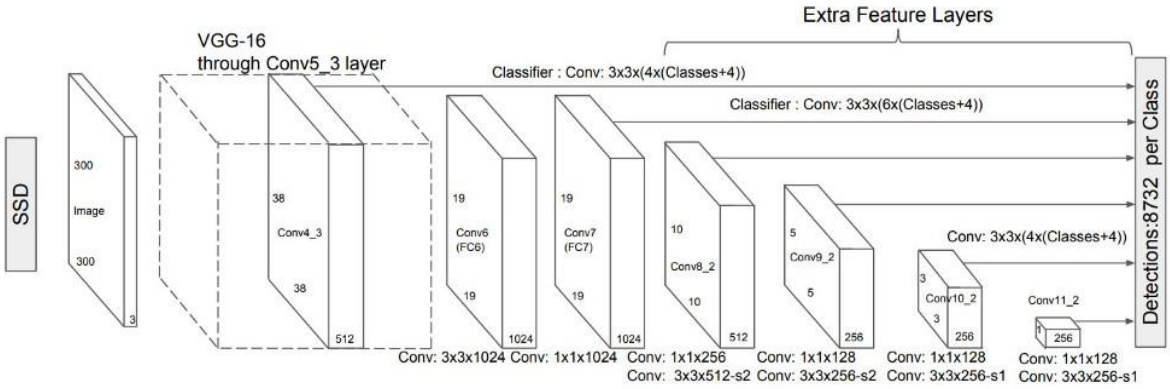


**Figura 2.8:** *Architettura di SSD*

Un'altra cosa importante da ricordare è che SSD utilizza la soppressione non massima per rimuovere le predizioni duplicate che puntano allo stesso oggetto. SSD classifica le previsioni ,infine, in base ai punteggi di fiducia. Partendo dalla previsione con la confidenza più alta, l'SSD valuta se una qualsiasi bounding box prevista in precedenza ha un IoU maggiore di 0,45 con la previsione attuale per la stessa classe. Se trovato, la previsione corrente sarà ignorata.

### 2.2.3 La Loss Function

La perdita di localizzazione è la mancata corrispondenza tra la bounding reale e la bounding box prevista. SSD penalizza solo le previsioni delle corrispondenze positive. Si desidera che le previsioni delle corrispondenze positive siano più vicine alla realtà. Le corrispondenze negative possono essere ignorate.

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

**Formula 1:** *Loss Function-SSD(I)*

La perdita di fiducia è la perdita di fare una previsione di classe. Per ogni previsione di corrispondenza positiva, la perdita è penalizzata in base al punteggio di fiducia della classe corrispondente. Per le previsioni di corrispondenza negative, la perdita è penalizzata in base al punteggio di confidenza della classe "0": la classe "0" classifica che nessun oggetto viene rilevato.[3]

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

**Formula 2:** *Loss Function-SSD(II)*

La funzione di perdita finale è calcolata come

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$$

**Formula 3:** *Loss Function-SSD(III)*

Dove N è il numero di corrispondenze positive e α è il peso della perdita di posizione [4].

## 2.3 MobileNet v1

In questo caso come prima rete SSD si è deciso di testare MobileNet v1.
MobileNetV1 utilizza la convoluzione separabile in profondità per ridurre la dimensione e la complessità del modello. È particolarmente utile per le applicazioni di visione mobile ed embedded.
I concetti chiave di questa rete sono:

- Dimensioni ridotte del modello: minor numero di parametri.
    - Minore complessità: meno moltiplicazioni e Multi-Acquisizione.

La convoluzione separabile in profondità è una convoluzione di profondità seguita da una convoluzione di punti come segue:

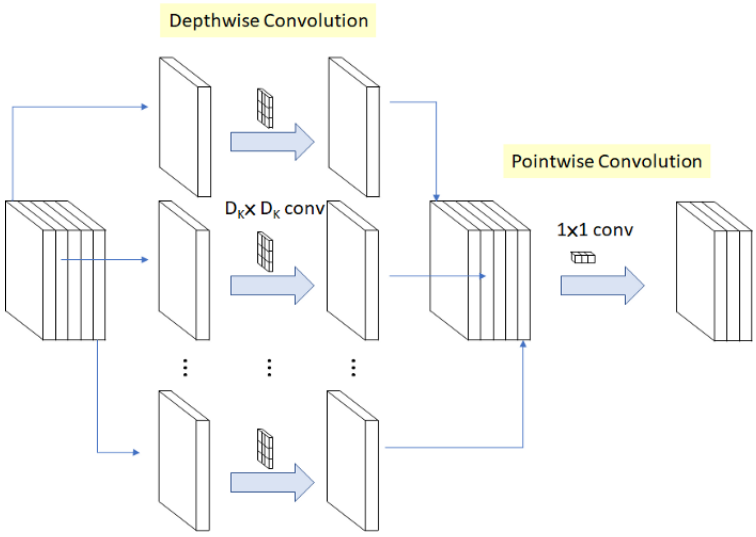


**Figura 2.9:** *Architettura di Mobilenetv1*

1. La convoluzione di profondità è la convoluzione spaziale DK×DK per canale. Si suppone che, nella figura precedente, ci siano 5 canali, quindi ci saranno 5 DK×DK di convoluzione spaziale.

2. La convoluzione dei punti è in realtà la convoluzione 1×1 per cambiare la dimensione.

Con l'operazione di cui sopra, il costo dell'operazione è:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

dove M: numero di canali di ingresso, N: numero di canali di uscita, DK: dimensione del kernel e DF: dimensione della feature map.
Per la convoluzione standard, è:

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$$

Pertanto, la riduzione del calcolo è:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F}$$
$$= \frac{1}{N} + \frac{1}{D_K^2}$$

Si osserva che Batch Normalization (BN) e ReLU sono applicati dopo ogni convoluzione [5] :

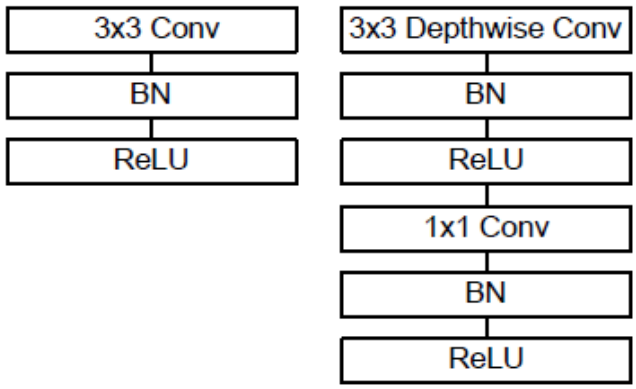


**Figura 2.10:** *Processamento di immagini di mobilenetv1*

Questa è l'architettura di rete completa:

| Type / Stride | Filter Shape | Input Size |
|---|---|---|
| Conv / s2 | $3 \times 3 \times 3 \times 32$ | $224 \times 224 \times 3$ |
| Conv dw / s1 | $3 \times 3 \times 32$ dw | $112 \times 112 \times 32$ |
| Conv / s1 | $1 \times 1 \times 32 \times 64$ | $112 \times 112 \times 32$ |
| Conv dw / s2 | $3 \times 3 \times 64$ dw | $112 \times 112 \times 64$ |
| Conv / s1 | $1 \times 1 \times 64 \times 128$ | $56 \times 56 \times 64$ |
| Conv dw / s1 | $3 \times 3 \times 128$ dw | $56 \times 56 \times 128$ |
| Conv / s1 | $1 \times 1 \times 128 \times 128$ | $56 \times 56 \times 128$ |
| Conv dw / s2 | $3 \times 3 \times 128$ dw | $56 \times 56 \times 128$ |
| Conv / s1 | $1 \times 1 \times 128 \times 256$ | $28 \times 28 \times 128$ |
| Conv dw / s1 | $3 \times 3 \times 256$ dw | $28 \times 28 \times 256$ |
| Conv / s1 | $1 \times 1 \times 256 \times 256$ | $28 \times 28 \times 256$ |
| Conv dw / s2 | $3 \times 3 \times 256$ dw | $28 \times 28 \times 256$ |
| Conv / s1 | $1 \times 1 \times 256 \times 512$ | $14 \times 14 \times 256$ |
| $5\times$ Conv dw / s1 | $3 \times 3 \times 512$ dw | $14 \times 14 \times 512$ |
| $5\times$ Conv / s1 | $1 \times 1 \times 512 \times 512$ | $14 \times 14 \times 512$ |
| Conv dw / s2 | $3 \times 3 \times 512$ dw | $14 \times 14 \times 512$ |
| Conv / s1 | $1 \times 1 \times 512 \times 1024$ | $7 \times 7 \times 512$ |
| Conv dw / s2 | $3 \times 3 \times 1024$ dw | $7 \times 7 \times 1024$ |
| Conv / s1 | $1 \times 1 \times 1024 \times 1024$ | $7 \times 7 \times 1024$ |
| Avg Pool / s1 | Pool $7 \times 7$ | $7 \times 7 \times 1024$ |
| FC / s1 | $1024 \times 1000$ | $1 \times 1 \times 1024$ |
| Softmax / s1 | Classifier | $1 \times 1 \times 1000$ |

**Figura 2.11:** *Architettura completa di Mobilenetv1*

### 2.3.1 Mobilenet v2

La grande idea dietro MobileNet V1 è che gli strati di convoluzione, che sono essenziali per i compiti di computer vision ma sono piuttosto costosi da calcolare, possono essere sostituiti dalle cosiddette convoluzioni separabili in profondità come gia abbiamo spiegato in precedenza.
Tutto il lavoro viene diviso sostanzialmente in due compiti, prima di tutto l'input viene preso da uno strato di convoluzione, dopo si da l output ad una convluzione semplice di 1x1 per creare nuove features.
L'architettura completa di MobileNet V1 consiste in una regolare convoluzione 3×3 come primo strato, seguita da 13 volte il blocco precedente.
MobileNet V2 usa ancora convoluzioni separabili in profondità, però la base è costituita e mostrata nella Figura 2.12.

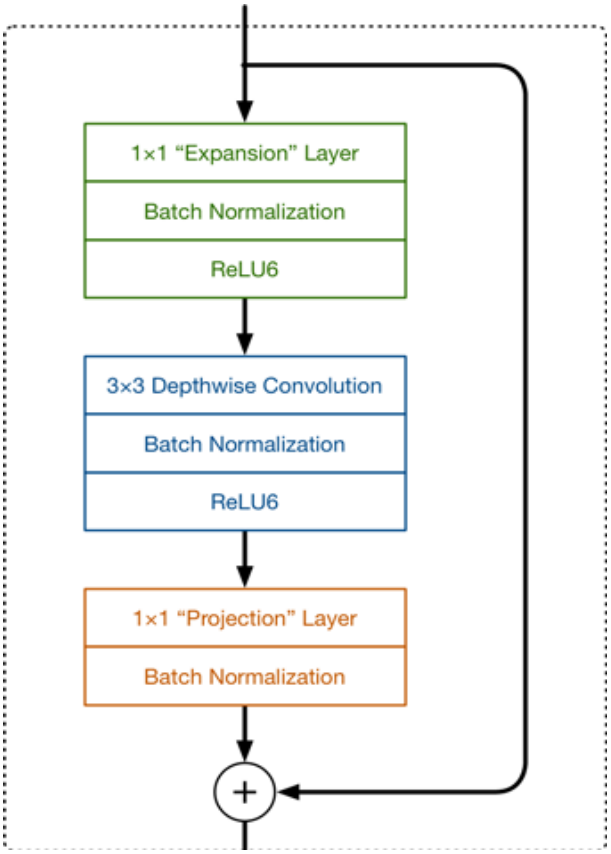


**Figura 2.12:** *Primi livelli di Mobilenet v2*

La struttura questa volta è formata da 3 strati rispettivamente:

- Due convoluzioni in profondità
- Una convoluzione punto punto

Inoltre sono state introdotte due grandi novità:

- La prima che anch’essa è una convoluzione 1x1 ha l’obbiettivo di aumentare il numero di canali prima che vengano dati input alla convoluzione profonda, facendo quindi la cosa contraria dello strato di proiezione. Tutto questo viene regolato da un certo parametro: il fattore di espansione, il quale decide appunto di quanto debbano essere espansi questi dati. In questo layer quindi sihanno sempre un maggior numero di dimensioni in uscita rispetto all’ingresso. Un eventuale esempio può essere il caso di un tensore che ha 24 dimensioni, inizialmente il tutto viene espanso facendo passare le dimensioni a 24*6=144 e infine il livello di proiezione proietta i 144 canali filtrati in un

numero più piccolo, 24.

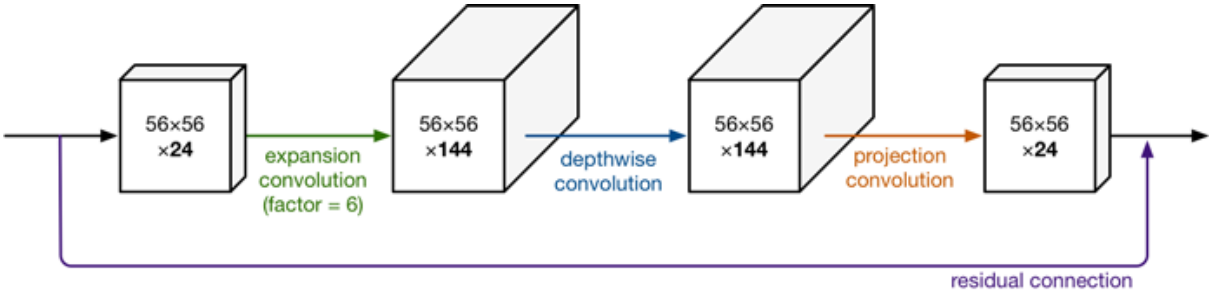


**Figura 2.13:** *Architettura di Mobilenet v2*

- La seconda riguarda il flusso dei gradienti attraverso la rete e viene chiamata connessione residua, la quale viene usata quando il numero di dimensioni di un canale in output è maggiore di quelle in ingresso. Ogni strato ha una normalizzazione batch e la funzione di trigger è ReLU6. Tuttavia, l'uscita dello strato di proiezione non ha una funzione di trigger applicata ad essa. Poiché questo strato produce dati bidimensionali, l'uso di una non linearità dopo questo strato distruggerebbe effettivamente le informazioni utili.

MobileNet V1 viene confrontato con V2, a partire dalle dimensioni dei modelli in termini di parametri appresi e quantità di calcolo richiesto:

| Version | MACs (millions) | Parameters (millions) |
|---|---|---|
| MobileNet V1 | 569 | 4.24 |
| MobileNet V2 | 300 | 3.47 |

5

**Figura 2.14:** *Comparazione tra MobileNet V1 y Mobilenet V2*

I "MAC" sono operazioni di moltiplicazione e accumulazione. Questo misura quanti calcoli sono necessari per eseguire l'inferenza su una singola immagine RGB 224×224. (Più grande è l'immagine, più MAC sono necessari).

Solo per il numero di MAC, V2 dovrebbe essere quasi il doppio più veloce di V1. Tuttavia, non si tratta solo del numero di calcoli. Sui dispositivi mobili, l'accesso alla

memoria è molto più lento dei calcoli. Ma qui V2 ha anche un vantaggio: ha solo l'80% del numero di parametri che ha V1 [6].

## 2.4 Riassunto

In questo capitolo sono stati discussi i diversi algoritmi utilizzati nelle fasi di test, YOLO e SSD, studiando anche la struttura delle reti utilizzate e il loro sviluppo attraverso le fasi di allenamento.
Questa parte del lavoro mirava a dare una spiegazione teorica e matematica dei modelli di computer vision.

# 3

# Disegno del progetto

Questo capitolo spiega il disegno del progetto, per completezza nel Class Diagram completo sono stati inseriti anche i moduli di Follow Me e Tracking Dinamico anche se non di mia competenza.

## 3.1 Descrizione del sistema

Per la descrizione e l'analisi del sistema, il progetto del sistema è diviso secondo la figura 3.1.

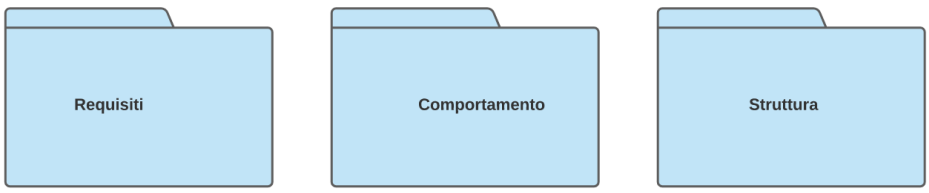


**Figura 3.1:** *Divisione del disegno*

## 3.2 Requisiti funzionali e non funzionali

Questo capitolo elenca i requisiti funzionali, i requisiti che definiscono le funzioni di un sistema o dei suoi sottosistemi, e i requisiti non funzionali, cioè i requisiti che specificano i criteri che possono essere utilizzati per giudicare le prestazioni del sistema.

Nella tabella Requisiti funzionali e non funzionali sono elencati tutti i requisiti:

**Tabella 3.1:** *Requisiti funzionali e non funzionali*

| ***ID Requisito*** | ***Spiegazione del requisito*** | ***Funzionale / Non Funzionale*** | ***Commento*** |
|---|---|---|---|
| *R01* | Capacità di riconoscere diversi tipi di oggetti | Funzionale | Le classi in questione sono quelle appartenenti al COCO-Dataset |
| *R02* | Parametrizzazione dello spazio di visione | Funzionale | |
| *R03* | El sistema deberá selecionar y seguir estáticamente un objeto target | Funzionale | Statico" si intende frame per frame |
| *R04* | Azione: possibilità di selezionare solo una classe di oggetti | Funzionale | |
| *R05* | Fornire un'interfaccia per la comunicazione con l'IA | Funzionale | |
| *R06* | Azione: assegnare diversi ID a ogni oggetto della classe selezionata | Funzionale | |
| *R07* | Azione: Inseguimento dinamico dell'oggetto target | Funzionale | Dinamico significa che l'IA ha una memoria frames per frames dell'obiettivo. |
| *R08* | Il sistema deve raggiungere un oggetto fermo o in movimento. | Funzionale | |
| *R9* | Il sistema deve permettere a una determinata classe di attivare e disattivare la funzione di tracciamento con un movimento specifico del corpo. | Funzionale | Funzione disponibile per gli oggetti di tipo persona |
| *R10* | Python 3.6 sarà usato come linguaggio di programmazione. | No Funzionale | Per mantenere la compatibilità con l'API della rete utilizzata |
| *R11* | Il sistema dovrà raggiungere una bassa latenza tra la produzione dell'immagine e l'uscita dell'algoritmo. | No Funzionale | Essenziale per l'uso dell'IA in tempo reale |
| *R12* | Il sistema permette di scegliere tra l'inseguimento statico e quello dinamico. | Funzionale | |
| *R13* | Il sistema applica il controllo di sicurezza per i casi limite nel tracciamento. | Funzionale | Casi: "target lost", "object collision" e "IDs switch" |

## 3.3 Descrizione del comportamento

### *3.3.1* Casi di uso

Prendendo come utente del sistema un High Level Software (in questo caso la GUI), si descrivono i casi d'uso di quest'ultimo sull'API:

#### 3.3.1.1 CU 1.1 Most central Object

- **Precondizione**: La rete è in funzione, con i moduli di rilevamento e tracciamento attivi. Nell'immagine deve essere presente (e quindi correttamente identificato) almeno un oggetto della classe "persona".
- **Descrizione**: Nella modalità di utilizzo dell'oggetto più centrale, l'IA imposta automaticamente la persona al centro dell'immagine come obiettivo, permettendovi di seguirla.
- **Sequenza normale**:
    1. La rete riconosce in un dato frame t=x una o più persone.
    2. Si controllano tutte le distanze di ogni persona riconosciuta e si prende solo la persona più vicina al centro.
    3. Questa persona è impostata come obiettivo.
    4. L'obiettivo è seguito.

- **Postcondizione:** La rete è sempre attiva, finché non perde il bersaglio e l'eventuale rover inizia a seguirlo.

#### 3.3.1.2 CU 1.2 Farthest Mode

- **Precondizione**: La rete è in funzione, con i moduli di rilevamento e tracciamento attivi. Nell'immagine deve essere presente (e quindi correttamente identificato) almeno un oggetto della classe "persona".
- **Descrizione**: Nella modalità Farthest Mode l'IA imposta automaticamente la persona al centro dell'immagine come obiettivo, permettendovi di seguirla.
- **Sequenza normale**:
    1. La rete riconosce in un dato frame t=x una o più persone.
    2. Si controllano tutte le distanze di ogni persona riconosciuta e si prende solo la persona più lontana dal centro.
    3. Questa persona è impostata come obiettivo.
    4. L'obiettivo è seguito.

- **Postcondizione:** La rete è sempre attiva, finché non perde il bersaglio e l'eventuale rover inizia a seguirlo.

#### 3.3.1.3 CU 1.3 Select Target

- **Precondizione**: L'utente sceglie attraverso l'interfaccia grafica la classe di oggetti da tracciare.La rete è in funzione, con i moduli di rilevamento e tracciamento attivi. Nell'immagine deve essere presente (e quindi correttamente identificato) almeno un oggetto della classe "persona".
- **Descrizione**: Nella modalità Select Target, l'IA seleziona automaticamente l'oggetto più vicino al centro tra tutti gli oggetti della classe scelta dall'utente e lo segue.
- **Sequenza normale**:
    1. La rete riconosce in un dato frame t=x uno o più oggetti della classe scelta dall'utente.
    2. Tutte le distanze di ogni oggetto

riconosciuto sono controllate e solo l'oggetto più vicino al centro è preso.
3. Questo oggetto è impostato come obiettivo.
4. L’obiettivo è seguito.

- **Postcondizione:** La rete è sempre attiva, finché non perde il bersaglio e l'eventuale rover inizia a seguirlo.

#### 3.3.1.4 CU 2.1 Check Target Lost

- **Precondizione**: La rete è in funzione, con i moduli di rilevamento e tracciamento attivi, e un obiettivo è già stato selezionato.
- **Descrizione**: Il seguente modulo permette di segnalare il caso in cui l'obiettivo viene perso durante il tracciamento. Perso significa che il bersaglio è fuori dall'immagine per più di N secondi.
- **Sequenza normale**:
    1. La rete riconosce la perdita dell'obiettivo a causa dell'assenza dell'obiettivo per troppo tempo.
    2. La rete de-seleziona l'oggetto con l'identificazione del bersaglio perso. Questo oggetto è impostato come obiettivo.
    3. Il sistema produce informazioni sul "bersaglio perso".

- **Postcondizione:** Il sistema imposta i parametri di perdita target e ritorna alla modalità scelta dall'utente.

#### 3.3.1.5 CU 2.2 Check Collision

- **Precondizione**: La rete è in funzione, con i moduli di rilevamento e tracciamento attivi, e un obiettivo è già stato selezionato.
- **Descrizione**: Il seguente modulo permette di segnalare il caso in cui la telecamera è vicina a un oggetto, sia il bersaglio che tutti gli oggetti rilevati in generale.
- **Sequenza normale**:
    1. Il sistema controlla se il punto più basso degli oggetti catturati nell'immagine è posizionato sotto un certo limite di soglia
    2. Se il punto 1 è controllato, viene emesso un messaggio di "stop" a causa della possibilità di una collisione con uno o più oggetti.
    3. Nel caso in cui l'ostacolo sia in movimento (persone o animali per esempio) il messaggio di stop viene mantenuto fino a quando la distanza da esso aumenta oltre la soglia accettabile.

- **Postcondizione:** Il sistema imposta i parametri relativi e ritorna alla modalità scelta dall'utente.

## *3.3.2* Diagrammi di sequenza

Per specificare la comunicazione e il comportamento di un sistema attraverso la sua interazione con gli utenti si faccia riferimento al diagramma del caso di uso e i diagrammi di sequenza. (*Figura 3.2*: Diagramma dei casi d'uso).

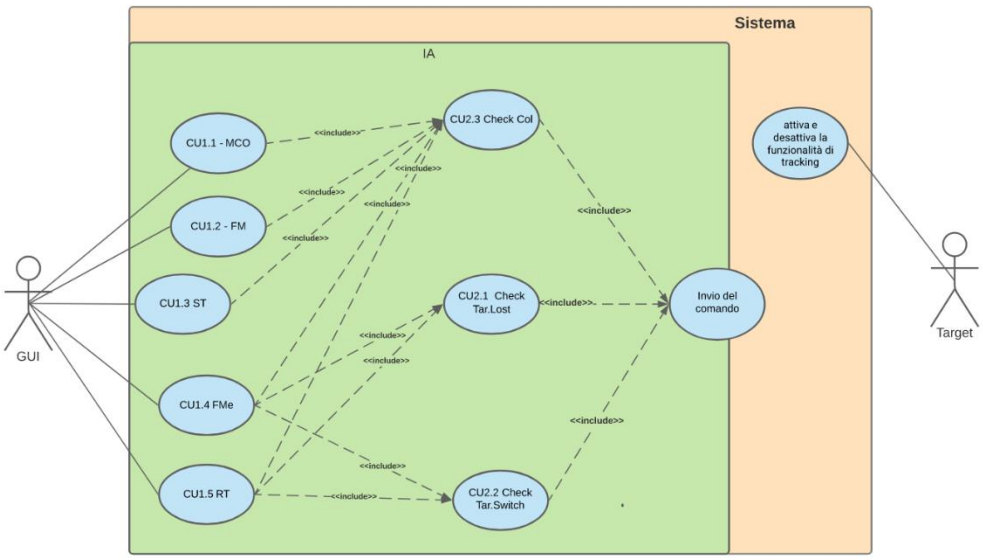


***Figura 3.2*:** *Diagramma dei casi d'uso*

### *3.3.3* Sequenze di esecuzione

I diagrammi seguenti mostrano il flusso delle operazioni che si eseguono in tutte le funzionalità descrivendole ad alto livello (eccetto il Farthest Mode, il cui grafico è molto simile a quello del Most Central Object).
Sono descritti nell'ordine:

- Most Central Object
- Select Detection Class Static

#### *3.3.3.1* Most Central Object

Per illustrare la serie di passi coinvolti, viene analizzata l'esecuzione della funzionalità Most Central Object:

1. L'utente, attraverso l'interfaccia GUI, seleziona la funzione Most Central Object, che è controllata dalla funzione Central_Mode() nel modulo di rilevamento degli oggetti.

2. La rete prende il frame corrente come input e restituisce un array con tutte le informazioni relative agli oggetti rilevati.

3. L'algoritmo calcola il centro di ogni scatola di delimitazione restituita dalla rete.

4. L'algoritmo seleziona quello più vicino al centro e lo imposta come obiettivo.

5. Attraverso la funzione check_collision() del modulo di sicurezza, che prende in input tutti gli oggetti restituiti dalla rete, rileva se c'è una collisione.

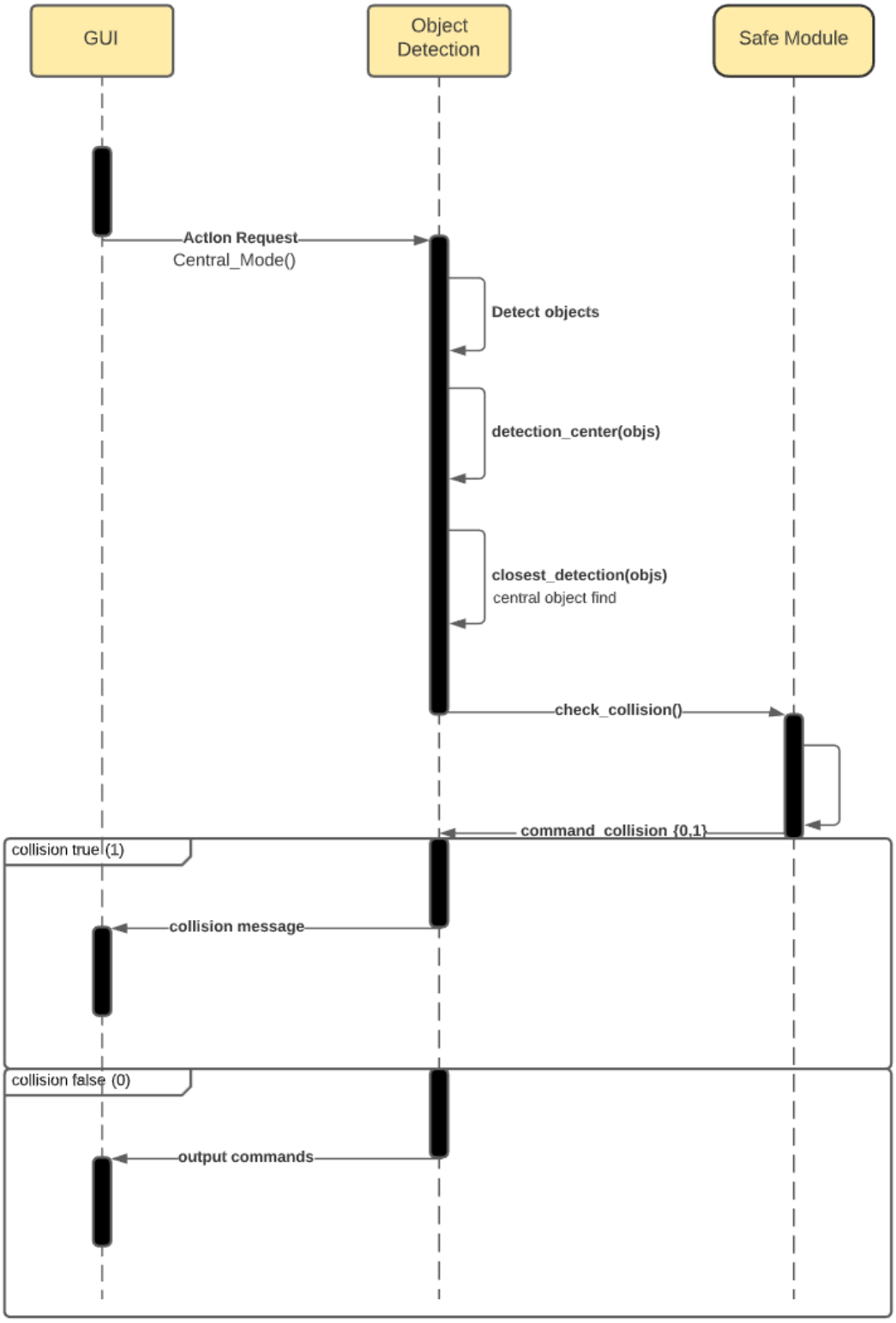


**Figura 3.3:** *Diagramma di esecuzione della Most Central Object*

#### *3.3.3.2* Select Detection Class Static

Per illustrare la serie di passi coinvolti, viene analizzata l'esecuzione della funzionalità Select Detection Class Static:

1. L'utente, attraverso l'interfaccia GUI, seleziona la modalità statica.

2. L'utente, attraverso il nuovo menu, seleziona la funzione Select detection class.

3. L'utente inserisce il nome della classe di richiesta dell'oggetto che è controllata dalla funzione personalized_mode() del modulo di rilevamento dell'oggetto.

4. La rete prende il frame corrente come input e restituisce un array con tutte le informazioni sugli oggetti rilevati.

5. L'algoritmo calcola il centro di ogni scatola di delimitazione restituita dalla rete.

6. L'algoritmo seleziona quello più vicino al centro e lo imposta come obiettivo.

7. Attraverso la funzione check_collision() del modulo di sicurezza, che prende in input tutti gli oggetti restituiti dalla rete, rileva se c'è una collisione.

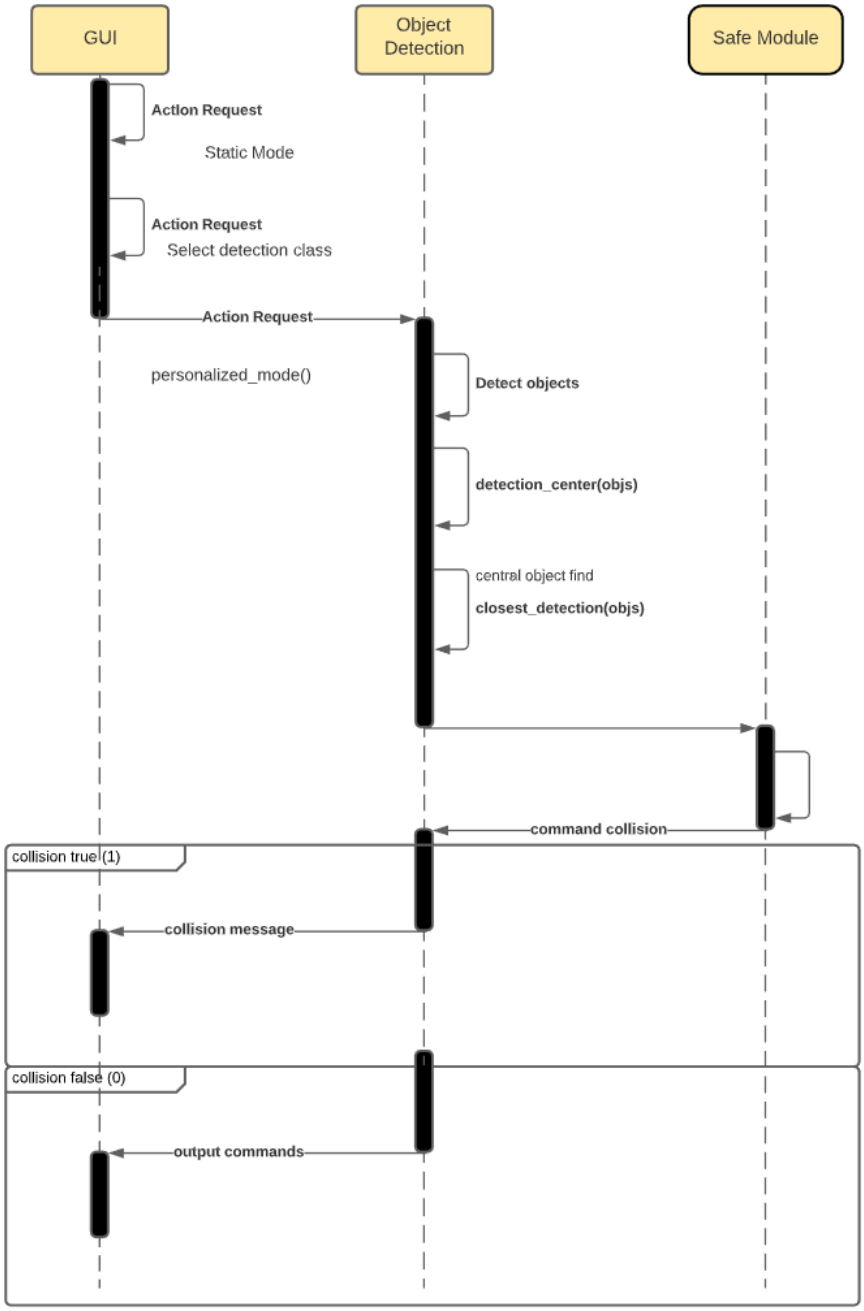


***Figura 3.4***: *Diagramma di esecuzione della Select Class Static*

## 3.4 Descrizione della struttura

Il sistema in esame è costituito da diversi componenti interconnessi che comunicano per il corretto funzionamento del sistema:

- **Microcontrollatore:** Questo componente è costituito dal microcontrollore stesso e dalle sue periferiche associate (come la telecamera) e comunica con l'intelligenza artificiale sia in entrata che in uscita.

- **Inteligenza Artificiale:** L'AI è responsabile dell'implementazione della rete artificiale e di tutti gli algoritmi associati al rilevamento e all'inseguimento degli oggetti. Attende ordini dal microcontrollore e riceve in input i valori passati dalla telecamera (componente del microcontrollore), restituendo informazioni sulla decisione da prendere, l'angolo dell'oggetto, ecc.
- **GUI:** Questo componente del sistema permette all'utente che desidera utilizzare le sue funzioni di scegliere tra diversi modi di utilizzare l'intelligenza artificiale. Permette anche di impostare alcune opzioni, per esempio, in certi casi, il tipo di classe dell'oggetto da tracciare.

- **Utente del sistema:** L'utente è l'attore che, attraverso la GUI, comunica in input con il sistema e che riceve in output le immagini dalla telecamera con le modifiche apportate dall'AI.

- **Obbiettivo Target:** L'oggetto target è l'attore che si riferisce al bersaglio che l'intelligenza artificiale prende di mira attraverso i suoi algoritmi di tracciamento. Il bersaglio non partecipa passivamente, ma, finché l'utente lo sceglie e appartiene alla classe "persona", può attivare e disattivare il tracciamento del bersaglio stesso.

Il sistema è espresso nelle figure seguenti (figura 3.5, figura 3.6, figura 3.7).

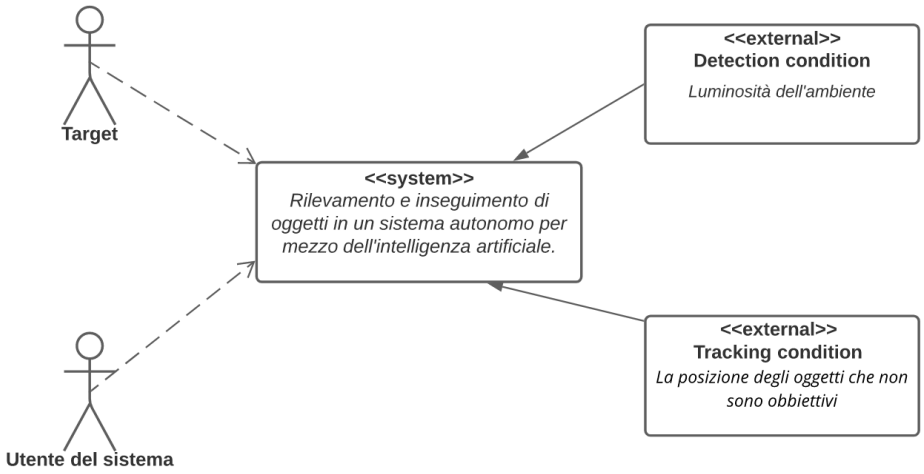


**Figura 3.5:** *Diagramma del contesto del sistema*

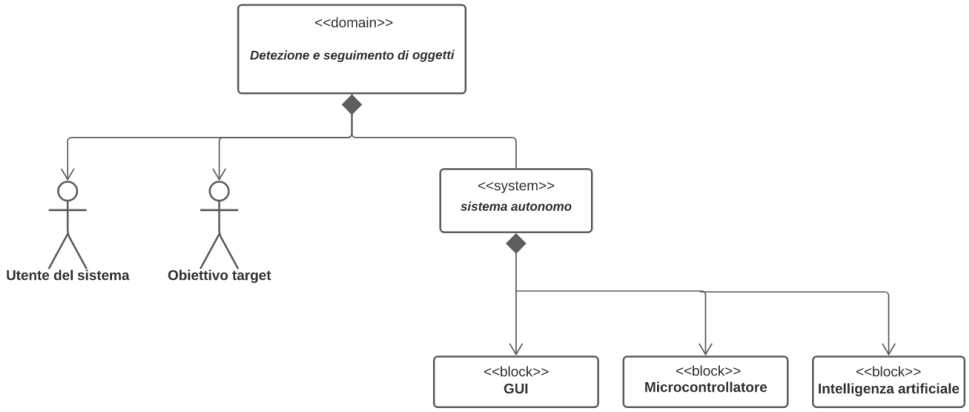


**Figura 3.6:** *High-level Block Definition Diagramm*

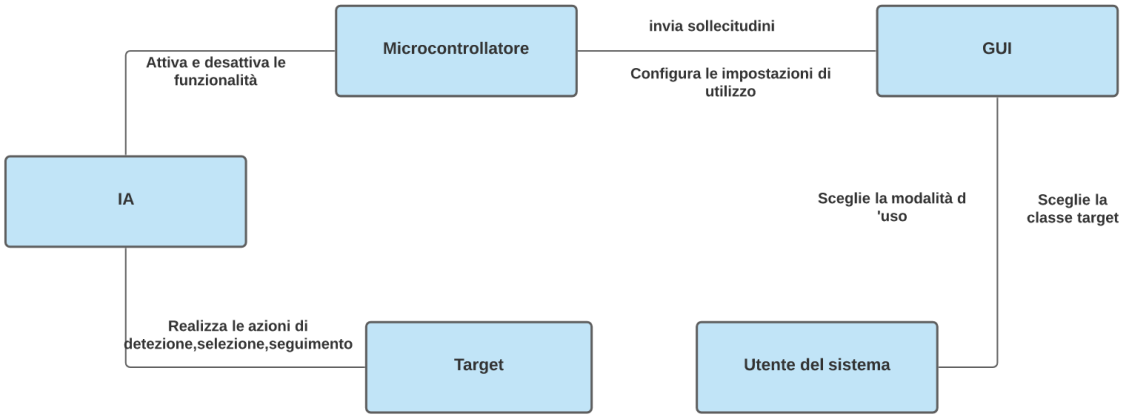


**Figura 3.7:** *Diagramma a blocchi interno*

## 3.5 Architettura della API

Il sistema consiste in:

- Un'interfaccia utente, che tramite pulsanti permette all'utente di utilizzare 5 funzioni divise in due gruppi: Modalità statiche e Modalità dinamiche.
- Un modulo Object Detection che gestisce la rete neurale e le 5 funzioni del sistema, inoltre da questo modulo sono controllati tutti gli altri.
- Un modulo di Tracking che gestisce dinamicamente frame per frame tutti i vari oggetti restituiti dalla rete e li segue.
- Un modulo Follow Me che permette a un possibile utente di essere seguito da un certo movimento e viceversa.
- Un modulo Safe per controllare le possibili collisioni, lo scambio tra l'obiettivo e altri oggetti della stessa classe e la perdita dell'obiettivo.

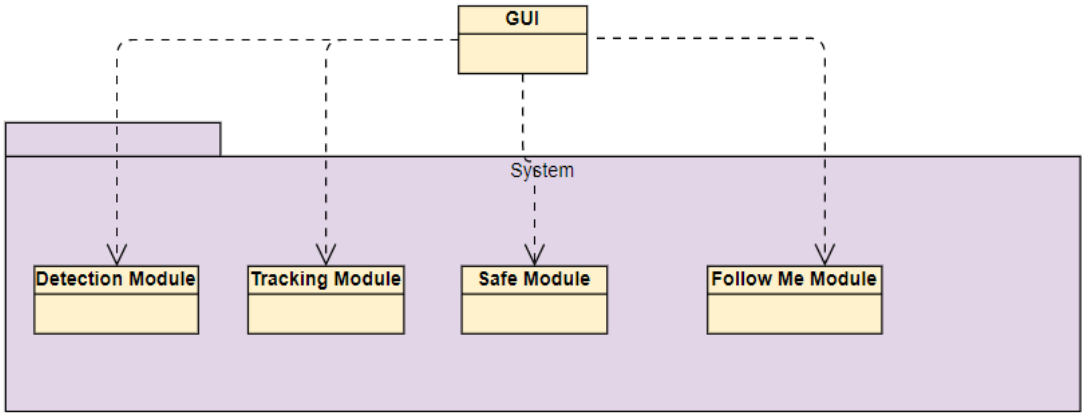


**Figura 3.8:** *Class Diagram di alto livello*

### 3.5.1 Class Diagrams

Prima di illustrare ogni singolo diagramma di classe, la figura 3.9 mostra il diagramma di classe completo:

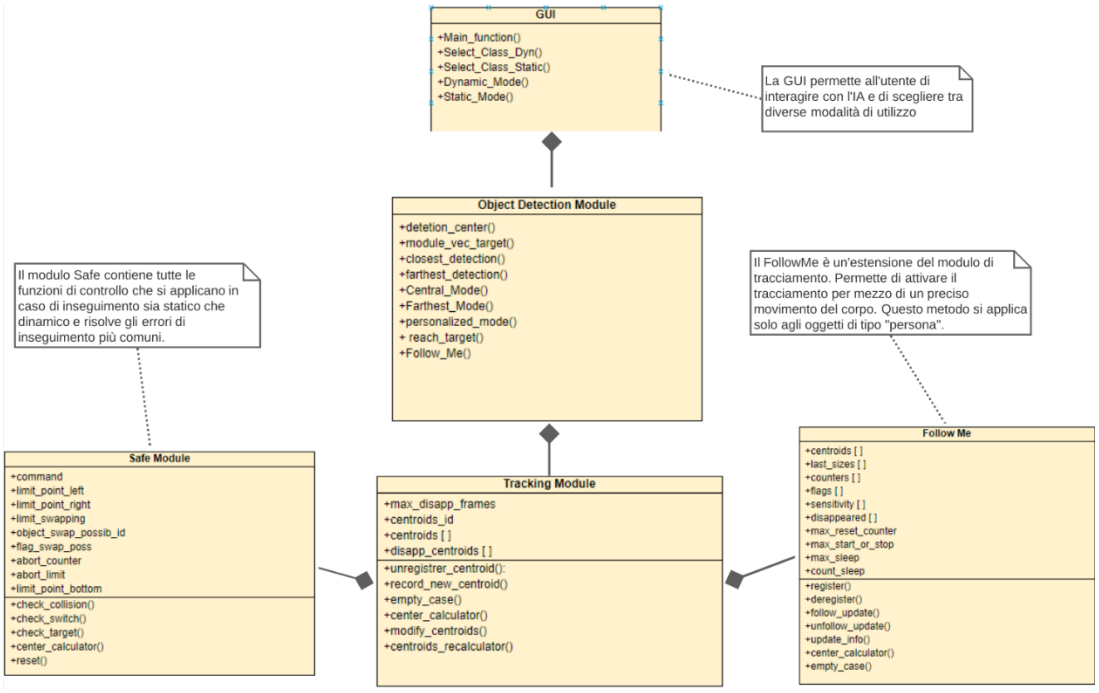


**Figura 3.9:** *Class Diagram completo*

Si spiegano i moduli di Object Detection e di Safe:

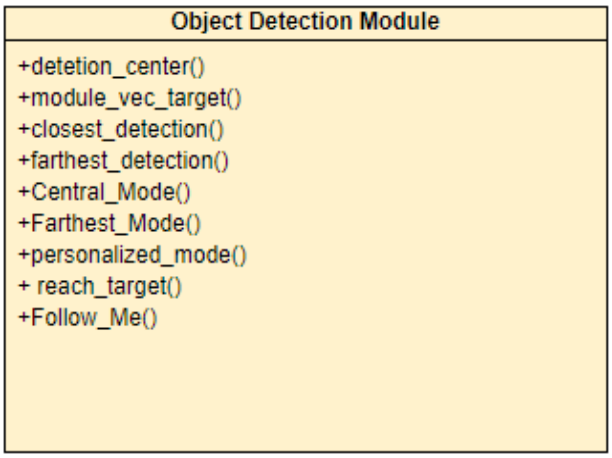


**Figura 3.10:** *Object Detection Module*

- *detection_center():* funzione che, dato un oggetto di tipo detection preso dalla rete, restituisce quello più vicino al centro.
- *closest_detection():* funzione che prende in input il vettore degli oggetti rilevati e restituisce il più vicino possibile al centro della telecamera.
- *farthest_detection():* funzione che prende in input il vettore degli oggetti rilevati e ritorna il più lontano possibile dal centro della telecamera.
- *Central_Mode() :* routine che gestiscono la funzionalità "Most central object".
- *Farthest_Mode() :* routine che gestiscono la funzionalità "Farthest Object".
- *Personalized_Mode() :* routine che gestiscono la funzionalità di "Select Detection Class Static".
- *reach_target()* : routine che gestisce la funzionalità di "Select Detection Class Dynamic".
- *Follow_Me() :* routine che gestisce la funzionalità di "Follow_Me".

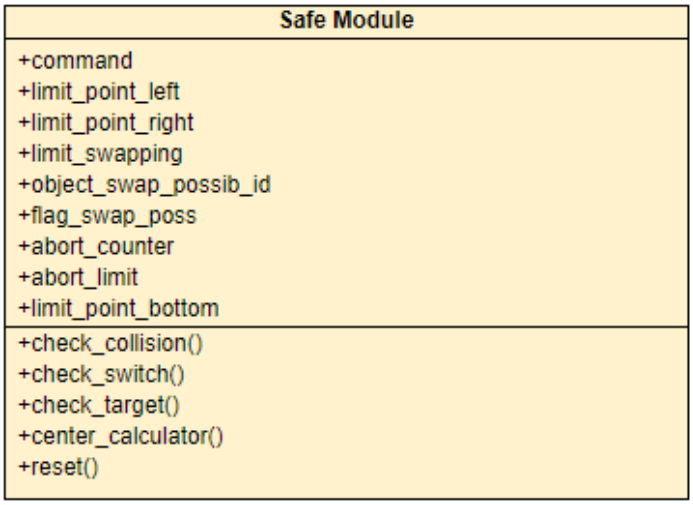


**Figura 3.11:** *Safe Module*

- *command* : parametro che assume un determinato valore in base alle funzioni del Safe Module
- *limit_point_left* : che imposta un limite sinistro basato sulla dimensione dello schermo in pixel.
- limit_point_right: parametro che imposta un limite sinistro basato sulle dimensioni dello schermo in pixel
- limit_swapping: parametro che indica la distanza limite in pixel prima che avvenga lo scambio
- object_swap_possib_id: parametro che indica che un certo ID ha una possibilità di swapping
- flag_swap_poss: parametro che se impostato a 1 indica che è probabile che avvenga uno swap
- abort_counter: contatore per interrompere la missione
- abort_limit: valore per indicare l'interruzione della missione
- limit_point_bottom: limite inferiore per controllare la collisione
- check_collision() : funzione che prende in input tutte le bounding box degli oggetti rilevati in un frame t=x e controlla se c'è probabilità di collisione.

- check_switch(): funzione che controlla in ogni frame se c'è una probabilità di swap.
- *check_target() :* funzione che controlla in ogni frame se l'obiettivo è stato perso.

# 4

# Object detection con NVIDIA Jetson Nano

In questo capitolo si discute l'hardware utilizzato e le due reti testate per il modulo di rilevamento degli oggetti che sono rispettivamente YOLO e SSD.

## 4.1 NVIDIA JETSON NANO

Il kit di sviluppo NVIDIA® Jetson Nano™ è stato scelto per questo progetto. Questa scheda è un piccolo e potente computer che permette di eseguire più reti neurali in parallelo per applicazioni come la classificazione delle immagini, il rilevamento di oggetti, la segmentazione e il processamento della voce. Il tutto in una piattaforma facile da usare che funziona con soli 5 watt. Proprio per questo motivo si è deciso di utilizzarlo per l'elaborazione delle immagini grazie al suo rapporto tra potenza di elaborazione e basso consumo energetico.

**Figura 4.1:** *NVIDIA Jetson Nano*

### 4.1.1 Porte e interfacce

- 4x USB 3.0 A
- USB 2.0 Micro B (Dispositivo)
- MIPI CSI-2 x2 (Connettore per telecamera flessibile a 15 posizioni)
- HDMI 2.0
- DisplayPort
- Gigabit Ethernet (RJ45)
- M.2 Key-E con PCIe x1
- Ingresso per MicroSD
- (3x) I2C, (2x) SPI, UART, I2S, GPIOs
- Processore ARM® Cortex®-A57 MPCore di 4 core
- Architettura NVIDIA Maxwell™ con 128 nuclei NVIDIA CUDA®
- 472 GFLOPS per eseguire rapidamente i moderni algoritmi AI

### 4.1.2 Confronto tra TFLOPs

Cosa sono i FLOPS?
FLOPS è un acronimo per operazioni in virgola mobile al secondo. È usato come misura generale delle prestazioni di un'unità di elaborazione. Le operazioni in virgola mobile sono necessarie quando si ha a che fare con un software che usa una grande varietà di numeri. Il software memorizza numeri esponenzialmente grandi o piccoli in una dimensione prevedibile codificata a 64 bit. La conversione di queste istruzioni a 64 bit in numeri grezzi richiede potenza di elaborazione. I processori possono essere misurati in FLOPS, che si riferisce a quanto velocemente possono convertire le istruzioni di bit in numeri[13].

Confronto della potenza di calcolo tra i moduli Jetson:

**Tabella 4.1:** *Comparazione della potenza di calcolo tra i moduli Jetson*

| | Jetson Nano | TX2 4gb | TX2i | Jetson Xavier NX | Jetson AGX Xavier Series |
|---|---|---|---|---|---|
| **AI Performance** | 472 GFLOPs | 1.33 TFLOPs | 1.26 TFLOPs | 21 TOPs | 32 TOPs |

Confronto della potenza di calcolo tra Jetson Nano e Raspberry Pi3 [14] :

**Tabella 4.2:** *Comparazione della potenza di calcolo tra Jetson Nano y Raspberry Pi3*

| | Jetson Nano Dev Board | Raspberry Pi 3A+ | Raspberry Pi 3B+ |
|---|---|---|---|
| AI Performance | 472 GFLOPS | 21.5 GFLOPs (est*) | 21.4 GFLOPs (est*) |
| CPU | 1.4 GHz 64-bit Quad-Core ARM Cortex-A57 MPCore | 1.4 GHz 64-bit Quad-core ARM Cortex-A53 | 1.4 GHz 64-bit quad-core ARM Cortex-A53 |
| GPU | 128-Core Nvidia Maxwell | Broadcom VideoCore IV | Broadcom VideoCore IV |
| RAM | 4GB LPDDR4 | 512MB LPDDR2 SDRAM | 1GB LPDDR2 SDRAM |
| GPIO Header | 40-pin | 40-pin | 40-pin |
| Board Dimensions | 100 X 79mm | 65 X 56mm | 85 x 56mm |
| Wireless | None | Dual-band 802.11ac wireless LAN, Bluetooth 4.2/BLE | Dual-band 802.11ac wireless LAN, Bluetooth 4.2 |
| Ports | 4x USB 3.0, wired ethernet 10/100/1000Mbps | 1 USB 2.0 | 4 USB 2.0, Wired Ethernet up to 330 Mbps |
| Multimedia | 2160p30 (H.264) | 1080p30 (H.264) | 1080p30 (H.264) |
| Video Output | HDMI, Display Port (4K) | HDMI, Display Serial Interface (DSI) | HDMI, Display Serial Interface (DSI) |
| Camera Serial Interface | YES | YES | YES |
| M.2 Key E Slot | YES | NO | NO |
| Price | $99 | ~$25 | ~$35 |

**Tabella 4.3:** *gflops dei processatori Intel*

| CPU model | Number of computers | Avg. cores/computer | GFLOPS/core | GFLOPs/computer |
|---|---|---|---|---|
| Intel(R) Xeon(R) CPU E5-1620 v2 @ 3.70GHz [x86 Family 6 Model 62 Stepping 4] | 24 | 7.71 | 4.74 | 36.55 |
| Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz [Family 6 Model 60 Stepping 3] | 232 | 7.88 | 4.73 | 37.31 |
| Intel(R) Core(TM) i5-9400F CPU @ 2.90GHz [Family 6 Model 158 Stepping 10] | 51 | 6.00 | 4.73 | 28.35 |
| Intel(R) Core(TM) i9-9900X CPU @ 3.50GHz [Family 6 Model 85 Stepping 4] | 10 | 19.00 | 4.72 | 89.61 |
| Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz [x86 Family 6 Model 58 Stepping 9] | 29 | 8.00 | 4.70 | 37.63 |
| Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz [Family 6 | 58 | 4.00 | 4.70 | 18.81 |

Model 60 Stepping 3]

Come si può vedere nelle tabelle, il Jetson Nano è quasi 20 volte più potente di un Raspberry Pi3 ed è anche paragonabile a un processore Intel Core i5 di 9a generazione [15].

Non è corretto confrontare due CPU o due GPU sui soli FLOPS, fondamentalmente perché il TFLOPS è una misura che non prende in considerazione nulla dell'architettura, ma piuttosto le unità di calcolo e la velocità o frequenza delle unità. Pertanto, lascia da parte qualsiasi parametro che influenza le prestazioni come ingressi e uscite, layout della cache, latenze, ALU, bus e così via.

Per fare un esempio facile e chiaro, la RX 5700 XT ottiene 9,754 TFLOPS, mentre la RTX 2070 ottiene 7,465 TFLOPS, il che riflette una differenza del 30,66% tra i due, ma le prestazioni sono praticamente le stesse nella vita reale.

- TFLOPS-> Shaders x 2 x frequenza in boost.
- RTX 2070-> 2304 x 2 x 1620 -> 7.464.960 FLOPS -> 7.465 TFLOPS
- RX 5700 XT -> 2560 x 2 x 1905 MHz -> 9.753.600 FLOPS -> 9.753 TFLOPS

Poiché AMD fa i suoi calcoli sul Boost Clock, ma si scopre che la frequenza non raggiunge mai veramente tali livelli ed è da qualche parte tra il Game Clock e il Base Clock, è più realistico per il confronto delle prestazioni prendere il secondo piuttosto che il primo, quindi la RX 5700 XT avrebbe circa 8.217 TFLOPS.

Oltre a questo, per confrontare è necessario conoscere le prestazioni per watt e l'architettura almeno, con tutte le varianti che implica. Questo vale per CPU, GPU, console, o qualsiasi componente che si rispetti, dove in molti casi la potenza di CPU e GPU sono sommate quando si tratta di SoC, il che complica ulteriormente il metro di misura.

### 4.1.3 Installazione di Jetson Nano

Per l'installazione si necessita per forza di cose:

- Una scheda microSD (32GB UHS-1 minimo)
- Tastiera e mouse USB
- Schermo del computer (HDMI o DP)

- Alimentazione micro-USB (5V⎓2A)

Il Jetson Nano Developer Kit utilizza una scheda microSD come dispositivo di avvio e per l'archiviazione primaria. È importante avere una scheda che sia veloce e abbastanza grande per il progetto; il minimo consigliato è una scheda UHS-1 da 32GB.

La prima cosa da fare è scrivere l'immagine del sistema operativo sulla scheda microSD.

Per questo si usa la Jetson Nano Developer Kit SD Card Image.

Una volta che il sistema operativo è stato scritto, la scheda microSD può essere inserita nella Jetson Nano[7].

Come sistema operativo si è deciso di utilizzare Ubuntu 18.04, sviluppando in Python con CUDA versione 10.0.

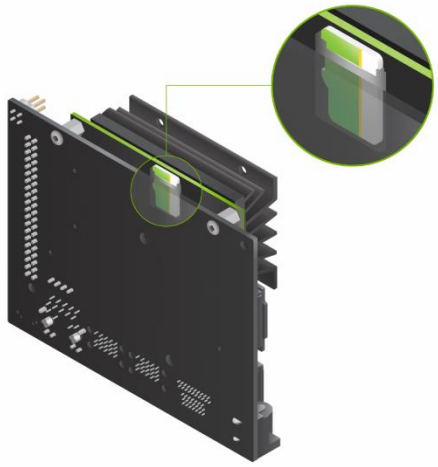

**Figura 4.2:** *Collocazione della microSD nella Jetson Nano*


**Figura 4.3:** *Primer boot*

### 4.1.4 Modalità di alimentazione corretta

U Utilizzando la microUSB con un power bank o un alimentatore da 5V⎓2A, all'avvio, si consiglia di impostare la modalità di alimentazione del Jetson Nano a 5W in modo che non si blocchi.
Ovviamente si può tornare sempre alla configurazione di default di 10W, dato però il numero di periferiche collegate (monitor, telecamera, ...) l'impostazione a 10W provoca il blocco del sistema perché non può garantire la potenza necessaria.

### 4.1.5 Configurare una partizione di swap

Per ridurre la pressione della memoria (e i crash), è una buona idea impostare una partizione di swap di 6GB (Nano ha solo 4GB di RAM).
In questo modo si vanno a prevenire eventuali crash durante la fase di testing degli algoritmi.
Più volte è stato necessario, anche facendo la partizione di swap, di chiudere tutte le applicazioni innecessarie (chrome in primis) per permettere un giusto funzionamento senza che il sistema venga affettato dalla poca quantità di ram disponibile.

## 4.2 YOLO Net

Questo paragrafo spiega come installare ed eseguire Yolo sulla Nvidia Jetson Nano utilizzando la sua GPU a 128 core.

### 4.2.1 Export del path Cuda

Il primo passo è esportare il giusto percorso di Cuda eseguendo determinati comandi da terminale, in questo modo tutti gli utenti del sistema della NVIDIA Jetson Nano saranno in grado di utilizzare la libreria CUDA nel giusto modo. Si ricorda che si è scelto di utilizzare la libreria CUDA 10.0

### 4.2.2 Scaricare Darknet and Yolo

Il prossimo passo da fare è quello di scaricare e copiare la cartella contenente Darknet dal link GitHub e accedere alla cartella Darknet:

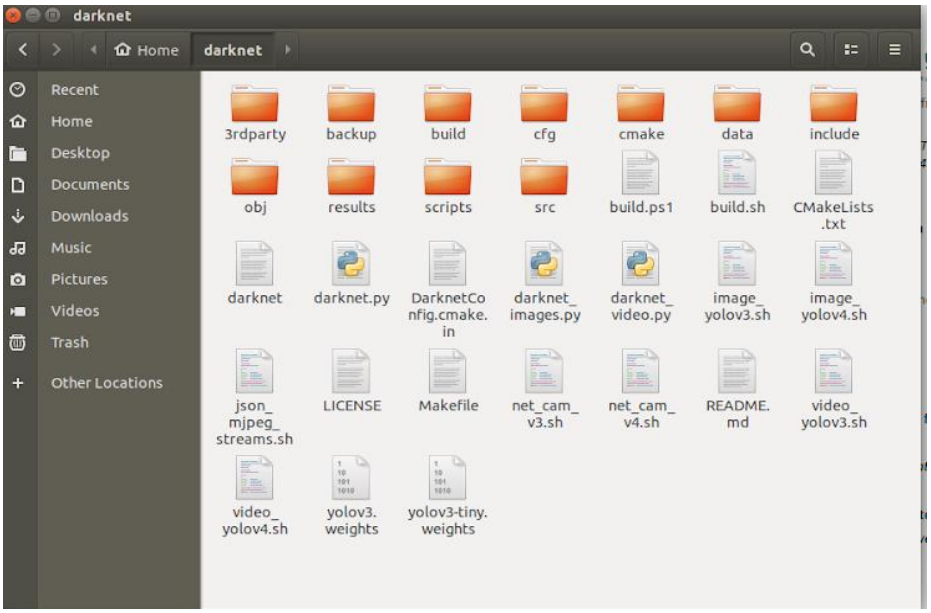


**Figura 4.4:** *Screenshot della directoy darknet*

Vengono inoltre scaricati il file dei pesi dell'algoritmo Yolo che contiene i pesi della rete addestrata sul dataset COCO.
Infine viene anche scaricata la versione YOLOv3-tiny la più piccola e veloce rispetto a YOLO.

### 4.2.3 Attivare la GPU

È necessario modificare Makefile per abilitare GPU, Cuda e Opencv
e impostando solo i valori GPU, CUDNN e OPENCV su uno.
Adesso il sistema è pronto per mandare in esecuzione l 'algoritmo YOLO, in più è possibile utilizzare la libreria di Opencv correttamente.

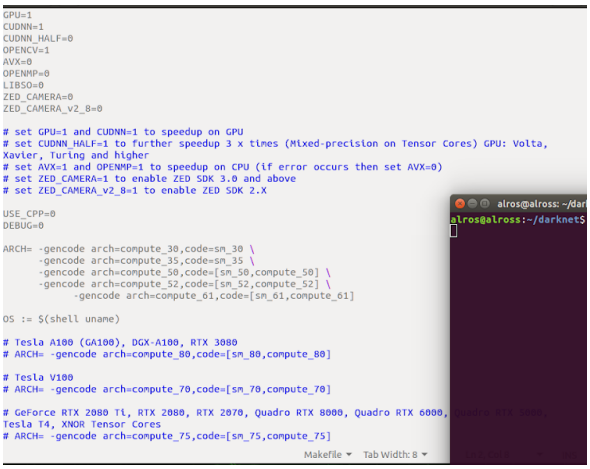


**Figura 4.5:** Modifica del file Makefile

### 4.2.4 I migliori parametri per YOLO v3

In Darknet, il parametro batch è la dimensione del mini-batch. Il parametro di suddivisione controlla quanti campioni entrano nella RAM (mini-batch/suddivisione = campioni caricati simultaneamente per passaggio) - un mini-batch. Una suddivisione non è un mini-batch nel senso convenzionale, perché in Darknet i pesi non sono aggiornati ogni batch (è qui che la denominazione è confusa) a meno che suddivisione == 1.

Per questo motivo questi parametri sono impostati nel file yolov3.cfg:

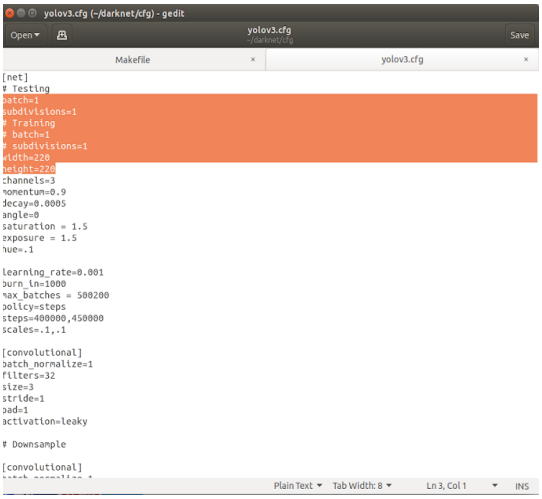


**Figura 4.6:** *modifica yolov3.cfg*

La larghezza e l'altezza dell'immagine di input è stata inoltre ridimensionata per diminuire il costo di calcolo e non sovraccaricare la GPU del Jetson Nano.

### 4.2.5 Analisi di YOLO v3 in Darknet

Eseguendo Darknet dalla riga di comando si possono vedere i risultati nella Figura 4.7.
:

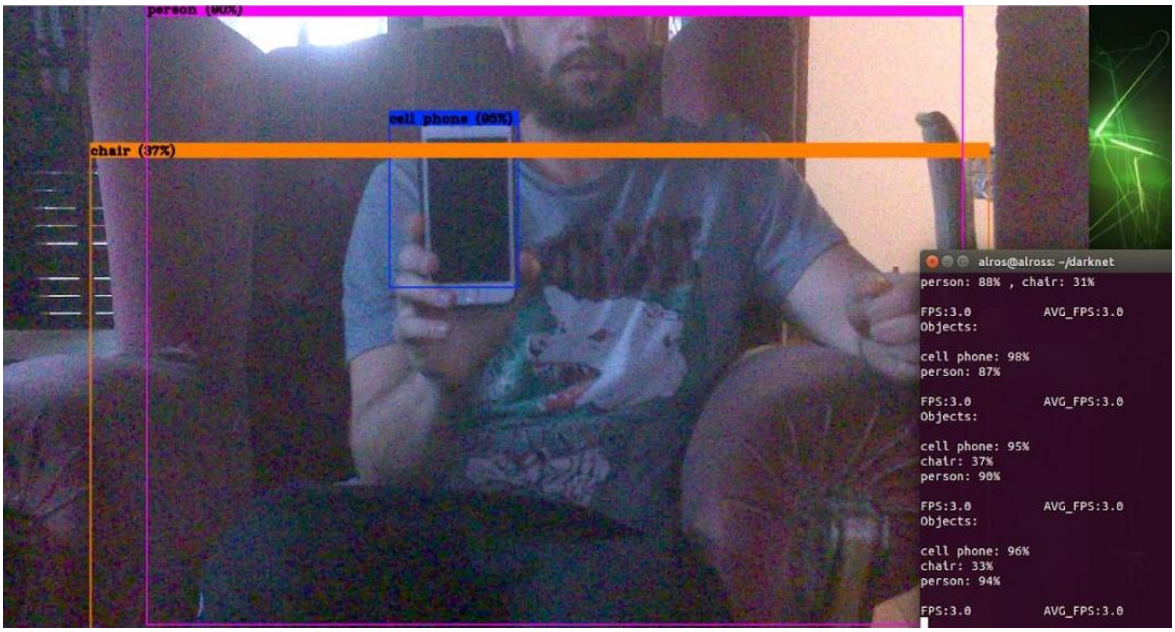


**Figura 4.7:** *Rilevamento oggetti YOLOv3*

Notate come l'immagine mostra le probabilità frame precedente.

Nell'immagine si può vedere come la rete può riconoscere correttamente le tre classi di oggetti e creare bounding box per i tre oggetti nell'immagine con le seguenti probabilità:

- telefono cellulare: 95%
- sedia: 37%
- persona: 90%
- sedia: 37%
- persona: 90%

Nonostante la buona precisione della rete, il numero di fps è così basso che non può essere applicato a una situazione in tempo reale sul rover, poiché l'intelligenza non sarebbe in grado di adattarsi alla velocità.

### 4.2.6 Analisi di YOLO v3 tiny

Per contribuire a rendere YOLO ancora più veloce, i creatori di YOLO hanno definito una variazione dell'architettura YOLO chiamata Tiny-YOLO. L'architettura Tiny-YOLO è circa il 442% più veloce dei suoi fratelli maggiori, raggiungendo oltre 244 FPS su una singola GPU.

Le piccole dimensioni del modello (< 50MB) e la velocità di inferenza rendono il rilevamento degli oggetti di Tiny-YOLO naturalmente adatto per la visione artificiale/dispositivi di apprendimento profondo come il Raspberry Pi, Google Coral e NVIDIA Jetson Nano. Tiny YOLO funziona sugli stessi principi di YOLO ma con un numero ridotto di parametri. Ha solo 9 strati convoluzionali, rispetto ai 24 di YOLO.

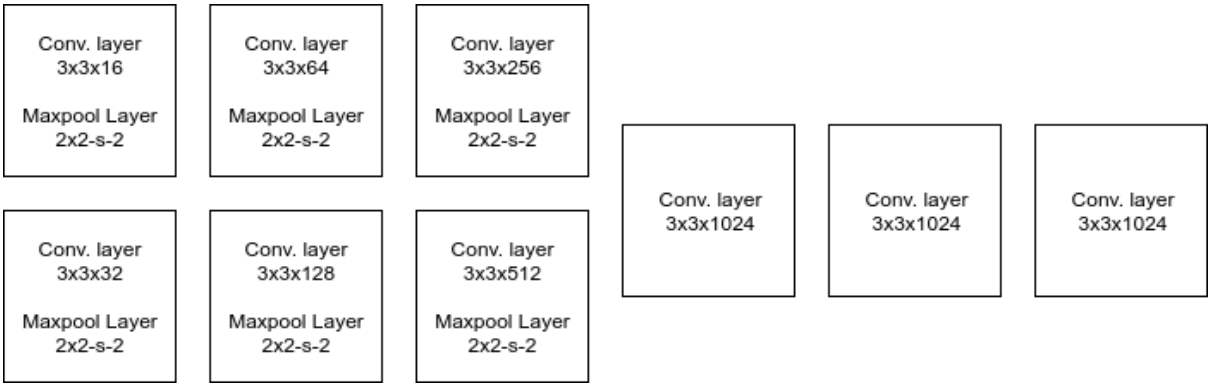


**Figura 4.8:** *Architettura di YOLO v3 tiny*

Nel file di configurazione cambiamo la larghezza e l'altezza e la impostiamo a 416x416, il numero di batch e suddivisioni è lo stesso di YOLO v3:

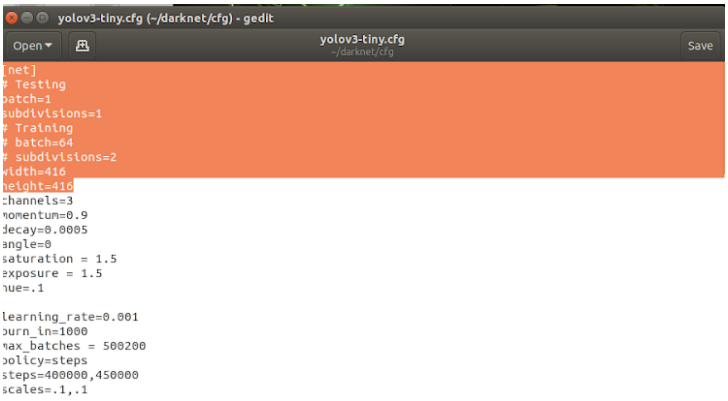


**Figura 4.9:** *Parametro yolov3 tiny*

Risultati ottenuti con YOLO v3-Tiny:

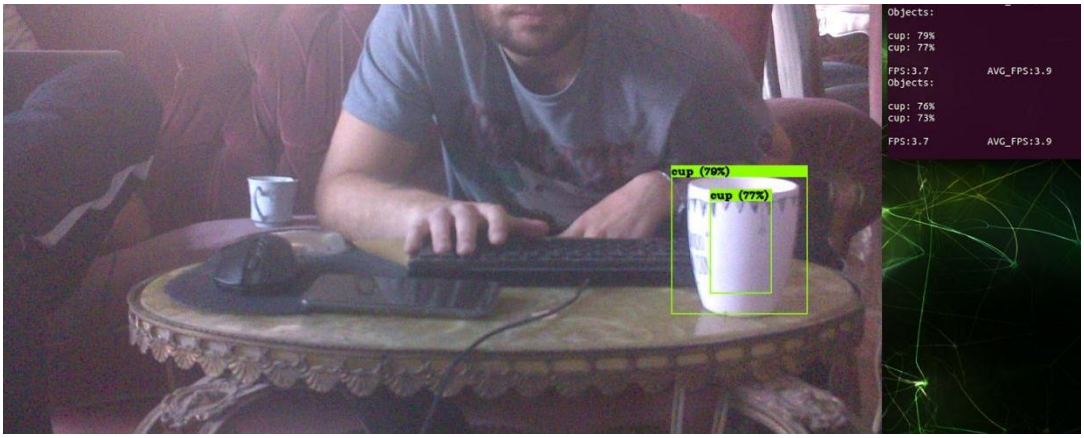


**Figura 4.10:** *Rilevamento oggetti YOLO v3-Tiny*

Nell'immagine si può vedere che l'algoritmo in questo caso è in grado di riconoscere due tazze, anche se in realtà ce n'è solo una con una precisione del 79%. Come si può vedere chiaramente nell'immagine, grazie a un'architettura più leggera della rete, è possibile raggiungere quasi 4 fps in media.
La precisione in questo caso diminuisce, tuttavia, anche questo risultato non raggiunge ancora le condizioni necessarie per un adattamento autonomo.

### 4.2.7 YOLO v4

La quarta generazione di YOLO è stata lanciata nell'aprile 2020. È stato presentato in un articolo intitolato "*YOLOv4: Optimal Speed and Accuracy of Object Detection*" a cura di Alexey Bochkovskiy.

#### 4.2.7.1 Differenza tra YOLO v3 y YOLO v4

Rispetto al precedente YOLOv3, YOLOv4 ha i seguenti vantaggi:

1. Si tratta di un modello di rilevamento degli oggetti efficiente e potente che permette a chiunque abbia una GPU 1080 Ti o 2080 Ti di addestrare un rilevatore di oggetti super-veloce e preciso.
2. È stata verificata la miglioria dei metodi di rilevamento degli oggetti "Bag-of-Freebies" e "Bag-of-Specials" durante l'addestramento del rilevatore.
3. I metodi all'avanguardia modificati, tra cui CBN (Cross Iteration Batch Normalization), PAN (Path Aggregation Network), ecc. sono ora più efficienti e adatti all'addestramento su singola GPU.

#### 4.2.7.2 Miglioramenti importanti di YOLO v4

YOLO v4 prende influenza dallo stato dell'arte BoF (bag of goodies) e da vari BoS (bag of specials). Il BoF migliora la precisione del rivelatore, senza aumentare il tempo di inferenza. Aumentano solo il costo del training. D'altra parte, BoS aumenta il costo di inferenza di una piccola quantità, tuttavia migliora significativamente l'accuratezza di rilevamento degli oggetti. [16]

#### 4.2.7.3 Analisi di YOLOv4

La prima cosa da fare è modificare i parametri di larghezza/altezza nel file cfg cfg/yolov4.cfg.
Con questi parametri è possibile ridurre il costo computazionale della GPU Jetson Nano.


**Figura 4.11:** *Parametri di YOLO v4*

La figura 4.14 mostra gli oggetti rilevati dall'algoritmo, con le loro probabilità ed etichette:

- diningtable 34%
- cup (dos) 92% 97%
- remote 27%
- person (dos) 70% 30%
- laptop 95%
- chair 30%

**Figura 4.12:** *Rilevamento oggetti YOLO v4*

Anche se in questo caso l'algoritmo è in grado di riconoscere un maggior numero di oggetti e ottenere una maggiore precisione, il numero di FPS è ancora troppo basso per essere implementato.
Notate come la rete rileva erroneamente certi oggetti confondendoli con altri (per esempio, la tastiera con il telecomando e la tazza con il bicchiere). Inoltre, la precisione è inferiore rispetto al caso precedente, anche se la rete è ora in grado di rilevare più oggetti.

## 4.3 SSD Mobilnet v2

Il primo passo da fare per installare ls SSD Mobilnet all' interno del sistema è quello di clonare il progetto jetson-inference [11] e di installare i seguenti pacchetti:

- Jetson.inference
- Jetson.utils

Dopo il processo di compilazione, i pacchetti jetson.inference e jetson.utils saranno disponibili per l'uso nei nostri ambienti Python.
Successivamente, si crea una directory di compilazione all'interno del progetto e si esegue il comando cmake per configurare la compilazione. Quando cmake viene eseguito, lancerà uno script (CMakePreBuild.sh) che installerà le dipendenze necessarie e scaricherà i modelli DNN.

Il progetto viene fornito con molte reti pre-addestrate che si possono scegliere e far scaricare e installare attraverso lo strumento Model Downloader (download-models.sh).

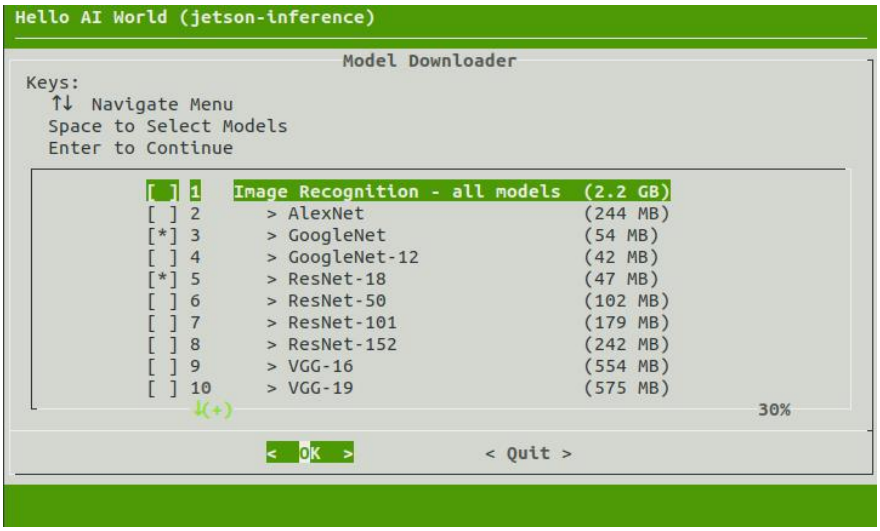


**Figura 4.13:** *Model Downloader*

In questa immagine si possono vedere i modelli di riconoscimento delle immagini, si ricorda che è stata scelta la rete MobileNet v2 SSD per il rilevamento degli oggetti.
Infine si esegue sudo make install per costruire le librerie e i binding di estensione Python.
Il progetto sarà costruito per jetson-inference/build/aarch64, con la seguente struttura di directory:

- *|-build*
- *\aarch64*
- *\bin where the sample binaries are built to*
- *\networks where the network models are stored*
- *\images where the test images are stored*
- *\include where the headers reside*
- *\lib where the libraries are build to*

Ora l'ambiente è pronto, si può procedere allo sviluppo del codice Python e usare le librerie appena installate.
Si è deciso di non usare le librerie Jetson perché non permettono una buona manipolazione dei frames. Per questo motivo si usa direttamente la libreria opencv2.

### 4.3.1 Implementazione di SSD

#### 4.3.1.1 Inizializzazione della rete

- Per lo sviluppo e l'implementazione si ha bisogno di opencv2 per la manipolazione di immagini/fotogrammi, la libreria time per gli FPS e Numpy per la manipolazione dei dati.
- Come primo passo si definisce una funzione che prende come parametri di input il numero float da troncare e il numero di cifre decimali che scegliamo di mantenere, questa funzione è stata scritta per decidere quante cifre decimali prendere dell'accuracy nel vettore delle previsioni.

- Subito dopo è necessario ottenere il tempo corrente, definendo il numero di FPS per inizializzare la rete. Un parametro importante della rete è la soglia, si decide di impostare la soglia a 0,7 per scartare tutti gli oggetti con una probabilità inferiore al 70%.

- Si imposta come si desidera visualizzare lo schermo, la rotazione della telecamera e il font durante il rilevamento in tempo reale.
- Vengono riportati nel dettaglio i parametri studiati per avere la miglior risoluzione possibile:

```
camSet='nvarguscamerasrc wbmode=3 tnr-mode=2 tnr-strength=1 ee-mode=2 ee-strength=1 ! video/x-raw(memory:NVMM), width=3264, height=2464, format=NV12, framerate=21/1 ! nvvidconv flip-method='+str(flip)+' ! video/x-raw, width='+str(dispW)+', height='+str(dispH)+', format=BGRx ! videoconvert ! video/x-raw, format=BGR ! videobalance contrast=1.5 brightness=-.2 saturation=1.2 ! appsink'
```

- *nvarguscamerasrc* con questo comando lanciamo la telecamera.
- *wbmode* = 3 si imposta la modalità di bilanciamento del bianco su 3 per avere la modalità fluorescente.
- *tnr-mode* = 2 si imposta la modalità di rumore del tempo su due, che è la modalità di alta qualità
- *tnr-strength* = 1 per regolare l'intensità della riduzione temporanea del rumore
- *ee-mode* =2 si fissa a due questo parametro per avere un'alta qualità di nitidezza
- *ee-strength* =1 per regolare la qualità dei contorni

- *video/x-raw(memory:NVMM)* usa il buffer DMA non de la CPU
- *width* larghezza dell'immagine
- *heigth* altezza dell' immagine
- *formato* = *NV12* setta il formato dell'immagine per la memoria

- L'altro comando mette semplicemente l'immagine nel formato corretto con una pipeline dal buffer DMA al buffer della CPU e infine mette il formato in modalità BGR.

#### 4.3.1.2 Routine principale

Uno dei problemi incontrati riguarda la compatibilità delle immagini provenienti da OpenCV che sono immagini Rosso-Blu-Verde, con una profondità in interi di 3 e 8 bit; il metodo Detect di SSD non è compatibile con questo tipo di immagini, quindi bisogna applicare una conversione prima di chiamarlo. La prima conversione è da BGR a RGBA. Il tutto è stato possibile con il metodo cvtColor dell'oggetto cv2, il quale permette di applicare la conversione desiderata applicando l'opzione COLOR_BGR2RGBA. Si deve anche dare il tipo del numero .astype(np.float32): un numero in virgola mobile a 32 bit che è il numero che dice qual è l'intensità del colore in ogni posizione.
Altro passo fondamentale nello sviluppo è stato quello di cambiare il tipo di tensore in modo che sia poi compatibile con l' architettura CUDA.
Inoltre dei parametri fondamentali per lo giusto sviluppo del modulo sono le variabili height e width che rappresentano la forma dell'immagine catturata dalla telecamera. Se queste due variabili non sono corrette si ha un crash del programma quando chiaramente si danno in input alla rete.
Questo succede perché la SSD è stata impostata a prendere in input solo immagini con una certa dimensione, che in questo caso è 224x224.

La base che sta a fondo dell'algoritmo è quella di, essendo un rilevatore di oggetti in tempo reale, comportarsi come una routine attiva fino a quando il parametro di escape 'q' viene chiamato.
Infatti per ogni frame con una relativa funzione viene catturato il frame attuale corrente e si restituisce un bool (True/False). Chiaramente se il fotogramma viene letto correttamente, sarà True.

All ‘interno del corpo principale viene quindi messa la rete neurale, la quale per ogni frame attraverso il metodo Detect va a restituire un array di tutti gli oggetti nel frame corrente. Si itera quindi attraverso tutti gli oggetti catturati.
Per ogni iterazione, vengono lette le informazioni sull'oggetto corrente:

- ID, l'indice dell'oggetto.
- Confidenza, la probabilità che l'oggetto sia della classe prevista.
- Top, left, bottom, right (top,left,bottom,right) sono i valori della posizione dell'oggetto nell'immagine.
- Il nome della classe relativa all'ID dell'oggetto (item).

### 4.3.2 Analisi di SSD Mobilenet

Eseguendo l’algoritmo vengono mostrati nella figura 4.14 i risultati:

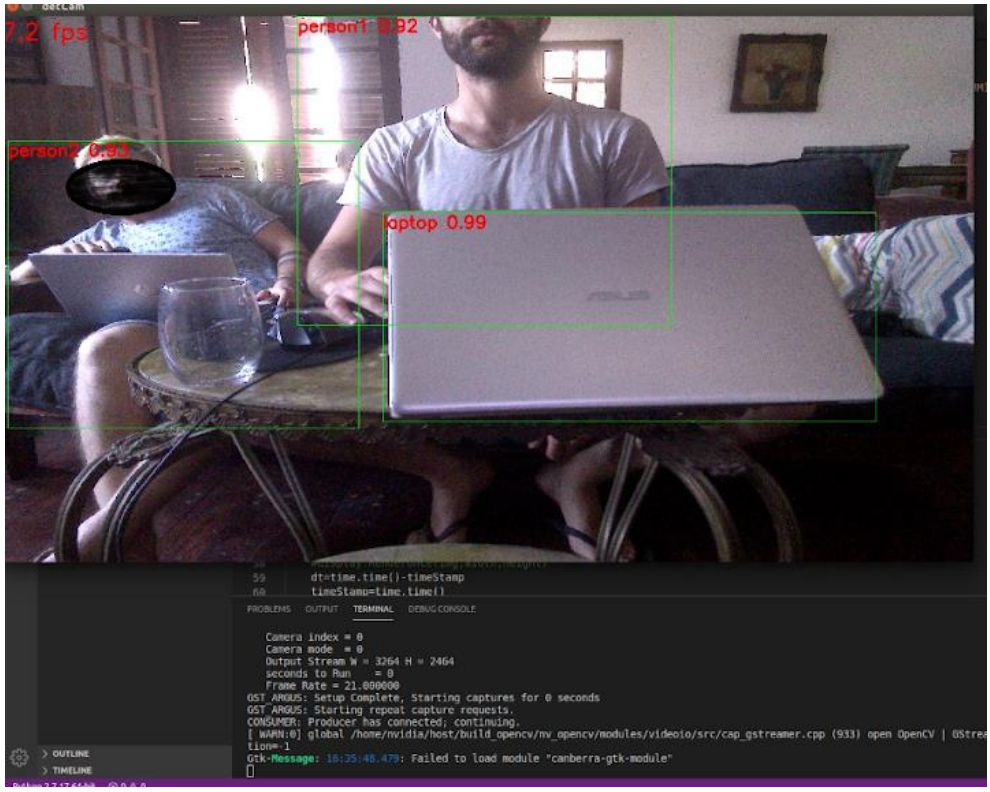


**Figura 4.14:** *Screenshot del rilevamento di oggetti*

L'algoritmo è abbastanza accurato, infatti può riconoscere la persona1 con il 92% di confidenza e la persona2 con il 93% di confidenza. Nell'immagine l'algoritmo può riconoscere più di una persona per fotogramma ed enumerarle per scegliere il tipo di oggetto da seguire.

La prima persona nel frame è etichettata come persona1, così come tutte le altre persone nel frame corrente.
Questo può essere un problema in una possibile realizzazione fisica:
Supponiamo per esempio che il sistema autonomo veda due persone nel frame t=0 e che si rivolga anche alla persona1.
Se la persona1 in qualsiasi fotogramma t=0+n lascia la telecamera, quando riappare nell'immagine nel fotogramma t=n+1 non sarà più etichettata come persona1 ma come persona2.

## 4.4 Conclusioni tra SSD e YOLO

In questo capitolo si è deciso di testare 4 diversi tipi di reti neurali per capire quale fosse il migliore da implementare in base alle caratteristiche hardware del sistema.

- YOLO
  - YOLO v3
  - YOLO v3 tiny
  - YOLO v4

- SSD Mobilenet v2

Come già discusso, la rete completa YOLO v3, anche se ha mostrato una maggiore precisione rispetto al Mobilenet SSD, non è in grado di funzionare a un numero minimo di FPS sull'hardware offerto dal Jetson Nano, rendendo praticamente impossibile un'applicazione pratica in tempo reale.
Si pensava che YOLO v3 tiny potesse superare questo problema di FPS grazie al suo numero molto più basso di parametri di rete rispetto agli altri, anche questo è stato smentito poiché i test hanno mostrato un aumento di un solo FPS che ancora una volta non permette un'applicazione reale. YOLO v4 è chiaramente il migliore in termini di precisione, ma cade sul lato delle prestazioni in tempo reale.
Infine, l'implementazione di un modulo di rilevamento di oggetti utilizzando SSD Mobilenet e Python nel framework Darknet scritto in C ha permesso di raggiungere una media di 8 fps permettendo di sviluppare finalmente il sistema. Per questo motivo tutti i moduli aggiuntivi sono stati scritti in Python e non in C dalla rete SSD.

# 5

# Tracking e implementazione dei moduli

In molte situazioni, ci sono più obiettivi nell'immagine a cui siamo interessati. Non vogliamo solo classificarli, ma anche ottenere le loro posizioni specifiche nell'immagine.

Idealmente, vorremmo utilizzare un sistema autonomo con telecamere che hanno una visione a 360 gradi. Tuttavia, il nostro studio è limitato a una vista di circa 63 gradi del sistema, dovuta all'uso di una singola telecamera.

Nel rilevamento degli oggetti, di solito usiamo un bounding box per descrivere la posizione del bersaglio. Il rettangolo di delimitazione è un box rettangolare che può essere determinato dalle coordinate degli assi x e y nell'angolo superiore sinistro e le coordinate degli assi x e y nell'angolo inferiore destro del rettangolo. L'origine delle coordinate nell'immagine qui sopra è l'angolo superiore sinistro dell'immagine (TOP-LEFT), e a destra e in basso sono le direzioni positive dell'asse x e dell'asse y, rispettivamente (il valore degli assi è in pixel)[13].

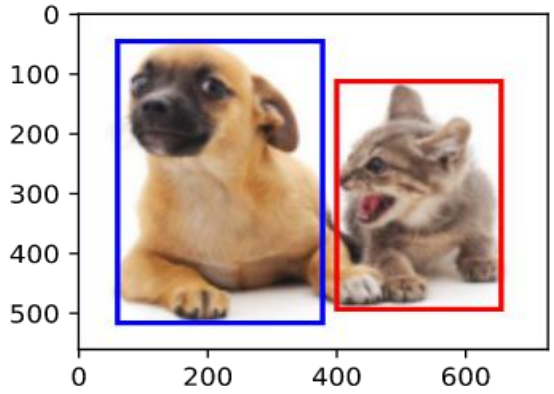


**Figura 5.1:** *Esempio di bounding box*

# 5.1 Tracking statico

## 5.1.1 Most Central Object Mode

Per uno scopo pratico, si è assunto che il parametro di selezione di un oggetto (rispetto a N oggetti della stessa classe) è la distanza del centro della bounding box dal centro dell'immagine. Quindi si è cambiato l'origine degli assi cartesiani, in modo che possa essere collocata al centro dell'immagine.

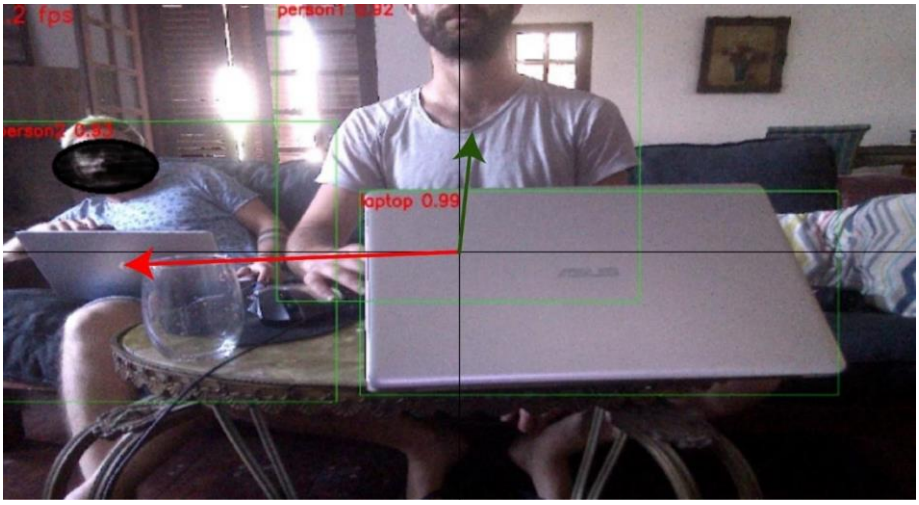


**Figura 5.2:** *Explicación de la distancia desde el centro*

Per calcolare la distanza tra il centro della scatola e il centro dell'asse cartesiano usiamo la seguente formula:

$$dist = \sqrt{(x - x_0)^2 + (y - y_0)^2}$$

**Fórmula 4:** *Euclidean Distance*

dove x0 e y0 sono i centri dell'asse cartesiano.

$$x_0 = x_{max}/2$$

$$y_0 = y_{max}/2$$

**Fórmula 5:** *Calculacion del centroide*

Quindi l'obiettivo selezionato sarà quello con lo standard più basso rispetto a tutti gli altri.

Il Most Central Object è stato creato per permettere al rover di seguire sempre la persona più centrata rispetto al suo campo visivo, questo le permette di seguirla e di autocorregersi con rate di 8 volte al secondo. La modalità è stata pensata per identificare

e selezionare l'oggetto con la classe "persona".

### 5.1.1.2 Creazione dell'angolo e della direzione

Per guidare il sistema autonomo, il sistema di assi cartesiani viene cambiato di nuovo impostando l'origine al centro dell'immagine per l'asse x, e alla base dell'immagine per l'asse y.
Se l'oggetto cade a sinistra dell'asse x, allora il robot sterzerà a sinistra (e viceversa permettendo così all'automa di centrare l'oggetto target nel suo campo visivo) per un totale di gradi calcolati secondo la formula:

$$atan\left(\left|x - x_c\right| / \left(y^*\right)\right)$$

**Fórmula 6:** *Formula para calcular el ángulo*

$x_c$ è la coordinata centrale sull'asse x del riquadro di delimitazione dell'oggetto in esame.

$y^*$ è la coordinata che rappresenta il valore più basso sull'asse y, cioè il fondo della scatola.

Una volta calcolato l'angolo, bisogna tener conto che la telecamera di sistema ha una vista orizzontale di 62,2 gradi, quindi il risultato ottenuto è proporzionale ai gradi di vista di questa telecamera.

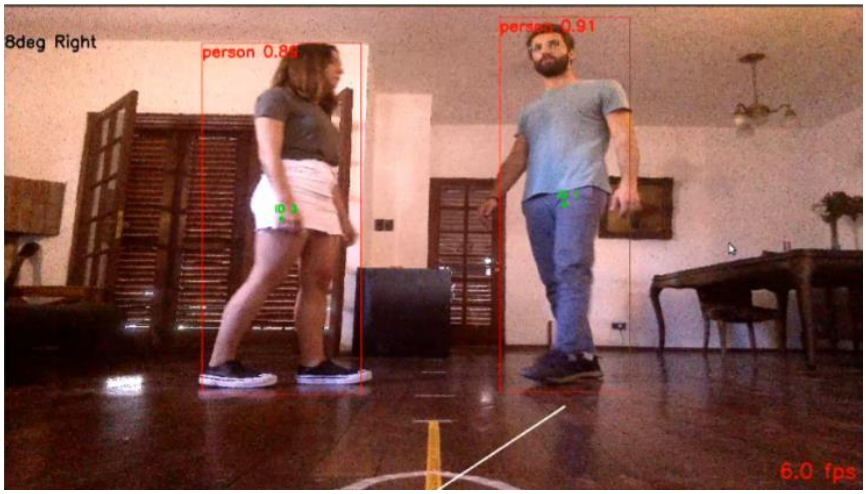


**Figura 5.6:** *Visualizzazione dell angolo e direzione*

## 5.2 Implementazione del Modulo Safe

Il modulo di sicurezza consiste in un insieme di metodi che gestiscono il comportamento del sistema in determinati casi limite. Le funzioni controllano sia la posizione del target e in generale degli oggetti nell'immagine (posizione rispetto agli assi e rispetto agli altri centroidi), sia la dimensione delle loro caselle di delimitazione. Più specificamente, le funzioni controllano:

### 5.2.1 Parametri del Modulo Safe

- ***command*** : parametro che assume un certo valore in base alle funzioni del Safe Module:command=0----> Non è stato rilevato nessun problema
  - command=1----> Stop probabilita di collisione
  - command=2----> Probabilità di scambio nelle seguenti tabelle con il seguente obiettivo
  - command=3----> Obbiettivo perso
  - command=4----> Switch: abort della missione

- ***limit_point_left*** : parametro che imposta un limite sinistro basato sulla dimensione dello schermo in pixel.

- ***limit_point_right*** : parametro che imposta un limite sinistro basato sulla dimensione dello schermo in pixel.

- ***limit_swapping***: parametro che indica la distanza limite in pixel prima che lo scambio abbia luogo

- ***object_swap_possib_id*****:** che indica che un certo ID ha probabilita di swap con un altro oggetto della stessa classe

- ***flag_swap_poss***: che se impostato a 1 indica che c'è un probabile swap.

- ***abort_counter*** : contatore per interrompere la missione

- ***abort_limit***: parametro per indicare l'aborto di missione

- ***limit_point_bottom***: limite inferiore per il controllo della collisione

- **Controllo del tempo di perdita dell'obiettivo.**

Un obiettivo, essendo un centroide, dopo un numero di secondi che può essere impostato, è stato perso. Questo porta alla perdita dell'oggetto target e quindi al riavvio della ricerca di un nuovo target.

- **Controllo della distanza tra l'oggetto e la telecamera**

La distanza limite tra la telecamera e l'oggetto è calcolata utilizzando la sua posizione rispetto all'asse e impostando un limite di prossimità accettabile per evitare una collisione. L'applicazione di questa soluzione è utilizzata sia per capire se c'è una possibilità di collisione con un oggetto durante l'inseguimento del bersaglio, sia per definire se l eventuale rover ha raggiunto il bersaglio.

- **Controllo di un possibile *Identity Switches***

Il caso di collisione tra due centroidi, a livello pratico, può essere visto secondo questa serie di eventi:

i. I due centroidi x,y sono progressivamente avvicinati fino a raggiungere una distanza minima selezionata con la variabile *limit_swapping*.

ii. Una volta superata questa distanza minima, i due oggetti si sovrappongono e solo un nuovo oggetto con il suo centro sarà rilevato, e quindi uno dei due ID sarà in "sospeso" per n frames.

iii. Dopo n frame, l'assegnazione degli ID sarà randomizzata.

## 5.3 Implementazione del Modulo di Object Detection

L’implementazione del modulo di object detection è stato pensato come punto nevralgico del sistema.

La gestione di tutti gli altri moduli è affidata a questo modulo, a livello comportamentale è essenzialmente una routine senza fine la quale viene interrotta da diversi interrupt che arrivano da tutti gli altri moduli.

Chiaramente viene sempre data la precedenza ai codici di errori che arrivano dal Modulo di Safe, per non mettere in situazioni critiche l'eventuale rover.

### 5.3.1 Parametri del Modulo Object Detection

- **Rete neurale:** tutti i parametri di gestione e dell'inizializzazione della rete neurale sono affidati al modulo di Object Detection.
- **Inizializzazione telecamera:** vengono gestiti in tutti i parametri per la gestione di una corretta qualità dell'immagine.
- **END_OF_THE_MISSION:** in base ad un determinato codice che arriva dal modulo di Safe questo flag viene settato
- **error_message:** parametro che cattura i codici di errore provenienti da tutti gli altri moduli, in base al codice si decide cosa fare.
- **counter_message:** con questo valore si esaminano gli errori provenienti dal modulo di Follow Me.

### 5.3.2 Funzioni del Modulo Object Detection

- **Controllo e gestione di tutti gli altri moduli**

Come già spiegato in precedenza a questo modulo viene affidata la gestione di tutti gli altri essendo la connessione principale con tutti gli altri moduli. Vengono quindi gestiti tutti i codici di interrupt provenienti dagli altri moduli come per esempio quelli del Modulo di Safe, tutti gli errori del modulo di Follow Me e in base al valore di questi parametri vengono prese delle decisioni, come per esempio dire al rover di fermarsi se il codice restituito dal Modulo di Safe della check collision sia True.

- **Parametrizzazione dello spazio**

Viene gestita la parametrizzazione dello spazio in pixel per dare una giusta visione dell'ambiente al rover. Viene quindi calcolato il giusto angolo da dare in output al rover e tutta la metrica che riguarda su quale sia l'oggetto più lontano o più vicino al centro di visione della telecamera.

- **Gestione della rete neurale**

Il modulo prende in esame tutti i parametri della rete neurale, prendendo tutto quello che restituisce come input(compresa la gestione degli errori) e la sua visualizzazione in real-time.

# 6

# Test

Una volta che i vari miglioramenti introdotti sono stati spiegati in dettaglio, così come gli obiettivi da raggiungere, si dovrebbe effettuare un processo di test per determinare se questi obiettivi sono stati raggiunti. In questo modo, si potrebbe determinare quali sono i limiti e quale lavoro futuro dovrebbe essere fatto per migliorarli.

## 6.1 Sperimentazione delle funzionalità statiche

Per valutare le prestazioni della rete nei casi di rilevamento statico, vengono testati i casi d'uso Most Central Mode e Farthest Mode, determinando il corretto svolgimento dei vari casi d'uso.
Si noti che il software di registrazione dello schermo abbassa ulteriormente il frame-rate di tre fotogrammi al secondo.

### 6.1.1 *Most Central Mode*

**Tabella 6.1:** Test della Most Central Mode

| |
|---|
| **Titolo**: Esperiemto Most Central Mode |
| **Descrizione**: Selezione da parte della rete dell'oggetto "persona" più vicino al centro dell'immagine. I valori dei messaggi di ritorno sono valutati in caso di presenza e assenza di oggetti. Infine, viene controllato il corretto funzionamento del controllo di collisione del bersaglio. |

| | |
|---|---|
| **Nome del video:** Static functionalities testing<br>**Secondi:** 0:08 - 0:48<br>**Link:**https://www.youtube.com/watch?v=GQC6uDjVuXY&ab_channel=ProyectoIntegradorCali%C3%B2-Forese | |
| **Luogo**: salone<br>**Illuminazione**: artificiale e naturale | **Posizione della telecamera**: 27 cm sopra il suolo<br>**Angolo della visione**: 62.2 x 48.8 deg<br>**Posizione dell'oggetto rispetto alla telecamera**: tra 0.8 e 4 metri |
| **Limite inferiore (collisione)**: 700px (1 metro dalla telecamera) | **Numero di oggetti**: 2<br>**Classe di oggetti**: persone |

Dal test si può vedere che l'esecuzione dell'esperimento è corretta: la rete sceglie la persona più centrale e la segue fino a raggiungerla. In alto a sinistra ci sono i valori di output (10 e 2 gradi) che indicano quanti gradi il sistema dovrebbe ruotare per centrare l'oggetto. La figura 6.1 mostra una sequenza di tre fotogrammi non consecutivi.

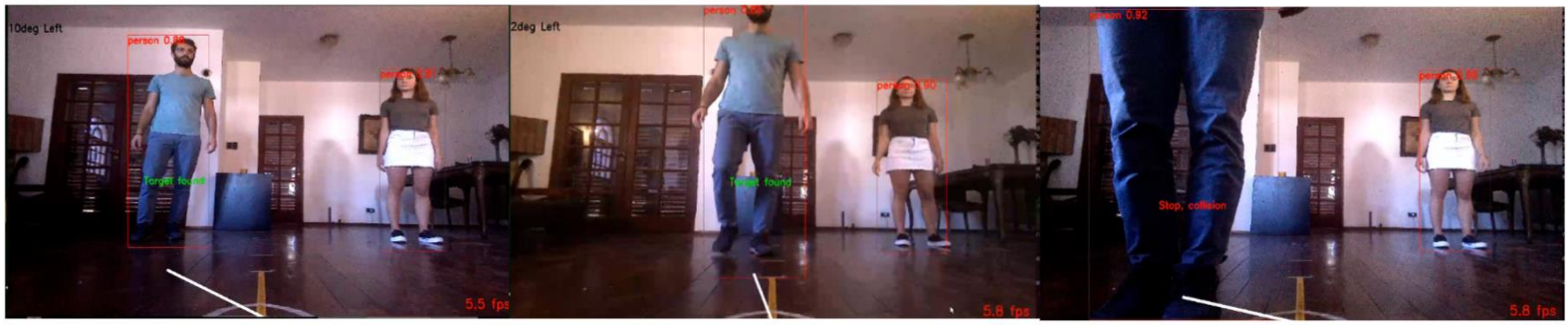


**Figura 6.1:** *Test della Most Central Mode*

## 6.1.2 *Farthest Mode* (single object)

**Tabella 6.2:** Esperimento Farthest Mode

| |
|---|
| **Titolo**: Esperimento Farthest Mode |
| **Descrizione**: Selezione, da parte della rete, dell'oggetto "persona" più lontano dal centro dell'immagine. I valori dei messaggi di ritorno sono valutati in caso di presenza e assenza di oggetti. Infine, il controllo di collisione del bersaglio è controllato per il corretto funzionamento. |
| **Nome del video:** Static functionalities testing<br>**Secondi:** 0:49 - 1:17<br>**Link:**https://www.youtube.com/watch?v=GQC6uDjVuXY&ab_channel=ProyectoIntegradorCali%C3%B2-Forese |

| | |
|---|---|
| **Luogo**: salone<br>**Illuminazione**: artificiale e naturale | **Posizione della telecamera**: 27 cm sopra il suolo<br>**Angolo di visione**: 62.2 x 48.8 deg<br>**Posizione dell'oggetto rispetto alla telecamera**: tra 0.8 e 4 metri |
| **Limite inferiore (collisione)**: 700px (1 metro dalla telecamera) | **Numero di oggetti**: 2<br>**Classe di oggetti**: persona |

In questa modalità d'uso, il sistema seleziona fotogramma per fotogramma il bersaglio più lontano dall'origine dell'asse. Nella prima immagine della figura 6.2 si può vedere come si raggiunge un frame-rate di 5.1, che è basso ma ancora sufficiente per gli scopi dell'esperimento. Quando la soglia impostata viene superata a un metro di distanza, la rete segnala correttamente il messaggio in output "Stop,Collision".

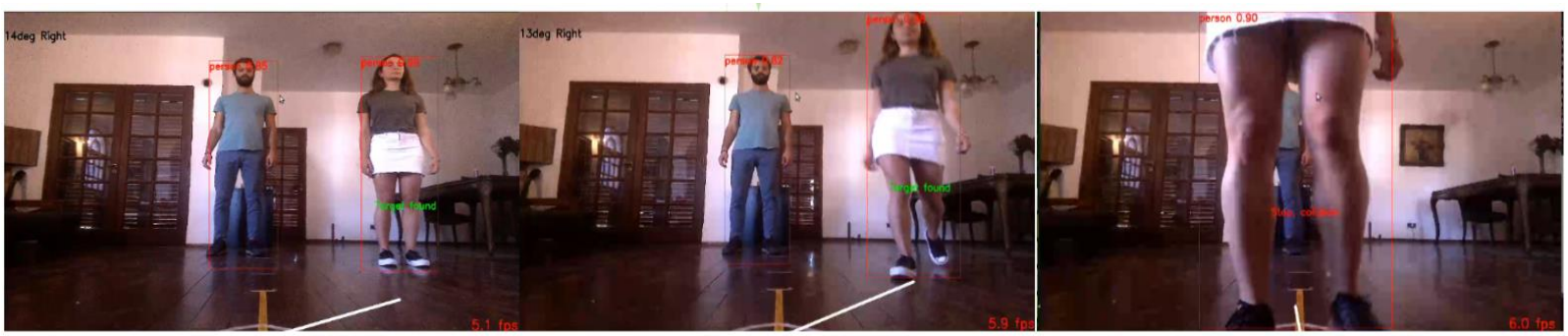


**Figura 6.2:** *Esperimento Farthest Mode*

## 6.2 Test del modulo di Safe

### 6.2.1 *Target Lost*

**Tabella 6.4:** Esperimento Safe (Target Lost)

| |
|---|
| **Título**: Experimento Safe (Target Lost) |
| **Descrizione**: Frame dopo frame, controllo della possibilità di perdita dell'obiettivo. Viene valutata la correttezza dei valori di output. |
| **Nome del video:** Follow Me Testing |

| | |
|---|---|
| **Secondi:** 1:05 - 1:26<br>**Link:**https://www.youtube.com/watch?v=-i_stxDzrNo&ab_channel=ProyectoIntegradorCali%C3%B2-Forese | |
| **Luogo**: salón<br>**Iluminazione**: artificial | **Posizione della camera**: 27 cm sopra il suolo<br>**Angolo di visione**: 62.2 x 48.8 deg<br>**Posizione dell'oggetto rispetto alla telecamera**: entre 1.5 y 2.5 metros |
| **max_reset_counter**:<br>4 frames<br>**max_disapp_frames**:<br>30 frames | **Numero di oggetti**: 2<br>**Classe di oggetto**: persona |

L'esperimento mostra che il controllo della perdita dell'obiettivo è corretto. Il timer di scadenza dell'identificazione del bersaglio è impostato su 30 frames, in modo che dopo 5-6 secondi l'identificazione del bersaglio sia cancellata e la funzione di inseguimento del target termini.


**Figura 6.4:** *Esperimento Safe (Target Lost)*

## 6.3 Migliori parametri per l'algoritmo basati sulla sperimentazione, i casi d'uso e i requisiti funzionali

Sono elencati in dettaglio tutti i migliori parametri del sistema che hanno permesso di raggiungere una condizione di stabilità accettabile e una buona tolleranza all'errore.

- **threshold =** 0.7

Dopo diversi test è stato deciso di impostare la soglia della rete a 0,7 in modo che vengano restituiti solo gli oggetti con una probabilità superiore al 70%.
In questo modo, gli oggetti a bassa probabilità non vengono restituiti e questo assicura che il sistema associato non si confonda.

- **limit_left =** 300, **limit_right** = 1000, **limit_bottom** = 700

Grazie a queste distanze di pixel, se il sistema rileva un oggetto che si trova a meno di 1 metro di distanza, si ferma e avverte l'utente che ci può essere una possibilità di collisione.

- **limit_swap** = 200

Impostando il limite di distanza per lo scambio a 200 pixel, lo scambio può essere riconosciuto con un margine di errore del 3%.

- **abort_limit_count** = 8

Impostando il contatore de abort mission per lo scambio a un secondo si può avere un'ottima reazione al problema dello swap.

Come mostrato nei test e nella tabella, entrambi i requisiti funzionali e non funzionali sono stati soddisfatti, così come gli obiettivi fissati all'inizio del progetto.

**Tabella 6.5:** Matrice di tracciabilità

| RF ID | Descrizione del RF | ID del Test | Status |
|---|---|---|---|
| **RF01** | Il sistema deve essere in grado di rilevare diversi tipi di oggetti | 7.1.1 | Passato |
| **RF02** | Il sistema deve parametrizzare lo spazio di visualizzazione per assegnare una posizione all'oggetto. | 7.2.1, 7.2.2.1, 7.2.2.2 | Passato |
| **RF03** | Il sistema deve selezionare e tracciare staticamente un oggetto target. | 7.1.1, 7.1.2 | Passato |
| **RF04** | Il sistema deve essere in grado di selezionare solo un tipo di classe di oggetto | 7.2.1 | Passato |
| **RF05** | Il sistema deve fornire un'interfaccia per la comunicazione con l'AI. | Tutti i test | Passato |
| **RF06** | Il sistema dovrà assegnare diversi ID per identificare oggetti della stessa classe | 7.2.1, 7.2.2.1, 7.2.2.2 | Passato |
| **RF07** | Il sistema dovrà selezionare e seguire dinamicamente l'oggetto di destinazione | Aldo's test | Passato |
| **RF08** | Azione: raggiungere un oggetto fermo o in movimento | Tutti i test | Passato |
| **RF09** | Permettere a una persona di attivare e disattivare la funzione di tracciamento con un movimento specifico del corpo. | Aldo's test | Passato |
| **RF12** | Permettere la selezione tra tracciamento statico e dinamico | Aldo's test | Passato |
| **RF13** | Applicare il controllo di sicurezza per i casi limite nel tracciamento | 7.1.1, 7.1.2, 7.2.2.1, 7.2.2.2, | Passato |

### 6.3.1 Parametri limiti per i requisiti non funzionali

- **Capacità di rilevare gli oggetti:**

  ❖ L'IA può rilevare perfettamente gli oggetti fino a 3 metri di distanza dalla telecamera.

  ❖ Se gli oggetti sono a piu di 3 metri è in grado di rilevare il movimento ogni 8 fotogrammi in media.

  ❖ Oltre i 4 metri si può vedere dalle prove che il sistema rileva il movimento in media ogni 12 fotogrammi, però se il bersaglio rimane fermo si perde.

- **Bassa latenza tra la produzione dell'immagine e l'uscita dell'algoritmo:**

Questo requisito non funzionale è soddisfatto grazie alla scelta dell'SSD Mobilenet v2, che raggiunge 8fps garantendo un'applicazione in tempo reale del sistema.

I test e la spiegazione del progetto si trovano in questo canale:

https://www.youtube.com/channel/UCU2Crx70ouY-IAGcFeVRv6g?view_as=subscriber

# 7

# Conclusione

All'inizio del progetto è stato proposto di implementare attraverso una rete neurale un algoritmo di Intelligenza artificiale capace di riconoscere diversi tipi di oggetti e di tracciarli nell'immagine di una telecamera, in modo da poterli seguire in un contesto reale in tempo reale.
Prima di fissare gli obiettivi, si è pensato alla possibilità di permettere al sistema di distinguere diversi oggetti e seguirne alcuni nel suo "campo visivo", adatto ad essere installato in un sistema di automi fisici (come un robot, un veicolo autonomo o un veicolo lunare).
Nel lavoro svolto, sono state testate diverse reti neuronali su una scheda che supporta vari componenti necessari per sviluppare algoritmi di apprendimento automatico (NVIDIA Jetson Nano).

I diversi modelli (più specificamente YOLO, nelle sue versioni v3, v4 e Tiny-v4, e SSD, attraverso la rete MobileNet v2), hanno generalmente riscontrato una velocità di elaborazione relativamente bassa calcolata in FPS (fotogrammi al secondo), considerando l'applicazione in contesti reali, come in una macchina.
Tuttavia, considerando l'uso di una scheda come la Jetson-Nano, alcuni dei risultati ottenuti possono essere considerati validi per delle applicazioni considerando un eventuale automa. I test effettuati con YOLO attraverso la sua rete neurale Darknet sono inadeguati perché la rete è troppo pesante per l'architettura e quindi inutile per essere applicata ad un rover, anche se in alcuni casi si è dimostrata sufficientemente precisa.
L'algoritmo Single Shot Detector (SSD), invece, implementato in Python si è dimostrato più veloce, fino ad un massimo di 8 FPS, e in media meno sensibile agli errori come il cambiamento di scala dell'oggetto, l'oggetto sfocato o distorto e i cambiamenti di illuminazione.

Essere in grado di rilevare diversi oggetti abbastanza velocemente e con un errore

sufficientemente basso, e quindi ottenere informazioni come la posizione, il tipo e la dimensione, ha permesso un'ulteriore parametrizzazione dello spazio dell'immagine creando regole di selezione degli oggetti. L'intelligenza è stata in grado di calcolare l'angolo e il modulo della posizione dell'oggetto bersaglio, in modo tale da poter trasmettere informazioni utili a un sistema automatizzato.
Il tutto è stato poi completato dallo sviluppo di un Modulo di Safe che ha permesso all'intelligenza di arrivare a studiare determinati casi limite dell'algoritmo, arrivando così a gestire casi di estrema importanza se pensati ad uno sviluppo fisico in tempo reale, come il riconoscimento di un eventuale collisione, la perdita del target, l'abort della missione e il riconoscimento di un possibile swap tra il bersaglio e un altro oggetto della stessa classe.
Tutti i requisiti funzionali e non funzionali fissati come obiettivo all'inizio del progetto sono stati soddisfatti da test precisi che hanno reso necessario lo studio dei vari parametri utilizzati fino alla migliore configurazione spiegata nella sezione Test.

Infine, sono state sviluppate funzioni per diversi casi d'uso, in modo che un possibile utente possa controllare il comportamento del proprio sistema e possa chiedere che diversi oggetti siano riconosciuti e tracciati in diverse aree dell'immagine della telecamera, in modalità dinamica con memoria, tutto questo aggiungendo funzioni di sicurezza in casi limite dell'algoritmo. I casi d'uso proposti nell'obiettivo del lavoro e implementati in questo progetto, servono come esempio per creare una base per estendere la funzionalità di utilizzo degli utenti e il comportamento del sistema autonomo.

## 7.1 Sviluppi futuri

Il sistema sviluppato serve come base per una serie di miglioramenti in diverse parti del progetto.

Per quanto riguarda lo sviluppo della rete, il prossimo passo sarebbe quello di analizzare e testare algoritmi di tracciamento avanzati, come Mosse, KLK, Camshift, che permettono all'IA di risolvere il problema del passaggio tra ID.
Per questo si consiglia l'uso di Dlib, una libreria creata da Davis King che contiene algoritmi di apprendimento automatico e strumenti che sono utilizzati in una vasta gamma di domini, tra cui la robotica e i dispositivi embedded.

L'idea è che l'automa sia in grado di diventare progressivamente più autonomo e possa quindi soddisfare diverse richieste degli utenti.

Tra i miglioramenti che possono essere aggiunti al progetto, il miglioramento dell'hardware gioca certamente un ruolo importante. L'implementazione dell'intelligenza nel Jatson-Nano è adatta al caso studio, ma dobbiamo pensare al fatto che aggiungendo nuove caratteristiche l'algoritmo perderà velocità, diminuendo notevolmente il valore fps.
Per ottenere un ambiente con sufficiente potenza e, quindi, velocità di calcolo, si consiglia di utilizzare una NVIDIA Jetson TX2, che, rispetto al Nano, ha una GPU con il doppio dei core (256 invece di 128) e il doppio della memoria (8 GB).
Anche una telecamera con una risoluzione migliore migliorerebbe le prestazioni del sistema, soprattutto considerando che alcuni dei problemi nella fase di test sono legati alla scarsa qualità delle immagini.

# Annesso [redacted]

A1

# Loss Function YOLO

La Loss Function è formata da 4 parti:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$

**Formula 1:** *Loss function-YOLO(I)*

La prima equazione calcola la perdita relativa alla posizione prevista del bounding box (x,y). La funzione calcola una somma su ogni predittore di bounding box (j = 0.. B) di ogni cella della matrice (i = 0 .. S^2). $\mathbb{1}$ obj è definito come segue:
● 1, se un oggetto è presente nella cella i della matrice e il predittore del j-esimo riquadro di delimitazione è "responsabile" di tale predizione.

● 0, altrimenti

Dal paper si può prendere questo:
*YOLO predicts multiple bounding boxes per grid cell. At training time we only want one bounding box predictor to be responsible for each object. We assign one predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth.[2]*

Che cos'è l'IoU?

Intersection over Union è una metrica di valutazione utilizzata per misurare la precisione di un rilevatore di oggetti in un dato dataset.

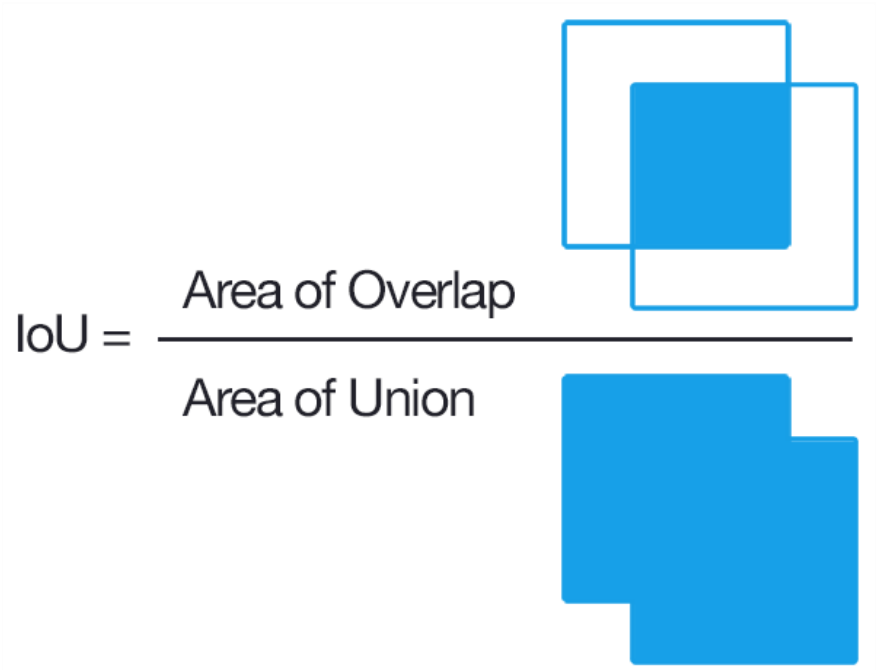


**Figura A.1:** *Spiegazione di IoU*

Tornando alla funzione di perdita, gli altri termini dell'equazione dovrebbero essere facili da capire: (x, y) sono la posizione del bounding box predetto e (x̂, ŷ) sono la posizione effettiva dei dati di allenamento.

Passiamo alla seconda parte:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2$$

**Formula 2:** *Loss Function-YOLO(II)*

Questa è la perdita relativa alla larghezza/altezza della box prevista. L'equazione è simile alla prima, tranne che per la radice quadrata.
Citando di nuovo il paper:
*Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this, we predict the square root of the bounding box width and height instead of the width and height directly.[2]*

Passando alla terza parte:

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2$$

**Formula 3:** *Loss Function-YOLO(III)*

Qua si calcola la perdita associata al punteggio di fiducia per ogni predittore bounding box. C è il punteggio di fiducia e Ĉ è l'intersezione sull'unione del bounding box previsto con quello della verità. $\mathbb{1}$ obj è uguale a uno quando c'è un oggetto nella cella, e 0 altrimenti. $\mathbb{1}$ noobj è il contrario.

I parametri λ elencati qui e anche nella prima parte sono usati per pesare diversamente le parti delle funzioni di perdita. Questo è necessario per aumentare la stabilità del modello. La penalità più alta è per le previsioni di coordinate (λ coord = 5) e la più bassa per le previsioni di confidence quando nessun oggetto è presente (λ noobj = 0,5).

L'ultima parte della funzione di perdita è la perdita di classificazione:

$$\sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

**Formula 4:** *Loss Function-YOLO(IV)*

È simile a un normale errore quadratico per la classificazione, tranne che per il termine $\mathbb{1}$ obj. Questo termine è usato perché non penalizza l'errore di classificazione quando nessun oggetto è presente nella cella (da cui la probabilità condizionata di classe discussa sopra).

# Bibliografia


- [1] Metodo Incrementale https://obsbusiness.school/es/blog-project-management/metodologias-agiles/caracteristicas-y-fases-del-modeloincremental#:~:text=El%20modelo%20incremental%20de%20gesti%C3%B3n,por%20el%20cliente%20o%20destinatario (ultima visita ottobre 2020)
- [2] "YOLO You Look Only Once " https://arxiv.org/pdf/1506.02640.pdf (ultima visita agosto 2020)
- [3] "Overview of YOLO" https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0 (ultima visita agosto 2020)
- [4] "SSD Single Shot Detector" https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab (ultima visita agosto 2020)
- [5] "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" https://arxiv.org/pdf/1704.04861v1.pdf (ultima visita agosto 2020)
- [6] "MobileNet v2" https://arxiv.org/pdf/1801.04381v4.pdf (ultima visita agosto 2020)

- [7]"Jetson Nano Developer Kit" https://developer.nvidia.com/embedded/jetson-nano-developer-kit (ultima visita agosto 2020)
- [8] "Darknet" https://pjreddie.com/darknet/ (ultima visita agosto 2020)
- [9] "CUDA" http://cuda.ce.rit.edu/cuda_overview/cuda_overview.htm (ultima visita settembre 2020)
- [10] "Darknet GitHub" https://github.com/AlexeyAB/darknet (ultima visita agosto 2020)
- [11] "Jetson Inference GitHub" https://github.com/dusty-nv/jetson-inference (ultima visita settembre 2020)
- [12] "Bounding Box" https://d2l.ai/chapter_computer-vision/bounding-box.html (ultima visita septiembre 2020)

- [13] “Comparacion Modules Intel” https://developer.nvidia.com/embedded/jetson-modules (ultima visita ottobre 2020)
- [14] “Comparación entre Jetson Nano y Raspberry Pi3” https://www.maketecheasier.com/nvidia-jetson-nano-vs-raspberry-pi/ (ultima visita ottobre 2020)
- [15]“CPU Performance” https://setiathome.berkeley.edu/cpu_list.php (ultima visita octubre 2020)
- [16]https://towardsdatascience.com/whats-new-in-yolov4-323364bb3ad3 (utlima visita settembre 2020)

# Referenza immagini