



**UNIVERSITA' POLITECNICA DELLE MARCHE**  
**FACOLTA' DI INGEGNERIA DELL'INFORMAZIONE**

---

Corso di Laurea triennale in **Ingegneria Elettronica**

**Studio dell'architettura e delle performance delle reti neurali YOLO e  
FOMO nell'ambito di applicazioni di guida autonoma su sistema  
embedded OpenMV Cam H7 Plus**

**Study of the architecture and performance of YOLO and FOMO neural  
networks in the context of autonomous driving applications on the  
OpenMV Cam H7 Plus embedded system**

Relatore:

**Prof. Laura Falaschetti**

Tesi di Laurea di:

**Francesco Pierucci**

**A.A. 2023 / 2024**

*Un ringraziamento speciale va alla mia famiglia, per il loro sostegno e tutto ciò che hanno fatto per me, anche quando i tempi erano bui ed io avevo perso le speranze, ma loro no. Un ringraziamento va alla mia fidanzata Claudia, che mi sei stata vicina in questo periodo turbolento pieno di salite e grazie poi per essere rimasta con me ad ammirare il panorama che si trovava alla fine. Un grazie anche a tutti i miei amici che mi sono stati vicino, a tutti quelli con cui ci si ritrovava al sushi o al Moon's e, o con del pesce crudo o con birra sul tavolo, ci aggiornavamo sulle nostre vite e ci divertivamo come matti. Grazie a tutti, sia gli amici che sono riuscito a frequentare di più, sia quelli con cui il tempo e gli impegni ci è andato contro. Ognuno di voi ha fornito un suo contributo, chi più e chi meno, e questo per me conta davvero tanto. Grazie, vi voglio bene.*

Ancona, Luglio 2024

Francesco Pierucci

# Indice

|  |           |
|--|-----------|
| <b>Elenco delle figure</b>                           | <b>3</b>  |
| <b>INTRODUZIONE</b>                                  | <b>5</b>  |
| <b>RETI NEURALI ARTIFICIALI: LE BASI</b>             | <b>9</b>  |
| 1.1 Rete neurale artificiale                         | 9         |
| 1.1.1 Il neurone artificiale                         | 9         |
| 1.1.2 Funzione di attivazione                        | 11        |
| 1.1.3 L'architettura di una rete neurale artificiale | 12        |
| 1.1.4 Apprendimento di una rete neurale              | 15        |
| 1.1.4.1 Funzione di costo                            | 15        |
| 1.1.4.2 Gradient descent                             | 16        |
| 1.1.4.3 Learning rate                                | 17        |
| 1.1.5 Convolutional Neural Network (CNN)             | 18        |
| 1.2 Applicazione in computer vision                  | 20        |
| <b>RETE NEURALE YOLO</b>                             | <b>22</b> |
| 2.1 Architettura di YOLO                             | 22        |
| 2.2 Funzionamento di YOLO                            | 24        |
| 2.3 Vantaggi di YOLO                                 | 25        |
| 2.4 Limiti di YOLO                                   | 25        |
| 2.5 Applicazioni                                     | 26        |
| 2.5.1 Veicoli Autonomi                               | 26        |
| 2.5.2 Videosorveglianza                              | 27        |
| 2.5.3 Robotica                                       | 27        |

|   |    |
|---|----|
| <b>RETE NEURALE FOMO</b>                                    | 29 |
| 3.1 Architettura e funzionamento di FOMO                    | 29 |
| 3.2 Vantaggi di FOMO  | 30 |
| 3.3 Limiti di FOMO  | 31 |
| <br>  |    |
| <b>CONFRONTO YOLO E FOMO SU EDGE IMPULSE</b>                | 32 |
| 4.1 Realizzazione reti neurali YOLO e FOMO con Edge Impulse | 32 |
| 4.2 Conclusioni   | 39 |
| <br>  |    |
| <b>Glossario</b>  | 41 |
| <br>  |    |
| <b>Bibliografia</b>   | 45 |

# Elenco delle figure

|   |    |
|---|----|
| 1.1 Schematizzazione di un Percettrone, come descritto da F. Rosenblatt.....  | 9  |
| 1.2 In grafico si mostrano le funzioni tanh (in verde) e sigmoide (in blu).....   | 12 |
| 1.3 Schematizzazione di una semplice rete neurale con due strati nascosti.....  | 13 |
| 1.4 Schema di una rete neurale con uno strato nascosto ricorrente a se stesso e allo strato di output.....  | 14 |
| 1.5 Grafici che paragonano un learning rate basso e uno alto, causando rispettivamente undershooting e overshooting.....  | 17 |
| 1.6 Schematizzazione generale del funzionamento di una rete CNN.....  | 19 |
| 1.7 Confronto tra gli output prodotti per le tecniche di: object recognition, semantic segmentation, object detection, instance segmentation.....   | 21 |
| 2.1 L'illustrazione mostra come YOLO divide l'immagine in una griglia, con ogni cella che prevede bounding boxes e le rispettive probabilità di classe. Le frecce indicano il flusso dai dati di input attraverso i livelli convoluzionali fino all'output con i bounding boxes sugli oggetti rilevati..... | 23 |
| 4.2 Tipologie di reti YOLO supportate dal tool Edge Impulse: una ideata dalla Renesas, una dalla Texas Instruments e quella che è stata creata dalla community di Edge Impulse che è stata scelta per questo progetto.....  | 35 |
| 4.3 Errore mostrato dal terminale del tool Edge Impulse.....  | 36 |
| 4.4 Nuove feature dopo aver impostato il formato RGB.....   | 37 |
| 4.5 Tabella mostrante i risultati del training in formato RGB.....  | 38 |
| 4.6 Tabella che raffigura i risultati pessimi del testing.....  | 38 |
| 4.7 Risultano discrete prestazioni qualora la rete neurale fornita dal tool, riuscisse ad implementare le elaborazioni di immagini e video in scala di grigi.....   | 40 |

## INTRODUZIONE

L'intelligenza artificiale sta cambiando in maniera evidente il modo di vivere la nostra vita quotidiana e lavorativa, sollevando molti vantaggi collegati a un miglior controllo, sicurezza ed efficienza di produzioni industriali, ma anche questioni etiche legate all'argomento. Nei capitoli a seguire si affronterà l'enorme potenziale del machine learning e degli enormi vantaggi associati al suo utilizzo. In particolare si affronterà l'ambito del machine learning applicato al mondo automobilistico, finalizzato alla guida autonoma e al rilevamento di pedoni e ostacoli.

Uno degli ambiti più affascinanti in cui il machine learning sta facendo la differenza è la guida autonoma. Sistemi avanzati, come quelli sviluppati da Tesla, Waymo e altre aziende leader, utilizzano algoritmi di machine learning per analizzare enormi quantità di dati provenienti da sensori, telecamere e radar montati sui veicoli. Questi dati vengono utilizzati per "insegnare" alle auto come riconoscere ostacoli, segnaletica stradale, pedoni e altri veicoli, permettendo loro di prendere decisioni in tempo reale e migliorare continuamente le loro capacità di guida.

La rete neurale **YOLO** ("You Only Look Once") e **FOMO** ("Faster Objects, More Objects") stanno emergendo come due delle tecnologie più avanzate e cruciali nel campo della rilevazione di pedoni e ostacoli, con un impatto

significativo su vari settori, in particolare su quello automobilistico. Queste innovative architetture di reti neurali convoluzionali sono progettate per effettuare il riconoscimento di oggetti in tempo reale.

Una delle applicazioni più critiche di YOLO e FOMO è nella sicurezza dei veicoli autonomi. La capacità di rilevare pedoni, veicoli e altri ostacoli in tempo reale è fondamentale per prevenire incidenti e garantire la sicurezza su strada. YOLO e FOMO eccellono in questo ambito grazie alla loro velocità e accuratezza. Essendo in grado di processare immagini ad alta velocità, permettono ai veicoli di reagire rapidamente ai cambiamenti nell'ambiente circostante, riducendo il rischio di collisioni.

Le tecnologie YOLO e FOMO sono anche integrate in dispositivi portatili e applicazioni mobili, ampliando le possibilità di utilizzo in diversi contesti quotidiani. Per esempio, nei dispositivi di assistenza alla guida, YOLO e FOMO possono fornire avvisi in tempo reale sui pericoli imminenti, migliorando la consapevolezza del conducente e riducendo il rischio di incidenti.

## Capitolo primo

### RETI NEURALI ARTIFICIALI: LE BASI

#### 1.1 Rete neurale artificiale

Prima di esaminare la ricerca svolta (capitolo 4, pag.30), verranno spiegati i componenti e i meccanismi di funzionamento alla base delle reti neurali.

##### 1.1.1 Il neurone artificiale

Il cervello umano è un complesso sistema neurobiologico in grado di elaborare informazioni, composto fondamentalmente da neuroni. I processi elettrochimici che hanno sede all'interno di queste cellule, processano e modulano le informazioni provenienti da altri neuroni per poi inviare nuovi segnali ai neuroni connessi.

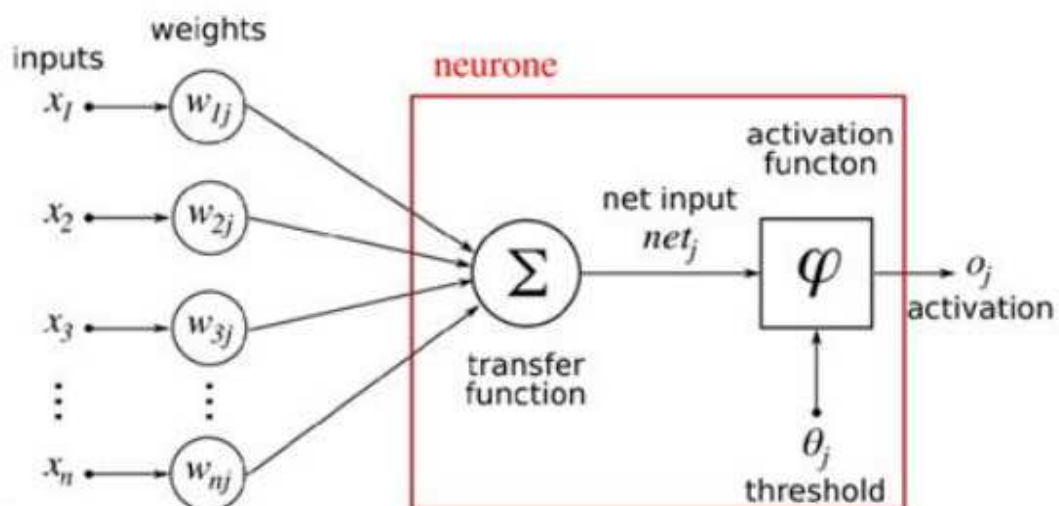


Fig. 1.1: Schematizzazione di un Perceptrone, come descritto da F. Rosenblatt.



L'idea alla base di tutto è di simulare direttamente sul computer il funzionamento del cervello, quindi di realizzare dei neuroni artificiali in grado da essere eseguiti da un calcolatore.

Nel 1958, Rosenblatt propose il “perceptrone”, schematizzato in figura 1.1, un neurone artificiale a soglia a singola uscita. Il perceptrone è un tipo di classificatore binario che associa un insieme di input ad un output scalare  $y$  (di tipo reale) che viene calcolato come:

$$y = \theta\left(\sum_{i=1}^n w_i x_i - b\right)$$

Questo neurone artificiale è capace di fare una sua previsione  $y$ , dove

$\sum_{i=1}^n w_i x_i - b$  è la somma pesata delle corrispondenti  $x_i$  del vettore degli input,

dove il peso (o weight) è definito da  $w_i$  e  $\theta$  rappresenta una funzione chiamata funzione di attivazione. Il valore  $b$ , detto bias, permette di far slittare la funzione di attivazione a destra o a sinistra per interpretare al meglio i dati in input.

Questa fase appena descritta, che consiste nella propagazione delle informazioni attraverso la rete, dall'input nel primo strato fino a ottenere un output, ovvero la previsione, si chiama *forward propagation*.

### 1.1.2 Funzione di attivazione $\theta$

Uno degli obiettivi principali delle funzioni di attivazione è *introdurre la non linearità* all'interno della rete. Operazioni lineari come l'addizione e la moltiplicazione possono modellare solo relazioni lineari. Tuttavia, molti problemi del mondo reale presentano schemi non lineari, e le funzioni di attivazione permettono alla rete di catturare e rappresentare queste relazioni piuttosto intricate. Applicando trasformazioni non lineari ai dati in input, le funzioni di attivazione consentono alla rete di apprendere mappature complesse tra input e output.

Un'altra caratteristica cruciale delle funzioni di attivazione è la loro capacità di *normalizzare* l'output di ciascun neurone. La normalizzazione garantisce che l'output dei neuroni rimanga entro un determinato intervallo, generalmente tra 0 e 1 o tra -1 e 1. Questo aiuta a stabilizzare il processo di apprendimento e previene che l'output dei neuroni diventi troppo grande o troppo piccolo man mano che la rete si approfondisce. Di seguito sono descritte due delle funzioni di attivazione più comuni:

- **Sigmoide**: come illustrato in figura 1.2, questa funzione mappa l'input su un valore compreso tra 0 e 1. È ampiamente utilizzata nei problemi di classificazione binaria, dove l'obiettivo è distinguere tra due classi.
- **Tanh**: mostrata anch'essa in figura 1.2, la funzione tangente iperbolica mappa l'input su un valore compreso tra -1 e 1. È considerata un

miglioramento rispetto alla sigmoide perché è centrata sullo zero, il che facilita l'apprendimento della rete. La funzione Tanh è spesso utilizzata nelle reti neurali ricorrenti (RNN) e nelle reti neurali convoluzionali (CNN).

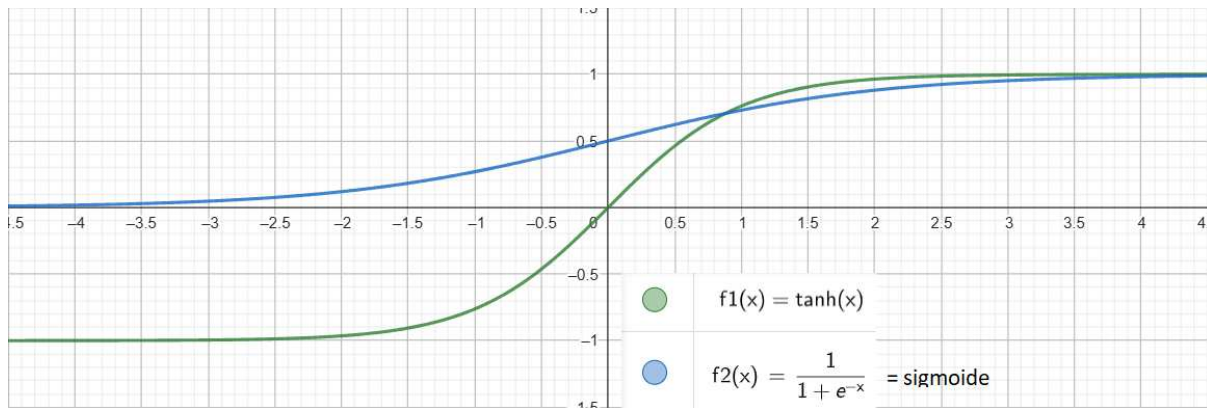


Figura 1.2: In grafico si mostrano le funzioni tanh (in verde) e sigmoide (in blu).

### 1.1.3 L'architettura di una rete neurale artificiale

L'architettura delle reti neurali prende spunto dal modello matematico del perceptrone. In una struttura di base, questa è costituita da una sequenza di strati interconnessi formati da perceptroni. Ogni perceptrone acquisisce il valore determinato dalla sua funzione di attivazione, la quale riceve in input la somma pesata degli input provenienti dai perceptroni dello strato precedente, con pesi determinati dalla connessione che li collega al perceptrone corrente. Questo processo si ripete strato dopo strato, fino a raggiungere quello di output, il quale fornisce informazioni rilevanti riguardo al problema per cui la rete neurale è stata utilizzata. Un esempio di una semplice rete neurale è mostrato nella figura 1.3.

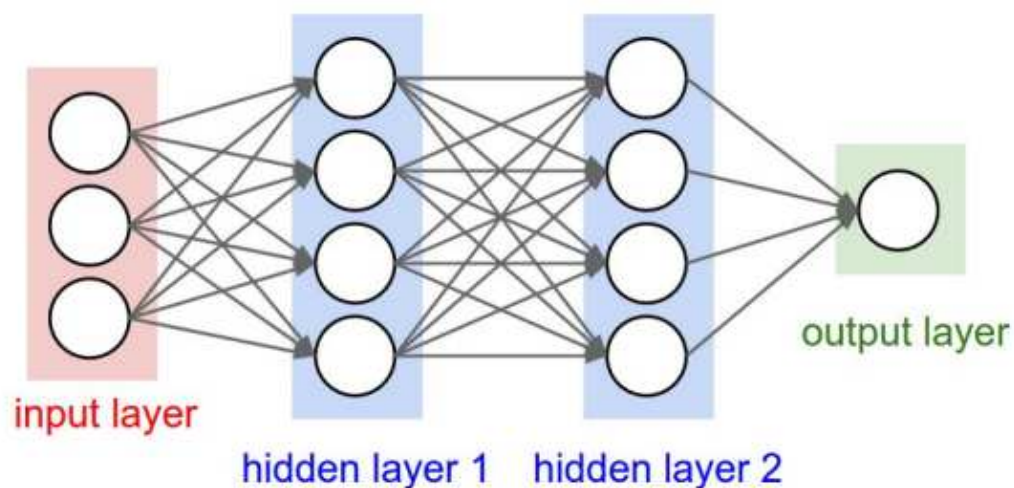


Figura 1.3: Schematizzazione di una semplice rete neurale con due strati nascosti.

Le principali tipologie di reti neurali sono:

- Reti Feed-Forward: la classica rete neurale, rappresentata da vari strati, o layer, completamente interconnessi dal precedente al successivo;
- Reti Ricorrenti (RNN): sono reti che utilizzano connessioni cicliche, dagli strati successivi a quelli precedenti oppure nello stesso strato, per permettere uno stato stabile tra gli input; sono utilizzate principalmente per elaborare dati sequenziali, come in una linea temporale. Un esempio di rete neurale ricorrente viene illustrato in figura 1.4.

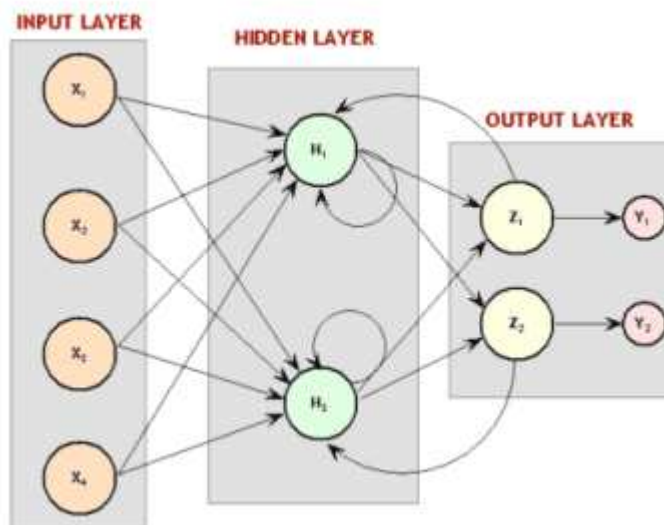


Figura 1.4: Schema di una rete neurale con uno strato nascosto ricorrente a se stesso e allo strato di output.

- Reti convoluzionali (CNN): le reti neurali convoluzionali rappresentano una tecnologia potente e versatile che ha migliorato drasticamente le capacità dei sistemi di intelligenza artificiale sia nell'interpretare che nel

comprendere i dati visivi, aprendo la strada a innovazioni in numerosi settori. Verranno affrontate meglio al paragrafo 1.1.5.

## 1.1.4 Apprendimento di una rete neurale

### 1.1.4.1 Funzione di costo

Per comprendere l'errore generato dalle predizioni di una rete neurale durante la fase di allenamento, si utilizza una formula che calcola la differenza tra il valore effettivo e quello previsto. La formula più comune è la seguente:

$$MSE = \frac{1}{n} \left( \sum_{i=1}^n y_i - \bar{y}_i \right)^2$$

Dove  $n$  rappresenta il numero di campioni da prevedere,  $y_i$  è il valore reale del campione  $i$ -esimo e  $\bar{y}_i$  è il valore previsto per lo stesso campione. Questa formula è nota come errore quadratico medio (“Mean Squared Error”, MSE). Applicando questa formula di errore al modello matematico del perceptrone, definito nella sezione 1.1.1, al posto di  $\bar{y}_i$  si ottiene:

$$Loss(w) = \frac{1}{n} \left[ \sum_{i=1}^n y_i - \theta \left( \sum_{i=1}^n w_i x_i - b \right) \right]^2$$

Questa formula viene chiamata anche funzione di costo (o di perdita) e definisce l'errore di perdita, il cui obiettivo è trovare il valore ottimale di  $w$  che minimizzi

la  $Loss(w)$ . Questo processo viene eseguito utilizzando l'algoritmo della discesa del gradiente (*gradient descent*).

#### 1.1.4.2 Gradient descent

La discesa del gradiente (*gradient descent*) è un algoritmo di ottimizzazione utilizzato in machine learning per minimizzare una funzione di costo. Questo processo è fondamentale per l'addestramento di modelli di machine learning, come le reti neurali.

Si parte con un insieme iniziale di pesi, che può essere scelto casualmente. Si calcola l'errore utilizzando la funzione  $Loss(w)$ .

Di seguito, si calcola il gradiente della funzione di costo rispetto ai parametri del modello.

I parametri vengono aggiornati nella direzione opposta al gradiente. Questo significa che ci si muove nella direzione che riduce la funzione  $Loss(w)$ , ovvero dove l'errore diminuisce. L'aggiornamento viene effettuato usando la formula:

$$w_i = w_{i-1} - \alpha \nabla Loss(w_i)$$

dove  $w_{i-1}$  rappresenta il peso precedente,  $\alpha$  è il learning rate (o tasso di apprendimento, un piccolo valore scalare che determina la dimensione del passo) e  $\nabla Loss(w_i)$  è il gradiente della funzione di perdita rispetto ai pesi.

Questa fase di aggiornamento dei pesi si chiama ***backpropagation***, in quanto prevede il confronto tra l'output predetto e quello reale, propagando l'errore

all'indietro attraverso ogni layer della rete. Questo processo viene iterato molte volte e consente, così, di aggiustare i pesi, migliorando progressivamente la predizione della rete fino a ottenere l'output desiderato.

### 1.1.4.3 Learning rate

Il learning rate viene utilizzato per regolare la velocità di apprendimento della rete in modo da raggiungere i minimi locali di errore, ma come mostrato in figura 1.5, un buon learning rate deve essere scelto in modo da evitare i due principali problemi che ne derivano:

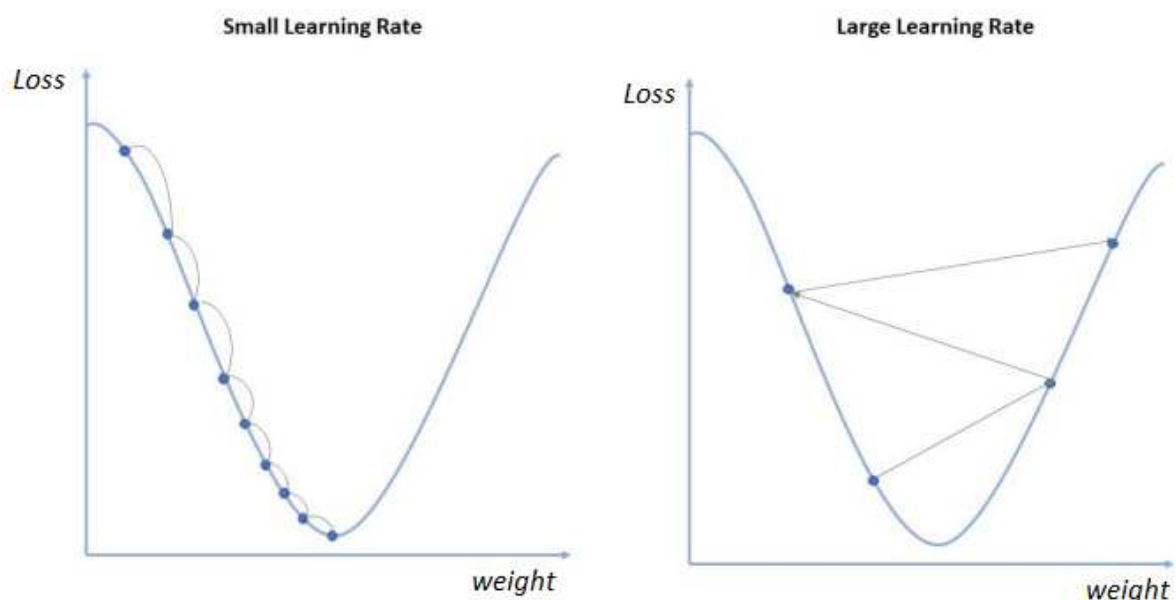


Figura 1.5: Grafici che paragonano un learning rate basso e uno alto, causando rispettivamente undershooting e overshooting.



- *Overshooting*: per un learning rate troppo elevato ne corrisponde una velocità di apprendimento altrettanto elevata, ciò significa che l'aggiornamento dei pesi sarà sempre inadeguato; quindi, dal punto di vista matematico viene saltato il minimo locale.
- *Undershooting*: se invece la velocità di apprendimento è troppo lenta può accadere un rallentamento nell'apprendimento, seguito dal suo stesso blocco nel primo minimo locale trovato.

### **1.1.5 Convolutional Neural Network (CNN)**

Le reti neurali convoluzionali ("Convolutional Neural Networks", CNN) rappresentano una classe avanzata di reti neurali artificiali particolarmente efficaci nell'elaborazione e nell'analisi di dati visivi. Sono ispirate alla struttura e al funzionamento della corteccia visiva degli animali, il che le rende particolarmente adatte per compiti come il riconoscimento di immagini, la classificazione, la segmentazione e l'analisi video.

Queste reti sono costituite da diversi layer che lavorano insieme per estrarre e interpretare le caratteristiche presenti in un'immagine. I principali tipi di strati utilizzati in una CNN sono:

- Strato Convoluzionale: è il cuore di una CNN e utilizza filtri convoluzionali che elaborano l'immagine di input. Ogni filtro è responsabile dell'estrazione di particolari caratteristiche, come bordi, angoli o texture. Il risultato di questa operazione è una mappa delle caratteristiche che evidenzia le parti rilevanti dell'immagine.

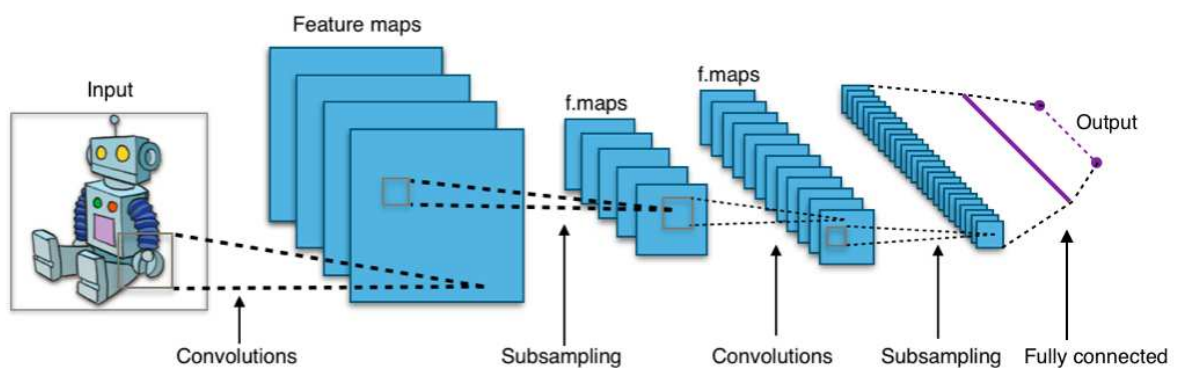


Figura 1.6: Schematizzazione generale del funzionamento di una rete CNN.

- Strato di Pooling: dopo lo strato convoluzionale, viene solitamente inserito uno strato di pooling per ridurre la dimensionalità delle mappe delle caratteristiche. Questo processo riduce la complessità computazionale e aiuta a rendere il modello più resistente alle variazioni e alle distorsioni nell'immagine.
- Strato di Attivazione: le funzioni di attivazione non lineari vengono applicate dopo la convoluzione per introdurre non linearità nel modello.

Questo è essenziale per consentire alla rete di apprendere e rappresentare relazioni complesse nei dati.

- *Strato Completamente Connesso (Fully Connected Layer)*: verso la fine della CNN, gli strati convoluzionali e di pooling sono generalmente seguiti da uno o più strati completamente connessi. In questi strati, ogni neurone è connesso a tutti i neuroni del precedente strato e servono a combinare le caratteristiche estratte in modo da effettuare la classificazione finale.

## **1.2 Applicazione in computer vision**

L'applicazione del machine learning nel campo della computer vision ha dato impulso a numerose aree di ricerca e all'uso di modelli di reti neurali per il riconoscimento di oggetti. Questo ha portato alla produzione di una varietà di output, come mostrato in figura 1.7, che sono generati dalle seguenti tecnologie:

- *Object recognition*: questo è stato il primo metodo di machine learning sviluppato per il riconoscimento di oggetti. Consente di identificare gli oggetti nelle immagini, fornendo come output i nomi delle classi e la probabilità che corrispondano alle classi previste.

- Object detection: evoluzione dell'object recognition, permette di riconoscere diversi oggetti in un'immagine, definendo anche la loro posizione e dimensione tramite bounding boxes.
- Semantic segmentation: il passo successivo nel riconoscimento di oggetti, assegna una classe a ogni pixel dell'immagine.
- Instance segmentation: l'evoluzione della semantic segmentation, non solo assegna una classe a ogni oggetto, ma assicura anche di distinguere i vari oggetti tra loro.

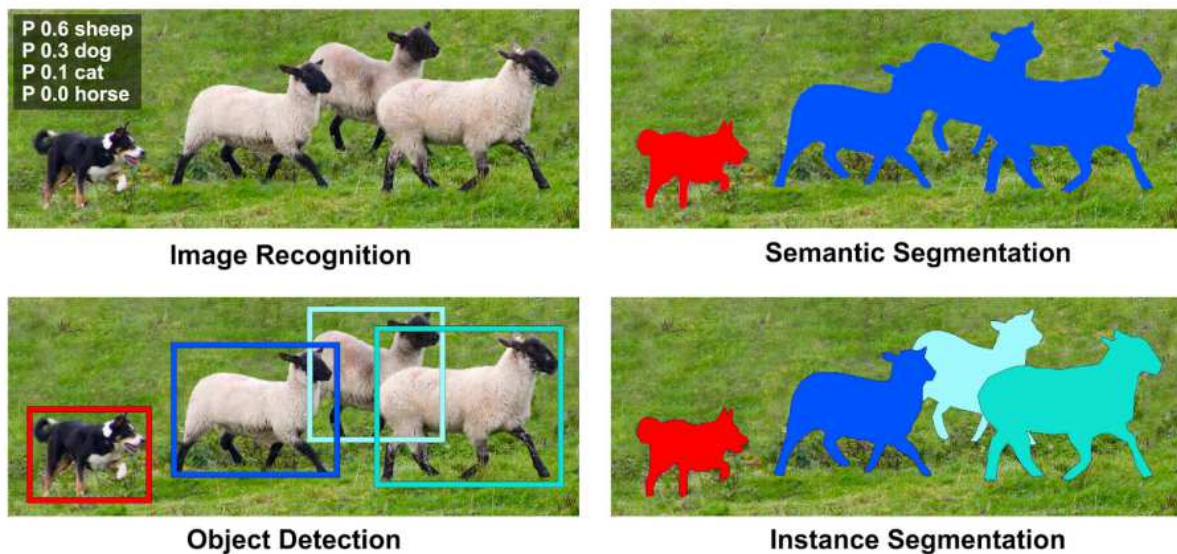


Figura 1.7: Confronto tra gli output prodotti per le tecniche di: object recognition, semantic segmentation, object detection, instance segmentation.

## Capitolo secondo

### RETE NEURALE YOLO

YOLO ("You Only Look Once") è una rete neurale convoluzionale progettata per il rilevamento di oggetti in tempo reale. Sviluppata inizialmente da Joseph Redmon nel 2015, YOLO ha rivoluzionato il campo del rilevamento degli oggetti grazie alla sua velocità e precisione. Contrariamente ad altri approcci che richiedono più fasi di elaborazione, YOLO effettua il rilevamento degli oggetti in una singola passata attraverso la rete neurale, rendendola estremamente efficiente per applicazioni in tempo reale.

#### 2.1 Architettura di YOLO

La rete YOLO è composta da diverse parti chiave che possono essere schematizzate per semplicità in maniera seguente:

- Input Image (Input Layer): l'immagine di input viene ridimensionata a una dimensione fissa (ad esempio, 416x416) per essere elaborata dalla rete.
- Convolutional Layers (Hidden Layers): la rete consiste in numerosi strati convoluzionali che estraggono caratteristiche rilevanti dall'immagine;

questi strati utilizzano filtri convoluzionali per identificare bordi, texture, e altre informazioni di basso livello.

- Bounding Boxes and Class Predictions (Output Layer): l'output finale della rete è una griglia SxS dove ogni cella prevede un certo numero di bounding box e le rispettive probabilità di classe per ciascun box.

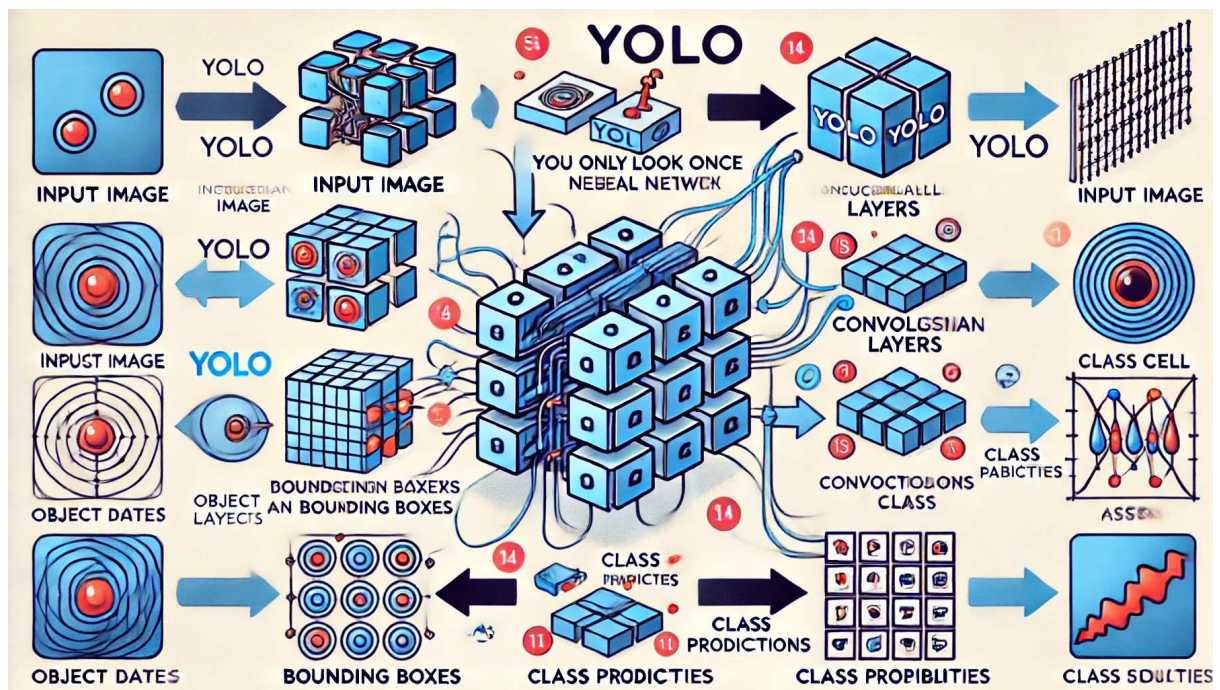


Figura 2.1: L'illustrazione mostra come YOLO divide l'immagine in una griglia, con ogni cella che prevede bounding boxes e le rispettive probabilità di classe. Le frecce indicano il flusso dai dati di input attraverso i livelli convoluzionali fino all'output con i bounding boxes sugli oggetti rilevati.

## 2.2 Funzionamento di YOLO

Come mostrato in figura 2.1, YOLO divide l'immagine di input in una griglia SxS. Ogni cella della griglia è responsabile del rilevamento di oggetti il cui centro ricade all'interno della cella stessa. Ognuna di esse prevede:

- Bounding Boxes: rettangoli che racchiudono gli oggetti rilevati. Ogni bounding box è definito da 5 valori: le coordinate x e y del centro del box, la larghezza e l'altezza del box, e una confidenza che indica la probabilità che il box contenga un oggetto.
- Class Predictions: le probabilità che l'oggetto racchiuso nel bounding box appartenga a ciascuna delle classi considerate.

Inoltre, YOLO utilizza una funzione di costo che combina tre componenti principali:

- Localization Loss: penalizza le differenze tra i box previsti e i box reali in termini di posizione e dimensioni.
- Confidence Loss: penalizza le differenze tra la confidenza prevista e quella reale (1 se il box contiene un oggetto, 0 altrimenti).
- Class Probability Loss: penalizza le differenze tra le probabilità di classe previste e quelle reali.

## 2.3 Vantaggi di YOLO

Se YOLO, ad oggi, rimane una delle reti neurali più promettenti e utilizzate nel campo del machine learning, è dovuto ai suoi vantaggi che non sono proprio da sottovalutare:

- Velocità: grazie alla sua architettura one-pass, YOLO è estremamente veloce e ciò la rende ottima per applicazioni in tempo reale.
- Precisione globale: considerando l'immagine nel suo complesso, YOLO tende a fare meno previsioni false positive rispetto ad altri approcci che si concentrano su regioni locali.
- Generalizzazione: YOLO ha una buona capacità di generalizzare su nuove immagini, rendendola adatta per applicazioni nel mondo reale.

## 2.4 Limiti di YOLO

Come abbiamo visto finora, YOLO è un'ottima rete neurale, ma ha anch'essa delle limitazioni. Infatti, nel suo utilizzo è probabile riscontrare due problematiche principali:

- Rilevazione di Oggetti di Piccole Dimensioni: YOLO può avere difficoltà a rilevare oggetti molto piccoli che occupano una porzione minima dell'immagine.



- Overlapping Objects: il rilevamento di oggetti sovrapposti può essere meno accurato rispetto ad approcci più complessi come quelli basati sulla region proposals.

## **2.5 Applicazioni**

YOLO rappresenta un approccio innovativo al rilevamento degli oggetti, combinando velocità e precisione in un'unica architettura. Nonostante alcuni limiti, la sua capacità di operare in tempo reale la rende una scelta ideale per molte applicazioni pratiche. Per questo motivo, YOLO è utilizzata in una vasta gamma di applicazioni riportate nei seguenti paragrafi.

### **2.5.1 Veicoli Autonomi**

YOLO può essere impiegato nei veicoli autonomi e questo per rilevare diversi oggetti che appaiono sulla strada e classificarli nella categoria a cui appartengono con l'aiuto delle bounding boxes [3], [4], [5], [6], [7], [15]. Questa sua applicazione è ancora un mondo nuovo da scoprire e ci sono vari studi che cercano di migliorare ancora di più la velocità di rilevamento avendo un'ottima precisione [4], [7], lavorando anche su un miglior riconoscimento di piccoli oggetti [4],[7].

### **2.5.2 Videosorveglianza**

YOLO ha anche applicazione nella videosorveglianza, per il monitoraggio di attività sospette e il conteggio delle persone. Si può implementare nel riconoscere pedoni che hanno un'arma in mano [8],[11], con l'intento di ridurre i crimini come rapina e taccheggio. Oppure, se consideriamo una smart city, può servire per il conteggio delle persone [12],[13].

Sono molte le sfide per ottenere degli ottimi risultati, perché, a causa dei loro elevati costi computazionali, è difficile applicare questi metodi su dispositivi edge con risorse limitate per applicazioni in tempo reale. Infatti ci sono vari studi che propongono sistemi innovativi per migliorare le prestazioni [9],[10].

### **2.5.3 Robotica**

L'utilizzo di YOLO in ambito della robotica, contribuisce alla navigazione e l'interazione dei robot con l'ambiente circostante. Infatti si può applicare insieme alla Microsoft Kinect così, dopo aver riconosciuto gli oggetti, si procede ad identificare la distanza dell'oggetto dal robot, in modo da poter creare un algoritmo di controllo per la navigazione di un robot mobile [1].

Oppure applicato al robot umanoide [2], dove, tramite una telecamera, si vuole determinare non solo la classe dell'oggetto, ma anche la sua dimensione. Consentendo così di afferrare l'oggetto di interesse al braccio meccanico, con l'aiuto di una telecamera di profondità. Se invece non ci fosse un braccio

meccanico e una palla come oggetto da individuare e da seguire, allora stiamo parlando di “soccer robot” [14].

## Terzo capitolo

### RETE NEURALE FOMO

FOMO (“Faster Objects, More Objects”) è una rete neurale progettata specificamente per il rilevamento efficiente degli oggetti su dispositivi con risorse limitate. Sviluppata da Edge Impulse, FOMO mira a fornire il rilevamento in tempo reale degli oggetti con requisiti computazionali e di memoria significativamente ridotti rispetto ai modelli tradizionali come YOLO.

#### 3.1 Architettura e funzionamento di FOMO

FOMO modifica l'architettura dei modelli di deep learning per il rilevamento degli oggetti concentrandosi sui centroidi degli oggetti anziché sulle bounding box. Questo approccio prevede:

- Strati convoluzionali (Input Layer): l'immagine di input viene passata attraverso gli strati convoluzionali, che estraggono caratteristiche significative.
- Generazione della Heatmap (Hidden Layers): invece di utilizzare strati completamente connessi per prevedere le bounding box, FOMO genera una heatmap. Questa evidenzia le aree dell'immagine dove gli oggetti

sono presenti, fornendo una mappa delle probabilità per ciascuna regione dell'immagine. In sostanza, in questo modo riscontriamo la posizione anziché le dimensioni degli oggetti.

- Predizione dei Centroidi (Output Layer): la heatmap viene utilizzata per determinare i centroidi degli oggetti presenti nell'immagine. Questa tecnica è meno complessa rispetto alla predizione delle bounding box, riducendo così il carico computazionale e il consumo di memoria.

Inoltre, la FOMO usufruisce di una funzione di costo tutta sua, ovvero una funzione di costo personalizzata. La funzione Loss risultante aiuta a preservare la località nel livello finale della rete, permettendo al modello di prevedere con precisione i centroidi degli oggetti.

### 3.2 Vantaggi di FOMO

L'approccio di FOMO è particolarmente adatto per i dispositivi edge, offrendo diversi vantaggi:

- Efficienza: prevedendo i centroidi degli oggetti, FOMO riduce la complessità e i requisiti di dati del modello. Utilizza fino a 30 volte meno potenza di elaborazione e memoria rispetto a modelli come YOLOv5.
- Rilevamento in Tempo Reale: FOMO può eseguire il rilevamento degli oggetti a frame rate elevati anche su dispositivi a bassa potenza. Ad

esempio, può raggiungere 60 fps su un Raspberry Pi 4 e 30 fps su un Arduino Nicla Vision con memoria limitata.

### 3.3 Limiti di FOMO

L'uso di FOMO presenta diverse limitazioni, soprattutto se confrontato con i modelli tradizionali di rilevamento degli oggetti come la YOLO:

- Assenza di Bounding Box: FOMO non genera bounding box per gli oggetti rilevati, ma si limita a individuare i centroidi. Questo può essere limitante in applicazioni dove la dimensione e la forma degli oggetti sono cruciali.
- Difficoltà con Oggetti Sovrapposti: può avere difficoltà nel rilevare oggetti con centroidi sovrapposti. Ogni cella nella heatmap funziona come un classificatore indipendente, il che rende problematico il rilevamento di oggetti che si sovrappongono nello spazio.
- Precisione Limitata: la precisione di FOMO nel rilevamento degli oggetti potrebbe essere inferiore rispetto ai modelli che utilizzano bounding box, specialmente in scenari complessi dove è necessario un rilevamento preciso e dettagliato degli oggetti.

## Quarto capitolo

### CONFRONTO YOLO E FOMO SU EDGE IMPULSE

#### 4.1 Realizzazione reti neurali YOLO e FOMO con Edge Impulse

Qui tratteremo in dettaglio la realizzazione di una rete neurale di tipo YOLOv5 per il riconoscimento di pedoni tramite immagini fornite dal dataset FLIR. La creazione della rete neurale in questione avviene con l'aiuto del tool Edge Impulse, una piattaforma progettata per facilitare la creazione, l'addestramento e il dispiegamento di modelli di machine learning per dispositivi embedded e Internet of Things (IoT).

Il tool è semplice e intuitivo, il che permette di realizzare varie AI facilitando anche l'apprendimento. Una volta registrati, creato un progetto e datogli un nome, si può passare agli step più importanti per ogni rete neurale:

- Collezionare/importare dati: ovvero costruzione del dataset. Per farlo ho considerato solo 150 immagini dal dataset FLIR con pedoni, autovetture ed entrambi insieme nello stesso scatto. Queste sono in scala di grigi e hanno una risoluzione 640 x 512 pixel.

Una volta caricate tutte sul sito, ho scelto di usare l'80% delle immagini per il training e il 20% per il testing, ovvero rispettivamente 118 e 32, per favorire un apprendimento di maggiore qualità.

Il motivo della scelta di così poche foto, risiede nei limiti della versione gratuita di Edge impulse. Senza la versione enterprise (quella a pagamento), abbiamo solo 20 minuti per allenare la nostra rete. Un dataset di maggiori dimensioni, avrebbe precluso un numero di epoche più elevato per un buon training e quindi un superamento più probabile del limite temporale.

- Etichettare i dati: in ogni foto si “etichetta” il soggetto di nostro interesse, che vogliamo determinare poi con la rete nelle foto dedicate al testing. Etichettiamo quindi i pedoni ed anche le autovetture per avere un confronto fra le feature.
- Pre-processare i dati: prima di creare le feature, si pre-processano i dati conferendo una qualità inferiore. Le foto sono state elaborate fino ad avere una risoluzione con 96x96 pixel, con lo scopo di renderli più accessibili alla CPU del dispositivo preso in considerazione. Infine impostiamo il formato delle foto come scala di grigi.



- Creazione feature: in figura 4.1 vi è il grafico con le feature.

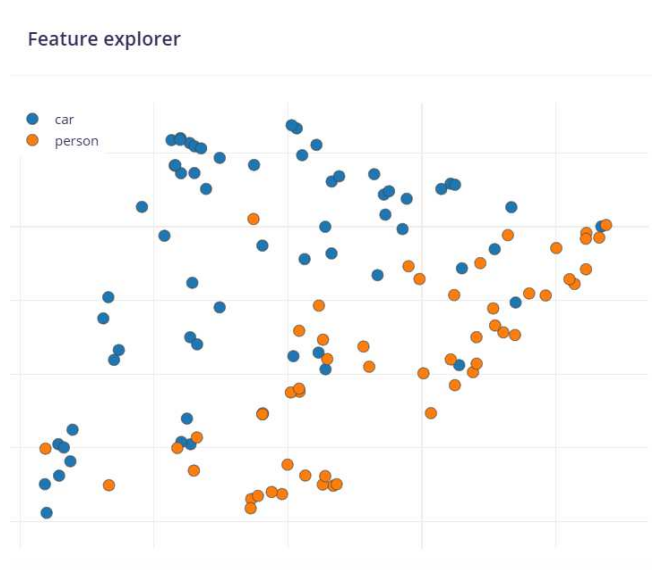


Fig. 4.1 Feature riguardo automobili (blu) e pedoni (arancione).

Step molto importante, visto che determina gran parte delle prestazioni del modello. Nel caso considerato, il risultato non è molto soddisfacente, diciamo sufficiente. Un po' me l'aspettavo, è difficile ottenere risultati migliori con sole 118 immagini e varie casistiche di pedoni/autovetture fotografate vicine, lontane o visibili solo in parte perché oscurate da pali, siepi o alberi.

- Training: scelgo 30 epoche, per completare l'allenamento in tempi più o meno corti, e un learning rate di 0.001. Inoltre, scelgo la quantizzazione intero a 8 bit (o int8), che riduce ulteriormente le dimensioni del modello e i requisiti di calcolo convertendo i numeri a virgola mobile a 32 bit in interi a 8 bit, con lo scopo di ottimizzare i tempi di calcolo. Infine, fra le

possibili reti neurali YOLO, mostrate in figura 4.2, scelgo la YOLOv5 creata dalla community, visto che le altre due sono dedicate esclusivamente a dispositivi ideati da Renesas e Texas Instruments.

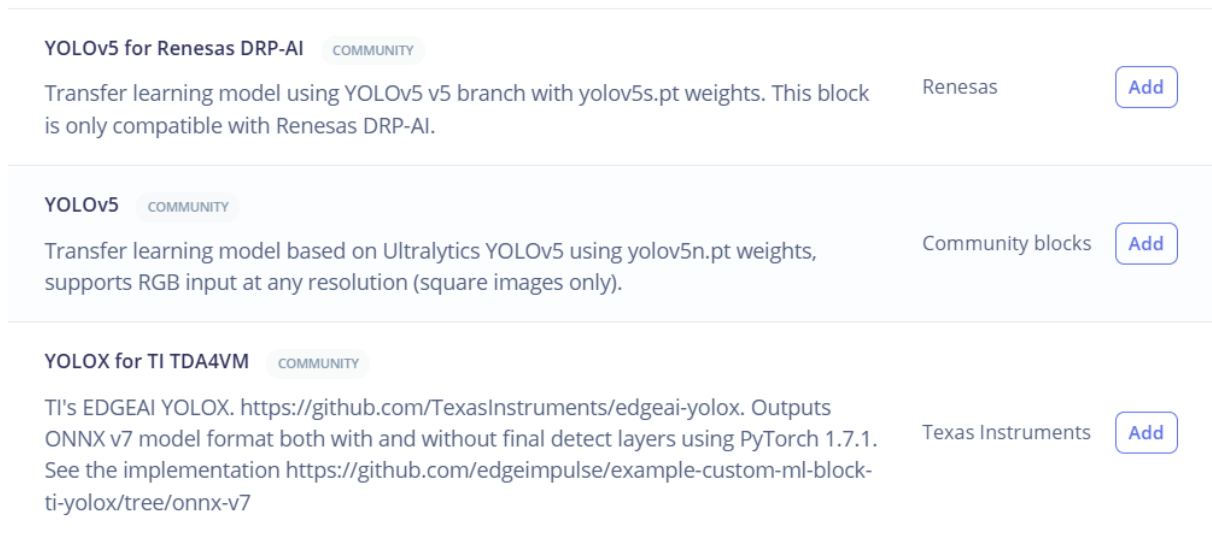


Fig. 4.2 Tipologie di reti YOLO supportate dal tool Edge Impulse: una ideata dalla Renesas, una dalla Texas Instruments e quella che è stata creata dalla community di Edge Impulse che è stata scelta per questo progetto.

A questo punto, inizio l'allenamento che, però, finisce subito a causa di un errore evidenziato dal terminale che riporto qua sotto in figura 4.3.

```
TypeError: Cannot handle this data type: (1, 1, 1), |u1  
Job failed (see above)
```

Fig. 4.3 Errore mostrato dal terminale del tool Edge Impulse.

Mi si accende una lampadina, vado a ricontrollare la rete YOLO che avevo selezionato e, infatti, questa supporta immagini e video solo RGB, che non potrà mai riprodurre una termocamera FLIR. A questo punto, provo comunque a selezionare il formato RGB, aspettandomi quasi sicuramente scarse prestazioni e bassissima precisione. Questo perché in formato RGB, le immagini sono vettori 3D con tre assi dove abbiamo rosso, verde e blu, mentre in formato scala di grigi (o bianco e nero), le immagini sono in 2D con due assi costituiti dal bianco e nero. Ciò significa che togliendo una dimensione, ma provare comunque a elaborare le immagini su tre dimensioni, andrà a discapito della precisione e qualità nel risultato.

Visto che il formato ora è cambiato, prima di tutto è necessario ricalcolare le feature, così da ottenere il grafico in figura 4.4, che sembra quasi lo speculare del precedente.

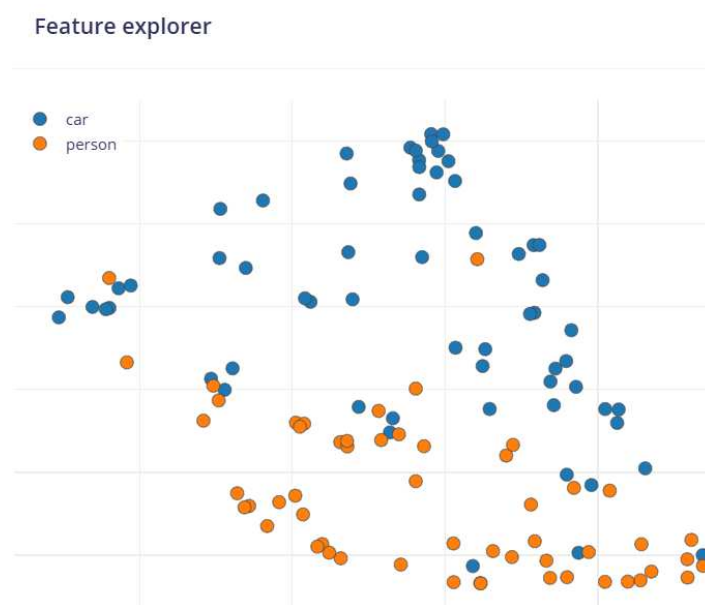




Fig. 4.4 Nuove feature dopo aver impostato il formato RGB.

Il peggio viene una volta effettuato il training, visto i risultati che sono evidenziati in figura 4.5: il massimo a cui possiamo ambire in queste condizioni è una precisione di 9.9%. Insomma, è la rete neurale meno prestante creata.

| <div style="text-align: center;">  <p><b>PRECISION SCORE</b> ⓘ<br/><b>9.9%</b></p> </div> <p><b>Metrics</b> (validation set)</p> <table border="1"> <thead> <tr> <th>METRIC</th> <th>VALUE</th> </tr> </thead> <tbody> <tr><td>mAP ⓘ</td><td>0.00</td></tr> <tr><td>mAP@[IoU=50] ⓘ</td><td>0.01</td></tr> <tr><td>mAP@[IoU=75] ⓘ</td><td>0.00</td></tr> <tr><td>mAP@[area=small] ⓘ</td><td>0.00</td></tr> <tr><td>mAP@[area=medium] ⓘ</td><td>0.01</td></tr> <tr><td>mAP@[area=large] ⓘ</td><td>-1.00</td></tr> <tr><td>Recall@[max_detections=1] ⓘ</td><td>0.00</td></tr> <tr><td>Recall@[max_detections=10] ⓘ</td><td>0.01</td></tr> <tr><td>Recall@[max_detections=100] ⓘ</td><td>0.03</td></tr> <tr><td>Recall@[area=small] ⓘ</td><td>0.02</td></tr> <tr><td>Recall@[area=medium] ⓘ</td><td>0.10</td></tr> <tr><td>Recall@[area=large] ⓘ</td><td>-1.00</td></tr> </tbody> </table> | METRIC   | VALUE | mAP ⓘ | 0.00 | mAP@[IoU=50] ⓘ | 0.01 | mAP@[IoU=75] ⓘ | 0.00 | mAP@[area=small] ⓘ | 0.00 | mAP@[area=medium] ⓘ | 0.01 | mAP@[area=large] ⓘ | -1.00 | Recall@[max_detections=1] ⓘ | 0.00 | Recall@[max_detections=10] ⓘ | 0.01 | Recall@[max_detections=100] ⓘ | 0.03 | Recall@[area=small] ⓘ | 0.02 | Recall@[area=medium] ⓘ | 0.10 | Recall@[area=large] ⓘ | -1.00 | <div style="text-align: center;">  <p><b>ACCURACY</b> ⓘ<br/><b>0.00%</b></p> </div> <p><b>Metrics for Object detection</b></p> <table border="1"> <thead> <tr> <th>METRIC</th> <th>VALUE</th> </tr> </thead> <tbody> <tr><td>mAP ⓘ</td><td>0.00</td></tr> <tr><td>mAP@[IoU=50] ⓘ</td><td>0.00</td></tr> <tr><td>mAP@[IoU=75] ⓘ</td><td>0.00</td></tr> <tr><td>mAP@[area=small] ⓘ</td><td>0.00</td></tr> <tr><td>mAP@[area=medium] ⓘ</td><td>-1.00</td></tr> <tr><td>mAP@[area=large] ⓘ</td><td>-1.00</td></tr> <tr><td>Recall@[max_detections=1] ⓘ</td><td>0.00</td></tr> <tr><td>Recall@[max_detections=10] ⓘ</td><td>0.00</td></tr> <tr><td>Recall@[max_detections=100] ⓘ</td><td>0.00</td></tr> <tr><td>Recall@[area=small] ⓘ</td><td>0.00</td></tr> <tr><td>Recall@[area=medium] ⓘ</td><td>-1.00</td></tr> <tr><td>Recall@[area=large] ⓘ</td><td>-1.00</td></tr> </tbody> </table> | METRIC | VALUE | mAP ⓘ | 0.00 | mAP@[IoU=50] ⓘ | 0.00 | mAP@[IoU=75] ⓘ | 0.00 | mAP@[area=small] ⓘ | 0.00 | mAP@[area=medium] ⓘ | -1.00 | mAP@[area=large] ⓘ | -1.00 | Recall@[max_detections=1] ⓘ | 0.00 | Recall@[max_detections=10] ⓘ | 0.00 | Recall@[max_detections=100] ⓘ | 0.00 | Recall@[area=small] ⓘ | 0.00 | Recall@[area=medium] ⓘ | -1.00 | Recall@[area=large] ⓘ | -1.00 |
|---|--|-------|-------|------|----------------|------|----------------|------|--------------------|------|---------------------|------|--------------------|-------|-----------------------------|------|------------------------------|------|-------------------------------|------|-----------------------|------|------------------------|------|-----------------------|-------|---|--------|-------|-------|------|----------------|------|----------------|------|--------------------|------|---------------------|-------|--------------------|-------|-----------------------------|------|------------------------------|------|-------------------------------|------|-----------------------|------|------------------------|-------|-----------------------|-------|
| METRIC  | VALUE  |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| mAP ⓘ   | 0.00   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| mAP@[IoU=50] ⓘ  | 0.01   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| mAP@[IoU=75] ⓘ  | 0.00   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| mAP@[area=small] ⓘ  | 0.00   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| mAP@[area=medium] ⓘ   | 0.01   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| mAP@[area=large] ⓘ  | -1.00  |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| Recall@[max_detections=1] ⓘ   | 0.00   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| Recall@[max_detections=10] ⓘ  | 0.01   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| Recall@[max_detections=100] ⓘ   | 0.03   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| Recall@[area=small] ⓘ   | 0.02   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| Recall@[area=medium] ⓘ  | 0.10   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| Recall@[area=large] ⓘ   | -1.00  |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| METRIC  | VALUE  |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| mAP ⓘ   | 0.00   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| mAP@[IoU=50] ⓘ  | 0.00   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| mAP@[IoU=75] ⓘ  | 0.00   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| mAP@[area=small] ⓘ  | 0.00   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| mAP@[area=medium] ⓘ   | -1.00  |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| mAP@[area=large] ⓘ  | -1.00  |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| Recall@[max_detections=1] ⓘ   | 0.00   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| Recall@[max_detections=10] ⓘ  | 0.00   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| Recall@[max_detections=100] ⓘ   | 0.00   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| Recall@[area=small] ⓘ   | 0.00   |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| Recall@[area=medium] ⓘ  | -1.00  |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| Recall@[area=large] ⓘ   | -1.00  |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |
| <p>Fig.4.5 Tabella mostrante i risultati del training in formato RGB.</p>   | <p>Fig. 4.6 Tabella che raffigura i risultati pessimi del testing.</p> |       |       |      |                |      |                |      |                    |      |                     |      |                    |       |                             |      |                              |      |                               |      |                       |      |                        |      |                       |       |   |        |       |       |      |                |      |                |      |                    |      |                     |       |                    |       |                             |      |                              |      |                               |      |                       |      |                        |       |                       |       |

- *Testing*: per confermare quanto è inadatta la rete appena ideata, nella tabella mostrata in figura 4.6, mostriamo i risultati ottenuti nel testing, quindi quando l'AI avrebbe dovuto individuare pedoni e autovetture sulle 32 immagini dedicate.

## 4.2 Conclusioni

La rete ideata con l'aiuto del tool Edge Impulse è totalmente inadatta a svolgere il compito di object detection. Di tutta quest'esperienza però, posso anche dire che mi aspetto sicuramente buoni risultati qualora avessi usato una rete YOLO in grado di elaborare ottimamente anche foto e video in scala di grigi. A conferma di ciò, per dimostrare che se si usa una rete che su Edge Impulse supporta il formato scala di grigi si ottengono risultati migliori, utilizzo per scrupolo il servizio EON tuner. Quest'ultimo è uno strumento avanzato per l'ottimizzazione di modelli di machine learning. Infatti questa IA dedicata, andando a tentativi, riesce a consigliare la migliore rete e le migliori impostazioni per il suo utilizzo. Ciò che consiglia, come mostrato in figura 4.7, è l'utilizzo della FOMO che elabora foto e video in bianco e nero.



Fig 4.7 Risultano discrete prestazioni qualora la rete neurale fornita dal tool, riuscisse ad implementare le elaborazioni di immagini e video in scala di grigi.

# Glossario

## B

### *Bias*

Il bias algoritmico, impiegato nelle reti neurali, fornisce un valore predefinito a ciascun livello. Funziona in modo simile a un perceptrone, ma il suo valore è fisso e non dipende dall'output dello strato precedente. Un bias troppo alto può portare la rete neurale a riconoscere le classi solo per input molto simili a quelli del dataset, mentre un bias troppo basso può far sì che la rete generalizzi troppo, commettendo errori nelle predizioni delle classi o rilevando classi inesistenti.

### *Bounding boxes*

La bounding box rappresenta il rettangolo che circonda un oggetto per il suo rilevamento. Questo formato è impiegato dalle reti neurali per la rilevazione di oggetti e include le dimensioni e le coordinate del rettangolo stesso. Le coordinate generalmente indicano l'angolo in alto a sinistra o a destra della bounding box, sebbene in alcuni casi, come nel modello YOLO, indichino il centro del rettangolo.

## C

### *Centroide*

È il punto che rappresenta la posizione centrale di un oggetto rilevato in un'immagine.

### *Computer Vision*

La computer vision è un campo interdisciplinare che sviluppa metodi per consentire ai computer o ai sistemi automatizzati, come le intelligenze artificiali, di estrarre informazioni da immagini o video.

### ***Convoluzione***

Nell'ambito delle intelligenze artificiali, la convoluzione si riferisce all'applicazione di connessioni sinaptiche ricorrenti in una rete neurale, che avvengono da uno strato a quello precedente o all'interno dello stesso strato.

## **D**

### ***Deep Learning***

Il deep learning è una branca del machine learning che ottiene prestazioni superiori e maggiore flessibilità rappresentando l'ambiente osservato come una gerarchia di concetti annidati. Ogni concetto è definito in relazione a concetti più semplici e rappresentazioni astratte, che vengono ulteriormente elaborate in concetti meno astratti. Questo approccio è utilizzato nell'object detection per estrarre parametri e migliorare la precisione nella predizione delle diverse classi di oggetti.

### ***Dispositivi edge***

Dispositivi che eseguono elaborazioni dati e intelligenza artificiale direttamente sul luogo di raccolta dei dati, piuttosto che inviare questi dati a un server centrale per l'elaborazione.

## **E**

### ***Edge Impulse***

Una piattaforma di machine learning progettata per semplificare la creazione, l'addestramento e il deployment di modelli di intelligenza artificiale su dispositivi edge.

### ***Epoca***

Un'epoca nel machine learning è un singolo passaggio attraverso l'intero set di dati di addestramento da parte dell' algoritmo di apprendimento. Durante un'epoca, il modello viene esposto a ciascun esempio di addestramento una volta, permettendogli di aggiornare i pesi e i parametri sulla base degli errori



calcolati. Spesso, è necessario eseguire molte epoche per ottimizzare il modello e migliorare la sua accuratezza, poiché il modello apprende gradualmente dai dati e perfeziona le sue previsioni ad ogni epoca.

## **F**

### ***Feature***

Le features nelle reti neurali rappresentano le informazioni astratte che vengono trasferite tra gli strati di neuroni. Queste caratteristiche rilevanti sono correlate a quelle apprese durante la fase di addestramento e sono utilizzate per il riconoscimento delle classi.

## **H**

### ***Heatmap***

Una rappresentazione visiva che mostra l'attività o la densità di oggetti rilevati in un'immagine o una scena. Nel contesto delle reti neurali e del rilevamento degli oggetti, la heatmap viene utilizzata per evidenziare le aree dove la rete ha rilevato maggiore presenza o attività di oggetti

## **L**

### ***Label***

La label include una bounding box, che specifica le dimensioni e le coordinate del rettangolo che racchiude un oggetto in un'immagine, insieme all'etichetta della classe corrispondente.

## **M**

### ***Machine Learning***

Il machine learning è una disciplina dell'intelligenza artificiale che sviluppa sistemi per consentire ai computer di apprendere simulando ripetutamente su dati raccolti. Questi sistemi apprendono il comportamento desiderato o riconoscono gli elementi di un ambiente specifico attraverso tali simulazioni.

### ***Microsoft Kinect***

Un dispositivo di sensori di movimento originariamente progettato per la console di gioco Xbox 360. Ha visto anche numerose applicazioni in ambito di ricerca e sviluppo perché, grazie alla camera RGB e ai vari sensori e microfoni, consente di rilevare il movimento, riconoscere il viso e la voce delle persone e di creare anche rappresentazioni tridimensionali di oggetti e persone nell'ambiente circostante.

## **Y**

### ***YOLOv5***

E' la quinta versione della rete neurale YOLO.

# Bibliografia

1. Jiyuan Cai, Lingkun Luo & Shiqiang Hu. (2020) Bi-direction Direct RGB-D Visual Odometry. *Applied Artificial Intelligence* 34:14, pages 1137-1158.
2. Tian, L., Thalmann, N.M., Thalmann, D., Fang, Z., Zheng, J. (2019). Object Grasping of Humanoid Robot Based on YOLO. In: Gavrilova, M., Chang, J., Thalmann, N., Hitzer, E., Ishikawa, H. (eds) *Advances in Computer Graphics. CGI 2019. Lecture Notes in Computer Science()*, vol 11542. Springer, Cham. [https://doi.org/10.1007/978-3-030-22514-8\\_47](https://doi.org/10.1007/978-3-030-22514-8_47).
3. A. Sarda, S. Dixit and A. Bhan, "Object Detection for Autonomous Driving using YOLO [You Only Look Once] algorithm," 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), Tirunelveli, India, 2021, pp. 1370-1374, doi: 10.1109/ICICV50876.2021.9388577. keywords: {Industries; Deep learning; Roads; Object detection; Classification algorithms; Autonomous vehicles; Faces; YOLO; autonomous vehicles; object detection; computer vision}.
4. Zhou, Yan, Sijie Wen, Dongli Wang, Jiangnan Meng, Jinzhen Mu, and Richard Irampaye. 2022. "MobileYOLO: Real-Time Object Detection

Algorithm in Autonomous Driving Scenarios" *Sensors* 22, no. 9: 3349.  
<https://doi.org/10.3390/s22093349>.

5. A. Sarda, S. Dixit and A. Bhan, "Object Detection for Autonomous Driving using YOLO algorithm," 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2021, pp. 447-451, doi: 10.1109/ICIEM51511.2021.9445365. keywords: {Training; Navigation; Computational modeling; Buildings; Object detection; Bicycles; Cameras; YOLO; custom training; computer vision Autonomous vehicles}.
6. S. Chen and W. Lin, "Embedded System Real-Time Vehicle Detection based on Improved YOLO Network," 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 2019, pp. 1400-1403, doi: 10.1109/IMCEC46724.2019.8984055. keywords: {YOLO; target detection; deep learning; real-time; embedded system}.
7. Nabavian, M., Osia, S. A., & Khoury, R. (2021). Practical Adversarial Attacks on Computer Vision with YOLO and DNN. arXiv preprint arXiv:2112.11798. Retrieved from <https://arxiv.org/abs/2112.11798>.
8. R. Garg and S. Singh, "Intelligent Video Surveillance Based on YOLO: A Comparative Study," 2021 International Conference on Advances in Computing, Communication, and Control (ICAC3), Mumbai, India, 2021, pp. 1-6, doi: 10.1109/ICAC353642.2021.9697321. keywords: {Deep learning; Image recognition; Art; Weapons; Computational modeling; Neural networks; Object detection; Handheld

Weapon (Pistol); Deep Learning; YOLOV3; YOLOV4; YOLOV5; Transfer Learning}.

9. Jha, S., Seo, C., Yang, E. et al. Real time object detection and trackingsystem for video surveillance system. *Multimed Tools Appl* 80, 3981–3996 (2021). <https://doi.org/10.1007/s11042-020-09749-x>.

10.H. H. Nguyen, T. N. Ta, N. C. Nguyen, V. T. Bui, H. M. Pham and D. M. Nguyen, "YOLO Based Real-Time Human Detection for Smart Video Surveillance at the Edge," 2020 IEEE Eighth International Conference on Communications and Electronics (ICCE), Phu Quoc Island, Vietnam, 2021, pp. 439-444, doi: 10.1109/ICCE48956.2021.9352144.

keywords: {Image edge detection; Video surveillance; Real-time systems; Safety; Security; Task analysis; Testing; Edge computing; smart video surveillance; YOLO; human detection}.

11.Ashraf, Abdul Hanan, et al. "Weapons detection for security and video surveillance using cnn and YOLO-v5s." *CMC-Comput. Mater. Contin* 70 (2022): 2761-2775. Narejo, Sanam, Pandey, Bishwajeet, Esenarro vargas, Doris, Rodriguez, Ciro, Anjum, M. Rizwan, Weapon Detection Using YOLO V3 for Smart Surveillance System, *Mathematical Problems in Engineering*, 2021, 9975700, 9 pages, 2021. <https://doi.org/10.1155/2021/9975700>.

12.P. Ren, W. Fang and S. Djahel, "A novel YOLO-Based real-time people counting approach," 2017 International Smart Cities Conference (ISC2), Wuxi, China, 2017, pp. 1-2, doi: 10.1109/ISC2.2017.8090864.

keywords: {Real-time systems; Object detection; Streaming media;

Conferences; Computer vision; Smart cities; Neural networks; People-counting; Boundary-selection; YOLO}.

13. Ren, Peiming, et al. "A novel squeeze YOLO-based real-time people counting approach." *International Journal of Bio-Inspired Computation* 16.2 (2020): 94-101.
14. H. Soebhakti, S. Prayoga, R. A. Fatekha and M. B. Fashla, "The Real-Time Object Detection System on Mobile Soccer Robot using YOLO v3," 2019 2nd International Conference on Applied Engineering (ICAE), Batam, Indonesia, 2019, pp. 1-6, doi: 10.1109/ICAE47758.2019.9221734. keywords: {YOLO V3; Mobile Soccer Robot; Object Detection}.
15. Liu, Jun, et al. "BiGA-YOLO: A Lightweight Object Detection Network Based on YOLOv5 for Autonomous Driving." *Electronics* 12.12 (2023): 2745.