ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA

DIPARTIMENTO di

INGEGNERIA DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE

"Guglielmo Marconi"

DEI

CORSO DI LAUREA MAGISTRALE IN

INGEGNERIA ELETTRONICA

TESI DI LAUREA

in

Sistemi Elettronici ad Alta Affidabilità M

ANALISI E IMPLEMENTAZIONE DI STRATEGIE DI INTELLIGENZA ARTIFICIALE PER IL CONTEGGIO DI PERSONE PER LA GESTIONE DI SITUAZIONI DI EMERGENZA

CANDIDATO

Alberto Abbruzzese

RELATORE

Chiar.mo Prof. Ing. Martin Eugenio Omana

CORRELATORI

Prof.ssa Ing. Cecilia Metra

Dott. Sara Cretì

Ing. Gianni Borelli

Anno Accademico

2022/2023

Sessione III

Indice

1. Introduzione 4
1.1 Motivazioni 4
1.2 Metodologia e Approccio 4
2. Concetti preliminari 7
2.1 Intelligenza Artificiale 7
2.1.1 Machine Learning 7
2.1.2 Deep Learning e Reti Neurali 9
2.2 Affidabilità e Soft Error 16
2.2.1 Affidabilità e Tolleranza ai Guasti 16
2.2.2 Guasti Transitori e Soft Error 19
3. Implementazione di Strategia IA per il Conteggio di Persone 22
3.1 Scelta della rete 22
3.1.1 YOLO e alcune sue Versioni 23
3.1.2 YOLO per il Conteggio di Persone 23
3.2 Test e Valutazione del modello 30
3.2.1 Datasets nel Pre-Allenamento e nel Test 31
3.2.2 Metrica di Valutazione 31
3.2.3 Valutazione del Modello 33
4. Analisi dell'Affidabilità della Strategia IA per il Conteggio di Persone 38
4.1 Metodologia per la Simulazione dei Soft Error 38
4.2 Analisi dei Risultati 40
4.2.1 Iniezione dei SE sul bit di segno 41
4.2.2 Iniezione dei SE sul bit più significativo dell'esponente 43
4.2.3 Iniezione dei SE sul bit più significativo della mantissa 46
5. Conclusioni 49

Bibliografia 52
Indice delle figure 56
Indice delle tabelle 57
Indice equazioni 57

1. Introduzione

1.1 Motivazioni

L'identificazione e il conteggio delle persone in una folla rappresentano una sfida di rilevanza significativa nell'ambito della visione artificiale. Tale compito ha una vasta gamma di applicazioni pratiche, spaziando dalla pianificazione urbana al rilevamento di comportamenti anomali, dalla videosorveglianza alla gestione della sicurezza pubblica, dalla difesa alla salute pubblica e alla gestione dei disastri [1]. In situazioni di emergenza come incendi, terremoti, alluvioni e altri disastri che purtroppo accadono spesso, la rapidità e l'efficacia delle operazioni di soccorso sono cruciali per minimizzare danni e salvare vite umane. Di fronte a tali situazioni, è essenziale adottare strategie di risposta e gestione delle emergenze efficaci ed efficienti.

Uno degli elementi chiave per una risposta ottimale è il conteggio accurato delle persone coinvolte nelle situazioni di emergenza. Il presente studio si propone di esplorare l'importanza del conteggio delle persone come dato fondamentale nelle operazioni di gestione delle emergenze, e di come questo possa essere fatto mediante una rete neurale in grado di acquisire immagini video, e di restituire in output questo dato. Inoltre, verrà mostrata un'analisi quasi totale della rete per valutare la sua robustezza ai *soft error*, errori che stanno assumendo una criticità crescente in relazione alla continua miniaturizzazione dei circuiti attuali [26].

1.2 Metodologia e Approccio

Una prima attività di ricerca ha permesso di individuare quelle che sono le reti più adatte alla risoluzione del problema, ovvero le Reti Neurali Convoluzionali (CNN). Questa tipologia di rete è adatta per l'elaborazione di immagini. Perciò, inizialmente sono state

implementate reti neurali come ResNet50 [4], VGG16 [4], AlexNet [2], e allenate con alcuni *datasets* [5][6][7].
Ovviamente lo scopo era allenare le reti per riuscire a contare le persone in qualsiasi immagine che le si proponeva.
Tuttavia, queste reti non erano adatte allo scopo previsto. In particolare, si riusciva a raggiungere un errore molto basso con alcuni *dataset*, ma la rete riusciva a rilevare le persone solamente nelle immagini utilizzate in allenamento.
Questo è dovuto al fatto che le reti sopracitate sono per lo più impiegate per i problemi di *classificazione* delle immagini e non per il rilevamento di persone.

In merito al rilevamento delle persone le possibilità di scelta della rete possono essere molteplici, anche se la complessità strutturale aumenta. Tra le reti più utilizzate ci sono le R-CNN (Region-based Convolutional Neural Network), in particolare le Faster R-CNN, una sua versione più veloce. Ma, in merito a velocità, sicuramente la scelta migliore ricade sulla YOLO (You Only Look Once) rete in grado di processare le immagini molto velocemente [8].
Ovviamente la velocità varia molto in base a dove viene implementata la rete (CPU, GPU o altro).
In particolare, la rete finale utilizzata è una YOLOv3 pre-allenata [9][10]. Per testarla è stato usato come linguaggio di programmazione *Phyton*, con un suo framework, *Tensorflow*. Inoltre, è stata utilizzata la piattaforma di Google Colab (Google Colaboratory 2024) [12]; piattaforma che mette a disposizione dell'utente gratuitamente (con risorse limitate) o a pagamento, delle risorse esterne come GPU, RAM e altre.
Per il processamento di un'immagine implementando la rete su CPU (Il nome della CPU non è accessibile dalla piattaforma) la rete impiega 1.3 secondi. Per il test sulla velocità in GPU, la rete è stata testata usando le GPU disponibili su Colab. Ovvero, la T4, la V100 e l'A100. Con la T4 la rete riesce a processare l'immagine in 134ms; con la V100 la velocità aumenta, processando un'immagine in 92ms. Mentre, con l'A100 la rete riesce a processare l'immagine in 89ms.

Naturalmente, è importante considerare che questi dati potrebbero non essere completamente rappresentativi, poiché, si stanno utilizzando delle risorse esterne, e quindi, la velocità di elaborazione delle immagini varia in base alla connessione Internet. Sicuramente, utilizzando delle risorse interne la velocità della rete migliorerebbe di molto. Seguendo alcune ricerche, infatti, implementando una rete YOLOv8, con circa gli stessi parametri della rete utilizzata in questa tesi, su una GPU A100, questa impiegherebbe 3.53ms a valutare un'immagine; riuscendo quindi, a dare una valutazione in tempo reale [11]. Bisogna però considerare che una YOLOv8 ha delle caratteristiche diverse rispetto alla YOLOv3, e una tra queste è proprio la velocità.

Il lavoro di tesi svolto è strutturato come segue. Innanzitutto, si esporranno concetti preliminari su cos'è un'Intelligenza Artificiale, alcuni suoi sottoinsiemi e cosa sono le reti neurali; in particolare si approfondiranno le Reti Neurali Convoluzionali (CNN). Si introdurranno inoltre, concetti sull'*affidabilità* di un sistema, e i *soft error*. Successivamente, si esporranno i risultati ottenuti testando la rete su un *dataset* contenente frames dove vengono ritratte persone. E come ultimo punto verrà mostrata un'analisi della rete ai *soft error* per valutare quali siano le parti più critiche della rete e quali sono i possibili rimedi per evitare un conteggio errato, evitando che questo possa incidere negativamente quando ci si trova in situazioni di emergenza.

2. Concetti preliminari

Prima di introdurre i risultati del lavoro svolto, è conveniente per il lettore conoscere alcuni concetti preliminari sul campo delle Intelligenze Artificiali, e alcuni concetti riguardanti l'*affidabilità* dei sistemi.

2.1 Intelligenza Artificiale

L' Artificial Intelligence Management System (AIMS) definisce l'Intelligenza Artificiale (IA) come "la capacità di un sistema di mostrare capacità umane quali il ragionamento, l'apprendimento, la pianificazione e la creatività" [40].

Di base, perciò, un'IA è una tecnologia in grado di simulare alcune competenze umane, e quindi, di risolvere problemi in autonomia. Ciò ha contribuito a diffondere sempre più le Intelligenze Artificiali negli ultimi anni, rendendole una presenza costante nelle nostre vite e diffondendosi in ogni ambito.

2.1.1 Machine Learning

Un ramo delle IA è il Machine Learning (o Apprendimento Automatico). Questo permette a un sistema artificiale di apprendere automaticamente dai dati che gli vengono forniti. Quindi, mentre in un Intelligenza Artificiale generica la conoscenza per risolvere determinati problemi può essere fornita programmando un sistema a svolgere una serie di algoritmi, nel Machine Learning questa è data da dati che noi forniamo alla macchina, e sarà la macchina stessa ad apprendere in maniera automatica. Questo apprendimento automatico viene eseguito tramite un processo di addestramento nel quale la macchina stessa cerca di rilevare relazioni all'interno dei dati, e lo fa attraverso l'applicazione di algoritmi di apprendimento che li analizzano e cercano di estrarre informazioni significative da essi.

Quindi, in sostanza per un'IA generica l'algoritmo per risolvere il problema è il programmatore a darlo; nel Machine Learning questo algoritmo è la macchina stessa a generarlo.

Le modalità di apprendimento nel Machine Learning sono 3 [25]:

- Apprendimento *supervisionato*: modalità che prevede un labeling dei dati di input utilizzati nella fase di addestramento. Quindi, sostanzialmente, la macchina apprende valutando l'errore che sta commettendo basandosi sulle etichette. Nell'apprendimento supervisionato, le due principali categorie di compiti sono la *classificazione* e la *regressione*. Nella *classificazione*, l'obiettivo è assegnare una classe a ciascun input, mentre nella *regressione* si cerca di predire un valore numerico in base alle caratteristiche degli input. Questi due approcci forniscono una struttura definita per l'addestramento dei modelli, poiché si dispone di un set di dati di addestramento etichettato che guida il processo di apprendimento.

- Apprendimento *non supervisionato*: in questa modalità, le etichette degli input non vengono fornite; perciò, la macchina dovrà trovare una relazione tra i dati di input, e ricavarne l'output desiderato. Nell'apprendimento non supervisionato, uno dei principali compiti è il clustering, che presenta una sfida fondamentale: dividere i dati in gruppi quando non sono accompagnati da etichette di classificazione. In questa modalità, la macchina si basa esclusivamente sulle caratteristiche intrinseche degli input per organizzarli in cluster.

- Apprendimento con *rinforzo*: la macchina apprende interagendo con l'ambiente in cui opera. Una volta concluso il suo compito, riceve un feedback dall'operatore che valuta se ha agito correttamente o meno. Nel caso di un feedback negativo, sarà la macchina a valutare, a seconda delle scelte prese, cosa ha sbagliato e su come correggersi. Questa modalità è spesso utilizzata quando la macchina è un agente autonomo.

Esiste in realtà una quarta modalità di apprendimento, che è una combinazione tra le prime due modalità. Ed è l'apprendimento *semi-supervisionato*, che prevede che solo una parte degli input siano etichettati. Questo consente di sfruttare le qualità dei due metodi per migliorare le prestazioni della macchina.

2.1.2 Deep Learning e Reti Neurali

Compreso nel Machine Learning vi è il Deep Learning, che è una branca che si basa su le reti neurali artificiali profonde, cioè reti neurali con più strati di neuroni interconnessi.

Il termine "Deep", ovvero profondo, si riferisce proprio alla presenza di più strati o layers di neuroni interconnessi tra di loro, che permettono l'estrapolazione di informazioni sempre più complesse dai dati di input.

Una rete è considerata profonda quando ha più di un layer nascosto (hidden layer).

Ma cos'è una rete neurale? E qual è il processo di apprendimento?

1. *Struttura di una rete neurale*

Una rete neurale artificiale è una struttura che comprende l'insieme di neuroni artificiali che emulano il funzionamento del cervello umano.

Quindi, alla base delle reti ci sono i neuroni artificiali che simulano il comportamento dei neuroni biologici. Il compito di ogni neurone è quello di effettuare dei calcoli sugli ingressi e produrre degli output, applicando una *funzione di attivazione*.

Il processo di calcolo si sviluppa moltiplicando gli ingressi per dei pesi (*weights*). I prodotti vengono accumulati e sommati a un *bias*, e successivamente il risultato (*numero di attivazione*) verrà processato da una *funzione di attivazione*, producendo l'output [24][25].

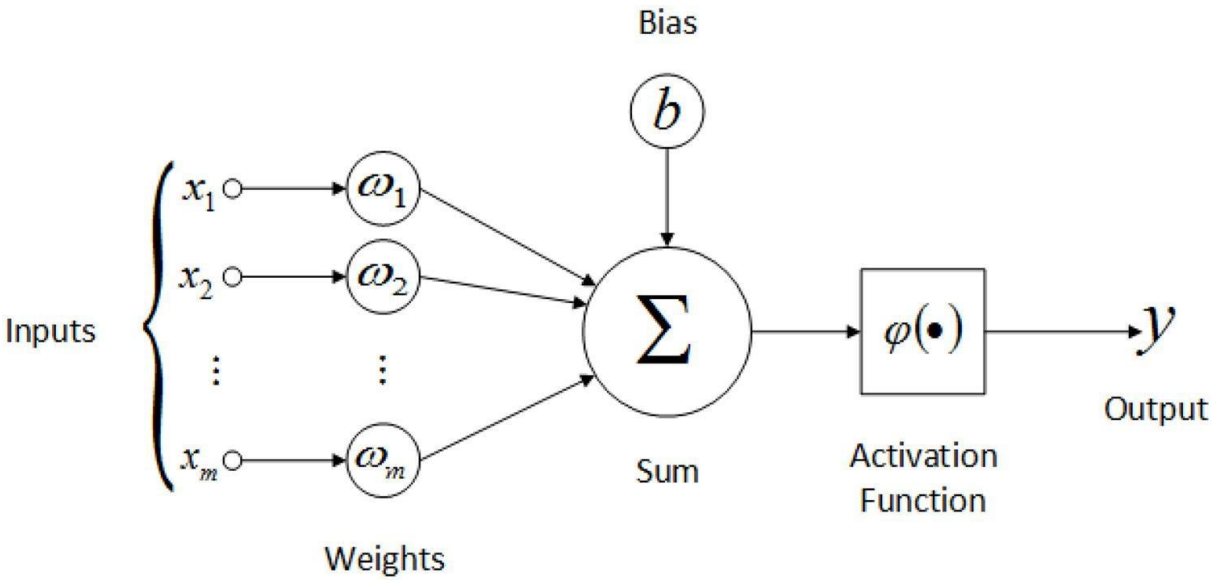


Figura 1. *Processamento degli Input da un neurone. [13]*

L'espressione matematica che produce il numero di attivazione del j-esimo neurone è la seguente:

$$num\_act_j = \sum_i x_i * w_{ij} + b_j \quad (1)$$

Mentre, l'output del neurone sarà:

$$Out_j = \varphi(num\_act_j) \quad (2)$$

Con φ *funzione di attivazione*.

Le *funzioni di attivazione* sono molteplici. È compito del programmatore decidere quale sia quella più adatta alla risoluzione del problema. Oltretutto, questa influisce anche nella velocità del processo di allenamento della rete, quindi, è di fondamentale importanza individuare la funzione più adatta.

Con layer invece si intente un'unità che raggruppa un insieme di neuroni. Ogni layer riceve un input dall'esterno o dall'output di altri layer e produce un output che viene poi passato al layer successivo. Quindi, in una rete i layer sono organizzati in modo sequenziale formando appunto dei veri e propri strati di neuroni.

Tra lo strato di input e quello di output ci sono degli strati interni alla rete, detti strati nascosti o hidden layers, e sono questi, come già detto in precedenza, a definire una rete profonda.

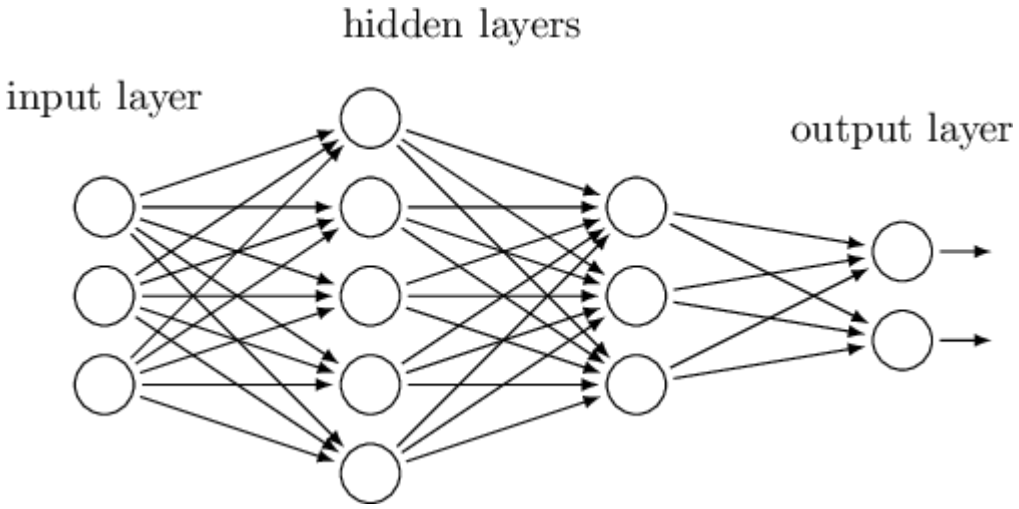


Figura 2. *Struttura a strati di una rete neurale. [14]*

Introdotta la struttura di una rete neurale generica ora non ci resta che definire il processo di allenamento.

2. *Allenamento di una rete*

In questa fase i valori di *pesi* e *bias* vengono modificati mediante un algoritmo di ottimizzazione. Esistono diversi algoritmi disponibili e spetta al programmatore selezionare quello più appropriato per l'applicazione specifica.

Inizialmente, vengono dati alla rete degli esempi etichettati (caso Supervisionato); l'algoritmo ne calcola una *funzione di costo* (errore), scelta sempre dal programmatore, per poi andare a modificare i pesi seguendo il processo di ottimizzazione scelto. Questo processo viene fatto in maniera iterativa per più esempi e per più volte (epoche). Successivamente, al termine di ogni epoca il processo valuta con dei dati di validazione, dati che non sono stati usati nel processo di ottimizzazione, quanto è l'effettivo errore

commesso dalla rete. Questo procedimento è utile per valutare il sovra-allenamento (*overfitting*) della rete. Fenomeno che causa un adattamento eccessivo della rete ai dati di allenamento. In particolare, si ha un sovra-allenamento quando l'errore generato valutando i dati di validazione non diminuisce con il passare delle epoche, ma anzi, aumenta. Una rete sovra-allenata, quindi, sarà in grado di distinguere solo i dati utilizzati durante l'allenamento, risultando in un funzionamento errato in presenza di nuovi dati o situazioni non incontrate durante l'allenamento. Caso contrario invece, è il sotto-allenamento (*underfitting*), situazione in cui le epoche sono troppo poche per permettere alla rete di allenarsi adeguatamente. Perciò, è fondamentale definire un numero adatto di epoche per l'allenamento.

Tornando agli algoritmi di ottimizzazione, quelli più usati sono l'algoritmo di *discesa del gradiente*, con tutte le sue variazioni; l'algoritmo *Adaptive Moment Estimation* (Adam) e l'algoritmo *Root Mean Square Propagation (*RMSprop) basato sul primo algoritmo.

Le *funzioni di costo* invece sono molte, e variano in funzione del tipo di problema che si sta affrontando. Per esempio, se si sta affrontando un problema di *classificazione* una canonica funzione di costo usata è la *Categorical Cross-Entropy Loss*. Mentre, per i problemi di regressione, problemi in cui si vuole predire un valore, le funzioni più utilizzate sono il *Mean Squares Error* (MSE), *Root Mean Square Error* (RMSE), *Mean Absolute Error* (MAE).

Di seguito sono riportate le espressioni delle funzioni prima citate.

$$CCE = \sum_i y_i * \log(\hat{y}_i) \qquad (3) \qquad MSE = \frac{1}{n}\sum_i (y_i - \hat{y}_i)^2 \qquad (4)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_i (y_i - \hat{y}_i)^2} \qquad (5) \qquad MAE = \frac{1}{n}\sum_i |y_i - \hat{y}_i| \qquad (6)$$

Dove y_i è il vero valore dell'i-esimo esempio mentre $\hat{y}_i$ è il suo valore predetto.

Perciò, il processo di allenamento mira a ridurre il più possibile la *funzione di costo*, fino ad ottenere un risultato soddisfacente.

3. *Reti Neurali Convoluzionali*

La famiglia di reti impiegata principalmente per l'elaborazione di immagini sono le reti neurali convoluzionali.

Questa categoria di reti è *Feedforward*, cioè, ogni layer ha l'input collegato direttamente all'output del layer precedente; mentre, il suo output è connesso all'input del layer successivo.

In questa sezione toccheremo la struttura base di questa famiglia di reti, e di come questa riesca a riconoscere oggetti ed esseri viventi all'interno di un'immagine.

Il fulcro della rete è il *layer convoluzionale*, che permette l'estrazione del pattern dell'immagine. Con pattern si intende quella configurazione delle feature (contorni, texture, colori…) che descrive il contenuto dell'immagine. Il layer applica uno o più filtri a una sotto porzione dell'immagine alla volta. Ogni filtro è una matrice di pesi che viene moltiplicata per i valori dei pixels in ingresso [21]. Questo processo genera un output che è la somma pesata dei pixel nell'area di input coperta dal filtro. Fatto ciò, il filtro viene poi applicato alla sotto porzione successiva. Il passo con cui si sposta il filtro da sotto porzione a sotto porzione è lo *stride*, ed è definito dal programmatore, come la dimensione del filtro. Inoltre, si può decidere se aggiungere un *padding*, in modo da mantenere la dimensione dell'output uguale a quella dell'input. Effettuata l'operazione di convoluzione, l'output potrebbe passare attraverso a una funzione di attivazione per introdurre una non linearità e permettere di ricavare il pattern dell'immagine. Altro layer fondamentale nelle reti convoluzionali è il *layer di pooling* che permette una riduzione della dimensionalità dei dati di output. Questo permette di poter applicare più layer convoluzionali, mantenendo un numero di pesi allenabili ragionevole.

Alternando l'utilizzo di layers convoluzionali e di pooling in sequenza, vengono ricavate le feature dell'immagine [21]. Questi dati in uscita vengono indirizzati a dei layers densi tramite un *layer flatten*, che permette l'appiattimento dei dati, disponendoli in una sola dimensione.

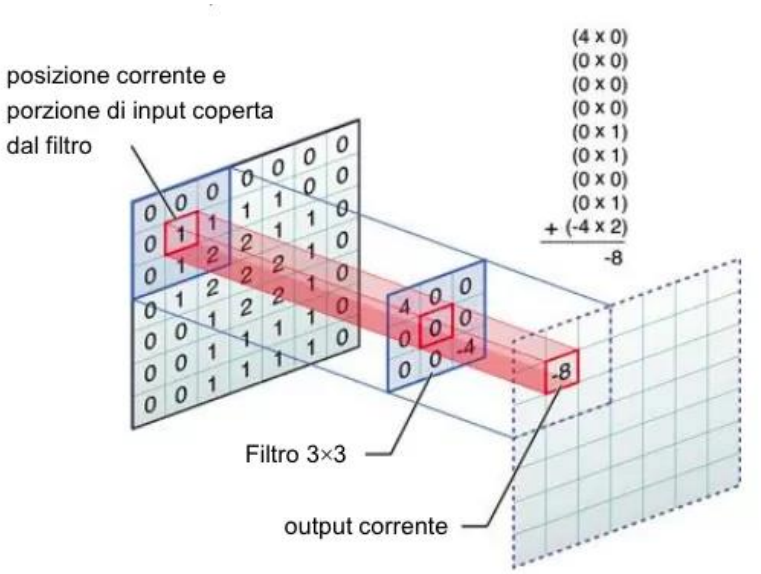


Figura 3. *Applicazione di un filtro convoluzionale. [19]*

I *layers densi* permettono di classificare, sulla base delle feature estratte dai layers precedenti, gli elementi all'interno dell'immagine. Lo stadio finale, quindi, sarà costituito da un numero di strati densi determinato dal progettista, che può scegliere se renderli interamente connessi o parzialmente connessi. Ovviamente, utilizzare dei layers densi parzialmente connessi implica una riduzione di parametri e, quindi, una velocità maggiore in fase di allenamento. Bisogna però prestare attenzione a un flusso corretto dei dati rilevanti attraverso la rete, garantendo che le informazioni più importanti non vengano bloccate [23].

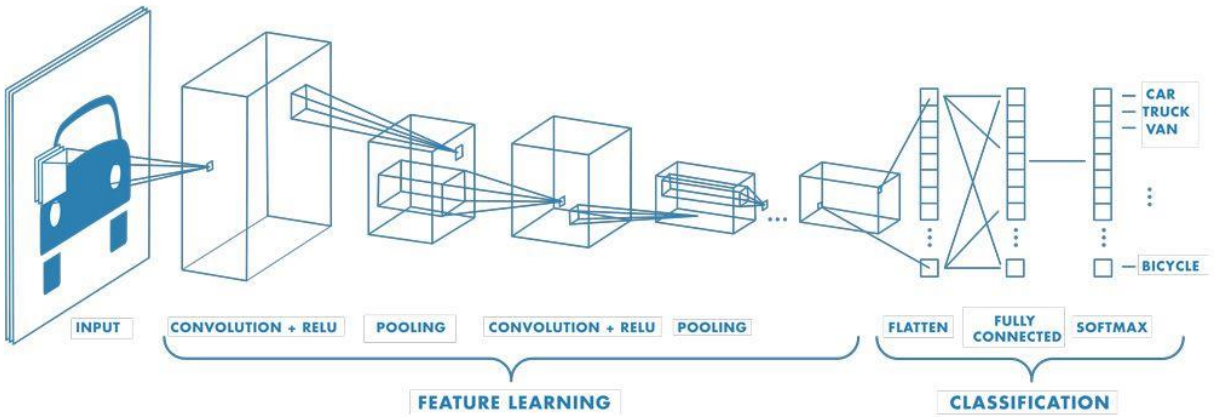


Figura 4. *Struttura di una Rete Neurale Convoluzionale. [15]*

Nella figura soprastante si può notare la generica struttura di una CNN. In output ai layers convoluzionali i dati vengono elaborati da una funzione di attivazione *Rectified Linear Unit* (ReLU), definita come:

$$ReLU = \max(0, input) \tag{7}$$

In questo tipo di funzione poiché la derivata è semplicemente 1 per input positivi e 0 per input negativi, il flusso del gradiente è più stabile durante l'addestramento. Questo aiuta a risolvere problemi come quello della scomparsa del gradiente, che comporta una lenta o addirittura assenza di aggiornamenti dei pesi più significativi.

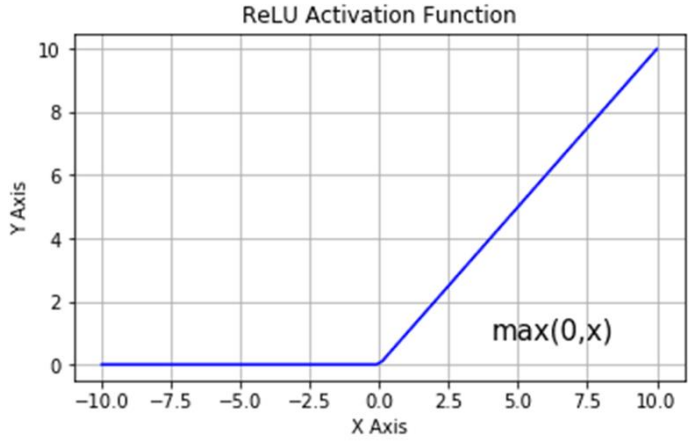


Figura 5. *Rappresentazione grafica funzione ReLU. [16]*

Qui di seguito si riportano altre funzioni di attivazione molto utilizzate.

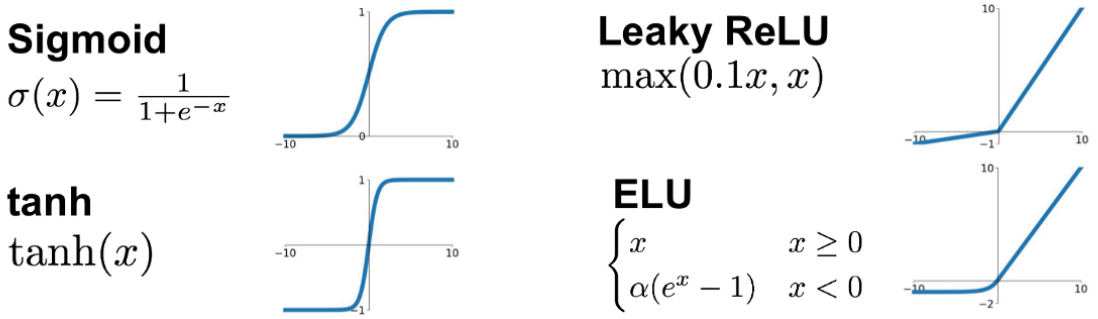


Figura 6. *Rappresentazione grafica di alcune funzioni di attivazione. [17]*

Sempre in Figura 4 si può notare come lo strato finale della rete sia un layer di *softmax*. La funzione *softmax* converte un vettore di valori reali in un vettore di probabilità, in modo che la somma di tutte le probabilità sia uguale a uno. Questo è utile in situazioni in cui si desidera assegnare un'etichetta di classe a un input, dove ci sono più classi tra cui scegliere. Come classe predetta si prenderà naturalmente l'uscita con la probabilità più alta.

Oltre ai layers citati, una CNN può includere layer di *Batch Normalizzation* e *Dropout*. Il primo prevede una normalizzazione dei dati di input nei vari livelli, consentendo un allenamento più veloce e più stabile. Il layer di *Dropout* invece, serve a ridurre la tendenza della rete a memorizzare i dati di addestramento (*overfitting*) [22], piuttosto che ricavare il pattern necessario a classificare l'immagine. Il funzionamento di questo layer prevede un inattivamento di alcuni neuroni in maniera casuale durante l'allenamento. Questo fa sì che i neuroni non si specializzino a riconoscere solo i dati utilizzati in allenamento e che non dipendano troppo da specifici neuroni vicini, rendendo l'allenamento più omogeneo tra i vari neuroni. Questo processo, come già detto, viene praticato in maniera randomica, e a ogni iterazione del processo di addestramento i neuroni inattivati saranno diversi.

2.2 Affidabilità e Soft Error

Come anticipato, nell'introduzione di questa tesi, la rete in esame è stata testata ai *soft error*, errori causati dai guasti *transitori* (TF).
Perciò, ci sembra opportuno introdurre alcuni concetti riguardanti le tecniche di *fault tolerance* e i guasti *transitori*, per poi passare a descrivere i *soft error*.

2.2.1 Affidabilità e Tolleranza ai Guasti

Per definizione l'*affidabilità* è la probabilità che un sistema funzioni correttamente per un determinato intervallo di tempo. Questo termine ovviamente può essere riferito ai sistemi elettronici, che è il caso di nostro interesse.
Esiste un insieme di tecniche che mira a garantire un corretto funzionamento di un sistema, anche in presenza di guasti. Questo genere di tecniche permette di avere dei dispositivi elettronici ad alta *affidabilità*. Ci si sta riferendo alle tecniche di *fault tolerance*.
Ma cosa si intende con dispositivo affidabile?

Il concetto di *affidabilità* di un dispositivo elettronico è sempre più rilevante al giorno d'oggi, specialmente se si considera il loro crescente impiego nelle applicazioni critiche e nella vita quotidiana. Un dispositivo elettronico affidabile è un dispositivo che è in grado di eseguire le sue funzioni in modo corretto e consistente nel tempo, senza guasti o malfunzionamenti significativi. Questo implica che il dispositivo possa operare nelle condizioni previste per un periodo di tempo appropriato senza subire danni o riduzioni delle prestazioni.

Ovviamente, il concetto di alta *affidabilità* viene meno se il dispositivo in questione non ha degli impatti eccessivamente negativi nella vita delle persone dovuti al suo malfunzionamento.

Nel nostro caso, poiché, il dispositivo è un hardware che implementa una rete neurale in grado di contare le persone in situazioni di emergenza, è ovvio che questi concetti siano di fondamentale importanza per la salvaguardia di vite umane.

Oltre al concetto di *affidabilità*, bisogna tener conto della *dependability*, che è la sintesi degli attributi di *relitbility*, *availability*, *safety* e *security.*

Con *reliability* si intende l'*affidabilità* di un sistema, concetto che abbiamo già avuto modo di analizzare. L'*availability* (o disponibilità) è invece, la probabilità istantanea che il sistema funzioni correttamente. Questo concetto riguarda per lo più quei dispositivi che possono permettersi un disservizio temporaneo (es. bancomat, servizio di trasporto pubblico, sistemi di sicurezza domestica, ecc…).

Per descrivere il concetto di *safety* possiamo affidarci a un esempio pratico, come quello di un semaforo. Se il semaforo si blocca sul rosso in modo controllato, mantenendo il traffico in uno stato di attesa sicuro, allora potremmo considerare il dispositivo come "safe" in quanto continua a svolgere il suo scopo primario di regolare il traffico in modo sicuro. Tuttavia, se il semaforo si blocca sul verde, creando potenzialmente situazioni di pericolo incrociato, allora il dispositivo non è più considerato "safe" perché non rispetta il suo scopo primario e può causare conseguenze potenzialmente pericolose. In definitiva, un dispositivo "safe" è un dispositivo che funziona secondo il suo scopo previsto e che è in grado di prevenire o mitigare eventuali rischi per l'ambiente o gli utenti durante il suo funzionamento normale o in caso di malfunzionamento.

La *security* invece, riguarda tutti quei dispositivi che elaborano e memorizzano dati. Un sistema ha un'alta *security* se ha un'alta probabilità di mantenere integri e riservati i dati che tratta.
Come detto, quel che vorremmo è che il nostro sistema abbia un'alta *reliability*.
Questa può essere espressa come:

$$Reliability = P(nf) + P(co, f) * P(f) \quad (8)$$

Dove P(nf) è la probabilità che non si verifichino guasti, P(co,f) è la probabilità di un corretto funzionamento del sistema anche in presenza di guasti, e P(f) è la probabilità che si verifichi un guasto.
Le tecniche di *fault tolerance* servono proprio a massimizzare la P(co,f), poiché, aumentare P(nf) non sempre basta (tecniche di *fault avoidance*).
Rendere un sistema ad alta affidabilità attraverso l'uso di queste tecniche implica l'introduzione di forme di ridondanza, per le quali è necessario essere disposti a investire in termini di hardware, software, o introducendo ridondanza d'informazioni o temporale.
Le tecniche hardware *fault tolerance* tradizionali sono la *ridondanza modulare*, *l'online testing & recovery*, e l'utilizzo di *codici a correzione d'errore* (ECC).
La prima tecnica prevede l'utilizzo di un minimo di tre moduli gemelli. L'output dei tre moduli viene inviato a un *voter*, che fornirà come sua uscita il valore logico di maggioranza.
Il secondo metodo comprende appunto l'*online testing*, che serve sostanzialmente a rilevare l'errore, e la *recovery* che permette di ristabilite il corretto funzionamento.
L'*online testing* può essere svolto con le tecniche di *duplicazione e confronto*, o impiegando i *circuiti di self-checking*.
Come tecnica di *recovery* vi è la *roolback & reatry*, che permette di far tornare il sistema in uno stato precedente (checkpoint). Se il guasto verificato era transitorio, allora il sistema continua a funzionare. Se invece, l'errore continua a persistere si può effettuare un *repair online* o *offline*. Il *repair online* permette un continuamento del servizio. In quello *offline* c'è bisogno che la macchina smetta di funzionare per ripararla.

Utilizzando i *codici a correzione d'errore* invece, non è necessario un *repair* poiché, è il codice stesso a correggere l'errore, e a permettere un continuo funzionamento della macchina [27].

2.2.2 Guasti Transitori e Soft Error

Con guasti *transitori* (TF) si intendono tutti quei guasti che causano la comparsa di una transizione rapida su un nodo. Essendo l'effetto quello di alterare temporaneamente la tensione o la corrente, causando un glitch, gli apparati più sensibili a questi tipi di guasti sono le memorie e i circuiti che prevedono un campionamento dei dati.

La generazione di un errore causato da questi guasti rappresenta un *soft error* (SE). Quindi, la causa è il TF, mentre l'effetto è il SE. Poiché, il sistema per implementare la rete neurale prevede l'utilizzo di una memoria in cui memorizzare i *pesi* e i *bias*, è opportuno prestare attenzione a questi errori. Inoltre, per accelerare i processi di calcolo, si potrebbe usufruire di un hardware accelerator, e, secondo alcuni studi, questi sono molto sensibili ai guasti *transitori*.

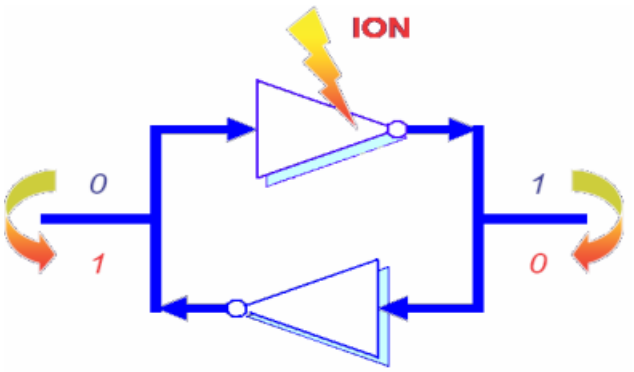


Figura 7. *Soft error su una cella RAM. [27]*

La rete impiegata, avendo molti parametri, necessita sicuramente di un supporto hardware per velocizzare il processo di elaborazione delle immagini per avvicinarsi il più possibile a un monitoraggio in tempo reale.

Per esempio, si potrebbe usare un hardware accelerator che permette di ottenere i risultati molto più velocemente.

I SE costituiscono una minaccia distintiva per questi tipi di supporto poiché il parallelismo di questi acceleratori può diffondere un singolo errore a più processi di calcolo successivi fino a quando l'output delle previsioni del modello ne risulta estremamente influenzato [20]. Inoltre, questi apparati lavorano a una frequenza elevata. Quindi, è più probabile che il guasto transitorio venga campionato e che si generi un SE.

Le cause principali dei guasti *transitori* sono le *particelle α* e i *raggi cosmici*, che, colpendo il silicio nei circuiti elettronici, generano delle coppie elettrone-lacuna che causano un'alterazione della carica e, quindi, una variazione del potenziale.

Le *particelle α* sono generate dal decadimento radioattivo di alcuni isotopi instabili, come le impurità di Torio o l'Uranio compresi all'interno del package di un chip.

I *raggi cosmici* invece, sono in generale dei neutroni emessi dal sole, che incidendo con gli atomi di silicio, possono dare il via a una ionizzazione *indiretta*, generando *particelle α* all'interno del silicio stesso.

Sono le *particelle α* quindi, a generare le coppie elettrone-lacuna. E lo fanno tanto più queste riescono a penetrate all'interno del silicio. Ovviamente, la profondità con la quale queste riescono a penetrare è funzione dell'energia che queste particelle hanno all'inizio della penetrazione.

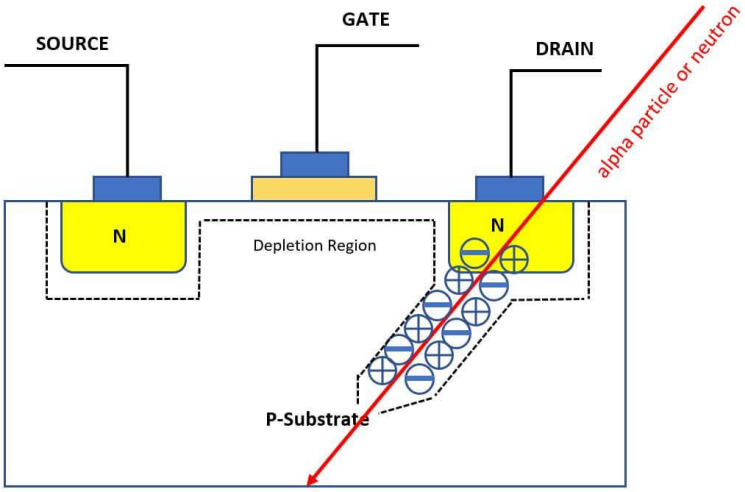


Figura 8. *Generazione coppie elettrone-lacuna da parte di una particella α. [18]*

Nel dettaglio, la particella perde 1.11eV a ogni coppia elettrone-lacuna generata

Addirittura, se la particella perfora drain e source di un MOSFET, potrebbe causare una temporanea conduzione del canale (effetto ALPEN). La particella però genera una carica totale neutra, perciò, il glitch avverrà solo se si è in presenza di un campo elettrico. Le zone che forniscono un elevato campo elettrico

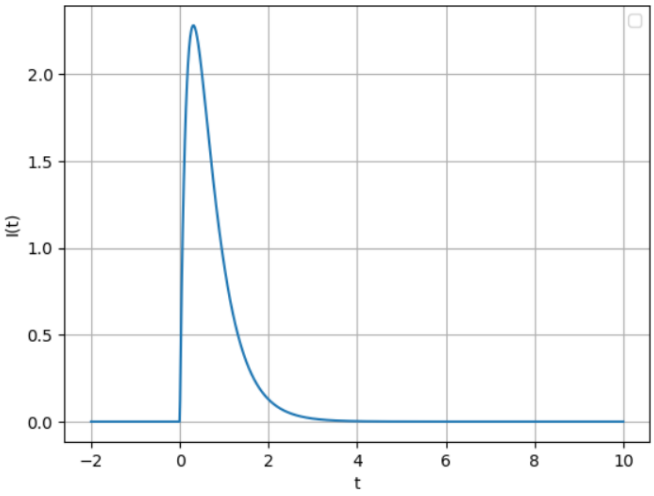


Figura 9. *Glitch di corrente*

in un MOSFET sono le regioni di svuotamento, dove l'accumulo di carica avviene per drift (effetto funneling). Mentre, al di fuori della zona svuotata l'accumulo avviene per diffusione.

Questi fenomeni generano, come detto, un glitch di tensione o corrente con equazione:

$$I(t) = I_o(e^{\frac{-t}{\tau_1}} - e^{\frac{-t}{\tau_2}}) \quad (9)$$

In cui I_0 dipende dall'energia della particella incidente, τ_1 rappresenta la costante di tempo di charge collection della giunzione, e dipende dalla tecnologia utilizzata e dalle condizioni di lavoro del transistor, τ_2 è la costante di tempo relativa al tempo di formazione degli ion-track, dipendente dalla tecnologia e dall'energia della particella.

Mentre, la carica accumulata o raccolta non è altro che l'area al di sotto della curva I(t).

$$Q = \int_0^\infty I(t)dt = I_0(\tau_1 - \tau_2) \quad (10)$$

Inoltre, l'accumulo di carica diventa sempre più significativo quanto più la tecnologia è miniaturizzata, la tensione di alimentazione è bassa e la capacità dei nodi è ridotta.
Per questo motivo, bisogna prestare attenzione a questa varietà di guasti con le tecnologie odierne [26].

3. Implementazione di Strategia IA per il Conteggio di Persone

In questo capitolo si parlerà in primo luogo delle motivazioni che hanno spinto alla scelta di questa rete e di come effettivamente la rete funzioni, con la scelta di alcuni parametri che hanno permesso a una riduzione dell'errore. In un secondo momento si introdurrà la metrica utilizzata e la valutazione del modello con un dataset adatto.

3.1 Scelta della rete

Un iniziale studio bibliografico ha evidenziato che per l'elaborazione delle immagini, le reti più adatte sono appunto le CNN introdotte nei concetti preliminari. Quindi, un primo obiettivo della presente attività di tesi è stato ricercare la rete adatta al nostro scopo.

Le reti testate sono state la ResNet50, l'AlexNet e la VGG16. Tutte quante non sono riuscite ad allenarsi adeguatamente per raggiungere un errore soddisfacente nel conteggio. Questo, come anticipato è dovuto alla struttura di queste reti, adatte per risolvere problemi di classificazione. Con problema di classificazione si intende appunto classificare il contenuto dell'immagine.

Dunque, bisognava individuare una rete adatta al rilevamento delle persone, e la scelta è ricaduta appunto sui modelli YOLO. Questa rete, per questioni strutturali che vedremo, è molto veloce nel rilevamento; e poiché la rete impiegata ha bisogno di elaborare immagini video molto velocemente per assicurare un conteggio il più possibile corretto, la velocità dev'essere garantita.

Infatti, secondo alcuni studi implementando l'inferenza di una YOLOv3 su una GPU NVIDIA GeForce RTX 3050 questa impiegherebbe 33,1ms a elaborare un'immagine (30.2fps) [35].

3.1.1 YOLO e alcune sue Versioni

La prima versione della YOLO è stata sviluppata presso la Cornell University da Joseph Redmon, Santosh Divvala, Ross Girshick e Ali Farhadi nel 2015.
Questa tipologia di rete proponeva una rapida elaborazione delle immagini. Addirittura, in un aggiornamento del 2016, si propone una sua versione più rapida in grado di lavorare a 155fps. Per le tecnologie che erano presenti all'epoca fu una rivoluzione. La caratteristica principale che la contraddistingue è proprio la velocità. Infatti, ha delle prestazioni in velocità non paragonabili ad altre architetture di quegli anni. Bisogna però tener conto che commette più errori di localizzazione ma ha molte meno probabilità di prevedere falsi rilevamenti [28].
Una seconda versione di YOLO fu rilasciata nel 2016 da Joseph Redmon, Ali Farhadi sempre presso l'Università di Cornell. Questa versione riesce a raggiungere i 67 fps, riuscendo a distinguere fino a 9000 categorie di oggetti. Questo grazie all'introduzione dei layer di batch normalizzation visti nella sezione 2.1.2, ma soprattutto grazie all'introduzione delle *anchors* (caselle di ancoraggio), che permettono un migliore rilevamento [29][30].
La YOLOv3 fu rilasciata sempre dagli stessi autori nell'Aprile del 2018. Questa versione è un po' più pesante, avendo più *pesi* e *bias*. Impiega delle ancore multiple; permettendo di rilevare oggetti in maniera più efficiente, mantenendo una precisione alta [29][31].
Le versioni YOLO ad oggi arrivano fino all'ottava. Ci sembra opportuno concludere la descrizione fino alla terza versione poiché, questa è la versione impiegata.

3.1.2 YOLO per il Conteggio di Persone

Questa rete è nota, come detto, per la sua velocità e precisione ed è utilizzato in vari contesti, come la sorveglianza video, la guida autonoma, l'analisi delle immagini mediche e molto altro ancora. Questo anche grazie alla sua struttura, che permette di classificare e rilevare in una volta sola gli oggetti, da qui il nome You Only Look Once. Questa precisazione è dovuta al fatto che le altre reti usate per l'*Object Detection* hanno bisogno di più passaggi per il rilevamento e la classificazione. Ci basti pensare alle

Region-based Convolutional Neural Networks (R-CNN). Queste forniscono risultati più accurati ma con una maggiore lentezza. Questa rete, infatti, prima propone delle Regioni di Interesse (ROI), e in un secondo momento queste verranno classificate [8].

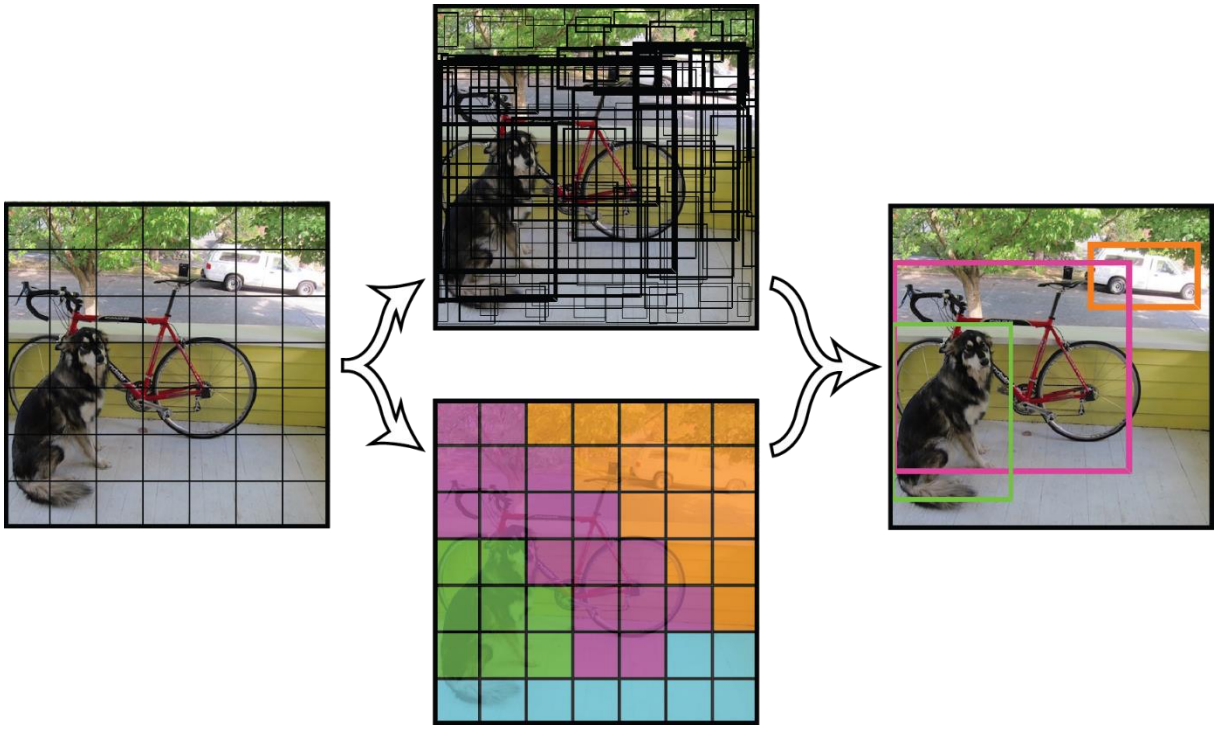


Figura 10. *Classificazione e rilevamento in una YOLO. [36]*

1. YOLOv3

La struttura della YOLO può essere suddivisa in più componenti. In primo luogo, la rete si affida a un *Backbone* per l'estrazione delle feature dell'immagine. Questa solitamente è composta da una CNN. In particolare, la YOLOv3 è stata progettata con un *Backbone* di Darknet-53 [31]. Darknet-53 è una variante dell'architettura ResNet. Con 53 strati convoluzionali è stata progettata proprio per l'object detection [38].

In serie a questo componente vi è un componente per il rilevamento, il layer di *detection*. Sostanzialmente in questa fase l'immagine viene divisa in celle e per ogni cella vengono proposte delle *bounding box* (BB) sfruttando la dimensione delle ancore settate nella

rete. Le *bounding box* rappresentano un riquadro dell'oggetto rilevato. Nel dettaglio, questa è composta nel seguente modo (x, y, w, h) dove x e y sono le coordinate spaziali dell'angolo in alto a sinistra o del centro del riquadro (dipende da come è stata allenata la rete), mentre w e h rappresentano la larghezza e l'altezza. A ogni *bounding box* è associato un numero che definisce la classe predetta, e delle *confidences* che rappresentano le probabilità che l'oggetto rilevato appartenga a una determinata classe. Il numero di *bounding boxes* che viene proposto per cella è pari al numero di maschere con cui settiamo la rete. Quindi, ogni cella propone un numero di bounding boxes pari al numero di maschere e la forma delle *bounding boxes* è ricavata dalle ancore proposte. Le maschere vengono utilizzate per specificare quali anchor box di una cella sono responsabili di predire certe classi di oggetti. Questo aiuta il modello a focalizzarsi sulle caratteristiche degli oggetti che vuole rilevare. Ovviamente, la proposta delle BB è fatta sulla base delle feature estratte dalla Darknet-53. La parte finale consiste in un filtraggio delle BB mediante una soglia sulla *confidence*, per poi essere nuovamente filtrate da un algoritmo che permette di eliminare quelle associate allo stesso oggetto (Algoritmo NMS).

Di seguito viene mostrato un esempio grafico di come le BB vengono proposte.

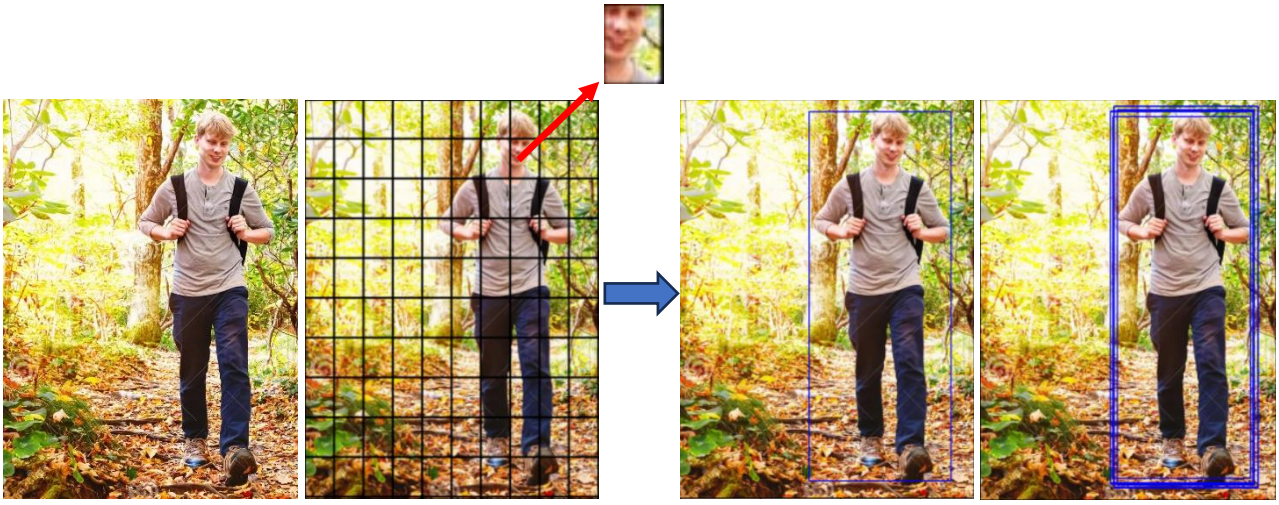


Figura 11. *Esempio proposta bounding boxes di una YOLO. [32]*

In Figura 9 sono mostrate le possibili le bounding box filtrate con una soglia sulla confidence. In realtà quelle proposte dalla rete sono molte di più (numero di celle* numero di maschere).

2. Bounding Boxes

Durante la fase di addestramento la rete utilizza come funzione di perdita la *binary cross-entropy* tra la *confidence* proposta e quella data dalla ground truth. Nel caso YOLO si parlerà di ground truth e non di label, poiché le label si riferiscono a un'etichetta nei problemi di classificazione. La funzione può essere espressa nel seguente modo:

$$BCE = \sum_{i=0}^{S^2} \sum_{j=0}^{B} W_{ij}^{obj} * [\hat{C}_i^j * \log(C_i^j) - (1 - \hat{C}_i^j) * \log(1 - C_i^j)] \quad (11)$$

Dove S^2 rappresenta il numero di celle in cui è stata divisa l'immagine, j si riferisce alla j-esima *bounding box*, e $\hat{C}$ e C sono rispettivamente la *confidence* predetta e quella ground truth dell'i-esima cella e j-esima bounding box, mentre, W è un peso associato ad essa [33].

Per la predizione della *confidence*, in fase di allenamento, la rete si affida al parametro Intersection over Union (IoU) tra la predizione e quella della ground truth, per far si che prevalgano le BB più simili a quelle che effettivamente noi vogliamo. L'espressione della *confidence* può essere ottenuta nel seguente modo:

$$C_i^j = P_{i,j}(Object) * IoU_{pred}^{truth} \quad (12)$$

Dove P è una funzione riferita all'oggetto che vogliamo rilevare [34].

La rete neurale, inoltre, prevede 4 coordinate per ogni BB: t_x, t_y, t_w, t_h. Queste coordinate rappresentano le trasformazioni da applicare alla *bounding box* di riferimento per ottenere la posizione e le dimensioni della *bounding box* predetta. Le equazioni mostrate di seguito descrivono come calcolare le coordinate della *bounding box* predetta basate sulle predizioni della rete neurale e sulle dimensioni dell'*anchor box*.

$$b_x = \sigma(t_x) + c_x \quad (13) \qquad b_y = \sigma(t_y) + c_y \quad (14)$$

$$b_w = p_w * e^{t_w} \quad (15) \qquad b_h = p_h * e^{t_h} \quad (16)$$

Dove c_x e c_y rappresentano le coordinate dell'angolo in alto a sinistra della cella rispetto all'angolo in alto a sinistra dell'immagine, σ è la funzione sigmoidea, e p_w e p_h rappresentano rispettivamente larghezza e altezza dell'ancora [31].

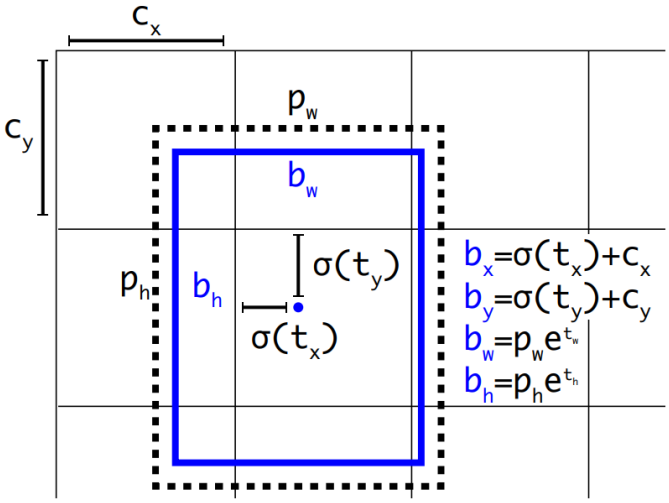


Figura 12. *Elaborazione di una bounding box. [31]*

Se g_x, g_y, g_w, g_h rappresentano la BB della ground truth, allora i valori t riferiti a essa saranno:

$$\sigma(\hat{t}_x) = g_x - c_x \quad (17) \qquad \sigma(\hat{t}_y) = g_y - c_y \quad (18)$$

$$\hat{t}_w = \log\left(\frac{g_w}{p_w}\right) \quad (19) \qquad \hat{t}_h = \log\left(\frac{g_h}{p_h}\right) \quad (20)$$

In fase di allenamento, per l'ottimizzazione della rete, si usa la somma degli *square error* come funzione di perdita [33].

$$SE = \sum_{i=0}^{S^2}\sum_{j=0}^{B} W_{ij}^{obj} * \left[\left(\sigma(t_x)_i^j - \sigma(\hat{t}_x)_i^j\right)^2 + \left(\sigma(t_y)_i^j - \sigma(\hat{t}_y)_i^j\right)^2\right] + \sum_{i=0}^{S^2}\sum_{j=0}^{B} W_{ij}^{obj} * \left[\left(\sigma(t_w)_i^j - \sigma(\hat{t}_w)_i^j\right)^2 + \left(\sigma(t_h)_i^j - \sigma(\hat{t}_h)_i^j\right)^2\right] \quad (21)$$

3. *Algoritmo NMS*

La fase finale, come anticipato, prevede un filtraggio delle *bounding boxes*. Inizialmente queste vengono filtrate con una soglia sulla *confidence*, prendendo, quindi, quelle con la probabilità più alta di contenere un oggetto d'interesse.

Poiché, le BB riferite allo stesso oggetto sono molteplici e con una *confidence* troppo alta per filtrarle in questo modo, è necessario filtrarle ulteriormente. Dunque, si è

pensato di utilizzare un algoritmo che mettesse in relazione le aree delle varie BB in modo da eliminare quelle relazionate allo stesso oggetto.

L'algoritmo di cui si parla è il Non-Maximum Suppression (NMS). Questo consiste nel calcolo dell'IoU tra le bounding boxes, filtrandole con una soglia proprio su questo parametro.

L'IoU non è altro che L'Intersection over Union:

$$IoU = \frac{Intersezione\ delle\ aree}{Unione\ delle\ aree} \tag{22}$$

Ma perché non si usa solo l'intersezione delle aree?

La risposta è semplice. Se una persona o un oggetto è in lontananza, la sua bounding box potrebbe essere racchiusa all'interno di un'altra bounding box riferita a una persona/oggetto che si trova più vicino. Allora, conviene usare anche il dato dell'unione delle aree, e quindi, l'IoU.

Inoltre, l'IoU non viene calcolato per bounding box di classi differenti. Questo è ovvio perché potremmo voler rilevare oggetti che potrebbero essere sovrapposti ad altri oggetti (es. persona all'interno di un'automobile, un cane e il suo guinzaglio, uomo con una cravatta…).

Bene, ora che abbiamo chiarito perché l'algoritmo usa questo parametro per il filtraggio non ci resta che capire come effettivamente funziona.

Supponiamo che in input all'algoritmo ci sia un elenco di BB (I) con le rispettive confidences (C) e una certa soglia (T), e che in uscita questo mi dia un elenco di BB filtrato (O), allora l'algoritmo seguirà i seguenti passi [34]:

1. Prende la bounding box con la confidence più alta in I, l'elimina da I e l'aggiunge a O.
2. Calcola l'IoU tra questa e tutte le altre bounding box.
3. Se l'IoU supera la soglia T allora la BB relativa a quell'IoU viene eliminata da I.
4. Ripeto l'operazione prendendo la BB con la confidence più alta in I…

Questo processo viene effettuato fino all'esaurimento di bounding boxes in I.

Di seguito si riporta un esempio grafico del processo dell'algoritmo.

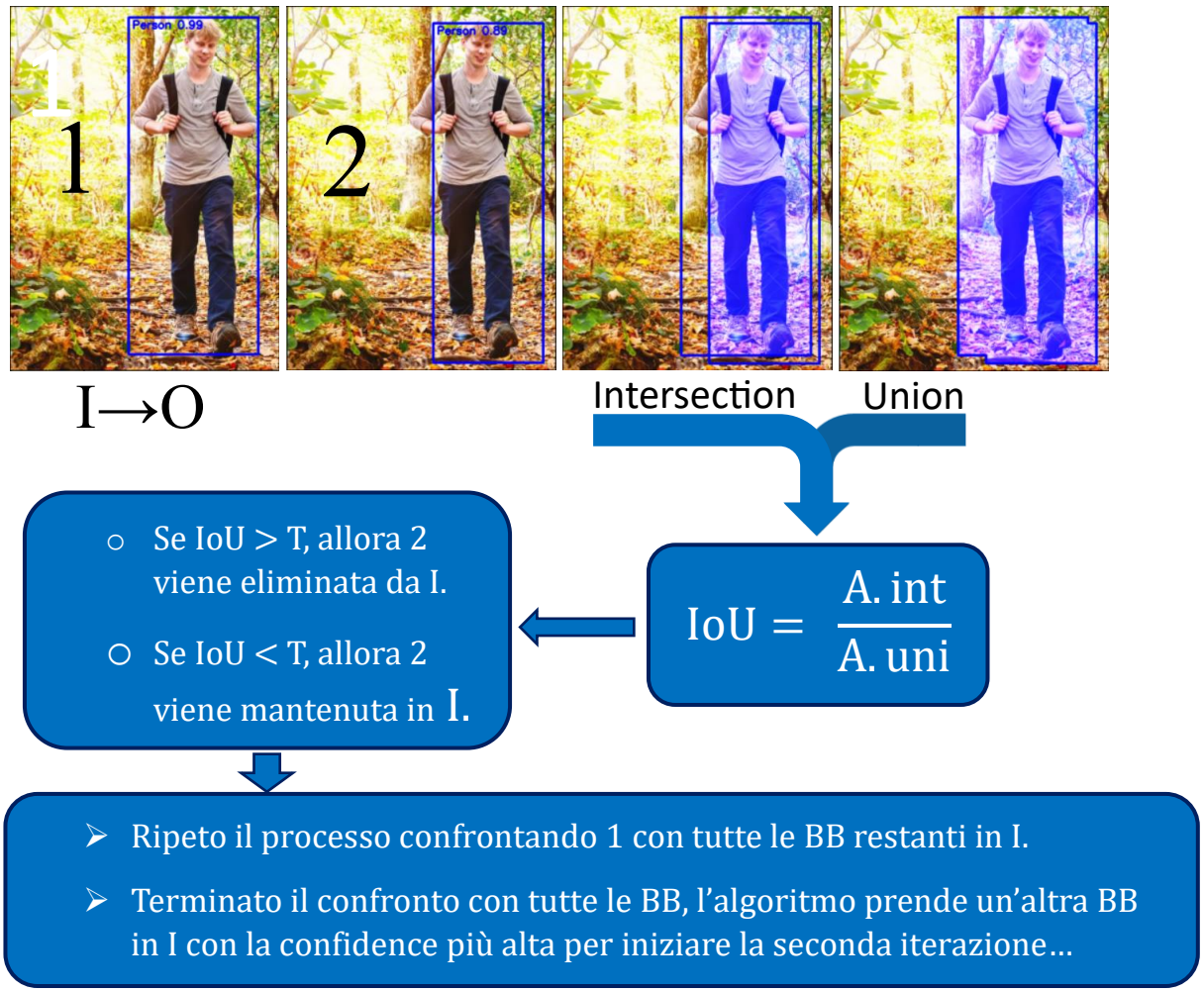


Figura 13. *Algoritmo NMS, rappresentazione grafica. [32]*

Per l'analisi della YOLOv3 è stato usato un threshold di 0,7. La scelta è stata fatta variandolo da 0 a 1 con un passo di 0,05 analizzando il comportamento della rete su un dataset di cui parleremo nella sezione 3.2. Per il threshold sulla *confidence* il procedimento è stato simile, arrivando alla conclusione che quello che provocava un errore più basso fosse una soglia di 0.2.

3. Informazioni sulla Rete Analizzata

La rete utilizzata prevede 278 layers tra cui 75 strati convoluzionali. Gli ultimi due livelli convoluzionali si dividono in tre vie. Il primo per la predizione delle bounding box, il secondo per la predizione della confidence, e l'ultimo per la predizione della classe. Inoltre, i layers finali prevedono un reshape per la manipolazione dei dati in uscita dagli strati convoluzionali. Questo per avere dei dati organizzati in maniera adeguata al calcolo delle BB.

Il numero di parametri della rete sono circa 62 milioni, e si suddividono in parametri allenabili e non allenabili.

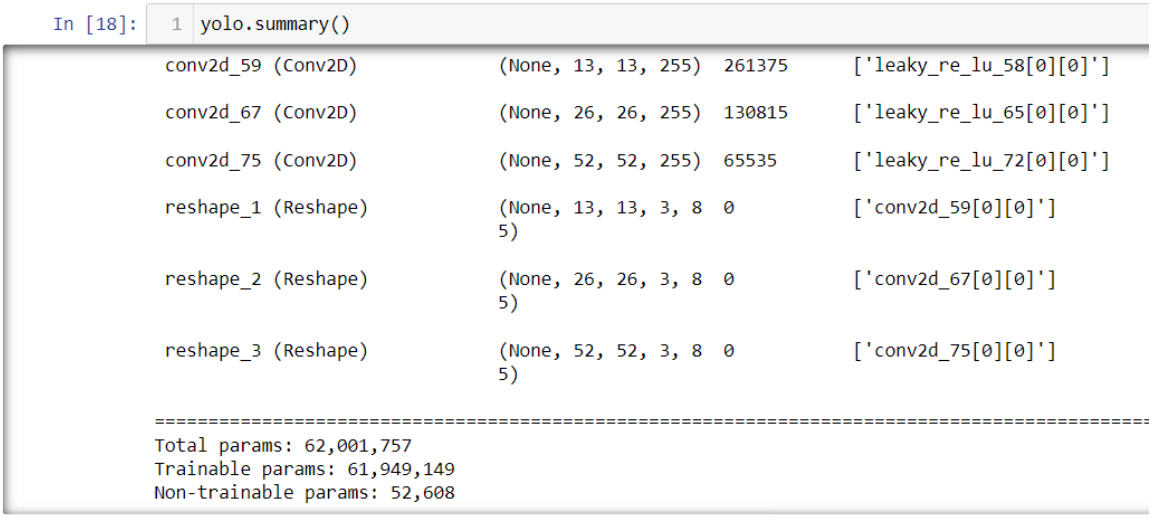


Figura 14. *Parametri e uscita della rete.*

I parametri non allenabili sono dei parametri incorporati nei layers, e che non possono essere modificati tramite un processo di ottimizzazione.

3.2 Test e Valutazione del modello

In questa sezione, si parlerà dei datasets utilizzati per la rete, la metrica impiegata per la valutazione del modello, e infine si mostreranno i risultati ottenuti.

3.2.1 Datasets nel Pre-Allenamento e nel Test

Il modello della rete analizzata, come già detto, è stata pre-addestrata [10]. Il dataset utilizzato in fase di addestramento è un dataset molto conosciuto nel campo dell'object detection; ed è il COCO Dataset [37]. Il dataset contiene più di 200k immagini etichettate, con circa 80 categorie di oggetti.

Il dataset utilizzato per la valutazione del modello è il MICC People Counting [6]. In questo dataset sono presenti circa 3350 frames video (640x480) di persone in una stanza in tre situazioni differenti.

- Situazione FLOW: in questi frames si vedono persone fluire all'interno della stanza. Come a simulare un ingresso a un'attività commerciale o un passaggio pedonale.
- Situazione GROUPS: in questa situazione vengono ritratti due gruppi di persone che parlano tra di loro e si muovono all'interno della stanza. Queste situazioni si propongono spesso in luoghi pubblici, dove le persone dialogano tra di loro in gruppo.
- Situazione QUEUE: nella terza e ultima situazione è stato chiesto alle persone di mettersi in fila. Come a simulare una coda a una cassa di un'attività commerciale.

Le ground truth prevedono la localizzazione delle persone all'interno dell'immagine, secondo lo schema che abbiamo visto prima: (x,y,w,h) dove x e y in questo caso si riferiscono al centro della bounding box.

Tra i frames disponibili ne sono stati utilizzati solo 180 per la valutazione del modello, per questioni di tempistiche e di confronto con l'analisi ai SE. La selezione è stata fatta prendendo casualmente 60 immagini per ogni situazione.

3.2.2 Metrica di Valutazione

Le metriche scelta nella valutazione del modello sono il Mean Absolute Error (MAE) e il Mean Squared Error (MSE), descritto nella sezione 2.1.2, l'*accuratezza* e il T*rue Positive Rate* (TPR).

Per il calcolo dell'accuratezza abbiamo dovuto far riferimento alla seguente formula matematica:

$$ACCURACY = \frac{TP+TN}{TP+TN+FP+FN} * 100 \quad (23)$$

Dove TP, TN, FP ed FN sono rispettivamente i True Positive, True Negative, False Positive, False Negative.

In particolare, gli indici sopracitati indicano rispettivamente:

- True Positive (TP): numero di persone contate correttamente dalla rete.
- False Positive (FP): numero di persone contate erroneamente nella rete anche se non presenti nell'immagine.
- False Negative (FN): numero di persone che, pur essendo presenti nell'immagine, non sono state contate dalla rete.
- True Negative (TN): numero di volte in cui la rete riconosce correttamente l'assenza di persone nell'immagine (conteggio nullo).

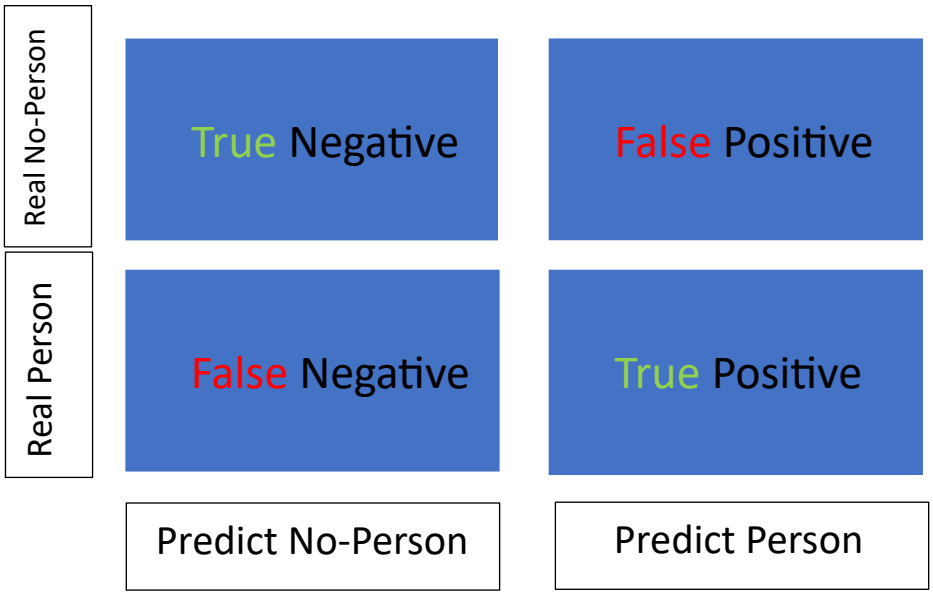


Tabella 1. *Organizzazione dei valori di TP, TN, FP, FN all'interno di una matrice di confusione.*

È stato inoltre valutato il tasso dei veri positivi (True Positive Rate, o TPR), con la seguente formula.

$$TPR\% = \frac{TP}{TP+FN} * 100 \quad (24)$$

Questo parametro ci aiuta a capire quante persone riesce effettivamente a contare la rete in funzione del numero totale di persone presente nell'immagine (TP+FN), essendo TP le persone contate correttamente e FN le persone che sono presenti nell'immagine ma che non sono state contate correttamente.

3.2.3 Valutazione del Modello

Per la valutazione, come detto nella sezione precedente, sono state utilizzate 180 immagini ridimensionate a 416x416, che è la reale dimensione dell'input della rete. Nel ridimensionamento è stato usato il metodo d'interpolazione spline cubica. Questo metodo utilizza l'interpolazione polinomiale cubica per stimare i valori dei pixel nell'immagine ridimensionata in base ai valori dei pixel circostanti nell'immagine originale. Questo può produrre risultati più lisci e dettagliati rispetto ad altri metodi di interpolazione.

I valori dei pixels sono stati normalizzati di 255 per avere un valore compreso tra 0 e 1 in formato float32, per tutti e tre i canali RGB. Questo perché la rete è stata allenata normalizzando i pixels. Questa pratica è molto utilizzata e aiuta a far convergere il modello più rapidamente in allenamento.

Inoltre, le ground truth sono state elaborate prendendo solamente il numero di persone. E quindi, contando il numero di BB all'interno di ogni file testo legato alle immagini utilizzate.

La scelta delle soglie è stata fatta valutando il modello sempre con i dati che abbiamo appena descritto (180 immagini). È stata fatta variare la soglia NMS fissando inizialmente una soglia sulla confidence di 0,5. La soglia NMS scelta è di 0,7 e, come anticipato, è stata scelta valutando la rete con soglie che andavano da 0,1 a 0,9 con passo 0,05. Una volta individuata la soglia NMS è stata fatta variare la soglia sulla confidence sempre con passo 0,05 sempre da 0,1 a 0,9. Ci si è accorti che la soglia con minor errore

era di 0,2. Una volta individuata la soglia sulla confidence è stata fatta variare nuovamente la soglia NMS per avere una conferma che la scelta fosse giusta.

Di seguito viene riportata la tabella per la valutazione della rete, utilizzando la metrica introdotta precedentemente che con le soglie appena descritte.

ACCURACY%	86,2%
TPR%	90,5%
MAE	0,72
MSE	1,29

Tabella 2. *Valutazione del modello.*

Si può notare che il TPR è più alto dell'accuratezza. Questo perché nel TPR non vengono presi in considerazione i falsi positivi, che comportano un conteggio eccessivo delle persone. L'accuratezza non è molto alta, ma mi indica che la rete riesce a contare correttamente 86 frames su 100. Inoltre, bisogna considerare anche di quanto effettivamente la rete sbaglia.

Il Mean Absolut Error è di 0,72. Questo indica che la rete sbaglia nel conteggio mediamente di quasi una persona sia in negativo che in positivo, senza quindi considerare la direzione. Il Mean Squared Error e anche relativamente basso, questo indica una bassa dispersione dei dati rispetto al valore corretto. Poiché il MSE penalizza maggiormente gli errori più grandi rispetto al MAE, un valore di 1.29 suggerisce che potrebbero esserci alcuni errori più grandi che contribuiscono a questo valore più alto rispetto al MAE.

Un valore del Mean Squared Error di una rete neurale maggiore rispetto al Mean Absolute Error suggerisce che gli errori predetti dalla rete sono maggiormente influenzati da errori di grandi dimensioni. Il MSE attribuisce un peso maggiore agli errori più grandi rispetto al MAE, poiché calcola la media dei quadrati degli errori anziché la media dei valori assoluti degli errori. Quindi, se il MSE è più alto del MAE, potrebbe indicare una maggiore sensibilità della rete agli errori di grandi dimensioni.

Nella Figura 15 invece, viene mostrata la matrice di confusione dei TP, TN, FP, FN del test effettuato.

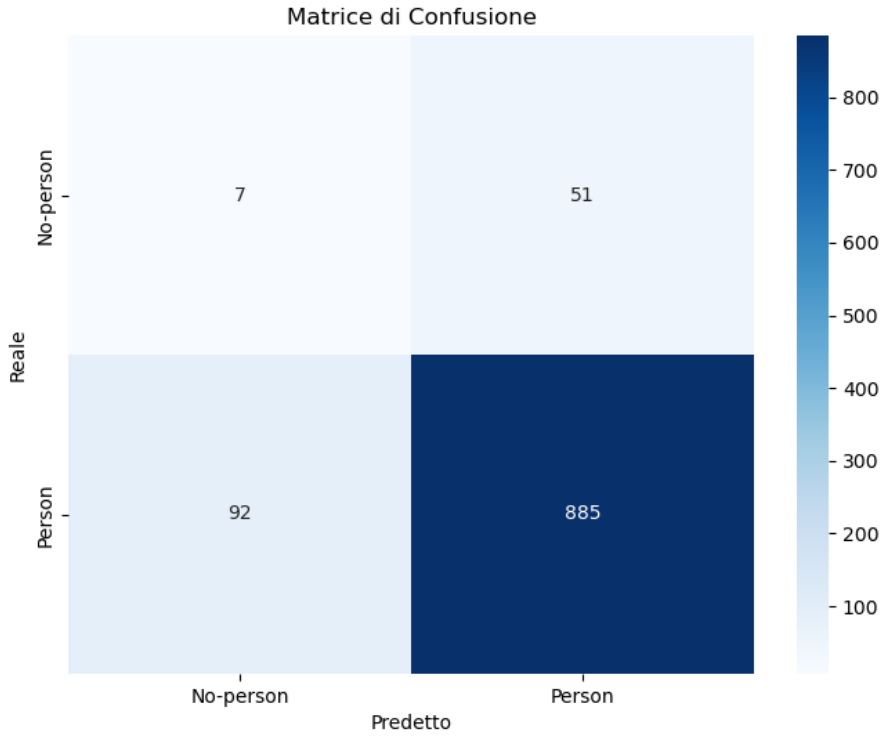


Figura 15. *Matrice di confusione caso fault free.*

La media delle persone proposte per immagine è di circa 5,4 ($(TP + FN)/180$).

Notiamo che TN è uguale a 7 che è uguale al numero di volte in cui l'immagine ha zero persone. Quindi, tutte le volte che alla rete gli si è presentata un'immagine con zero persone, questa ha contato correttamente. Dunque, i falsi positivi sono dovuti unicamente a un errata eliminazione delle bounding boxes multiple. Questo perché non ci sono oggetti che la rete identifica come persone.

Quindi, ricapitolando la rete conta correttamente 9 su 10 volte, e quando sbaglia lo fa con un errore assoluto poco meno di una persona. Sicuramente per il nostro scopo risulta soddisfacente. Anche se bisognerebbe valutare il suo comportamento in situazioni affollate. In quel caso l'accuratezza sicuramente si riduce. Questo è dovuto al fatto che la rete è stata allenata per riprodurre BB che inquadrino l'intero corpo del soggetto, e quindi, si avrà una maggiore sovrapposizione delle bounding boxes dovuta alla troppa vicinanza tra le persone, aumentandone l'errore nel conteggio. Questo perché un'eccessiva sovrapposizione causa uno scartamento di BB mediante l'algoritmo NMS.

Per valutare il modello a 360° è necessario anche valutare, secondo le metriche scelte, anche il suo comportamento al variare del numero di persone all'interno dell'immagine.

Di seguito viene mostrata una tabella che mostra la variazione del TPR e dell'accuratezza al variare del numero di persone.

Numero di persone	Accuratezza%	TPR%
1	100%	100%
2	82,75%	100%
3	76,53%	90,19%
4	77,27%	77,27%
5	88,33%	88,33%
6	100%	100%
7	90,88%	94,92%
8	73,86%	73,86%
9	87,3%	87,3%

Tabella 3. *Variazione TPR e Accuratezza al variare del numero di persone.*

Notiamo che entrambi i parametri di valutazione oscillano molto al variare del numero di persone con un minimo del 73,86% quando il numero di persone è 8, e un massimo del 100% quando le persone sono 1 o 6 (86,93 ± 13,07). Si ha un'oscillazione del 26% da un estremo all'altro.

È opportuno sottolineare che nel calcolo di TPR e delle accuratezze nelle singole situazioni, non è mostrato il caso in cui le persone sono pari a 0. Questo perché il TPR sarebbe indefinito essendo il denominatore pari a 0, e l'accuratezza invece è uguale al 100% poiché, le persone predette sono sempre 0, quindi TP, FP ed FN saranno pari a 0, perciò sia al numeratore che al denominatore resterà TN.

Bisogna inoltre notare come il TPR è sempre maggiore o uguale all'accuratezza. Questo perché nell'accuratezza incidono i falsi positivi e non si sta tenendo conto dei veri negativi (TN=0 nei singoli casi). Notiamo inoltre, che TPR e accuratezza spesso coincidono. È dovuto al fatto che in quei casi in cui questo accade non ci sono falsi positivi e quindi, le equazioni di accuratezza e TPR coincidono.

4. Analisi dell'Affidabilità della Strategia IA per il Conteggio di Persone

In questo capitolo verrà esposto l'analisi della rete ai soft error, una tipologia di errori causati da guasti transitori e introdotta nel capitolo sui concetti preliminari.

L'implementazione di una rete prevede l'utilizzo di una memoria RAM per memorizzare i pesi e, poiché, sappiamo che questo tipo di memorie sono sensibili ai guasti transitori, risulta opportuno simularli e valutare il loro impatto sul conteggio di persone.
L'effetto dei soft error sarà quello di generare un *bit flip* e, quindi, di variare uno o alcuni bit del codice che rappresenta il peso. Ovviamente, la variazione può essere positiva se la variazione è 0→1, o negativa se la variazione è 1→0.
Questo ci permetterà di poter attuare alcune strategie per rendere la rete più *reliable* e *safe* e, quindi, evitare un conteggio errato, producendo delle situazioni che, in caso di pericolo, potrebbero generare ancora più danni.
Inoltre, come già detto nei capitoli precedenti, la velocità di calcolo potrebbe essere supportata da degli hardware accelerators, anch'essi sensibili a questa tipologia di errori.

4.1 Metodologia per la Simulazione dei Soft Error

Essendo i pesi dei numeri decimali, bisogna fare una piccola introduzione allo standard IEEE 754 floating-point.
Questo standard rappresenta i numeri decimali in 3 diversi formati: 32 bit, 64 bit, 128 bit.
La rappresentazione scelta è quella a 32 bit (precisione singola), che è quella più conforme per la memorizzazione di molti pesi come nel nostro caso. Inoltre, vedremo che la rete non necessita di un'eccessiva precisione nella rappresentazione del peso.
Il peso W può essere rappresentato nel seguente modo:

$$W = (-1)^S \times 2^{E-B} \times (1, M) \tag{25}$$

Dove S è il bit di segno, E è l'esponente, M è la mantissa, e B rappresenta un bias previsto dallo standard di 127.
Ci aspettiamo già, quindi, che l'impatto maggiore dal punto di vista numerico lo daranno i bit dell'esponente. Sono quelli che cambiano radicalmente il valore del peso.
La suddivisione dei bit è rispettivamente:

Segno: 1 bit

Esponente: 8 bit

Mantissa: 23 bit

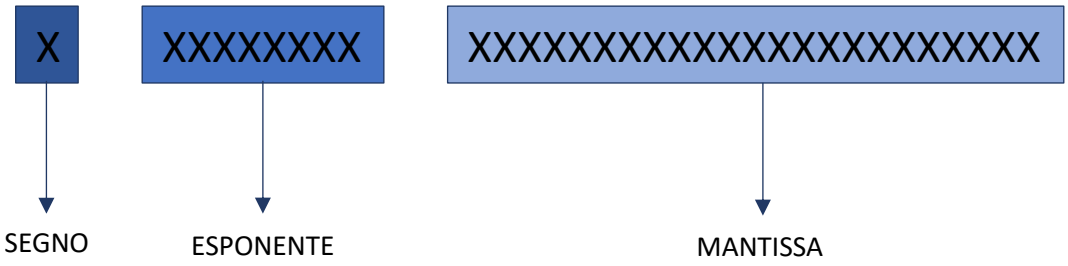


Figura 16. *Rappresentazione standard IEEE 754 floating-point.*

Poiché, il numero dei pesi è dell'ordine dei milioni, l'analisi ai SE non può essere fatta per tutti i bit di tutti i pesi.

I bit di particolare interesse sono il bit più significativo dell'esponente, quello della mantissa, e il bit di segno.

L'algoritmo previsto dall'analisi è il seguente:

1. Estrazione dei pesi di un layer mediante la funzione *get_weights()* (funzione disponibile su Tensorflow).
2. Salvataggio dei pesi in situazione fault free in un array.
3. Estrazione di un solo peso.
4. Conversione del peso da decimale a 32 bit seguendo lo standard visto.
5. Generazione di un SE in uno dei tre bit desiderati.
6. Conversione del peso da 32 bit a decimale.
7. Inserimento del peso alterato in una copia dell'array fault free.
8. Inserimento dei pesi nel layer.
9. Test e valutazione della rete vista nel capitolo 3.
10. Inserimento dell'errore commesso in un array.
11. Ripeto dal punto 3…

Questo procedimento si termina quando sono stati alterati tutti i pesi del layer. Ovviamente, il procedimento è stato fatto per più layers e in tutte e tre i contesti proposti.

Ovvero, alterazione del bit di segno, alterazione del bit più significativo dell'esponente, e alterazione del bit più significativo della mantissa.

Nella valutazione della rete in ogni singola situazione sono state utilizzate le metriche del capitolo 3. Ovviamente, poiché, i pesi sono molti, per ogni layer analizzato è stata fatta una media degli errori commessi salvati nell'array del punto 10.

4.2 Analisi dei Risultati

Questa sezione verrà suddivisa in tre ulteriori parti. Ovvero, l'analisi dei layers simulando il SE sul bit segno, simulandolo sul bit più significativo dell'esponente, e simulandolo sul bit più significativo della mantissa.

Poiché, la rete presenta molti layer (278), nell'analisi si sono inclusi solo i layer convoluzionali (75). I layers non analizzati sono layer di batch normalizzation e altri layer come zero_padding, leaky_relu e altri che però non prevedono pesi, essendo layer che sostanzialmente applicano una funzione o ridistribuiscono gli output del layer precedente diversamente. L'esclusione dei layers di batch normalizzation è dovuta al tempo necessario per l'analisi di tutti i layers che risulta eccessivo per la durata di questa ricerca di tesi. Inoltre, i layers convoluzionali sono il fulcro per l'estrazione del pattern dell'immagine. Perciò, sicuramente è necessario approfondire un'analisi di questi. C'è anche da dire che i layers convoluzionali comprendono la quasi totalità dei parametri di tutta la rete (61,85 mln su 62 mln).

4.2.1 Iniezione dei SE sul bit di segno

Poiché i layers sono molti risulta scomodo visualizzare la metrica tramite una tabella. Quindi, si è optato per una rappresentazione grafica che mostra sulle ascisse il numero del layer convoluzionale, e sulle ordinate l'accuratezza percentuale o il TPR percentuale.

Di seguito i grafici di accuratezza e True Positive Rate.

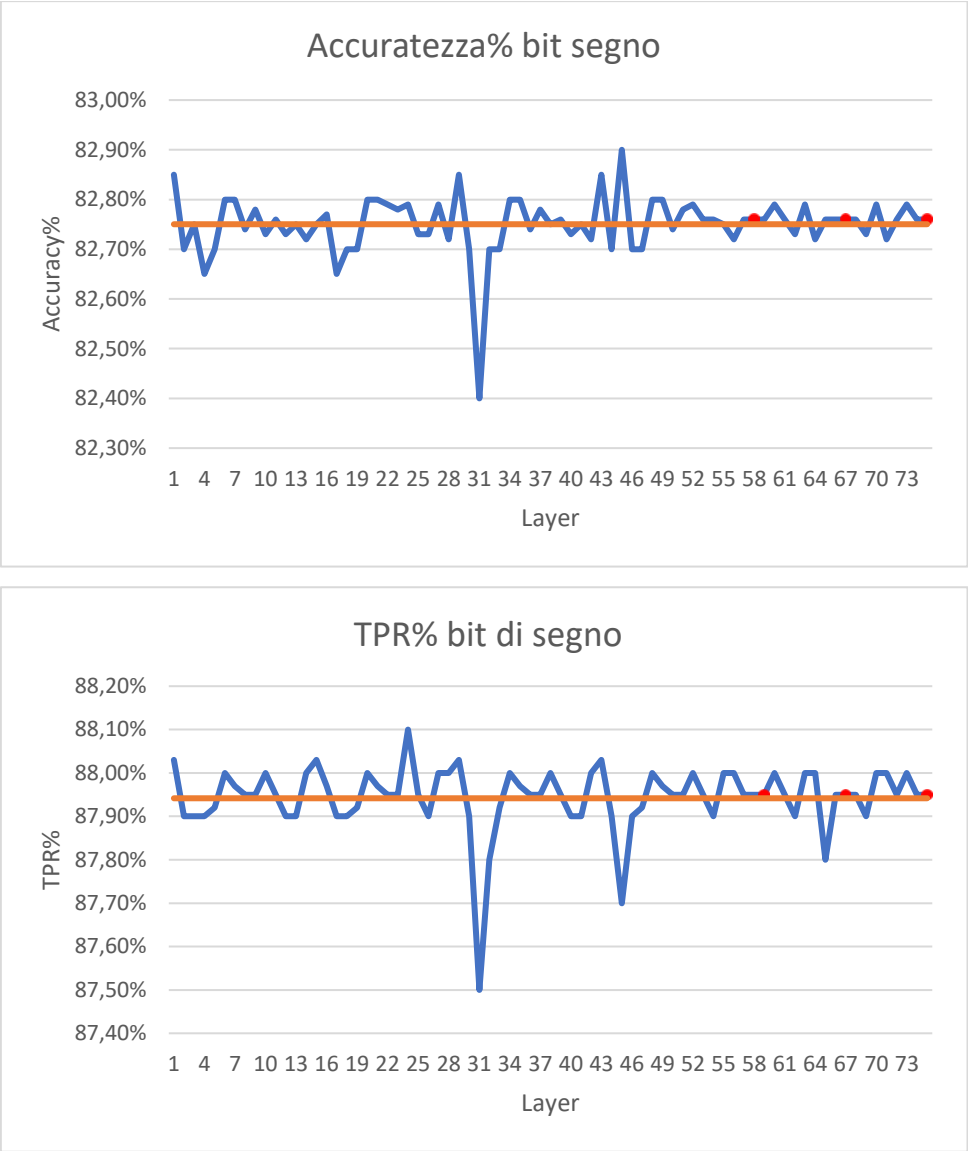


Figura 17. *Rappresentazione Accuratezza relativi alla simulazione dei SE bit di segno.*

In cui, l'accuratezza media e il TPR medio (linee arancioni) sono rispettivamente 82,75% e 87,94%.

Ricordiamo che le uscite sono tre (predizione bounding box, confidence e classe) e sarebbero gli output dei layers 59, 67 e 75 (punti rossi nei grafici).

Notiamo quindi, una piccola riduzione di queste rispetto al caso fault free, in particolare, del TPR. Questo indica che i falsi positivi in media sono aumentati. Questo è dovuto alle immagini in uscita dai filtri convoluzionali che sono lievemente distorte, poiché il peso del filtro è stato alterato.

Inoltre, bisogna anche considerare l'errore assoluto e la dispersione dell'errore rispetto alla condizione in assenza di guasti.

Anche questi, come previsto, peggiorano ma lievemente. Il MAE sale da 0,72 (caso fault free) a 0,795, mentre il MSE sale da 1,29 a 1,4. Nonostante il peggioramento la rete continua a sbagliare in media di quasi una persona.

Bisogna, considerare anche l'omogeneità delle metriche, che restano comunque in un range che va da 82,7% ±0,2% per l'accuratezza e 87,8% ±0,3% per il TPR per tutti i layers convoluzionali.

In egual modo per l'errore, con un MAE che va da 0,795±0,005 e un MSE di 1,4±0,01.

In conclusione, quindi, se si dovesse verificare un SE sul bit di segno dei pesi di questa rete, il risultato non è grave. Anzi, la rete continua a mantenere un'accuratezza discreta, continuando a contare le persone.

Per capire meglio cosa sta succedendo alla rete è opportuno valutare la relativa matrice di confusione (Figura 18). Per facilitare la visualizzazione dei risultati è stata fatta una somma di tutti i TP, TN, FP ed FN di tutte le immagini presentate, come nell'analisi fault free. Questo è stato fatto per ogni variazione dei pesi, e salvando i risultati in un array numpy. Successivamente è stata fatta la media dei risultati per ogni layer analizzato prendendo solamente la parte intera.

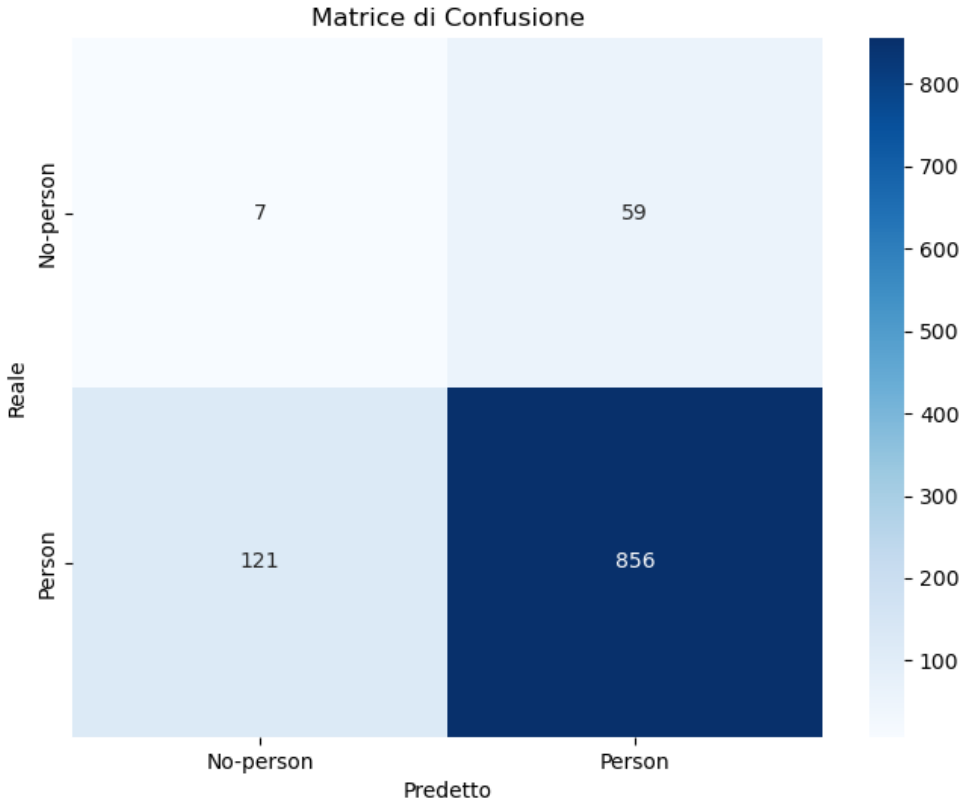


Figura 18. *Matrice di confusione relativa alle predizioni della rete nel caso di simulazione di SE nel bit segno.*

Notiamo che rispetto al caso fault free sono aumentati i falsi negativi, a discapito dei veri positivi ovviamente. Inoltre, sono aumentati anche se di poco i falsi positivi. Ciò, è dovuto al fatto che avendo un peso diverso, l'immagine in uscita dal layer risulta distorta e quindi, la rete non riesce a rilevare le persone come nel caso fault free.

4.2.2 Iniezione dei SE sul bit più significativo dell'esponente

Questa sezione descrive la parte critica dell'analisi. Si è notato che il bit più significativo dell'esponente di tutti i pesi era 0. Questo perché tutti i pesi, o almeno quelli dei layers convoluzionali, hanno un valore compreso tra ±2 estremi esclusi. Quindi, ribaltando positivamente il bit più significativo dell'esponente (0→1), il valore del peso raggiungeva cifre elevatissime. Non consentendo alla rete di contare.

Per dimostralo facciamo un esempio numerico. Supponiamo di voler rappresentare il numero 0.458, questo verrà codificato in 00111110111010100111111011111010.

Alterando il bit di segno questo cambierà solamente il suo segno. Quindi avrà una variazione di 0,916. Variando il bit più significativo della mantissa il valore del peso sarà circa 0.333 quindi si avrà una variazione in negativo di 0,125.

Variando il bit più significativo dell'esponente il valore del peso sale a $1{,}558\times10^{38}$, notiamo quindi che il peso viene alterato molto (ben 39 ordini di grandezza).

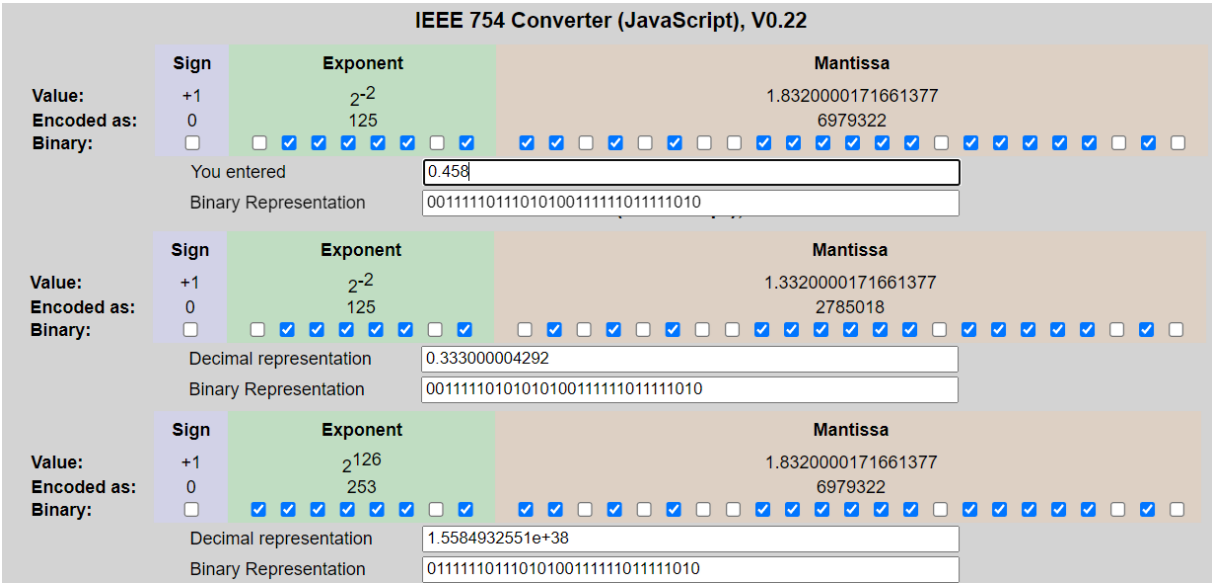


Figura 19. *Esempio variazione numero con bit flip.*

Mutando il bit più significativo dell'esponente quindi, ci si è accorti che la rete contava sempre 0 persone. Questo perché l'immagine in uscita dai filtri era talmente tanto alterata da non permettere alla rete il conteggio.

Questa situazione si è presentata per tutti i pesi e per tutti i layers. Quindi, è un qualcosa di critico. Riducendo accuratezza e TPR a 0,71% e prevedibilmente 0%, poiché non rileva persone. Lo 0,71 è dato perché rileva correttamente i veri negativi (tutte quelle volte che non ci sono persone nell'immagine).

Il MAE sale al 5,3 e il MSE al 25,2. Bisogna dunque, trovare una soluzione o almeno segnalare quando la macchina smette di funzionare. Poiché tutti i bit più significativi dell'esponente sono pari a 0, qualora questo fosse 1 andrebbe rilevato, segnalato e se possibile corretto.

Questo sicuramente riduce la *reliability*, la *availability* e la *safety* della rete. Se la rete è impiegata per scopi che vanno al di fuori di un'emergenza, come per esempio in un

esercizio commerciale, bisogna considerare la riduzione di affidabilità e disponibilità della macchina.

Quando invece la rete è impiegata per la gestione di emergenze, i concetti principali da tenere in conto sono sicuramente affidabilità e sicurezza. In particolare, la *safety* dovrebbe essere tale da garantire un efficiente e tempestivo intervento, non generando l'effetto contrario; ovvero, la generazione di ulteriori danni e perdite.

In quest'ultimo caso è quindi fondamentale garantire che il bit più significativo dell'esponente sia 0. Perciò, risulta necessario in fase di implementazione trovare una soluzione per garantire un corretto funzionamento.

Anche in questo caso è stata calcolata la matrice di confusione sempre con la stessa procedura. Prevedibilmente nella matrice di confusione tutti i TP sono andati a finire nei FN, mentre i FP sono scesi a 0.

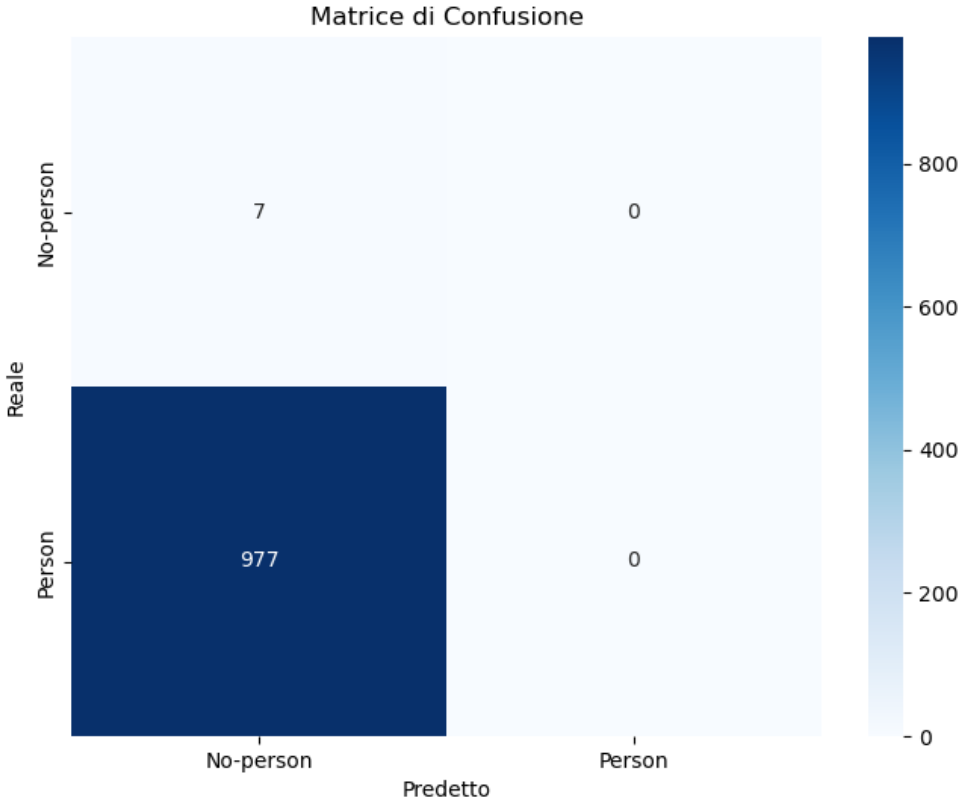


Figura 20. *Matrice di confusione relativa alle predizioni della rete nel caso di simulazione di SE nel bit più significativo esponente.*

Questo perché comunque il numero di persone totale (TP+FN) resta costante, e quindi, tutte le persone che nel caso fault free erano rilevate adesso non lo sono più. Da notare anche che i falsi positivi sono pari a 0 sempre per il motivo per cui la rete non riesce a rilevare persone.

Ma è solo il bit più significativo dell'esponente a dare problemi? Gli altri bit dell'esponente che ruolo giocano?

Da una piccola e veloce analisi si è notato che fortunatamente è solo il bit più significativo a dare problemi, mentre, perturbando il secondo le metriche di valutazione peggiorano, ma leggermente come nel caso dell'alterazione del bit di segno. Quindi, la rete continuerebbe a funzionare discretamente se si alterassero gli altri bit. Questo perché alterando il secondo bit il peso cambia ma non eccessivamente come nell'alterazione del bit più significativo.

4.2.3 Iniezione dei SE sul bit più significativo della mantissa

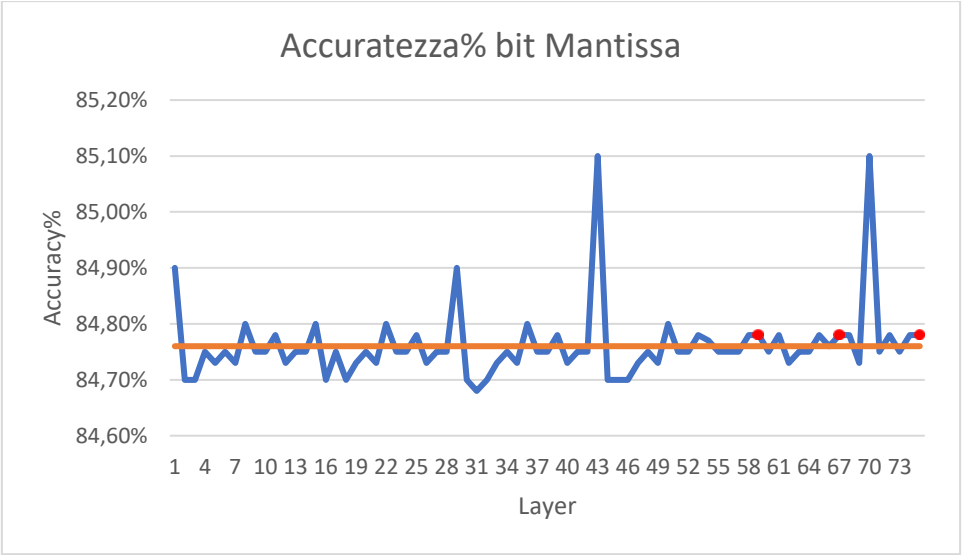
L'alterazione del bit più significativo della mantissa dovrebbe causare una leggera modifica del peso. Quindi, ci si aspettava già da prima del test un'accuratezza abbastanza buona.

Questa previsione è dovuta al fatto che il bit della mantissa sono quelli che incidono meno nella variazione del peso, alterandolo veramente di poco.

Infatti, variando il bit più significativo della mantissa dei pesi, la rete continua a contare con un'accuratezza che è superiore rispetto al test effettuato sul bit di segno.

Le reti neurali possono essere abbastanza robuste rispetto a piccole variazioni del peso. Quindi, anche se si altera il bit più significativo della mantissa, la rete potrebbe essere in grado di compensare questa variazione e produrre risultati accurati.

Nella pagina successiva i grafici che mostrano l'accuratezza e il TPR al variare del layer testato.



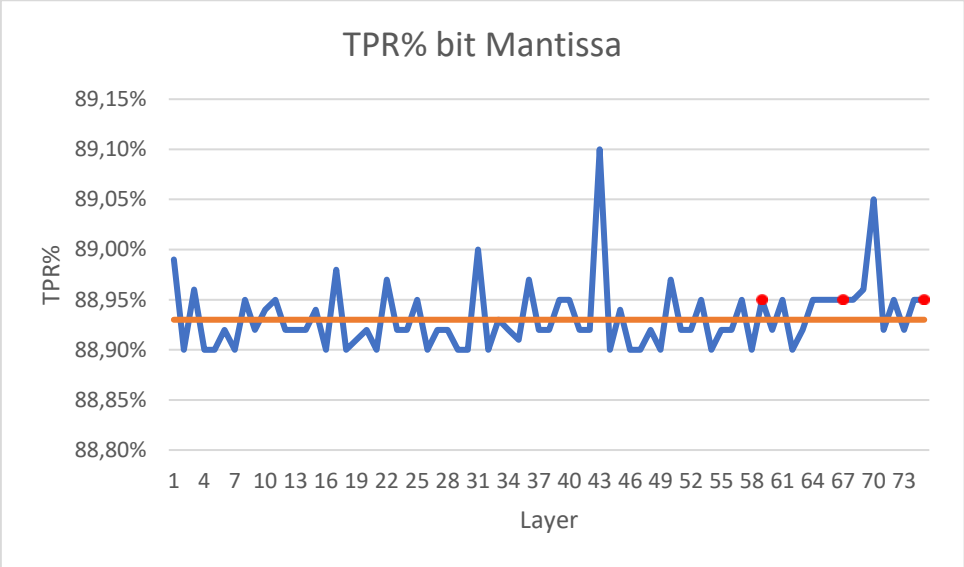


Figura 21. *Rappresentazione Accuratezza relativi alla simulazione dei SE del bit più significativo della mantissa.*

Notiamo che accuratezza% e TPR% salgono a una media di 84,76% e 88,93% che, come ci si aspettava, è molto vicino al caso fault free. La variazione di questi è 84,89% ±0,21% per l'accuratezza, mentre, per il TPR è 89% ±0,1%

Gli errori invece, sono rispettivamente 0,75 in media per il MAE e 1,35 per il MSE, con un excursus di 0,745±0,005 e 1,35±0,01.

Dunque, i bit sulla mantissa sono quelli che ci preoccupano meno se si verificano dei soft error. Bisogna però considerare che questi errori potrebbero accumularsi, e, quindi, alterare significativamente il peso con il passare del tempo.

Da i risultati ottenuti in termini di accuratezza e TPR ci aspettiamo una matrice di confusione molto simile al caso fault free.

Infatti, i TP diminuiscono solamente di 13 a favore dei FN, e i FP aumentano solo di 2.

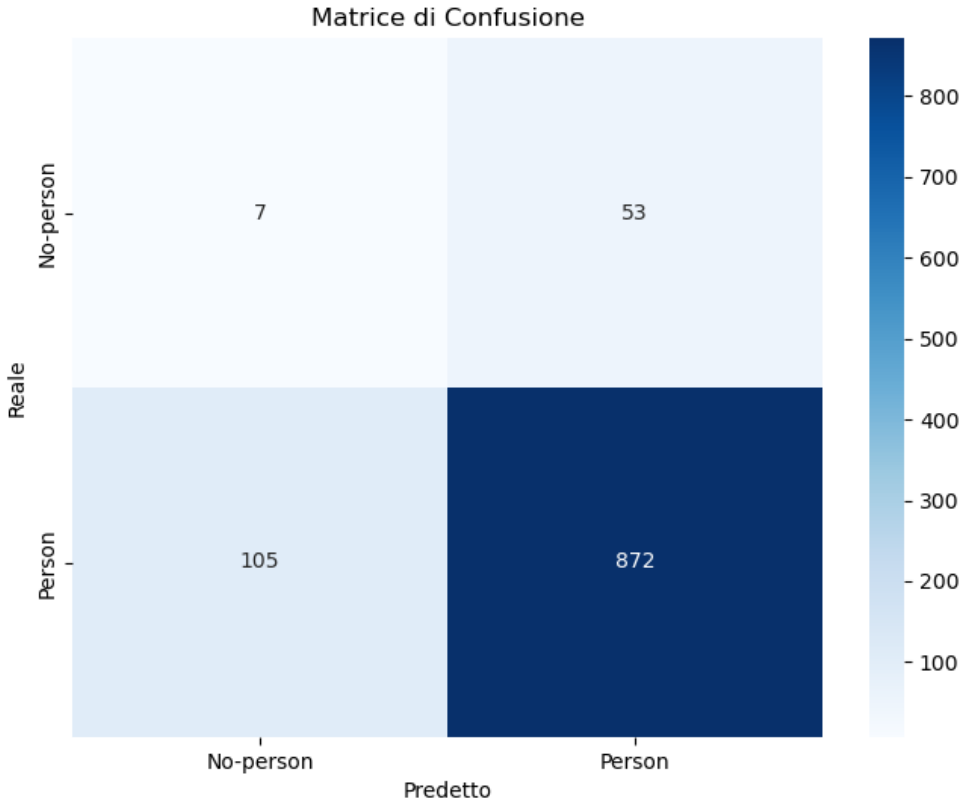


Figura 22. *Matrice di confusione relativa alle predizioni della rete nel caso di simulazione di SE nel bit più significativo della mantissa.*

Notiamo infatti che la matrice è molto simile a quella nel caso fault free. Inoltre, calcolando il TPR e l'accuratezza da questa, il risultato è molto simile alla media mostrata nella pagina precedente.

Bisogna inoltre, considerare che in tutti i casi la rete ha un TN che è uguale al caso fault free. Questo significa che anche in presenza di errori, se non ci sono persone nell'immagine, questa conta correttamente. Quindi, i falsi positivi, anche in questi casi, sono dovuti unicamente a un errore dovuto nella non eliminazione di bounding boxes multiple.

Conclusioni

La presente ricerca ha esaminato l'importanza del conteggio delle persone da immagini e video utilizzando l'intelligenza artificiale (IA) in una varietà di contesti. Attraverso un'analisi approfondita, sono emerse diverse applicazioni derivanti dall'impiego di sistemi di IA per questa attività.

In primo luogo, è stato evidenziato il ruolo fondamentale del conteggio delle persone nel monitoraggio e nella gestione delle folle. La capacità di rilevare e quantificare il numero di individui in luoghi affollati come stadi, concerti, e manifestazioni pubbliche è essenziale per identificare potenziali situazioni di sovraffollamento o emergenze, consentendo alle autorità di intervenire prontamente e prevenire potenziali incidenti. Questo aumenterebbe la *safety* di spazi sia pubblici che privati, consentendo di avere una maggiore sicurezza nelle situazioni di pericolo,

Inoltre, avere come dato il flusso delle persone può essere utile anche al di fuori di situazioni di emergenza. Quindi, è uno strumento che, se usato nel giusto modo, potrebbe migliorare significativamente le nostre vite.

Avere però una macchina che fornisce questo dato può implicare però anche uno studio sull'affidabilità di essa. Per questo ci teniamo a dire che, in particolare nelle situazioni di emergenza, è fondamentale un test approfondito sulla rete per un'accurata valutazione di essa.

Di essenziale importanza è anche il risultato ottenuto in questa tesi sull'analisi ai *soft error*, che indica che la parte più sensibile è sicuramente il bit più significativo dell'esponente. Questo fa si che la rete smetta di funzionare, generando delle potenziali situazioni di pericolo e, quindi, producendo l'effetto contrario.

Applicare le tecniche di fault tolerance citate nei concetti preliminari aumenta sicuramente la *dependability* della rete, ma cresce anche la complessità strutturale. Inoltre, oltre ad avere un apparato più complesso, questo comporta un maggiore dispendio di spazio e consumo, oltre che a un maggior costo economico. Perciò, l'analisi

e l’applicazione delle tecniche andrebbero fatte solamente se il campo applicativo lo richiede.

Bisogna anche aggiungere che il conteggio di persone analizzato prevede una ricezione dell’immagine frontale. Un conteggio con riprese dall’altro sicuramente renderebbe il dato in uscita molto più accurato per via della minore sovrapposizione delle bounding boxes, e quindi, un utilizzo migliore dell’algoritmo NMS. Alcuni studi hanno evidenziato che una rete allenata con immagini frontali riesce a contare meglio se gli si propongono riprese dall’alto, quindi, è evidente che, se la rete fosse allenata con immagini dall’alto, questa avrebbe un errore molto inferiore a quello proposto in questa attività di ricerca [39].

In conclusione, l'impiego di sistemi di IA per il conteggio delle persone riveste un ruolo fondamentale nella gestione delle situazioni di sicurezza e no, offrendo numerosi vantaggi in termini di prevenzione degli incidenti, risposta alle emergenze e analisi del flusso delle persone. Tuttavia, è essenziale considerare attentamente le implicazioni etiche e le sfide legate alla privacy nell'implementazione di tali tecnologie, garantendo un equilibrio adeguato tra sicurezza e rispetto dei diritti individuali.

Bibliografia

[1] Ilyas N, Shahzad A, Kim K. Convolutional-Neural Network-Based Image Crowd Counting: Review, Categorization, Analysis, and Performance Evaluation. Sensors (Basel). 2019 Dec 19;20(1):43. doi: 10.3390/s20010043. PMID: 31861734; PMCID: PMC6983207.

[2] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. Part of Advances in Neural Information Processing Systems 25 (NIPS 2012).

[3] Karen Simonyan ∗ & Andrew Zisserman. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. 4 Sep 2014.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. Submitted on 10 Dec 2015.

[5] Vishwanath A. Sindagi, Rajeev Yasarla, Vishal M. Patel. JHU-CROWD++: Large-Scale Crowd Counting Dataset and A Benchmark Method. Submitted on 7 Apr 2020 (v1), last revised 2 Nov 2020 (this version, v2)].

[6] Real-time people counting from depth imagery of crowded environments, Enrico Bondi, Lorenzo Seidenari, Andrew D. Bagdanov, Alberto Del Bimbo, Proc. of IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS), 2014, Seoul, Korea.

[7] https://personal.ie.cuhk.edu.hk/~ccloy/downloads_mall_dataset.html

[8] Fiza Joiya. OBJECT DETECTION: YOLO VS FASTER R-CNN. Peer-Reviewed, Open Access, Fully Refereed International Journal. Volume:04/Issue:09/September-2022. Impact Factor- 6.752.

[9] J Redmon, A Farhadi, YOLOv3: An Incremental Improvement, 2018

[10] https://medium.com/jungletronics/yolo-object-detection-2828800fd2a2

[11] Jocher Glenn, Chaurasia Ayush, Qiu Jing. Ultralytics YOLO. 2023-1-10

[12] https://colab.research.google.com/

[13] https://www.developersmaggioli.it/blog/le-reti-neurali-ricorrenti/

[14] https://www.ce.unipr.it/people/medici/geometry/node112.html

[15] https://it.mathworks.com/discovery/convolutional-neural-network.html

[16]https://www.nomidl.com/deep-learning/what-is-relu-and-sigmoid-activation-function/

[17]https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092

[18] https://lorit-consultancy.com/en/2020/10/are-we-going-soft-on-errors-part-1/

[19]https://www.domsoria.com/2019/10/come-funziona-una-rete-neurale-cnn-convolutional-neural-network/

[20] Younis Ibrahim, Haibin Wang, Junyang Liu, Jinghe Wei, Li Chen, Paolo Rech, Khalid Adam, Gang Guo,. Soft errors in DNN accelerators: A comprehensive review, Microelectronics Reliability, Volume 115, 2020, 113969, ISSN 0026-2714,

[21]https://www.ibm.com/it-it/topics/convolutional-neural-networks#:~:text=Ogni%20nodo%20si%20connette%20a,al%20livello%20successivo%20della%20rete.

[22] https://keras.io/api/layers/regularization_layers/dropout/

[23] Ayush Thakur. Keras Dense Layer: How to Use It Correctly. Last Updated: May 22, 2023.

[24] https://www.ibm.com/it-it/topics/neural-networks

[25] Crescenzio Gallo. Reti Neurali Artificiali: Teoria ed Applicazioni. Quaderno n. 28/2007. Dipartimento di Scienze Economiche, Matematiche e Statistiche Università degli Studi di Foggia

[26] C. Constantinescu, "Intermittent faults and effects on reliability of integrated circuits," *2008 Annual Reliability and Maintainability Symposium*, Las Vegas, NV, USA, 2008, pp. 370-374, doi: 10.1109/RAMS.2008.4925824.

[27] C. Metra, “Fault Tolerance”, Sistemi Elettronici ad Alta Affidabilità, Università di Bologna, 2021.

[28] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, You Only Look Once: Unified, Real-Time Object Detection, Submitted on 8 Jun 2015 , last revised 9 May 2016.

[29] https://docs.ultralytics.com/it

[30] Joseph Redmon, Ali Farhadi. YOLO9000: Better, Faster, Stronger. Submitted on 25 Dec 2016.

[31] Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. Submitted on 8 Apr 2018.

[32] https://it.dreamstime.com/fotografia-stock-uomo-che-cammina-su-un-sentiero-nel-bosco-image96414176

[33] Zhao, L.; Li, S. Object Detection Algorithm Based on Improved YOLOv3. *Electronics* 2020, *9*, 537.

[34] https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

[35] W. Pebrianto, P. Mudjirahardjo and S. H. Pramono, "YOLO Method Analysis and Comparison for Real-Time Human Face Detection," 2022 11th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), Malang, Indonesia, 2022, pp. 333-338, doi: 10.1109/EECCIS54468.2022.9902919.

[36] https://pjreddie.com/darknet/yolov1/

[37] Tsung-Yi Lin, Maire M, Belongie SJ, Bourdev LD, Girshick RB, Hays J, et al. Microsoft COCO: Common Objects in Context. CoRR [Internet]. 2014; abs/1405.0312. Available from: http://arxiv.org/abs/1405.0312

[38]https://www.v7labs.com/blog/yolo-object-detection#how-does-yolo-work-yolo-architecture

[39] M. Ahmad, I. Ahmed, K. Ullah and M. Ahmad, "A Deep Neural Network Approach for Top View People Detection and Counting," *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, New York, NY, USA, 2019, pp. 1082-1088, doi: 10.1109/UEMCON47517.2019.8993109.

[40]Norma ISO/IEC 42001:2023 Information technology - Artificial intelligence Management System (AIMS)

Indice figure

Figura 1. Processamento degli Input da un neurone. [13] 10
Figura 2. Struttura a strati di una rete neurale. [14] 11
Figura 3. Applicazione di un filtro convoluzionale. [19] 14
Figura 4. Struttura di una Rete Neurale Convoluzionale. [15] 14
Figura 5. Rappresentazione grafica funzione ReLU. [16] 15
Figura 6. Rappresentazione grafica di alcune funzioni di attivazione. [17] 15
Figura 7. Soft error su una cella RAM. [27] 19
Figura 8. Generazione coppie elettrone-lacuna da 20
Figura 9. Glitch di corrente 20
Figura 10. Classificazione e rilevamento in una YOLO. [36] 24
Figura 11. Esempio proposta bounding boxes di una YOLO. [32] 25
Figura 12. Elaborazione di una bounding box. [31] 27
Figura 13. Algoritmo NMS, rappresentazione grafica. [32] 29
Figura 14. Parametri e uscita della rete. 30
Figura 15. Matrice di confusione caso fault free 35
Figura 16. Rappresentazione standard IEEE 754 floating-point. 39
Figura 17. Rappresentazione Accuratezza e TPR relativi alla simulazione dei SE nel bit di segno 41
Figura 18. Matrice di confusione relativa alle predizioni della rete nel caso di simulazione di SE nel bit segno 43
Figura 19. Esempio variazione numero con bit flip. 44
Figura 20. Matrice di confusione relativa alle predizioni della rete nel caso di simulazione di SE nel bit più significativo dell'esponente 45
Figura 21. Rappresentazione Accuratezza e TPR relativi alla simulazione dei SE nel bit più significativo della mantissa 47
Figura 22. Matrice di confusione relativa alle predizioni della rete nel caso di simulazione di SE nel bit più significativo della mantissa. 48

Indice tabelle

Tabella 1. Organizzazione dei valori di TP, TN, FP, FN all'interno di una matrice di confusione. 32
Tabella 2. Valutazione del modello. 34
Tabella 3. Variazione TPR e Accuratezza al variare del numero di persone. 36

Indice equazioni

Eq.1 Precessamento di un neurone 10
Eq.2 Funzione di Attivazione 10
Eq.3 Categorical Cross-Entropy Loss (CCE) 12
Eq.4 Mean Squared Error (MSE) 12
Eq.5 Root Mean Squared Error (RMSE) 12
Eq.6 Mean Absolute Error (MAE) 12
Eq.7 Rectified Linear Unit (ReLU) 15
Eq.8 Reliability 18
Eq.9 Glitch di corrente 21
Eq.10 Carica accumulata Glitch di corrente 21
Eq.11 Binary Cross-Entropy (BCE) 26
Eq.12 Confidence 26
Eq.13 Coordinata x bounding box predetta 26
Eq.14 Coordinata y bounding box predetta 26
Eq.15 Coordinata w bounding box predetta 26
Eq.16 Coordinata h bounding box predetta 26
Eq.17 Coordinata x bounding box ground truth (g_x) 27
Eq.18 Coordinata y bounding box ground truth (g_y) 27
Eq.19 Coordinata w bounding box ground truth (g_w) 27
Eq.20 Coordinata h bounding box ground truth (g_h) 27
Eq.21 Square Error(SE) 27
Eq.22 Intersection over Union (IoU) 28
Eq.23 Accuracy% 32
Eq.24 TPR% 32
Eq.25 Rappresentazione IEEE 754 floating-point di un peso 39