



Università del Piemonte Orientale

Dipartimento di Scienze e Innovazione Tecnologica

Corso di Laurea in Informatica

Relazione per la prova finale

Intelligenza sull' "Edge" per la gestione dei parametri termici di una sala macchine

Tutore interno:

Prof. Marco Guazzone

Candidato:

Antonino Granata

Anno Accademico 2022/23

Abstract

Questo studio si concentra sulla gestione avanzata dei parametri tecnici, energetici e termici di una sala macchine, implementata con tecnologie Internet of Things (IoT). Lo scopo principale è sviluppare una soluzione innovativa per rilevare la presenza umana all'interno della sala utilizzando le telecamere di videosorveglianza, con un'implementazione specifica sull'“Edge”.

La tesi esplora tre approcci principali: Robot Collaborativi (Cobot), Item detection e Reti neurali, in particolare Yolov5, implementati su hardware come Nvidia Jetson AGX Orin e Rockchip Radxa RK3588 B. Tuttavia, i risultati della sperimentazione indicano sfide significative dovute alla mancanza di documentazione e allo scarso sviluppo dei sistemi esistenti.

Nonostante una sperimentazione di successo, proporre una soluzione definitiva rimane difficile a causa della documentazione e dello sviluppo insufficienti dei sistemi esistenti. La mancanza di informazioni chiare da fonti ufficiali e la complessità delle configurazioni dei dispositivi contribuiscono a questa sfida. Tuttavia, i vantaggi innegabili dei Cobot nella garanzia di una collaborazione sicura con gli operatori rappresentano un passo significativo verso l'automazione, migliorando l'efficienza e la produttività.

Lo studio sottolinea l'importanza di valutare l'impatto energetico del sistema attuale per ottimizzare l'efficienza e minimizzare i consumi. Nonostante alcune limitazioni, l'integrazione dei Cobot segna un progresso nell'automazione e nell'efficienza complessiva del sistema. Questo approccio consente una gestione efficiente dei parametri termici in loco, riducendo la necessità di trasferire dati estesi e il consumo energetico complessivo.

I Cobot, con la loro collaborazione sicura con gli operatori, giocano un ruolo cruciale nell'automazione delle operazioni, offrendo nuove possibilità in diverse industrie. La valutazione attenta del consumo energetico conferma che l'integrazione dei Cobot supportata dall'intelligenza sull'“Edge” è un passo positivo verso un'automazione crescente, offrendo un notevole miglioramento delle performance del sistema.

A tutti voi,

Fate sempre qualcosa di nuovo,

Guidati dalla curiosità,

E anche se una cosa non riuscite a vederla,

Come un programma che gira su un processore,

Provate a comprenderla e approfondirla.

Come recita una citazione a cui tengo particolarmente:

"Faccio sempre ciò che non so fare per imparare come va fatto."

- Vincent Van Gogh

Indice

1. Introduzione	10
1.1 Contesto e motivazione	10
1.2 Scopo della tesi.....	10
2. Approcci utilizzati.....	12
2.1 Cobot	12
2.2 Item detection	12
2.3 Reti neurali	13
2.3.1 CNN - Reti neurali convoluzionali	14
2.3.2 Rete neurale utilizzata da Yolov5	16
3. Tecnologie utilizzate	18
3.1 Linux e scripting shell for orchestration	18
3.2 Python.....	19
3.2.1 TreadingHTTPServer	19
3.2.2 PyTorch	20
3.2.3 Subprocess.....	21
3.2.4 YOLO	22
3.2.5 OpenCV	23
3.3 Node-RED	23
3.3.1 Nodo exec.....	25
3.3.2 Nodo read file	26
3.3.3 Nodo HTTP request	27
3.4 Docker e LXC container	28
3.4.1 Comandi Docker utilizzati	29
3.5 Json come formato di interscambio dati	30
3.6 Nvidia Jetson AGX Orin	31
3.6.1 GPU - CUDA 11.4.....	32
3.7 Rockchip Radxa Rock 5 B	32
3.7.1 NPU	33
3.8 Intel NUC Mele.....	33
3.9 Shelly Pro 2 PM.....	33
4. Spiegazione della sperimentazione	35
4.1 Scelta di YOLOv5	37
4.2 Inizializzazione Jetson AGX Orin e Radxa Rock 5 B	38
4.2.1 Installazione Yolov5 su container Docker – Jeston AGX Orin.....	39

4.2.1.1 Creazione e avvio del container Docker	39
4.2.1.2 Preparazione dell'ambiente YOLOv5	40
4.2.1.3 Esecuzione dell'inferenza di test.....	41
4.2.2 Installazione YOLOv5 su container Docker - Radxa Rock 5 B.....	41
4.2.2.1 Realizzazione e Avvio del Container Docker	41
4.2.2.2 Installazione di YOLOv5 ed esecuzione della demo	42
4.3 Creazione server per inferenza Jetson AGX Orin e Radxa Rock 5 B.....	43
4.3.1 L'inferenza sulla Jetson AGX Orin	44
4.3.2 L'inferenza sulla Radxa Rock 5 B	44
4.4 Creazione container LXC su Intel NUC	45
4.4.1 Implementazione flow Nodered.....	45
5. Risultati della sperimentazione.....	47
5.1 Risultati Jetson AGX Orin	48
5.1.1 Risultati server Jetson AGX Orin.....	50
5.2 Risultati Radxa Rock 5 B	51
5.2.1 Risultati server Radxa Rock 5 B	52
5.3 Risultati in termini di consumo.....	53
6. Conclusioni e sviluppi futuri.....	54
7. Bibliografia	56
8. Ringraziamenti.....	59

1. Introduzione

1.1 Contesto e motivazione

La presente attività si inserisce nel contesto della gestione avanzata dei parametri tecnici energetici e termici di una sala macchine, la quale è equipaggiata con tecnologie Internet of Things (IoT). In questo scenario, emerge come una sfida fondamentale la necessità di automatizzare il controllo della temperatura considerando la presenza umana all'interno dell'ambiente. Questo perché la corretta gestione della temperatura è fondamentale per garantire il corretto funzionamento e la durata delle apparecchiature elettroniche presenti nella sala macchine. Inoltre, assicurare un ambiente termico ottimale è essenziale per il comfort e la sicurezza degli operatori che lavorano in tale ambiente.

La complessità di questo contesto è ulteriormente accentuata dalla natura dinamica e mutevole dell'ambiente di lavoro di una sala macchine. La presenza umana all'interno di tale ambiente può variare nel tempo e può influenzare significativamente le condizioni ambientali, richiedendo un sistema di controllo della temperatura in grado di adattarsi rapidamente e in modo efficace a queste variazioni.

Di conseguenza, l'automatizzazione del controllo della temperatura in presenza umana si configura come obiettivo principale per ottimizzare l'efficienza energetica, garantire la sicurezza delle apparecchiature elettroniche e migliorare il comfort degli operatori. In questo contesto, l'utilizzo di tecnologie IoT e di sistemi di videosorveglianza si presenta come una soluzione promettente per affrontare questa sfida in modo efficace e innovativo.

1.2 Scopo della tesi

Lo scopo principale di questa tesi è lo sviluppo di una soluzione innovativa ed efficiente che sfrutti le telecamere di videosorveglianza per rilevare la presenza umana all'interno della sala macchine. L'obiettivo centrale è integrare le informazioni catturate dalle telecamere con i vari sistemi Internet of Things presenti nell'ambiente, allo scopo di automatizzare in modo intelligente e responsivo il controllo della temperatura.

In particolare, il sistema mira a realizzare un rilevamento in tempo reale della presenza o assenza di individui mediante le telecamere di videosorveglianza e a utilizzare tali informazioni per regolare automaticamente la temperatura dell'ambiente. Questo comporta

la capacità del sistema di adattare dinamicamente le impostazioni di temperatura in risposta alla presenza umana, garantendo un ambiente sicuro, confortevole ed efficiente sia per gli operatori che per le apparecchiature elettroniche presenti.

Parallelamente, parte significativa della tesi si concentra sull'analisi dell'evoluzione della facilità di programmazione, gestione e raccolta dati da parte dei dispositivi e delle case produttrici. Questa valutazione critica contribuirà a comprendere a che livello di accessibilità sono giunti i dispositivi e le soluzioni presenti sul mercato, fornendo una panoramica chiara delle opportunità e delle sfide nel campo della videosorveglianza e dell'IoT.

La soluzione proposta non solo mira a migliorare la sicurezza e il comfort nella sala macchine attraverso la rilevazione intelligente della presenza umana, ma anche a ottimizzare l'efficienza energetica grazie alla comunicazione e coordinamento tra le telecamere e i sistemi IoT esistenti. L'integrazione di queste componenti consente una gestione avanzata e sinergica degli ambienti industriali, contribuendo a un utilizzo più intelligente delle risorse e a una migliore qualità degli ambienti di lavoro.

Questo approccio è stato sviluppato per INRETE S.R.L, l'azienda che ha ospitato lo stage. INRETE S.R.L opera nel core business di supportare le aziende nella trasformazione dei servizi Internet in nuove opportunità, aumentare la competitività, migliorare i servizi e ridurre i costi. Fondata nel 1994 a Torino, INRETE è pioniera nel portare Internet nelle aziende e organizzazioni, con un'esperienza volta a sfruttare il web per ridurre la complessità attraverso soluzioni efficaci, mantenendo alti standard di qualità ed economicità [1].

Le loro soluzioni mirano a offrire strumenti intuitivi e approcci rivisti all'utilizzo della rete e dei servizi connessi, riducendo i tempi di risposta anche per attività complesse e conseguentemente diminuendo i costi [1]. L'azienda si distingue per fornire soluzioni semplici e solide, promuovendo affidabilità nel tempo [1].

2. Approcci utilizzati

2.1 Cobot

Un "Cobot", abbreviazione di "robot collaborativo", è un tipo di robot progettato per lavorare fianco a fianco con gli esseri umani in un ambiente di lavoro [2]. A differenza dei tradizionali robot industriali, che spesso operano in modo separato dagli esseri umani e sono isolati in aree o gabbie di sicurezza, i Cobot sono progettati per essere sicuri e interagire direttamente con le persone senza rischi per la loro incolumità. Questo permette di automatizzare compiti ripetitivi e faticosi, liberando gli operatori per compiti più complessi e creativi, aumentando così la produttività e l'efficienza in diversi settori industriali [2].

Un Cobot è in grado di rilevare la presenza umana nelle vicinanze e rispondere adeguatamente per garantire la sicurezza dell'ambiente di lavoro, utilizzando sensori che lo fermano automaticamente in caso di contatto con una persona. Questo è in linea con gli standard di sicurezza come la ISO 10218-1 [3].

Nel caso specifico della tesi, il Cobot è rappresentato da una sala server precedentemente domotizzata, la quale ha il compito di gestire le operazioni di controllo della temperatura in modo efficiente e sicuro. Questo sistema automatizzato è progettato per monitorare costantemente l'ambiente utilizzando le videocamere di videosorveglianza integrate nel sistema. Quando rileva la presenza umana all'interno della sala server, il Cobot regola dinamicamente la temperatura dell'ambiente in base alle necessità. Questa funzionalità permette al Cobot di garantire una temperatura adeguata alla presenza umana, ma anche di mantenere le condizioni ottimali di temperatura per il corretto funzionamento degli apparecchi elettronici, contribuendo così a preservarne l'integrità e la durata nel lungo termine.

2.2 Item detection

La "Item detection," o rilevamento di oggetti, rappresenta una componente cruciale nel campo della visione artificiale e dell'elaborazione delle immagini. Ad esempio, nella Figura 1, possiamo notare come gli oggetti come: "building, bag, pushcar, bicycle e human" vengano identificati. Questa tecnologia si concentra sull'identificazione e la localizzazione di oggetti specifici all'interno di un'immagine o di un frame video. L'obiettivo principale è individuare la presenza di elementi di interesse, consentendo alle applicazioni di interpretare e comprendere il contesto visivo circostante [4].

Il rilevamento di oggetti è ampiamente utilizzato in una vasta gamma di settori, tra cui la sicurezza, la sorveglianza, l'automazione industriale, la guida autonoma e l'analisi video. Algoritmi avanzati di Item detection, spesso basati su reti neurali convoluzionali (CNN) e tecniche di apprendimento profondo, hanno dimostrato di ottenere risultati eccezionali nella precisione e nella velocità di rilevamento [4].

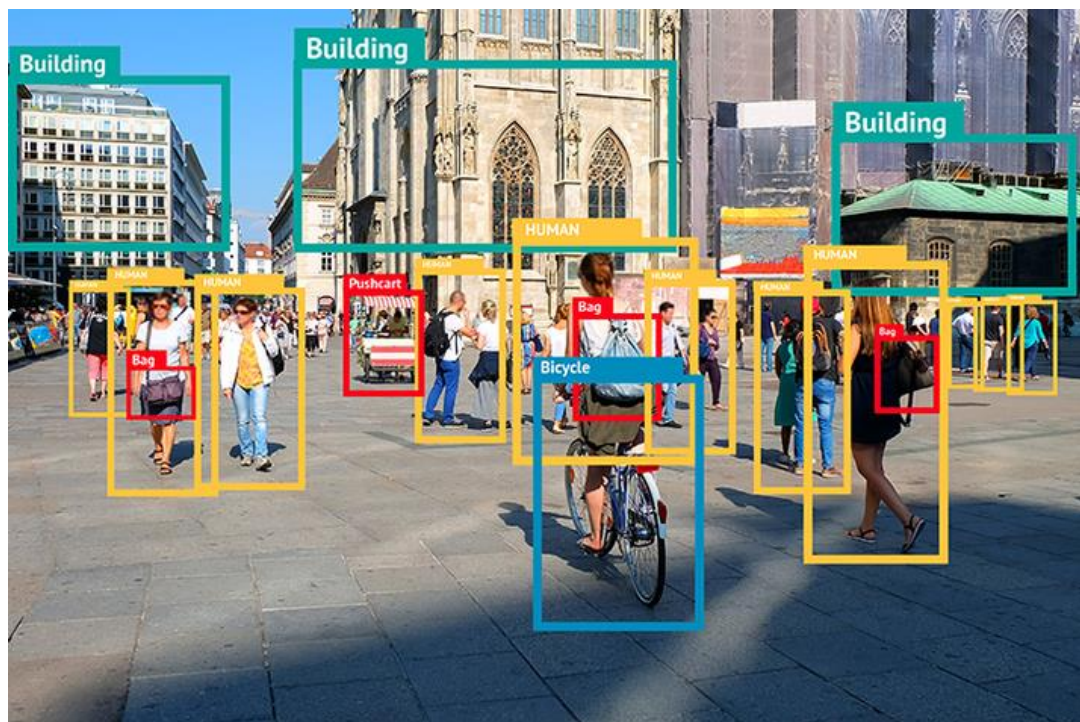


Figura 1: Item detection degli oggetti di un'immagine [4]

La fase di Item detection coinvolge la segmentazione dell'immagine per identificare e isolare singoli oggetti, seguita dalla classificazione per assegnare una categoria specifica a ciascun elemento rilevato. La combinazione di queste due fasi consente di ottenere una comprensione dettagliata del contenuto visivo, facilitando l'automazione di compiti complessi.

2.3 Reti neurali

Le reti neurali rappresentano il cuore pulsante della rivoluzione nell'ambito dell'apprendimento automatico e della visione artificiale [5]. Queste strutture computazionali sono modellate ispirandosi al funzionamento del cervello umano, dove neuroni interconnessi collaborano per elaborare informazioni complesse. Nell'ambito dell'informatica, le reti neurali artificiali sono state progettate per apprendere automaticamente pattern e rappresentazioni significative dai dati, consentendo di affrontare problemi complessi e di effettuare previsioni senza esplicita programmazione [5].

Le reti neurali possono assumere diverse architetture, ma le reti neurali profonde (DNN) hanno guadagnato particolare attenzione [5]. Queste reti, caratterizzate da strati multipli di neuroni, sono in grado di apprendere rappresentazioni sempre più complesse dei dati. Sulla base delle DNN si sono sviluppate le reti neurali convoluzionali (CNN), specializzate nel trattare i dati bidimensionali come immagini e le reti neurali ricorrenti (RNN) progettate per gestire sequenze di dati, come il linguaggio naturale [5].

L'allenamento di una rete neurale coinvolge il processo di ottimizzazione dei pesi delle connessioni tra neuroni in base ai dati di addestramento, in modo che la rete possa generalizzare correttamente su nuovi dati. L'emergere di strumenti e framework di deep learning, come TensorFlow e PyTorch, ha semplificato notevolmente lo sviluppo e l'implementazione di reti neurali, rendendo accessibile questa tecnologia a una vasta comunità di sviluppatori.

2.3.1 CNN - Reti neurali convoluzionali

Le CNN rappresentano un tipo specifico di rete neurale artificiale (ANN) particolarmente efficace nell'elaborazione di immagini e dati bidimensionali [6]. Ispirate dal sistema visivo umano, le CNN sfruttano l'operazione di convoluzione per estrarre caratteristiche e pattern significativi dalle immagini [6]. L'operazione di convoluzione consiste nell'applicare un filtro, chiamato kernel, a diverse regioni sovrapposte dell'input [6]. Il kernel scansiona l'input, moltiplica i valori corrispondenti e somma il risultato per generare una mappa delle caratteristiche, nota come mappa di attivazione, mostrata nella Figura 2.

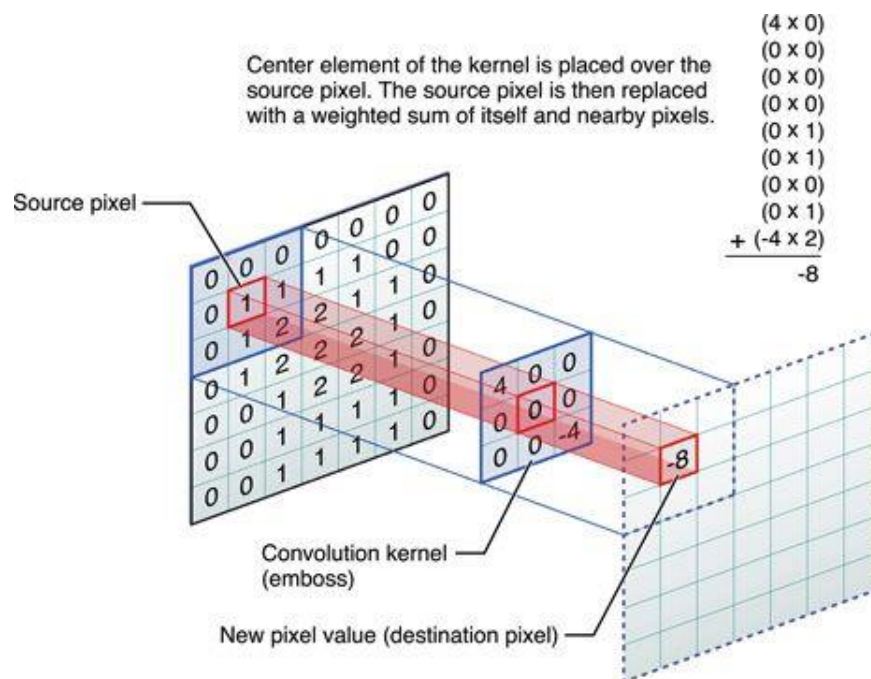


Figura 2: Operazione di convoluzione [6]

La struttura fondamentale delle CNN mostrata in Figura 3 comprende tre tipi principali di strati [6], ovvero:

- Strati Convoluzionali:
 - Applicano filtri alle regioni locali dell'immagine.
 - Catturano feature spaziali come bordi, texture e pattern.
 - L'operazione di convoluzione è essenziale per individuare queste caratteristiche.
- Strati di Pooling:
 - Riducono progressivamente le dimensioni spaziali dell'output della convoluzione.
 - Riducono la complessità computazionale.
 - Si concentrano sulle feature più rilevanti.
 - Possono utilizzare operazioni come il max pooling (seleziona il valore massimo in una regione) o il mean pooling (calcola la media).
- Strati Fully Connected (FC):
 - Collegano ogni neurone di uno strato con tutti i neuroni dello strato successivo.
 - Aprono la strada alla classificazione finale o all'output desiderato.
 - Sono responsabili della combinazione delle feature estratte per ottenere una rappresentazione globale dei dati.

Le CNN sono particolarmente adatte per il rilevamento di oggetti nelle immagini attraverso la capacità di apprendere feature gerarchiche, consentendo la rappresentazione di concetti complessi. Il processo di addestramento coinvolge la presentazione di immagini etichettate in ingresso alla rete, con l'ottimizzazione dei pesi delle connessioni per migliorare la capacità predittiva del modello [6].

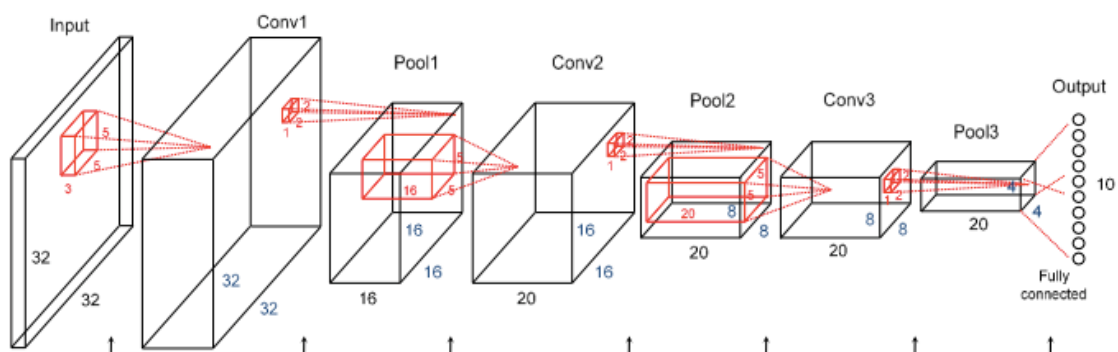


Figura 3: Struttura di una CNN [6]

2.3.2 Rete neurale utilizzata da YOLOv5

YOLO (You Only Look Once) utilizza diverse varianti di reti neurali per il rilevamento degli oggetti. Fin dalla sua prima versione, YOLO ha implementato reti neurali convoluzionali per il rilevamento di oggetti in tempo reale [7].

Le versioni iniziali di YOLO utilizzavano una rete neurale convoluzionale di base per il rilevamento di oggetti. Successivamente, con l'evoluzione della serie YOLO, sono state introdotte migliorie e cambiamenti nelle architetture delle reti neurali per migliorare la precisione e la velocità di rilevamento [7].

YOLOv5 utilizza una specifica architettura di rete neurale convoluzionale per il rilevamento di oggetti e prevede diverse varianti (YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x) come mostrato in Figura 4, differenziandosi principalmente per il numero di livelli della rete, ottenendo modelli di piccole e grandi dimensioni (dai 4MB per la versione n ai 166MB per la versione x), la complessità computazionale in termini di costi e velocità di esecuzione del modello (dai 6,3ms per la versione n ai 12,1ms per la versione x) e la loro precisione (dai 28,4 COCO mAP ai 50,7 COCO mAP).

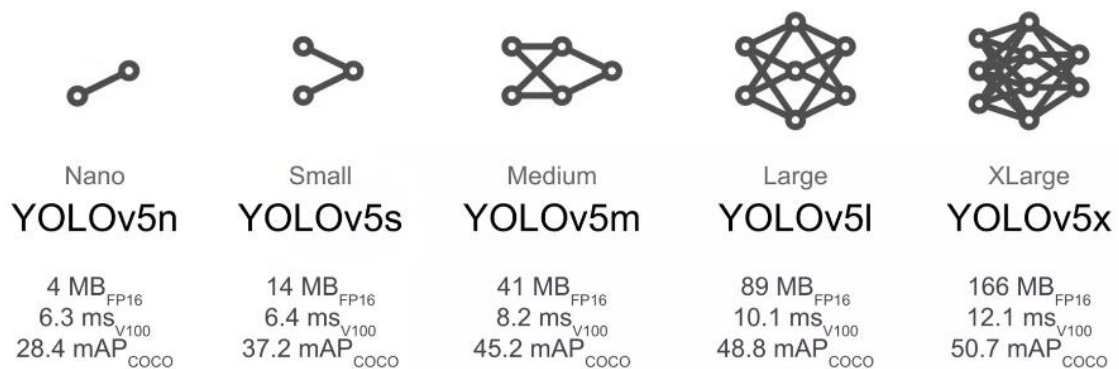


Figura 4: Versioni di YOLOv5 [7]

La loro grandezza definisce la quantità di neuroni e conseguenti connessioni sinaptiche per gli strati nascosti della rete neurale [8], ovvero:

- YOLOv5n: 3 strati nascosti, 128 neuroni per strato e 36 milioni di connessioni sinaptiche totali.
- YOLOv5s: 6 strati nascosti, 256 neuroni per strato e 144 milioni di connessioni sinaptiche totali.

- YOLOv5m: 6 strati nascosti, 512 neuroni per strato e 576 milioni di connessioni sinaptiche totali.
- YOLOv5l: 6 strati nascosti, 1024 neuroni per strato e 2,3 miliardi di connessioni sinaptiche totali.
- YOLOv5x: 6 strati nascosti, 2024 neuroni per strato e 9,2 miliardi di connessioni sinaptiche totali.

Queste varianti sono ottimizzate per adattarsi a diversi contesti di utilizzo, consentendo una maggiore flessibilità in termini di risorse computazionali e precisione richiesta.

3. Tecnologie utilizzate

3.1 Linux e scripting shell for orchestration

Linux, come sistema operativo open-source basato sul kernel Linux, si distingue per la sua natura altamente personalizzabile e adattabile. Le sue molteplici distribuzioni, o “distro”, rappresentano un'eccellente manifestazione di questa flessibilità. Ogni distribuzione è caratterizzata da un insieme specifico di software preinstallati e configurazioni di sistema, progettate per soddisfare le esigenze di un'ampia gamma di utenti e utilizzi [9].

Inoltre, la vasta comunità di sviluppatori e appassionati che contribuisce costantemente al miglioramento e all'espansione di Linux rappresenta un altro elemento chiave del suo successo. Questo modello di sviluppo collaborativo porta a continui aggiornamenti, correzioni di bug e nuove funzionalità, mantenendo Linux al passo con le esigenze in evoluzione degli utenti [9].

Gli scripting shell svolgono un ruolo importante nell'ecosistema Linux, consentendo agli utenti di interagire in modo flessibile e potente con il sistema operativo. Questi script, scritti per l'interprete della shell di Linux, possono automatizzare una vasta gamma di attività di sistema, semplificando compiti complessi e ripetitivi. Grazie alla loro capacità di integrare comandi di sistema, logica di programmazione e operazioni di input/output, gli script shell offrono agli utenti un controllo granulare sulle operazioni di sistema.

L'interfaccia testuale della shell permette agli utenti di eseguire comandi direttamente al kernel del sistema operativo, senza l'intermediazione di un'interfaccia grafica. Questo approccio offre un controllo più immediato e dettagliato sul sistema, consentendo agli utenti di automatizzare processi, gestire file e directory, monitorare risorse di sistema e altro ancora [9].

Nello studio è stato scelto l'utilizzo di Linux anche perché le macchine utilizzate nel progetto di tesi hanno sistemi operativi basati su Linux e architettura ARM64. Questa scelta si è rivelata particolarmente vantaggiosa data la compatibilità diretta con l'ambiente di lavoro e l'hardware utilizzato nel contesto dell'attività svolta. Inoltre, l'ampia disponibilità di risorse, documentazione e supporto per Linux e le sue distribuzioni ARM64 ha semplificato lo sviluppo e l'implementazione delle soluzioni necessarie per il progetto. Grazie alla flessibilità e all'adattabilità di Linux, è stato possibile affrontare con successo le sfide specifiche del lavoro, ottenendo risultati efficaci e affidabili.

3.2 Python

Python è un linguaggio di programmazione ad alto livello, interpretato e general-purpose, ampiamente utilizzato per lo sviluppo di software, scripting, analisi dei dati e sviluppo web [10].

Una delle principali ragioni dietro la scelta di Python per questo progetto è la sua vasta e attiva comunità, che fornisce un supporto prezioso e una ricchezza di risorse online.

Python si distingue per la gestione automatica della memoria (garbage collection), la sua natura orientata agli oggetti e un esteso insieme di librerie standard che coprono una vasta gamma di domini, tra cui networking, elaborazione di testo, intelligenza artificiale e interfacciamento con database [10].

La versatilità e la facilità d'uso di Python lo rendono adatto sia per principianti che per sviluppatori esperti. La sua comunità attiva e collaborativa contribuisce costantemente all'evoluzione del linguaggio, offrendo una vasta gamma di risorse, documentazione e supporto online [10].

La scelta di utilizzare Python per questo progetto è stata influenzata dalla disponibilità di risorse specifiche online per YOLOv5, tutte scritte in Python. Questo ha semplificato l'integrazione di YOLOv5 nel contesto del progetto, consentendo una rapida adozione e implementazione grazie alle risorse accessibili e alla facilità di utilizzo offerte da Python. Questo ha contribuito a rendere il processo di sviluppo più efficiente e accessibile, grazie alla vasta comunità e alle risorse specifiche disponibili online.

3.2.1 ThreadingHTTPServer

Il modulo ThreadingHTTPServer in Python rappresenta un componente fondamentale per la gestione delle richieste HTTP in modo concorrente ed efficiente [11]. Questo modulo, incluso nella libreria standard di Python, offre un'implementazione del server HTTP incorporata che utilizza threading per gestire multiple richieste simultaneamente [11]. Tale caratteristica risulta particolarmente rilevante in scenari in cui è necessario gestire concorrentemente richieste da più client, migliorando significativamente le prestazioni e la responsività del server.

L'utilizzo del `ThreadingHTTPServer` semplifica notevolmente lo sviluppo di applicazioni web in Python, permettendo agli sviluppatori di implementare server HTTP senza la necessità di utilizzare server web esterni o framework complessi. La capacità di gestire richieste concorrenti attraverso l'utilizzo di thread consente di mantenere elevati livelli di prestazioni anche in situazioni di carico elevato.

Attraverso il `ThreadingHTTPServer`, è possibile fornire risorse web dinamiche o statiche, rispondere a richieste di API e gestire diverse operazioni legate alla comunicazione HTTP [11]. Questo modulo offre una soluzione integrata, consentendo agli sviluppatori di concentrarsi sulla logica dell'applicazione senza dover gestire dettagli di basso livello legati al threading e alla gestione delle connessioni HTTP.

Nel caso specifico del progetto, il `ThreadingHTTPServer` si rivela uno strumento fondamentale per garantire l'accessibilità e l'efficacia dei servizi di Item detection della Jetson AGX Orin e della Radxa Rock 5 B (hardware utilizzato, introdotto nei capitoli 3.6 e 3.7), consentendo loro di essere facilmente accessibili e utilizzabili su una rete locale.

3.2.2 PyTorch

Il framework PyTorch, una delle principali librerie di machine learning in Python, si distingue per la sua flessibilità, facilitando lo sviluppo e l'implementazione di modelli avanzati nel campo dell'intelligenza artificiale [12]. La sua architettura modulare e la filosofia di design orientata alla ricerca ne fanno uno strumento potente e ampiamente utilizzato nella comunità scientifica e industriale [12]. Il modulo centrale, `torch`, offre una vasta gamma di funzionalità essenziali per la costruzione e l'addestramento di reti neurali [12].

Tra le caratteristiche distintive di PyTorch vi è il suo approccio dinamico al grafo computazionale, che permette la creazione di modelli più flessibili e adatti alla sperimentazione [12]. Il sistema di “autograd” integrato semplifica il calcolo dei gradienti, agevolando la retro-propagazione e l'ottimizzazione dei parametri del modello durante il training. La gestione efficiente dei tensori, rappresentata attraverso il modulo `torch.Tensor`, fornisce una solida base per la manipolazione dei dati in input e output [13].

PyTorch, tra le principali librerie di machine learning in Python, non solo si distingue per la sua flessibilità e potenza, ma offre anche una gestione efficiente delle operazioni su GPU attraverso il modulo `torch.cuda` [14]. Questo componente è particolarmente significativo per sfruttare le capacità di calcolo parallelo delle unità di elaborazione grafica, accelerando notevolmente le operazioni di addestramento dei modelli e consentendo una maggiore scalabilità computazionale.

Il modulo `torch.cuda` consente di trasferire tensori e modelli sulla memoria della GPU, sfruttando la sua potenza di calcolo per eseguire operazioni matematiche in parallelo [14]. Questo è particolarmente rilevante in scenari in cui la complessità computazionale richiede risorse significative, come nel caso di reti neurali profonde o di grandi set di dati.

3.2.3 Subprocess

Il modulo `subprocess` in Python si configura come uno strumento essenziale per l'interazione con processi esterni al programma in esecuzione [15]. Questa libreria offre un'interfaccia completa e potente per avviare, comunicare e controllare processi separati, consentendo una maggiore modularità e flessibilità nelle applicazioni Python [15].

Una delle classi principali all'interno del modulo è `subprocess.Popen`, che permette di avviare processi esterni e di gestire le loro comunicazioni con il programma Python in esecuzione [15]. Questo modulo è particolarmente utile quando è necessario eseguire comandi di sistema, lanciare script esterni o interagire con programmi che richiedono l'avvio in un processo separato.

Nel caso specifico, è stata utilizzata la libreria `subprocess` per montare la directory "image" sulla cartella `tmpfs` del sistema operativo, mediante la funzione `mount_all()`. Questa operazione coinvolge l'esecuzione del comando `mount -a` attraverso `subprocess.run()` che coinvolge l'istruzione `subprocess.Popen`. L'uso di `subprocess` in questo contesto consente di eseguire l'operazione di montaggio in un processo separato, assicurando una gestione efficiente e controllata dell'operazione.

Attraverso `subprocess`, è possibile catturare eventuali errori generati durante l'esecuzione del comando `mount -a` e gestirli in modo appropriato. Questa versatilità rende il modulo `subprocess` una scelta determinante per scenari in cui l'interazione con il sistema

operativo o l'esecuzione di comandi esterni è fondamentale, come nel caso dell'amministrazione del sistema e delle operazioni di montaggio di directory su tmpfs.

3.2.4 YOLO

YOLO è un framework di computer vision ampiamente utilizzato per il rilevamento in tempo reale di oggetti in immagini e video. Grazie alla sua architettura innovativa e alla sua elevata precisione, YOLO si è affermato come uno dei principali approcci per il rilevamento degli oggetti, offrendo prestazioni di livello professionale in termini di velocità e accuratezza [16].

In Python, YOLO è implementato attraverso diverse librerie e wrapper che semplificano l'utilizzo e l'integrazione del framework nelle applicazioni Python [16]. Uno dei più popolari è la libreria "darknet", che fornisce un'implementazione efficiente e ottimizzata dell'algoritmo YOLO e offre funzionalità avanzate per il rilevamento degli oggetti.

L'utilizzo di YOLO in Python è particolarmente vantaggioso per sviluppatori e ricercatori che desiderano integrare il rilevamento degli oggetti nelle loro applicazioni e progetti. Grazie alla sua facilità d'uso e alla sua scalabilità, YOLO consente di implementare sistemi di visione artificiale sofisticati e in tempo reale, adatti a una vasta gamma di casi d'uso, dall'analisi di sicurezza alla sorveglianza urbana, dal monitoraggio del traffico alla guida autonoma [16].

YOLOv5, in particolare, ha portato ulteriori miglioramenti alle capacità di rilevamento degli oggetti. Introdotta come una versione più leggera e più rapida rispetto alle sue precedenti controparti, YOLOv5 mantiene la sua efficienza e precisione, ma con un'enfasi particolare sulla facilità d'uso e sull'adattabilità a diversi scenari di applicazione.

La libreria PyTorch è stata utilizzata come base per l'implementazione di YOLOv5. La sua integrazione con PyTorch facilita l'addestramento su dataset personalizzati e l'estensione delle funzionalità di rilevamento degli oggetti per soddisfare esigenze specifiche. La versatilità di YOLOv5 si riflette nella sua capacità di adattarsi a molteplici contesti, inclusi progetti di Item detection.

L'Item detection rappresenta una delle applicazioni chiave di YOLOv5. Questa funzionalità consente di identificare e localizzare non solo categorie generiche di oggetti, ma anche particolari elementi specifici di interesse all'interno di un'immagine o video. Ciò è particolarmente rilevante in scenari in cui è necessario individuare e monitorare dettagli specifici, come la presenza di un operatore all'interno della sala macchine.

3.2.5 OpenCV

OpenCV (Open Source Computer Vision) è una libreria open source ampiamente utilizzata per la visione artificiale e l'elaborazione delle immagini [17]. L'obiettivo principale di OpenCV è fornire strumenti e funzionalità per sviluppare applicazioni di visione computazionale in vari settori, tra cui robotica, realtà aumentata, riconoscimento di oggetti, tracciamento di movimenti e molto altro [17]. L'interfaccia di programmazione (API) di OpenCV per Python è chiamata `opencv-python`. Questa API semplifica l'utilizzo delle potenti funzionalità di OpenCV all'interno di script Python, rendendo più accessibile la creazione di applicazioni di visione artificiale [17]. Un esempio di funzione utilizzata in OpenCV è `resize()`. Questa funzione consente di ridimensionare un'immagine secondo le specifiche desiderate. Ad esempio, è possibile ridimensionare un'immagine per adattarla a determinate dimensioni o rapporti di aspetto.

3.3 Node-RED

Node-RED è uno strumento di programmazione versatile progettato per facilitare l'integrazione e l'automazione di dispositivi hardware, API e servizi online [18]. La sua interfaccia di sviluppo basata su browser offre un editor intuitivo che semplifica la creazione e la gestione di flussi di lavoro complessi [18]. La "Palette Nodes" o tavolozza dei nodi fornisce una vasta gamma di componenti preconfigurati che possono essere facilmente collegati tra loro per creare flussi operativi complessi. Nella Figura 5 si vuol schematizzare come con Node-RED è possibile programmare dei dispositivi IoT visibili e facilmente raggiungibili dall'utente utilizzando la rete.

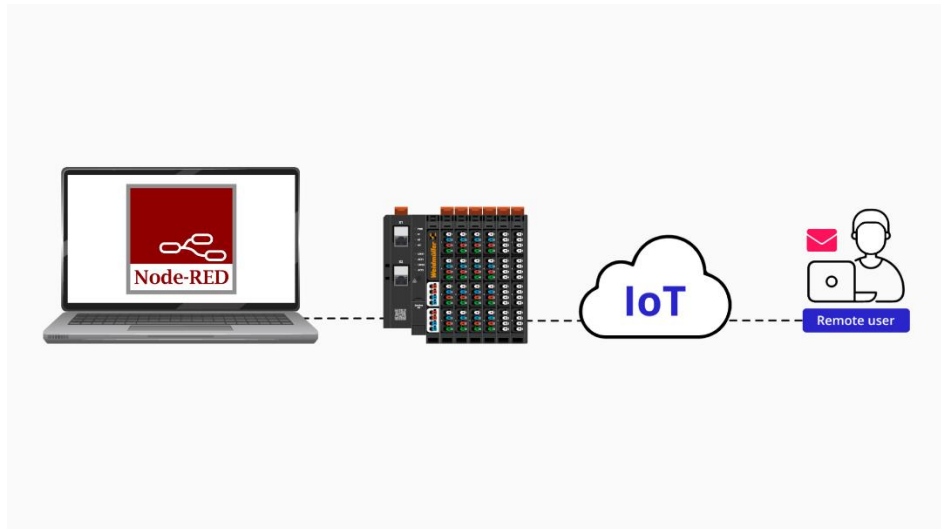


Figura 5: Node-RED

All'interno dell'editor, gli sviluppatori possono creare funzioni JavaScript per personalizzare il comportamento dei nodi o aggiungere logica specifica al flusso di lavoro. Una libreria integrata consente di salvare e organizzare funzioni, modelli o interi flussi per un facile riutilizzo in progetti futuri [18].

Il runtime di Node-RED è basato su Node.js, sfruttando appieno il modello non bloccante e basato sugli eventi di Node.js che consente al codice di eseguire operazioni in modo asincrono, senza attendere il completamento di ciascuna operazione prima di passare alla successiva [18]. Questo è particolarmente vantaggioso in scenari in cui è richiesta un'alta concorrenza e reattività, come nel caso di applicazioni IoT. Inoltre, Node.js è noto per la sua efficienza e leggerezza, il che lo rende adatto all'esecuzione su hardware a basso costo e in ambienti con risorse limitate.

Con una repository di pacchetti Node ricco di oltre 225.000 moduli, Node-RED offre una vasta estensibilità. Gli sviluppatori possono facilmente aggiungere nuove funzionalità alla tavolozza dei nodi per adattarsi alle esigenze specifiche del loro progetto [18].

Un altro vantaggio di Node-RED è la possibilità di archiviare i flussi creati utilizzando il formato JSON [18]. Questo consente un facile processo di importazione ed esportazione, facilitando la condivisione di flussi con altri utenti e la collaborazione su progetti comuni. In sintesi, Node-RED si presenta come un potente strumento per semplificare lo sviluppo e l'integrazione di soluzioni IoT e automazione.

3.3.1 Nodo exec

La crescente complessità delle applicazioni e dei flussi di lavoro all'interno di Node-RED ha evidenziato la necessità di esplorare approfonditamente le potenzialità offerte dal nodo "exec" [19]. Questo componente si presenta come una risorsa fondamentale quando è richiesto l'interfacciamento diretto con il sistema operativo o l'esecuzione di operazioni che coinvolgono comandi esterni. Nel contesto del mio progetto specifico, ho sfruttato appieno le caratteristiche avanzate di questo nodo, concentrandomi sulla sua configurabilità per l'esecuzione di comandi shell, in particolare l'utilizzo di comandi "curl" per richiedere al server il frame più recente e per pilotare la domotica della sala macchine. Nel caso della Figura 6 il nodo esegue una chiamata "curl" alla videocamera numero 2 `http://192.168.11.5:8082/camera2` chiedendo l'ultimo frame catturato e salvandolo con l'opzione `-o` sulla cartella `tmp` come "lastframe.jpg".

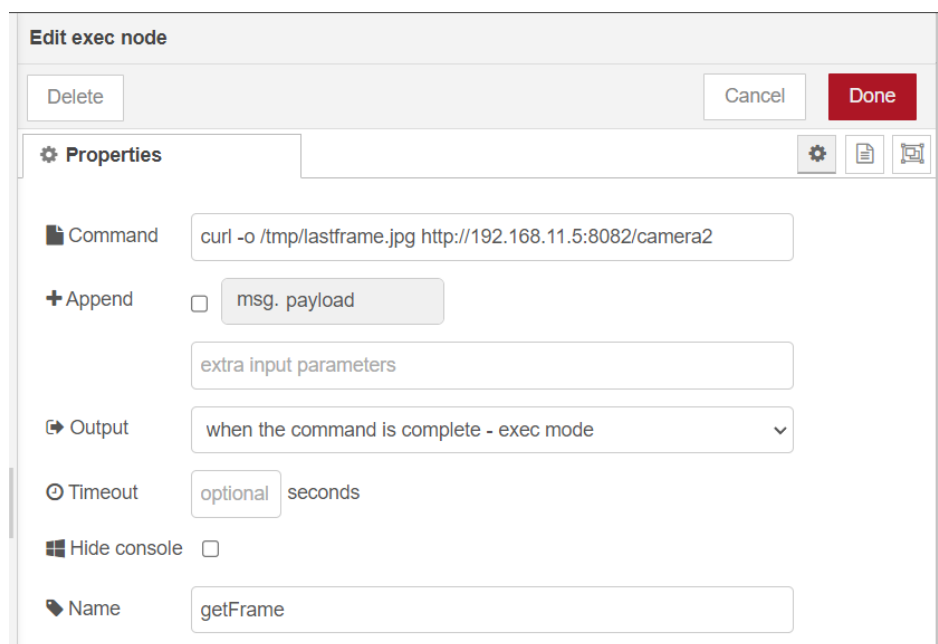


Figura 6: Nodo exec con curl

L'introduzione di parametri dinamici attraverso la sintassi dei modelli di Node-RED è stata una caratteristica particolarmente utile, permettendo la personalizzazione dinamica dei comandi "curl" in base alle condizioni e alle variabili di contesto del flusso di lavoro.

La gestione degli errori robusta del nodo "exec" è stata fondamentale per garantire l'affidabilità delle automazioni. Ho configurato il nodo in modo da gestire eventuali errori

critici in modo appropriato, interrompendo il flusso quando necessario o proseguendo in modo da non compromettere l'efficienza complessiva dell'applicazione.

Infine, la possibilità di selezionare modalità di esecuzione specifiche, come l'esecuzione in background o l'attesa del completamento, ha offerto un controllo preciso sul comportamento del nodo "exec". Questo ha consentito una personalizzazione avanzata del flusso di lavoro, adattandolo alle specifiche necessità del mio progetto di automazione e garantendo il recupero efficiente e tempestivo dei frame più recenti dal server remoto. In conclusione, l'analisi dettagliata del nodo "exec" in Node-RED ha contribuito in modo significativo alla comprensione approfondita delle sue funzionalità e ha dimostrato il suo ruolo centrale nell'implementazione di soluzioni automatizzate e flessibili, come nel caso delle "curl" usate per ottenere frame dal server remoto.

3.3.2 Nodo read file

Il nodo "Read File" in Node-RED si rivela come un componente essenziale per l'interazione con i file di dati all'interno dei flussi di lavoro. La sua funzionalità principale consiste nella lettura di file presenti nel sistema, incorporando tali dati nel flusso di Node-RED, nello specifico è stato utilizzato questo nodo prendendo spunto dalla lettura di un file CSV [20] modificando il flusso e utilizzando solo il nodo interessato. Questo nodo si dimostra particolarmente utile in contesti in cui è necessario acquisire informazioni da file di configurazione, dati di log o altri tipi di file memorizzati localmente. La Figura 7 rappresenta l'implementazione utilizzate nel progetto di tesi.



Figura 7: Nodo Read File

Nel mio caso specifico, ho sfruttato appieno le capacità del nodo "Read File" per leggere un file di immagine in formato jpg. Questo ha consentito l'integrazione diretta del contenuto dell'immagine nel flusso di lavoro di Node-RED, aprendo possibilità avanzate di manipolazione e analisi all'interno dell'ambiente di sviluppo. La versatilità del nodo si riflette nella sua capacità di gestire formati di file diversi, offrendo la flessibilità necessaria per leggere testo, JSON o altri formati specifici con facilità.

La configurabilità del nodo "Read File" è un elemento chiave, poiché consente di specificare dinamicamente il percorso del file da leggere. Questa caratteristica si è dimostrata cruciale nel mio progetto, poiché ha permesso un adattamento dinamico ai cambiamenti nel sistema di file. La gestione flessibile dell'output del nodo consente inoltre un'interazione agevole con altri nodi nel flusso, ampliando ulteriormente le possibilità di analisi e manipolazione dei dati letti all'interno dell'ambiente Node-RED.

3.3.3 Nodo HTTP request

Il nodo "HTTP Request" si configura come una risorsa fondamentale all'interno dell'ecosistema Node-RED, fornendo un meccanismo efficiente per la comunicazione con risorse esterne attraverso richieste http [21]. Questo componente si rivela di particolare utilità in scenari in cui è necessario interagire con servizi web, API e altri endpoint remoti [21].

La Figura 8 rappresenta la configurabilità avanzata del nodo "HTTP Request" che offre la flessibilità necessaria per definire parametri cruciali, come il metodo di richiesta, l'URL di destinazione, le intestazioni personalizzate e il payload dati [21]. Questa versatilità si estende alla gestione delle risposte HTTP, consentendo di manipolare e incorporare i dati ricevuti direttamente nel flusso di lavoro di Node-RED.

Nel mio caso specifico, ho sfruttato appieno le potenzialità del nodo "HTTP Request" per inviare richieste sia alla Jetson AGX Orin che alla Radxa Rock 5 B. Inviando i frame catturati dalle telecamere, ho ottenuto in risposta i risultati dell'Item detection sotto forma di JSON. Questa implementazione ha consentito di integrare in modo efficiente l'analisi degli oggetti rilevati nel contesto del mio progetto, arricchendo il flusso di lavoro di Node-RED con informazioni provenienti da risorse esterne.

La possibilità del nodo "HTTP Request" di impostare opzioni di sicurezza e autenticazione garantisce inoltre una comunicazione sicura con risorse protette, contribuendo alla robustezza complessiva del sistema implementato.

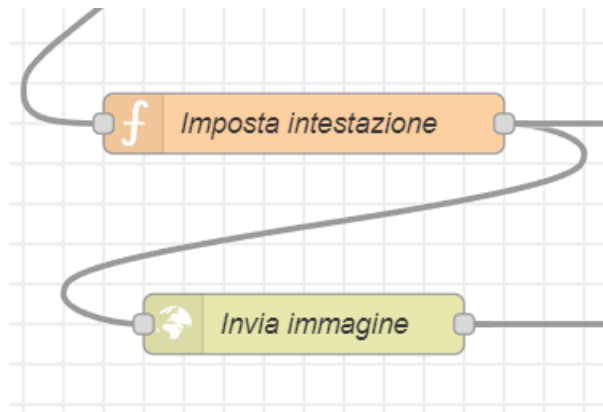


Figura 8: Nodo HTTP Request

3.4 Docker e LXC container

Docker Container [22] e LXC (Linux Containers) [23], rappresentati in Figura 9 sono due approcci distinti ma potenti per la virtualizzazione a livello di sistema operativo, offrendo ambienti isolati e portatili per l'esecuzione di applicazioni. Docker, una piattaforma di containerizzazione innovativa, ha guadagnato popolarità grazie alla sua facilità d'uso e alla gestione semplificata dei container [22]. Basato su Docker Engine, consente agli sviluppatori di confezionare le proprie applicazioni e le relative dipendenze in container leggeri e autonomi, garantendo consistenza tra ambienti di sviluppo e produzione [22].

LXC è un'implementazione di container basata sul kernel Linux, che offre una virtualizzazione a livello di sistema operativo [23]. LXC fornisce un approccio più "tradizionale", mettendo maggiormente l'accento sulla gestione manuale delle configurazioni e delle risorse di sistema [23]. Viene spesso scelto in scenari in cui è richiesto un maggiore controllo sulle risorse di sistema e dove la flessibilità di configurazione è prioritaria [23].

Entrambi i modelli offrono isolamento delle risorse, portabilità e scalabilità, ma differiscono nel modo in cui gestiscono la creazione e la distribuzione dei container [24]. Docker è noto per il suo vasto ecosistema e la facilità di utilizzo tramite Docker Compose per orchestrare container multi-servizio [22]. D'altra parte, LXC fornisce una maggiore personalizzazione e controllo diretto sulle risorse, consentendo agli utenti di adattare gli ambienti container alle esigenze specifiche.

Nel mio caso specifico, Docker e LXC sono stati utilizzati per implementare servizi contenuti all'interno di una macchina host consentendo di mantenere l'integrità della stessa senza apportare modifiche significative.

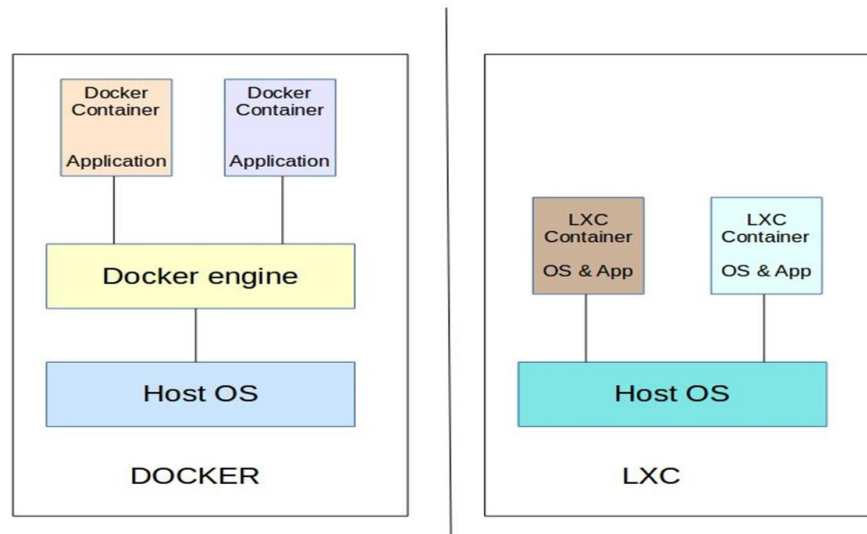


Figura 9: LXC e Docker [22]

3.4.1 Comandi Docker utilizzati

E' importante evidenziare alcune configurazioni chiave per la corretta comprensione del contesto di utilizzo [25]. In particolare, il comando `run` è stato impiegato per avviare nuovi container basati su specifiche immagini. Alcune configurazioni e opzioni degne di nota includono l'uso del flag `--gpus` per specificare l'assegnazione di risorse GPU (vedi capitolo 3.6.1) al container, agevolando carichi di lavoro che richiedono accelerazione hardware.

L'opzione `--privileged` che permette di avviare un container con privilegi elevati, come quello di accedere direttamente all'hardware, come l'NPU (vedi capitolo 3.7.1), o che modificano la configurazione del sistema.

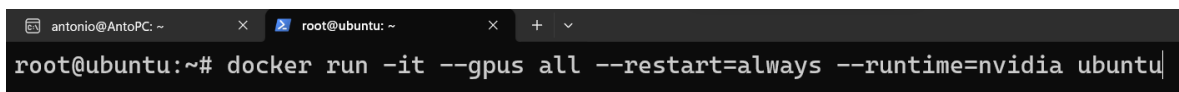
Inoltre, è stato utilizzato il comando `-p 8080:8080` per mappare la porta 8080 del container sulla porta 8080 della macchina host [25]. Questa configurazione facilita la comunicazione con il servizio all'interno del container attraverso la porta specificata sulla macchina host.

Un'altra scelta importante è stata l'opzione `--restart=always`, che configura il comportamento di riavvio automatico del container [25]. Questa opzione garantisce che il

container si avvii automaticamente in caso di arresto anomalo o riavvio del sistema, contribuendo alla continuità del servizio.

Queste configurazioni, integrate nel comando `run`, sono cruciali per garantire il corretto funzionamento del servizio Docker e la gestione efficiente delle risorse, fornendo un quadro chiaro delle scelte operative adottate nel contesto del progetto.

Tutti i comandi citati sono stati concatenati per la creazione del container Docker come in Figura 10.

A screenshot of a terminal window with a dark background. The window has two tabs: 'antonio@AntoPC: ~' and 'root@ubuntu: ~'. The active tab is 'root@ubuntu: ~'. The command prompt shows 'root@ubuntu:~#' followed by the command 'docker run -it --gpus all --restart=always --runtime=nvidia ubuntu|'. The cursor is at the end of the command.

```
root@ubuntu:~# docker run -it --gpus all --restart=always --runtime=nvidia ubuntu|
```

Figura 10: Terminale con comandi Docker

3.5 Json come formato di interscambio dati

Il formato di scambio dati JSON (JavaScript Object Notation) rappresenta una pietra angolare nell'ecosistema delle tecnologie web e nell'interscambio di informazioni tra applicazioni [26]. Questo formato leggero, basato su testo, offre una struttura chiara e flessibile per rappresentare dati complessi attraverso coppie chiave-valore e strutture annidate [26]. La sintassi di JSON è ispirata da JavaScript, ma è indipendente dal linguaggio, rendendolo universalmente comprensibile e facilmente interpretabile da una vasta gamma di linguaggi di programmazione [26].

L'approccio semplice di JSON lo rende ideale per la trasmissione di dati tra server e client, nonché per la memorizzazione di configurazioni e informazioni di configurazione. La sua struttura chiara e leggibile agevola la comprensione e la manipolazione dei dati, facilitando lo sviluppo di applicazioni interconnesse e la creazione di API (Application Programming Interface) efficienti [26].

JSON è particolarmente adatto per rappresentare dati complessi come array, oggetti e stringhe, consentendo di modellare una vasta gamma di informazioni in un formato standardizzato, come la risposta di un server Figura 11.

```

25/2/2024, 12:06:09 node: Risultato
msg.payload : string[123]
▼ string[123]
{
  "number_people": 2,
  "average_run_time": [
    118.03603172302246,
    70.86491584777832,
    7.0209503173828125
  ]
}

```

Figura 11: JSON output item detect

3.6 Nvidia Jetson AGX Orin

La Nvidia Jetson AGX Orin emerge come una soluzione hardware di fascia alta, nota per le sue prestazioni eccezionali nel campo dell'intelligenza artificiale e del machine learning [27]. Dotata di un processore NVIDIA Orin SoC, questa piattaforma offre una potente unità di elaborazione grafica (GPU) e Tensor Cores come in Figura 12 usati per accelerare le operazioni di deep learning. La presenza di connettività avanzata e la capacità di gestire carichi di lavoro complessi la rendono adatta a scenari di elaborazione intensiva, come il rilevamento di oggetti in tempo reale e l'analisi di immagini ad alta risoluzione [27].

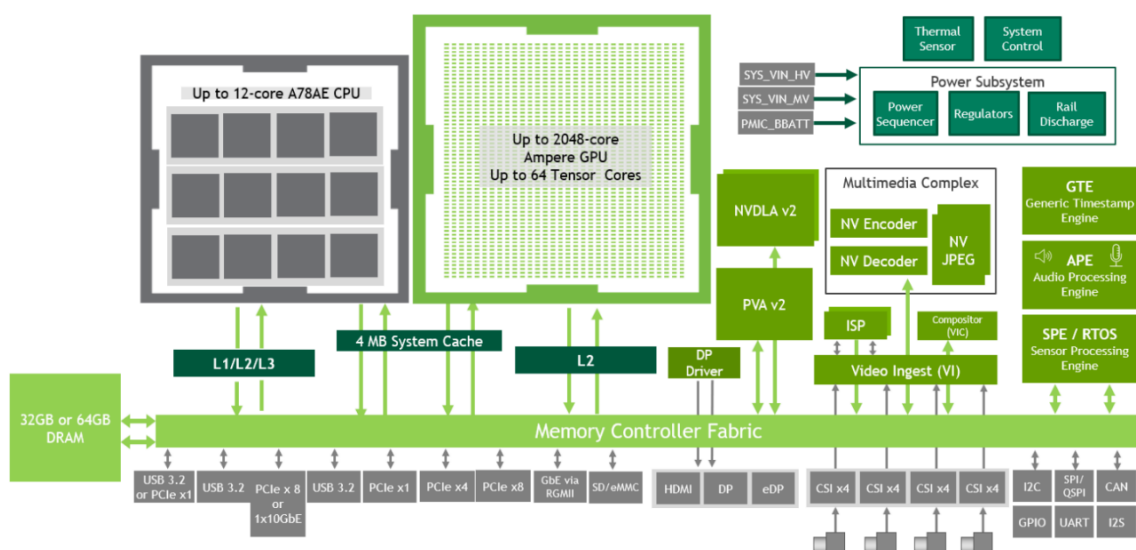


Figura 12: Nvidia Jetson AGX Orin Developer Kit [28]

3.6.1 GPU - CUDA 11.4

La Nvidia Jetson AGX Orin sfrutta la potenza di calcolo di CUDA 11.4, un ambiente di sviluppo parallelo creato da NVIDIA per la programmazione su GPU. CUDA (Compute Unified Device Architecture) consente agli sviluppatori di sfruttare appieno le capacità di calcolo parallelo della GPU presente sulla Jetson AGX Orin [29].

La versione 11.4 di CUDA porta con sé numerosi miglioramenti e ottimizzazioni, offrendo un ambiente di sviluppo avanzato per le applicazioni di intelligenza artificiale e machine learning [29]. Grazie a CUDA, è possibile sfruttare appieno le potenzialità della GPU e dei Tensor Core integrati nella piattaforma Jetson AGX Orin, accelerando notevolmente le operazioni di deep learning [29].

3.7 Rockchip Radxa Rock 5 B

Il Rockchip Radxa Rock 5 B rappresenta una piattaforma hardware versatile, basata su architettura ARM, che offre un equilibrio tra prestazioni e consumo energetico [30]. Equipaggiato con un processore Rockchip SoC RK3588, come in Figura 13, questa piattaforma si distingue per la sua capacità di gestire carichi di lavoro vari, rendendola idonea a una gamma diversificata di applicazioni [30]. La sua flessibilità è ulteriormente evidenziata dalla presenza di interfacce di connessione e opzioni di espansione, rendendola adatta a progetti con requisiti di personalizzazione e integrazione specifici [30].

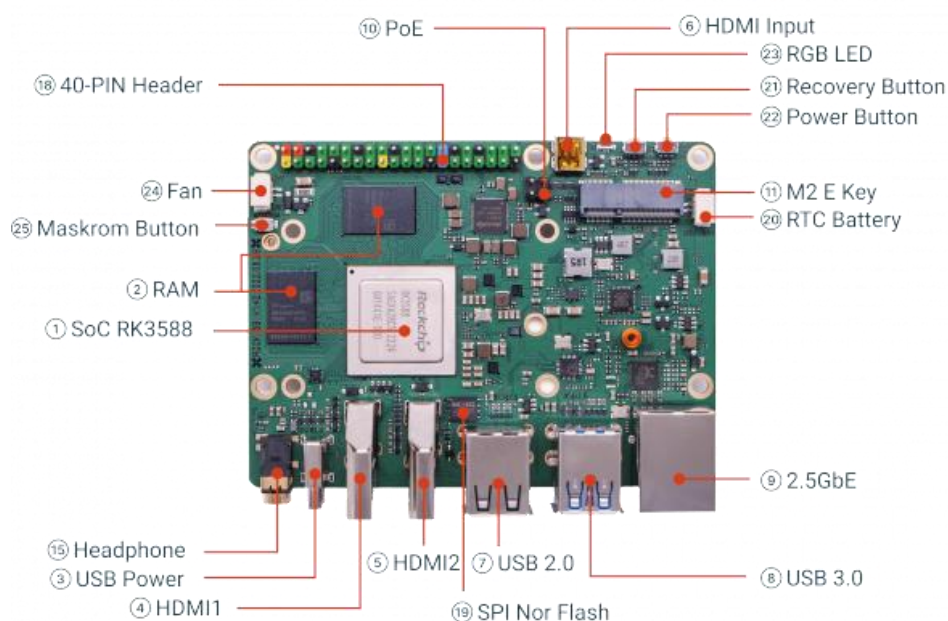


Figura 13: Rockchip Radxa RK3588 [31]

3.7.1 NPU

La Neural Processing Unit (NPU) integrata nel Rockchip Radxa Rock 5 B è progettata per accelerare operazioni specifiche delle reti neurali, la NPU ottimizza le prestazioni del dispositivo in scenari come il riconoscimento di immagini e il processamento del linguaggio naturale e contribuisce al risparmio energetico, ottimizzando il bilanciamento tra prestazioni e consumo energetico complessivo del sistema [32]. Questo aspetto è particolarmente rilevante in contesti in cui l'efficienza energetica è critica.

3.8 Intel NUC Mele

Gli Intel NUC Mele sono mini-PC compatti e potenti basati su architettura x86, offrendo un'opzione affidabile per applicazioni di dimensioni ridotte o ambienti con spazio limitato. Equipaggiati con processori Intel di ultima generazione, queste piattaforme forniscono una potenza di elaborazione notevole in un formato compatto [33]. La presenza di porte di connettività varie e la facilità di integrazione le rendono adatte a soluzioni embedded o progetti che richiedono una combinazione di prestazioni e compattezza [33].

3.9 Shelly Pro 2 PM

Lo Shelly Pro 2PM, ovvero la Figura 14, rappresenta una soluzione avanzata nel contesto dell'automazione domestica e industriale, offrendo un dispositivo versatile e ricco di funzionalità [34]. Progettato dalla casa produttrice Shelly, il Pro 2PM è un interruttore a due canali che consente di controllare e monitorare dispositivi elettrici in modo remoto attraverso la rete Wi-Fi [34].



Figura 14: Shelly Pro 2PM

Questo dispositivo è particolarmente notevole per la sua flessibilità e adattabilità, consentendo agli utenti di integrare facilmente l'automazione nell'ambiente lavorativo [34]. Dotato di due canali di commutazione, il Pro 2PM offre la possibilità di controllare separatamente due carichi elettrici, offrendo una gestione dettagliata e personalizzabile delle risorse energetiche [34].

4. Spiegazione della sperimentazione

L'idea di sperimentazione dalla quale si è partiti è quella di realizzare un Cobot che gestisce la temperatura di una sala macchine, per questo abbiamo utilizzato diversi dispositivi hardware e linguaggi di programmazione come Node-RED e Python.

In Node-RED è stata sviluppato tutto il sistema di richieste HTTP, ovvero “curl”, svolte sulle varie macchine, per esempio la “cattura” del frame più recente richiesta al server che gestisce le videocamere di sorveglianza o quelle che inviano la notifica della presenza o meno di un operatore all'interno dell'ambiente sottoposto a sperimentazione.

La sperimentazione si basa sull'uso di due dispositivi hardware, la Nvidia Jetson AGX Orin e la Rockchip Radxa 3588 B, che presentano notevoli differenze in termini di costi, consumi e prestazioni. Nonostante queste disparità, entrambi i dispositivi sono destinati a svolgere lo stesso compito, ossia identificare la presenza umana all'interno di una sala macchine. La scelta di utilizzare due hardware distinti consente di esplorare diverse soluzioni e approcci, fornendo un'analisi più approfondita delle potenzialità e delle sfide legate all'integrazione di tecnologie di videosorveglianza nell'ambito dell'automazione industriale.

Per la realizzazione del sistema presentato nella Figura 15, oltre alla Nvidia Jetson AGX Orin e alla Rockchip Radxa 3588 B, sono stati impiegati altri dispositivi, come il server di videosorveglianza con il quale catturavamo i frame da analizzare, la domotica della sala server e l'Intel NUC Mele impiegato per la comunicazione tra i diversi dispositivi. Infine, sono stati evidenziati in rosso i container LXC e in azzurro i container Docker realizzati. Inoltre, nello schema è stata evidenziata la direzionalità delle richieste HTTP utilizzando le frecce nere per l'invio della richiesta, mentre quelle rosse per la risposta. Nel caso specifico, il Container Node-red richiede l'ultimo frame catturato al server della videosorveglianza e successivamente lo invia sia al Container jetsonServerYolov5 e sia al Container radxaServerYolov5 i quali invieranno come risposta i risultati dell'Item detection. Se verrà rilevata la presenza di un operatore all'interno della sala server, sarà compito del Container Node-red informare il sistema di domotica della sala macchine, il quale si occuperà di regolare la temperatura di conseguenza.

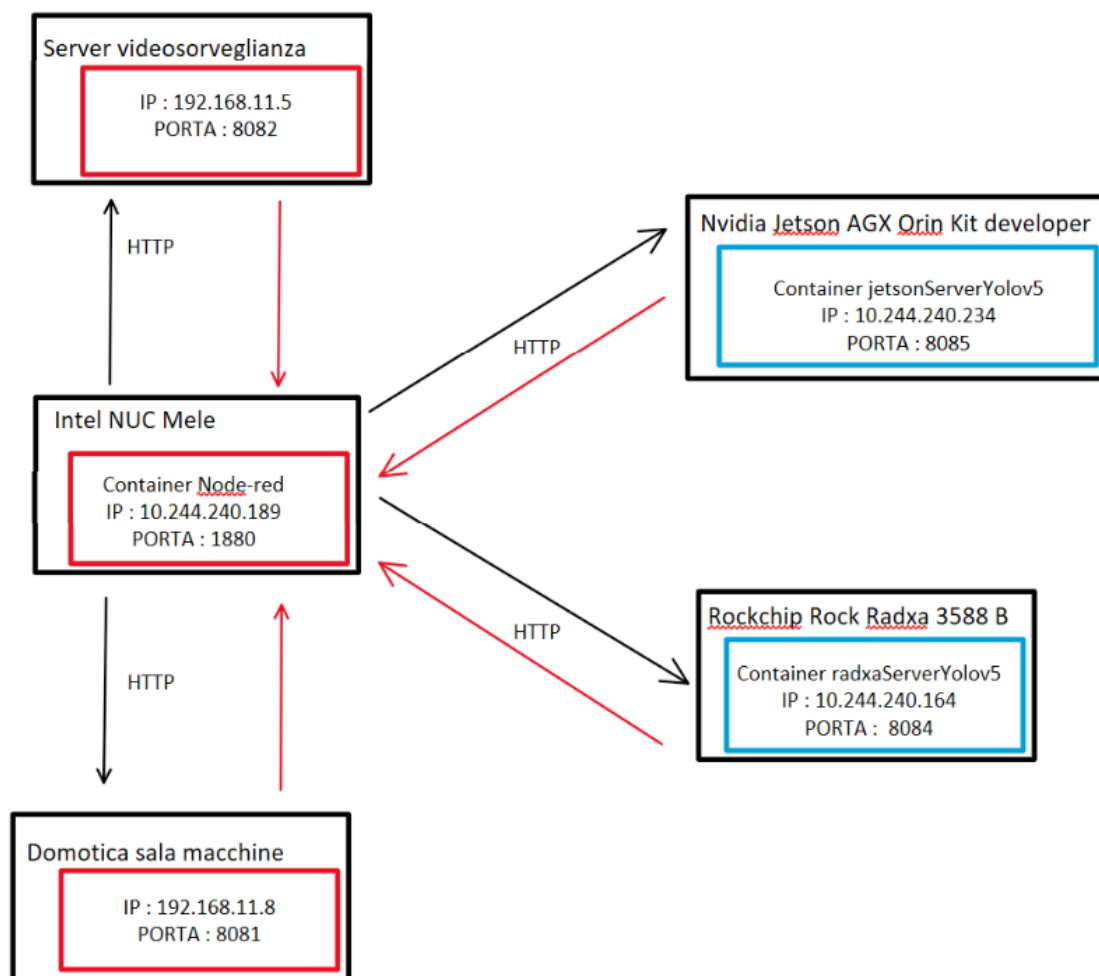


Figura 15: Il sistema implementato

Sono stati utilizzati differenti modelli per testare a pieno le capacità dei dispositivi, Nvidia Jetson AGX Orin e alla Rochchip Radxa 3588 B, in modo da avere un riscontro concreto in termini di frame al secondo e di veridicità del rilevamento delle persone.

Per monitorare i consumi relativi alla Nvidia Jetson AGX Orin e alla Rochchip Radxa 3588 B è stato utilizzato uno Shelly Pro 2 PM B, un magnetotermico intelligente in grado di fornire il consumo in watt.

La scelta di implementare il sistema sull'“Edge” anziché nel cloud è stata dettata da diverse considerazioni strategiche e pratiche. Innanzitutto, l'elaborazione dei dati direttamente sull'“Edge”, cioè nei dispositivi locali come la Nvidia Jetson AGX Orin e la Rochchip Radxa 3588 B, consente una risposta più immediata e tempestiva alle situazioni rilevanti. Questo è

particolarmente critico in ambienti industriali in cui la latenza è un fattore determinante per garantire una risposta in tempo reale.

Inoltre, l'elaborazione locale riduce la dipendenza dalla connessione internet, garantendo che il sistema possa operare anche in situazioni in cui la connessione potrebbe essere instabile o assente. Ciò contribuisce a migliorare l'affidabilità complessiva del sistema, specialmente in contesti in cui la continuità operativa è fondamentale, come nelle applicazioni industriali.

La sicurezza è un altro aspetto importante. L'elaborazione dei dati sull'“Edge” limita la trasmissione di informazioni sensibili verso il cloud, riducendo potenziali rischi legati alla sicurezza e alla privacy. In ambienti industriali, in cui la protezione dei dati è fondamentale, questa scelta può contribuire a mitigare possibili vulnerabilità.

La varietà nei dispositivi selezionati introduce un elemento di eterogeneità nella sperimentazione, consentendo di confrontare le performance, l'efficienza e la scalabilità delle soluzioni proposte. L'obiettivo è non solo identificare la presenza umana e modificare i parametri termici della sala macchine, ma anche valutare come le diverse caratteristiche hardware influiscano sulla capacità di elaborazione e sulla reattività dei sistemi proposti.

4.1 Scelta di YOLOv5

Analizzando il grafico delle performance delle varie versioni di YOLO, ossia in Figura 16, dove sull'asse delle ascisse troviamo la COCO mAP, ovvero il tempo necessario per elaborare un'immagine, mentre sull'asse delle ordinate troviamo Parameters(M), il numero dei parametri del modello. La scelta del modello migliore dipende dalle esigenze specifiche dell'applicazione. Se è importante la velocità, YOLOv5 potrebbe essere la scelta migliore. Se è importante la precisione, YOLOv8 potrebbe essere la scelta migliore.

Per la sperimentazione è stato scelto YOLOv5, perché come si può vedere dal grafico, YOLOv5 è più veloce di YOLOv8. Tuttavia, YOLOv8 è più preciso. In particolare, prendendo in analisi la variante “x” con dimensioni e prestazioni maggiori, YOLOv8 ottiene un punteggio COCO mAP di circa 55, mentre YOLOv5 ottiene un punteggio poco superiore a 50. In proporzione, le varianti più piccole sono simili.

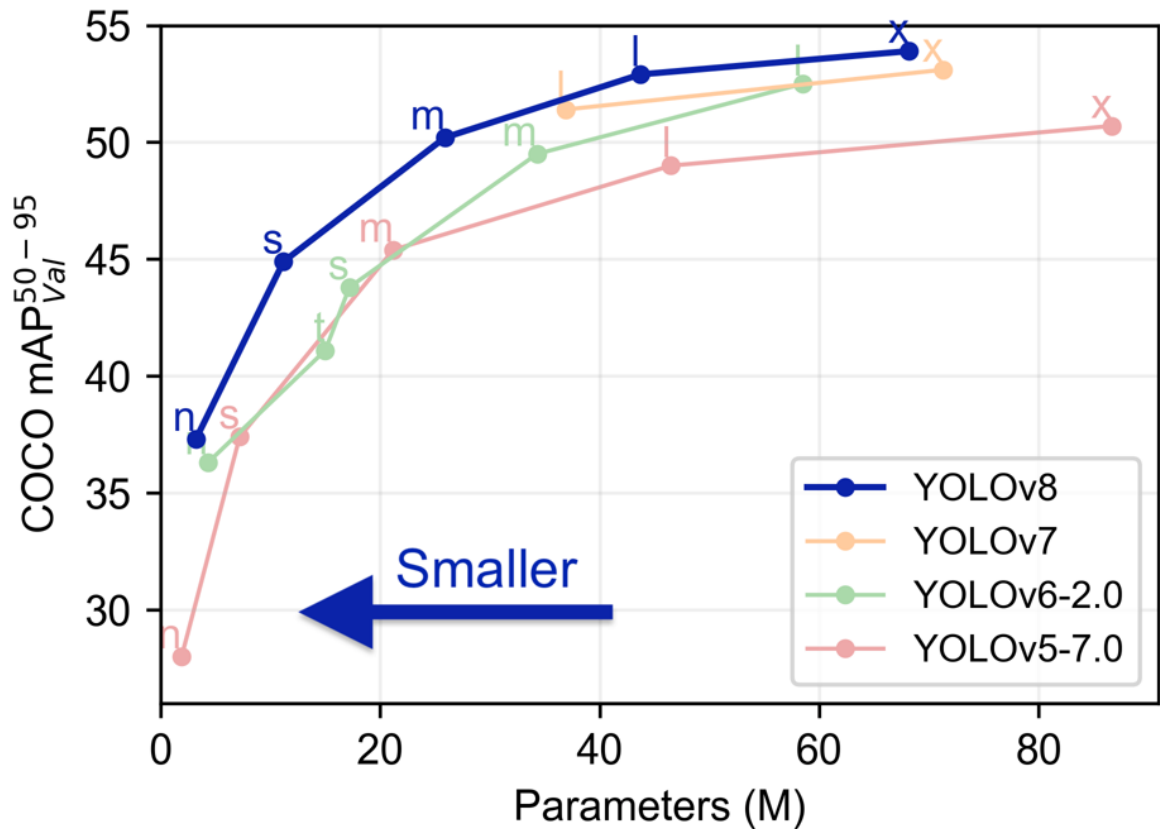


Figura 16: Confronto differenti versioni di YOLO [35]

4.2 Inizializzazione Jetson AGX Orin e Radxa Rock 5 B

Il processo di inizializzazione dei dispositivi Jetson AGX Orin e Radxa Rock 5 B è risultato complesso, con particolare enfasi sulla Jetson AGX Orin, la quale presenta una procedura più articolata rispetto alla tradizionale operazione di flashing e installazione del sistema operativo [36]. Nel caso della Jetson AGX Orin, come da manuale Nvidia, è stato necessario ricorrere all'utilizzo di una macchina virtuale con Ubuntu 18 installata su un altro dispositivo collegato tramite USB-C alla Jetson AGX Orin, data la necessità di utilizzare il Nvidia SDK Manager per Jetson AGX Orin, un'applicazione fondamentale per il download e l'installazione di tutti i pacchetti necessari, compreso CUDA 11.4 [36].

È importante notare che, alla fine, sulla Jetson AGX Orin è stato installato Ubuntu 20.04 Focal, mentre sulla Radxa Rock 5 B è stato utilizzato Ubuntu 22.04 Jammy. Questa specifica scelta di versioni di sistema operativo riflette l'adeguamento alle esigenze e alle compatibilità dei dispositivi, garantendo una configurazione ottimale e coerente con le richieste del progetto.

4.2.1 Installazione YOLOv5 su container Docker – Jetson AGX Orin

L'implementazione di YOLOv5 all'interno di un container Docker su Jetson AGX Orin è stata una scelta strategica al fine di massimizzare la portabilità, garantire l'isolamento dell'ambiente, semplificare la gestione delle dipendenze e assicurare la coerenza nell'esecuzione del modello di rilevazione di oggetti. Questa decisione riflette la necessità di creare un ambiente computazionale controllato, indipendente dalle variabili dell'host, per garantire una corretta esecuzione del software su diversi dispositivi e contesti. L'utilizzo del container Docker facilita la riproducibilità dell'ambiente di sviluppo, minimizzando i conflitti di dipendenze e offrendo una soluzione scalabile e facilmente gestibile per la rilevazione di oggetti su dispositivi Jetson AGX Orin.

4.2.1.1 Creazione e avvio del container Docker

E' stato realizzato un container Docker partendo dall'immagine fornita da Nvidia [37] come mostrato in Figura 17, contenente Cuda 11.4 e Pytorch preinstallati, utilizzando il comando Docker:

```
docker run -it --gpus all --runtime=nvidia  
nvcr.io/nvidia/l4t-pytorch:r35.2.1-pth2.0-py3
```

il quale riveste un ruolo importante nell'implementazione di un ambiente di esecuzione ottimizzato su dispositivi Jetson AGX Orin. Analizziamo dettagliatamente ciascun componente del comando per comprenderne l'importanza in una prospettiva più ampia e discorsiva.

L'opzione `-it` sottolinea l'aspetto interattivo dell'ambiente container, fornendo un punto di interazione diretta con il sistema operativo all'interno del container. Questa caratteristica è fondamentale quando si necessita di eseguire comandi manuali o controllare il flusso di esecuzione, aspetti particolarmente rilevanti durante lo sviluppo e la fase di debug.

La specifica `--gpus all` indica chiaramente l'intenzione di sfruttare tutte le unità di elaborazione grafica (GPU) disponibili. In un contesto di machine learning, dove le operazioni intensive di calcolo possono trarre vantaggio dalla potenza delle GPU, questa opzione rappresenta una scelta strategica per ottimizzare le prestazioni dell'applicazione PyTorch all'interno del container.

L'opzione `--runtime=nvidia` è essenziale per garantire una corretta comunicazione tra il container Docker e le GPU NVIDIA del sistema. Rappresenta un ponte fondamentale che abilita le operazioni di calcolo parallelo, contribuendo così a massimizzare l'utilizzo delle risorse hardware disponibili.

L'effettivo utilizzo della GPU da parte di PyTorch è stato verificato avviando un ambiente Python all'interno del container Docker. Ciò è stato realizzato importando la libreria `torch` e successivamente eseguendo il metodo `torch.cuda.is_available()`. Questo approccio ha permesso di confermare con certezza che PyTorch ha accesso e la capacità di sfruttare le unità di elaborazione grafica disponibili, un aspetto critico nell'ottimizzazione delle prestazioni dei modelli di machine learning implementati.

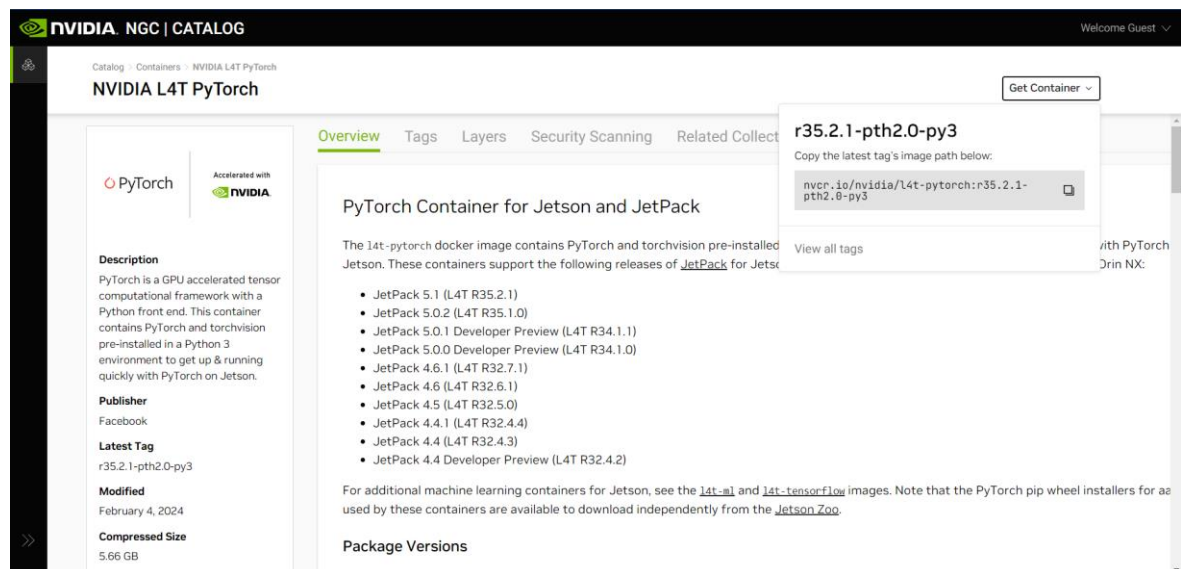


Figura 17: Download container Docker Jetson AGX Orin

4.2.1.2 Preparazione dell'ambiente YOLOv5

Nell'implementazione dell'ambiente YOLOv5 su dispositivi Jetson AGX Orin, è stato creato un ambiente dedicato nella directory utente denominata "user1". All'interno di questa directory, la repository YOLOv5 è stata clonata dal repository su GitLab [35]. Tuttavia, durante il tentativo di esecuzione di YOLOv5, sono emersi problemi legati alla libreria OpenCV.

La risoluzione di questi problemi ha coinvolto un processo dettagliato, nel quale è stata installata e compilata la libreria OpenCV [38]. Questa operazione è stata eseguita attraverso

l'uso del comando `cmake` seguito da `make -j 8`, che ha abilitato la compilazione sfruttando tutti gli 8 core del processore [38]. Questa fase critica ha permesso di superare le problematiche iniziali legate a OpenCV, consentendo così l'esecuzione corretta di YOLOv5 sul dispositivo Jetson AGX Orin.

4.2.1.3 Esecuzione dell'inferenza di test

Dopo una dettagliata ricerca, la versione specifica di Ultralytics selezionata per integrare con successo l'ambiente YOLOv5 è stata la 8.0.232. Una volta completata l'installazione di questa versione, il processo di configurazione è stato portato a termine. Questa fase ha posto le basi per effettuare il test di funzionamento eseguendo il file "detect.py" mediante l'interprete Python3.

L'avvio di "detect.py" rappresenta il momento culminante del processo di implementazione, in cui YOLOv5 entra in azione per rilevare oggetti all'interno delle immagini o dei video specificati. L'esecuzione di questo test è un passo fondamentale che conferma il corretto funzionamento dell'intero sistema implementato su dispositivi Jetson AGX Orin.

Questo momento segna la conclusione della fase di installazione e configurazione, aprendo la strada a successive analisi e valutazione delle prestazioni del modello YOLOv5 nell'ambiente specifico dei dispositivi Jetson AGX Orin. La scelta della versione specifica di Ultralytics e l'esecuzione del test "detect.py" riflettono la precisione e la cura dedicate alla realizzazione di un ambiente operativo efficiente e funzionale per la rilevazione di oggetti.

4.2.2 Installazione YOLOv5 su container Docker - Radxa Rock 5 B

Per creare un ambiente isolato e ottimizzato per eseguire YOLOv5 sul NPU del dispositivo Radxa Rock 5 B, utilizzando un container Docker in un ambiente basato su Ubuntu 22.04 LTS (jammy), è necessario seguire una serie di passaggi precisi [39].

4.2.2.1 Realizzazione e Avvio del Container Docker

Una fase fondamentale nell'implementazione di YOLOv5 su dispositivi Radxa Rock 5 B è la creazione e l'avvio del container Docker. Quest'operazione, all'apparenza apparentemente semplice, rappresenta il fondamento dell'ambiente di lavoro isolato e controllato in cui sarà orchestrato il sistema di rilevamento di oggetti YOLOv5. Partendo dall'immagine di Ubuntu 22.04 LTS (jammy), è stata plasmata un'istanza virtuale di sistema operativo mediante il comando :

```
docker run --privileged --name=rock -it ubuntu
```

Questo comando, con le sue opzioni distinte, apre una finestra sulla virtualizzazione, consentendo la creazione di un ambiente controllato e privilegiato. L'opzione `--privileged` è stata selezionata attentamente per garantire che il container abbia accesso completo alle risorse del sistema host, un aspetto essenziale per operazioni specializzate all'interno del container.

L'opzione `-it` rende il container interattivo, dotandolo di un terminale e agevolando l'interazione diretta con l'ambiente.

Infine, è stata controllata la versione del kernel utilizzando il comando: `uname -r` e verificato che fosse "5.10.160-rockchip"

4.2.2.2 Installazione di YOLOv5 ed esecuzione della demo

Dopo aver avviato il container Docker Radxa Rock 5 B e verificato la corretta versione del kernel, si è proceduto con la preparazione dell'ambiente per l'installazione di YOLOv5. Questo processo include la creazione di una directory dedicata, denominata "user1", in cui verranno eseguite le operazioni successive.

Successivamente, è stata scaricata l'implementazione di YOLOv5 da GitHub e copiata nella directory appena creata all'interno del container Docker utilizzando il comando `docker cp`. Una volta copiata, l'archivio è stato estratto nella directory "user1" e sono state configurate le variabili di ambiente necessarie per l'esecuzione di YOLOv5, aggiungendo l'export delle apposite librerie al file ".bashrc", ovvero :

```
export LD_LIBRARY_PATH=/user1/lib
/user1/rknn_yolov5_demo_linux
```

Inoltre, sono state scaricate e installate le librerie necessarie per il corretto funzionamento di YOLOv5 e per sfruttare al meglio le risorse hardware del dispositivo Radxa Rock 5 B. Tra queste librerie vi sono `librga2` e `libdrm2`, le cui versioni specifiche sono state scaricate da una repository di gitlab [40].

`Librga2` rappresenta una libreria grafica sviluppata da Rockchip, un'azienda rinomata per la produzione di processori ARM e soluzioni integrate. Il suo ruolo predominante è quello di

fornire un'interfaccia di programmazione delle applicazioni (API) per la gestione avanzata delle risorse grafiche su dispositivi basati su processori Rockchip.

Tra le sue funzionalità principali, librga2 spicca nella gestione dell'accelerazione hardware, decodifica video, e rendering grafico. Questa libreria è di particolare interesse per gli sviluppatori che lavorano su piattaforme Rockchip, offrendo strumenti essenziali per ottimizzare le prestazioni grafiche dei dispositivi.

Libdrm2, parte integrante del progetto DRM (Direct Rendering Manager), è una libreria cruciale nella gestione dei driver grafici su sistemi Linux. Contrariamente a una possibile confusione con l'acronimo DRM associato a Digital Rights Management, in questo contesto DRM si riferisce al Direct Rendering Manager.

Infine, una volta completate tutte le operazioni di preparazione dell'ambiente e installazione delle dipendenze, è stato eseguito il demo di YOLOv5 utilizzando il comando:

```
./rknn_yolov5_demo    ./model/RK3588/yolov5s-640-640.rknn  
./model/bus.jpg
```

seguito dai parametri necessari per l'esecuzione del modello su un'immagine di test.

In questo modo, l'ambiente Docker Radxa Rock 5 B è stato correttamente configurato e ottimizzato per l'esecuzione di YOLOv5, garantendo un'implementazione efficace e performante del sistema di rilevamento di oggetti su dispositivi Radxa Rock 5 B.

4.3 Creazione server per inferenza Jetson AGX Orin e Radxa Rock 5 B

Il processo di creazione del server per l'inferenza su dispositivi Jetson AGX Orin e Radxa Rock 5 B è stato attentamente pianificato e implementato attraverso l'uso di container Docker. Questa scelta strategica mira a rendere l'inferenza accessibile al container Nodered, integrandola nel nostro progetto in modo sinergico. Il server è configurato come un'istanza di tipo "ThreadingHTTPServer", che opera in modalità multi-thread per gestire più richieste contemporaneamente, garantendo così una maggiore efficienza nell'erogazione del servizio.

Durante l'avvio del server, è stata adottata una soluzione intelligente per minimizzare l'uso del disco: è stata montata una directory sulla tmpfs. Questa operazione consente di eseguire

le operazioni di salvataggio e ridimensionamento delle immagini direttamente in memoria, riducendo al minimo l'impatto sul disco. La procedura di avvio comprende la lettura dei dati byte per byte attraverso il server web, con la successiva conversione e salvataggio dell'immagine nel formato desiderato, in questo caso, ".jpg".

Il server rimane in ascolto su porte specifiche, utilizzando la porta 8084 per la Radxa Rock 5 B e la porta 8085 per la Jetson AGX Orin. Questo approccio consente una gestione chiara delle comunicazioni e facilita l'integrazione del server nell'ecosistema del nostro progetto complessivo.

4.3.1 L'inferenza sulla Jetson AGX Orin

Nel contesto della configurazione del server per l'inferenza sulla Jetson AGX Orin, un aspetto fondamentale consiste nel precaricare in memoria tutte le librerie necessarie sia per YOLOv5 che per il funzionamento stesso del server. Particolarmente rilevante è l'avvio del server con il caricamento immediato e ottimizzato del modello "yolov5s.pt" direttamente in memoria. Questa strategia mira a garantire un accesso rapido e senza intoppi al modello durante le fasi di inferenza.

L'ottimizzazione dell'inferenza sulla Jetson AGX Orin è stata implementata semplificando il processo nel modo più efficiente possibile, con una gestione attenta delle eccezioni. Questo approccio è finalizzato a garantire la stabilità del sistema durante le operazioni di rilevamento oggetti, semplificando al contempo il codice e migliorandone la leggibilità. L'attenzione alle eccezioni è di particolare importanza per garantire una maggiore affidabilità del server nell'affrontare scenari imprevisti o situazioni critiche durante l'inferenza.

4.3.2 L'inferenza sulla Radxa Rock 5 B

Nel contesto dell'inferenza sulla Radxa Rock 5 B, è stato necessario adottare un approccio differente rispetto alla Jetson AGX Orin. In questo caso, poiché avevamo a disposizione solo il compilato del modello "rknn" senza il sorgente, l'implementazione dell'inferenza si è basata su un processo specifico. Utilizzando il modello "rknn" e l'immagine su cui eseguire l'inferenza, abbiamo lanciato un comando dedicato. La risposta a questo comando consisteva in un file JSON contenente le informazioni rilevanti, in particolare il numero di persone individuate.

Successivamente, questo file JSON è stato elaborato all'interno del nostro sistema e il conteggio delle persone è stato inviato come risposta alla richiesta CURL proveniente da Nodered. Questa metodologia adottata ha consentito un'efficace integrazione tra il modello "rknn", l'immagine di input e il nostro sistema complessivo, garantendo al contempo un'adeguata gestione delle risposte e il trasferimento accurato delle informazioni rilevanti a Nodered.

4.4 Creazione container LXC su Intel NUC

L'Intel NUC è stato adoperato per istituire un ambiente di programmazione isolato, tramite un'implementazione container di LXC, rispetto agli altri dispositivi come la Jetson AGX Orin, la Radxa Rock 5 B, il server di videosorveglianza e il server della sala macchine. In questa configurazione, è stato creato un container LXC chiamato Nodered che ospita Debian 11, all'interno del quale è stato installato Node-RED insieme a tutti i nodi necessari per il progetto. I frame acquisiti e salvati sul container Nodered e saranno gestiti su una directory tmpfs, garantendo una gestione efficiente delle risorse e mantenendo elevate prestazioni senza sovraccaricare il disco. Questa scelta è stata effettuata per assicurare un ambiente di sviluppo isolato e ottimizzato per l'esecuzione di Node-RED, contribuendo alla coerenza e all'efficienza complessiva del sistema.

4.4.1 Implementazione flow Nodered

Per progettare il flusso di Node-RED di Figura 18, è stata posta particolare attenzione all'efficienza al fine di evitare sovraccarichi al server responsabile delle videocamere di sorveglianza. In tal senso, è stato implementato un nodo “Cattura Frame” che, con una periodicità di un secondo, esegue una richiesta curl al server per ottenere l'ultimo frame catturato e lo salva all'interno del container di Node-RED.

Parallelamente, si effettua la lettura del frame “Lettura file” memorizzato all'interno del container di Node-RED, che successivamente viene inviato sia alla Jetson AGX Orin che alla Radxa Rock 5 B per eseguire l'Item detection e rilevare la presenza di persone all'interno della sala server.

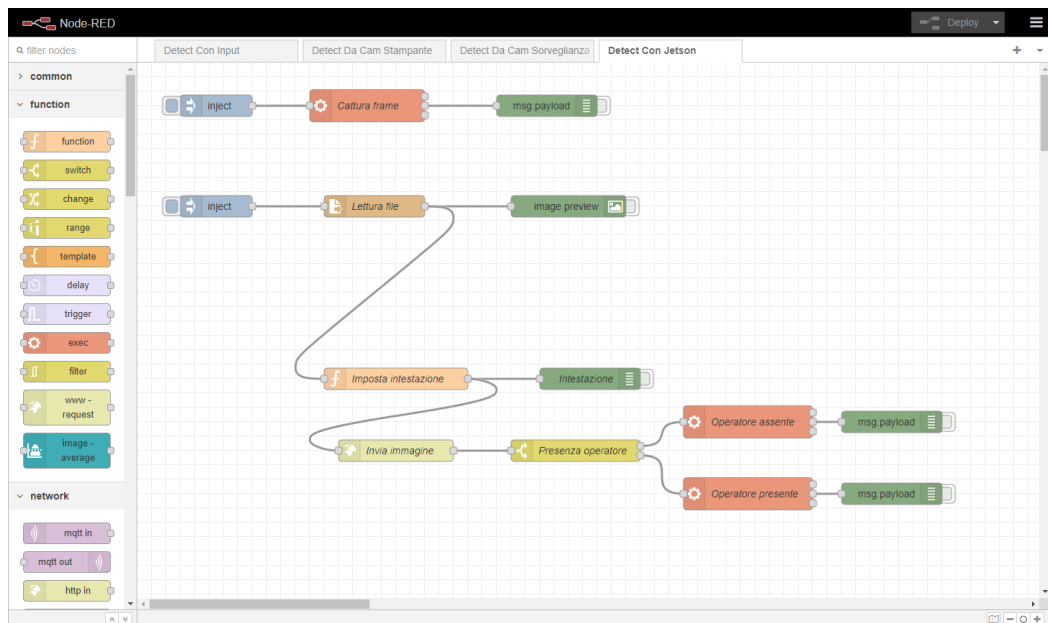


Figura 18: Flow Node-red

In caso di esito positivo, indicando la presenza di operatori nella sala, Node-RED comunica al server responsabile della gestione dell'ambiente la presenza rilevata e procede all'ottimizzazione della temperatura. Questa implementazione mira a garantire un monitoraggio efficiente e reattivo, evitando sovraccarichi inutili da parte del server delle videocamere, ma mettendo in sovraccarico o quasi i server Jetson AGX Orin o Radxa Rock 5 B che si occupano dell'inferenza.

5. Risultati della sperimentazione

I dati raccolti durante la fase di sperimentazione si sono rivelati fondamentali per valutare le performance delle diverse schede, focalizzando l'attenzione principalmente sulla Jetson AGX Orin, dove è stata possibile apportare modifiche dirette al codice sorgente, e sulla Radxa Rock 5 B, che ha utilizzato un codice precompilato. Questa distinzione è risultata cruciale per comprendere le potenzialità e le limitazioni di ciascuna piattaforma.

Durante la raccolta dati è stato utilizzato un unico frame di test, quello in Figura 19, così da poter ottenere i dati i più omogenei possibili.

Le metriche considerate per valutare le prestazioni dei dispositivi si concentrano principalmente sul numero di inferenze al secondo eseguite dalla Jetson AGX Orin e dalla Radxa Rock 5 B, offrendo un'analisi dettagliata delle capacità di elaborazione delle due schede. Inoltre, sarà considerato il consumo di potenza di entrambi i dispositivi durante le operazioni di inferenza. Questa analisi quantitativa fornirà una panoramica completa delle performance, consentendo di confrontare in modo esaustivo le due piattaforme.



Figura 19: Frame di test

5.1 Risultati Jetson AGX Orin

Nel processo di raccolta dati sulla Jetson AGX Orin, sono stati condotti vari test utilizzando il codice demo sorgente con diversi formati di modello, come `pytorch` e `torchscript`, e differenti rappresentazioni dei dati numerici, ovvero "int8" (integer 8-bit) e "half" (half-precision).

Lo studio è stato ampliato attraverso l'utilizzo e la conversione di differenti modelli di rete neurale YOLO, come Yolov5n e Yolov5s. Questi modelli hanno permesso di affrontare errori di tipo "tensor" e sono stati generati in due versioni: Yolov5n.torchscript con codifica int8 e Yolov5n.torchscript con codifica half, partendo da Yolov5n.pt; Yolov5s.torchscript con codifica int8 e Yolov5s.torchscript con codifica half, partendo da Yolov5s.pt.

Partendo da un frame catturato, Figura 19, dalla videocamera con 2 persone al lavoro in azienda, sono stati ottenuti i seguenti risultati, Figura 20, utilizzando Yolov5n:

```
root@bd7a11bcf7d0:/user1/yolov5# python detect.py --source=/user1/server/image/input.jpg --device=0 --weights=yolov5n.pt
detect: weights=['yolov5n.pt'], source=/user1/server/image/input.jpg, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000
es=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False
WARNING ⚠️ torchvision==0.14 is incompatible with torch==2.0.
Run 'pip install torchvision==0.15' to fix torchvision or 'pip install -U torch torchvision' to update both.
For a full compatibility table see https://github.com/pytorch/vision#installation
YOLOv5 🚀 v7.0-267-gc42aba5 Python-3.8.10 torch-2.0.0a0+ec3941ad.nv23.02 CUDA:0 (Orin, 62801MiB)

Fusing layers...
YOLOv5n summary: 213 layers, 1867405 parameters, 0 gradients
image 1/1 /user1/server/image/input.jpg: 544x640 2 persons, 2 chairs, 1 tv, 31.6ms
Speed: 1.9ms pre-process, 31.6ms inference, 4.4ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp9
root@bd7a11bcf7d0:/user1/yolov5# python detect.py --source=/user1/server/image/input.jpg --device=0 --weights=/user1/yolov5/int8s/yolov5n.torchscript
detect: weights=['/user1/yolov5/int8s/yolov5n.torchscript'], source=/user1/server/image/input.jpg, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.25,
op=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=exp, exist_ok=False, line_t
WARNING ⚠️ torchvision==0.14 is incompatible with torch==2.0.
Run 'pip install torchvision==0.15' to fix torchvision or 'pip install -U torch torchvision' to update both.
For a full compatibility table see https://github.com/pytorch/vision#installation
YOLOv5 🚀 v7.0-267-gc42aba5 Python-3.8.10 torch-2.0.0a0+ec3941ad.nv23.02 CUDA:0 (Orin, 62801MiB)

Loading /user1/yolov5/int8s/yolov5n.torchscript for TorchScript inference...
image 1/1 /user1/server/image/input.jpg: 640x640 2 persons, 2 chairs, 1 tv, 30.5ms
Speed: 3.6ms pre-process, 30.5ms inference, 6.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp10
*[[Arroot@bd7a11bcf7d0:/user1/yolov5# python detect.py --source=/user1/server/image/input.jpg --device=0 --weights=/user1/yolov5/halves/yolov5n.torchscript
detect: weights=['/user1/yolov5/halves/yolov5n.torchscript'], source=/user1/server/image/input.jpg, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.25,
op=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=exp, exist_ok=False, line_t
WARNING ⚠️ torchvision==0.14 is incompatible with torch==2.0.
Run 'pip install torchvision==0.15' to fix torchvision or 'pip install -U torch torchvision' to update both.
For a full compatibility table see https://github.com/pytorch/vision#installation
YOLOv5 🚀 v7.0-267-gc42aba5 Python-3.8.10 torch-2.0.0a0+ec3941ad.nv23.02 CUDA:0 (Orin, 62801MiB)

Loading /user1/yolov5/halves/yolov5n.torchscript for TorchScript inference...
image 1/1 /user1/server/image/input.jpg: 640x640 2 persons, 2 chairs, 1 tv, 30.1ms
Speed: 3.6ms pre-process, 30.1ms inference, 5.8ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp11
```

Figura 20: Risultati yolov5n

Utilizzando Yolov5n.pt in formato float32 il frame viene ridimensionato in 544x640, rilevando 2 persone, 2 sedie e 2 televisori, come mostrato in Figura 21, con un tempo di 31,6ms.

Yolov5n.torchscript in int8 il frame viene ridimensionato in 640x640, rilevando 2 persone, 2 sedie e 2 televisori in 30,5ms.

Yolov5n.torchscript in half il frame viene ridimensionato in 640x640, rilevando 2 persone, 2 sedie e 2 televisori in 30,1ms.

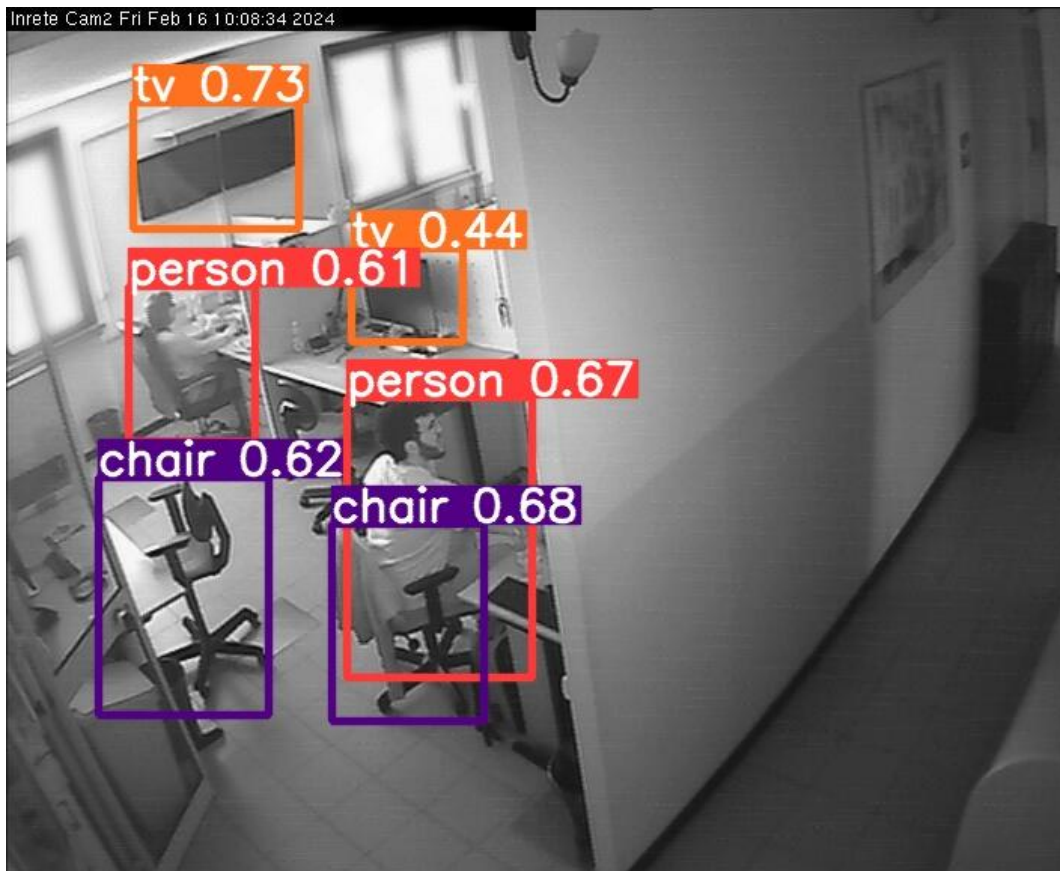


Figura 21: Frame - Item detection - Jetson AGX Orin

Utilizzando il frate di testi precedente e il modello di partenza Yolov5s sono stati ottenuti i seguenti risultati mostrati in Figura 22:

```
root@bd7a11bcf7d0:/user1/yolov5# python detect.py --source=/user1/server/image/input.jpg --device=0 --weights=yolov5s.pt
detect: weights=['/user1/yolov5s.pt'], source=/user1/server/image/input.jpg, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000
es=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False
WARNING ⚠ torchvision==0.14 is incompatible with torch==2.0.
Run 'pip install torchvision==0.15' to fix torchvision or 'pip install -U torch torchvision' to update both.
For a full compatibility table see https://github.com/pytorch/vision#installation
YOLOv5 🚀 v7.0-267-gc42aba5 Python-3.8.10 torch-2.0.0a0+ec3941ad.nv23.02 CUDA:0 (Orin, 62801MiB)

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients
image 1/1 /user1/server/image/input.jpg: 544x640 2 persons, 3 chairs, 2 tvs, 1 laptop, 33.9ms
Speed: 2.3ms pre-process, 33.9ms inference, 4.8ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp12
root@bd7a11bcf7d0:/user1/yolov5# python detect.py --source=/user1/server/image/input.jpg --device=0 --weights=/user1/yolov5/int8s/yolov5s.torchscript
detect: weights=['/user1/yolov5/int8s/yolov5s.torchscript'], source=/user1/server/image/input.jpg, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.25,
op=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=exp, exist_ok=False, line_t
WARNING ⚠ torchvision==0.14 is incompatible with torch==2.0.
Run 'pip install torchvision==0.15' to fix torchvision or 'pip install -U torch torchvision' to update both.
For a full compatibility table see https://github.com/pytorch/vision#installation
YOLOv5 🚀 v7.0-267-gc42aba5 Python-3.8.10 torch-2.0.0a0+ec3941ad.nv23.02 CUDA:0 (Orin, 62801MiB)

Loading /user1/yolov5/int8s/yolov5s.torchscript for TorchScript inference...
image 1/1 /user1/server/image/input.jpg: 640x640 2 persons, 3 chairs, 1 tv, 49.8ms
Speed: 3.6ms pre-process, 49.8ms inference, 6.1ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp13
root@bd7a11bcf7d0:/user1/yolov5# python detect.py --source=/user1/server/image/input.jpg --device=0 --weights=/user1/yolov5/halfs/yolov5s.torchscript
detect: weights=['/user1/yolov5/halfs/yolov5s.torchscript'], source=/user1/server/image/input.jpg, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.25,
op=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=exp, exist_ok=False, line_t
WARNING ⚠ torchvision==0.14 is incompatible with torch==2.0.
Run 'pip install torchvision==0.15' to fix torchvision or 'pip install -U torch torchvision' to update both.
For a full compatibility table see https://github.com/pytorch/vision#installation
YOLOv5 🚀 v7.0-267-gc42aba5 Python-3.8.10 torch-2.0.0a0+ec3941ad.nv23.02 CUDA:0 (Orin, 62801MiB)

Loading /user1/yolov5/halfs/yolov5s.torchscript for TorchScript inference...
image 1/1 /user1/server/image/input.jpg: 640x640 2 persons, 3 chairs, 1 tv, 49.4ms
Speed: 3.1ms pre-process, 49.4ms inference, 6.1ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp14
```

Figura 22: Risultati yolov5s

Analizzando il modello `yolov5s.pt` in `float32` il frame viene ridimensionato in 544x640, rilevando 2 persone, 3 sedie e 2 televisori e 1 notebook in 33,9ms.

`Yolov5s.torchscript` in `int8` il frame viene ridimensionato in 640x640, rilevando 2 persone, 3 sedie e 1 televisore in 49,8ms.

`Yolov5s.torchscript` in `half` il frame viene ridimensionato in 640x640, rilevando 2 persone, 3 sedie e 1 televisore in 49,4ms.

Il modello `YOLOv5s.pt` in `float32` sembra mostrare una migliore precisione di rilevamento, identificando un notebook in più rispetto agli altri modelli. Tuttavia, i modelli `YOLOv5n.torchscript` in formato `int8` e `half` mostrano tempi di rilevamento più lunghi rispetto al modello `YOLOv5s.pt` in `float32`. Questa differenza potrebbe essere attribuita alla conversione del modello in formati a precisione inferiore (`int8` e `half`), che, pur riducendo la precisione, comportano una maggiore efficienza computazionale.

5.1.1 Risultati server Jetson AGX Orin

Una volta analizzati i tempi di partenza, si è scelto di semplificare un po' il codice dell'inferenza della Jetson AGX Orin e successivamente di verificare la sua efficienza, nel caso specifico utilizzando il modello `Yolov5s.pt`.

Prendiamo i tempi di esecuzione dell'inferenza, dal primo all'ultimo, per il numero di inferenze come in Figura 23. Va sottolineato che l'immagine utilizzata è la stessa per ogni inferenza, come da test.

```
inferenze = 1000

start_time = time.time()
for _ in range(inferenze):
    infer_image(global_model, source=input_path, device='cuda')

end_time = time.time()

# Calcolo inferenza al secondo
total_time = end_time - start_time
inferenze_per_secondo = inferenze / total_time
```

Figura 23: Inferenze al secondo Jetson AGX Orin

Il risultato ottenuto è di 31.19 inferenze al secondo, vedi Figura 24, che è un tempo discreto per una macchina che sul mercato propone tempi come 100 inferenze al secondo e costa più di 3500 euro.

```
root@bd7a11bcf7d0:/user1/server# ls
antonio image models server_th.py start_server_yolo.sh test.py
root@bd7a11bcf7d0:/user1/server# ./test.py
YOLOv5 🚀 v7.0-267-gc42aba5 Python-3.8.10 torch-2.0.0a0+ec3941ad.nv23.02 CUDA:0 (Orin, 62801MiB)

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients
Number of inferences: 1000
Total time: 32.0661 seconds
Inferences per second: 31.19
root@bd7a11bcf7d0:/user1/server# |
```

Figura 24: Risultato Inferenze al secondo Jetson AGX Orin

5.2 Risultati Radxa Rock 5 B

Per la raccolta dati su Radxa Rock 5 B è stato utilizzato il frame di test di Figura 19 e il modello “yolov5s-640-640.rknn”, ottenendo risultati poco certi a volte quasi nulli per i frame catturati dalle videocamere, in quando il modello fornito era poco dettagliato.

In termini di millisecondi la Radxa Rock 5 B eseguite un’inferenza alla velocità di 28 ms con dei risultati delle volte poco accettabili. In Figura 25 possiamo vedere l’inferenza svolta sulla Radxa Rock 5 B.



Figura 25: Frame - Item detection – Radxa Rock 5 B

5.2.1 Risultati server Radxa Rock 5 B

```
command = "/user1/rknn_yolov5_demo_linux/run_yolo.sh"
inferenze = 1000

start_time = time.time()
for _ in range(inferenze):
    subprocess.run(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
end_time = time.time()

# Calcolo inferenza al secondo
total_time = end_time - start_time
inferenze_per_secondo = inferenze / total_time
```

Figura 26: Inferenze al secondo Radxa RK3588 B

Sulla Radxa Rock 5 B sono stati effettuati test simili calcolando il tempo di esecuzione delle inferenze come in Figura 26, ma non del tutto uguali in quanto non si è arrivati a trovare un sorgente da modificare, ma partendo dal modello di default ovvero: “yolov5s-640-640.rknn” e utilizzando il comando demo, abbiamo ottenuto il risultato di 1.20 inferenze al secondo, vedi Figura 27, che per un NPU e una macchina che costa più di 300 euro sono molto poche.

```

root@8b980f4d3ab7:/user1# ls
librerie  rknn_yolov5_demo_linux  rknn_
root@8b980f4d3ab7:/user1# ./test.py
Numeber of inferences: 10
Total time: 8.3152 seconds
Inferences per second: 1.20
root@8b980f4d3ab7:/user1# |

```

Figura 27: Risultato Inferenze al secondo Radxa RK3588 B

5.3 Risultati in termini di consumo

Con l'ausilio dello ShellyPro2PM_B, siamo stati in grado di monitorare il consumo energetico delle due macchine, Jetson AGX Orin e Radxa Rock 5 B, sia in modalità di standby che durante il normale funzionamento.

Dalla voce “Active Power” possiamo notare che durante la modalità “idle”, Figura 28 A, le due macchine mantengono una potenza utilizzata nell’ordine dei 7 W e dei 2 W, mentre in esecuzione o “run”,Figura 28 B, Jetson AGX Orin e Radxa Rock 5 B consumano esattamente il doppio della potenza, questo influisce sicuramente sulla scelta progettuale e sui costi.

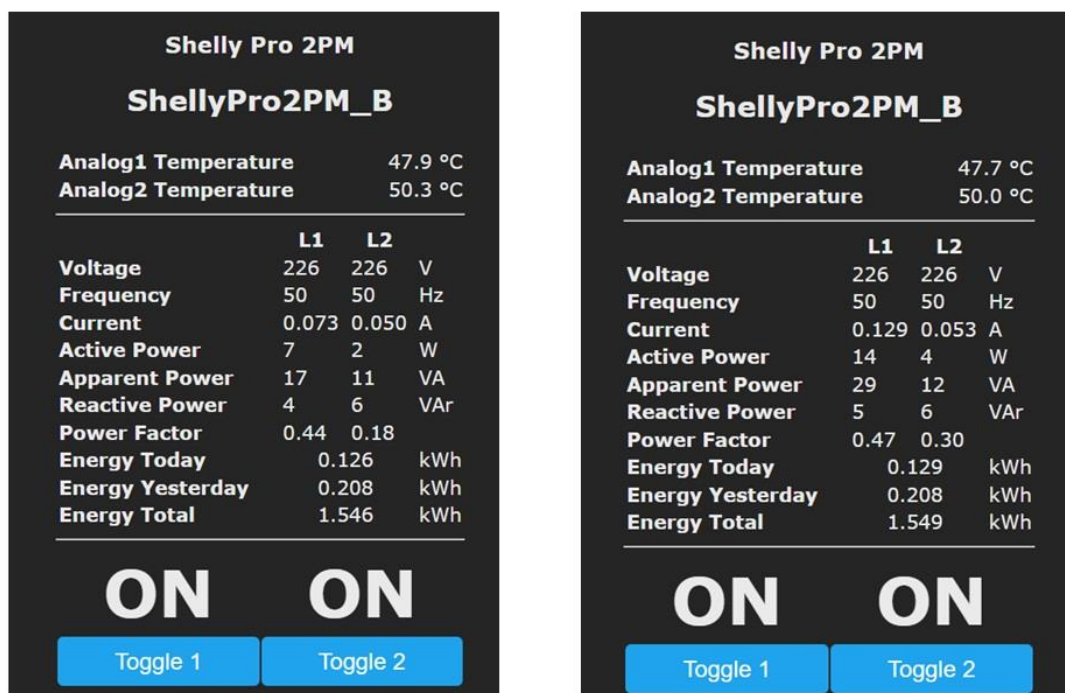


Figura 28 A : Stato “idle”, Figura 28 B : Stato “run”

6. Conclusioni e sviluppi futuri

Alla luce dei risultati ottenuti durante la sperimentazione, emerge la sfida nel proporre una soluzione definitiva e ottimale per affrontare il problema oggetto dello studio. Tale complessità è principalmente dovuta alla mancanza di documentazione adeguata e sviluppo dei sistemi utilizzati, i quali, pur essendo presenti sul mercato da diverso tempo, non soddisfano completamente le aspettative in termini di prestazioni e consumo. La ricerca di documentazione ha evidenziato informazioni spesso poco chiare e provenienti da fonti non ufficiali, con canali ufficiali privi di una soluzione specifica per i problemi e la corretta configurazione dei dispositivi. Ad esempio, nonostante Nvidia pubblici benchmark ufficiali per le sue schede Jetson AGX Orin, i risultati della sperimentazione non sempre rispecchiano tali dati. Solo verso la conclusione dello stage è emersa una documentazione leggermente più chiara, menzionando l'implementazione di un acceleratore chiamato DLA (Deep Learning Accelerator) per accelerare i carichi di lavoro di deep learning sulla GPU [41]. Anche Rockchip, per la scheda Radxa RK3588 B, offre documentazione scarsa e spesso in cinese, complicando l'accesso alle informazioni necessarie. Nonostante Nvidia abbia dedicato 12 anni allo sviluppo della Jetson AGX Orin e Rockchip un periodo simile per la Radxa RK3588 B, sembra che non si sia ancora giunti a ottenere un prodotto facilmente utilizzabile e ricco di risorse per una vasta gamma di funzioni.

Tuttavia, è importante sottolineare gli indiscutibili vantaggi dei Cobot. La collaborazione sicura con gli operatori rappresenta un elemento cruciale, in quanto i Cobot, grazie a sensori avanzati e tecnologie di sicurezza integrate, possono operare fianco a fianco con gli esseri umani senza rischi di incidenti, aprendo nuove prospettive di collaborazione nella produzione e in altri settori. L'attuale sistema implementato, nonostante presenti alcune limitazioni, offre un passo significativo verso l'automazione delle operazioni, con i Cobot che svolgono un ruolo chiave in questo contesto. La loro presenza consente di migliorare l'efficienza e la produttività, aprendo nuove opportunità di adattamento in diversi contesti lavorativi.

Rispetto ai consumi, è essenziale valutare attentamente l'impatto energetico del sistema attuale, con l'obiettivo di ottimizzare l'efficienza e minimizzare i consumi. Nonostante tali considerazioni, l'integrazione di Cobot rimane un passo positivo verso una maggiore automazione e miglioramento complessivo delle performance dei sistemi.

La sperimentazione ha affrontato con successo le sfide legate alla documentazione e allo sviluppo dei sistemi, mettendo in luce il potenziale trasformativo dell'integrazione dei Cobot nella gestione dei parametri termici di una sala macchine. L'implementazione dell'intelligenza sull'“Edge”, con particolare riferimento all'acceleratore DLA sulla GPU, rappresenta una pietra miliare significativa per migliorare le prestazioni di deep learning in modo localizzato. Nonostante le difficoltà incontrate con la documentazione delle schede Nvidia Jetson AGX Orin e Radxa RK3588 B di Rockchip, l'adozione dell'intelligenza sull'“Edge” si configura come una soluzione strategica per ottimizzare l'efficienza energetica. Questo approccio consente una gestione più efficiente dei parametri termici direttamente sul luogo, riducendo la necessità di trasferire grandi quantità di dati e così il consumo energetico complessivo.

Questa attività può essere estesa a tutti i locali dotati di un sistema di videosorveglianza, compresi uffici pubblici e privati, al fine di gestire autonomamente i parametri termici. Un possibile sviluppo futuro coinvolgerebbe l'evoluzione del sistema di videosorveglianza tramite l'introduzione di videocamere termiche. Questa innovazione consentirebbe non solo il rilevamento degli operatori, ma anche la gestione integrata dei parametri tecnici attraverso un'unica infrastruttura avanzata.

Inoltre, si potrebbe considerare l'utilizzo delle condizioni climatiche esterne per ottimizzare la temperatura interna, soprattutto in ambienti critici come le sale server. Questa ottimizzazione potrebbe essere implementata attraverso un sistema intelligente che bilancia in modo dinamico la temperatura interna, attingendo direttamente dalle risorse climatiche esterne, rilevate con l'ausilio di videocamere termiche esterne. Ciò contribuirebbe in modo significativo a migliorare l'efficienza complessiva del controllo termico, garantendo una gestione più precisa e responsiva dei parametri ambientali.

7. Bibliografia

- [1] Inrete - Internet to support your business, <https://www.inrete.it/>
- [2] Cobot, <https://it.wikipedia.org/wiki/Cobot>
- [3] EN ISO 10218-1: Requisiti di Sicurezza per Robot Industriali, <https://www.gt-engineering.it/normative-tecniche/normative-en-iso/en-iso-10218-1/>, 14 Luglio 2023
- [4] Object Detection Technology – How It Works And Where Is It Used?, <https://www.datasciencecentral.com/object-detection-technology-how-it-works-and-where-is-it-used/>, 14 Ottobre 2021
- [5] Cosa sono le reti neurali?, <https://www.ibm.com/it-it/topics/neural-networks>
- [6] Rete neurale convoluzionale, <https://www.domsoria.com/2019/10/come-funziona-una-rete-neurale-cnn-convolutional-neural-network/>
- [7] Introduzione a YOLO: rilevamento di oggetti in tempo reale, <https://hashdork.com/it/Yolo/>, 6 Maggio 2023
- [8] Ultralytics YOLOv5 Architettura, https://docs.ultralytics.com/it/yolov5/tutorials/architecture_description
- [9] Scripting della shell Linux e Crontab, <https://medium.com/@pradipkv247/linux-shell-and-shell-scripting-cron-and-crontab-bb591464f688>, 9 Giugno 2023
- [10] Python, <https://it.wikipedia.org/wiki/Python>
- [11] HTTP servers, <https://docs.python.org/3/library/http.server.html>
- [12] Pytorch, <https://github.com/pytorch/pytorch>, 26 Febbraio 2024
- [13] TORCH.TENSOR, <https://pytorch.org/docs/stable/tensors.html>
- [14] TORCH.CUDA, <https://pytorch.org/docs/stable/cuda.html>
- [15] Subprocess management, <https://docs.python.org/3/library/subprocess.html>
- [16] YOLO Object Detection with OpenCV and Python, <https://towardsdatascience.com/yolo-object-detection-with-opencv-and-python-21e50ac599e9>, 22 Agosto 2018
- [17] OpenCV, <https://it.wikipedia.org/wiki/OpenCV>
- [18] Node-RED, <https://nodered.org/>
- [19] FlowFuse – Exec, <https://flowfuse.com/node-red/core-nodes/exec/>
- [20] Node-RED, <https://flows.nodered.org/flow/5bca514f463320c82ad7aecce6457193>
- [21] Node-Red HTTP Request Node for Beginners, <https://stevesnoderedguide.com/node-red-http-request-node-beginners>, 24 Ottobre 2023
- [22] Docker overview, <https://docs.docker.com/get-started/overview/>

- [23] Container and virtualization tools, <https://linuxcontainers.org/>
- [24] LXC, LXD, Container, Docker, KVM, VM, Hypervisor, <https://medium.com/@nikhil24may/lxc-lxd-container-docker-kvm-vm-hypervisor-eae6cd2ec515>, 3 Settembre 2020
- [25] Docker (base command), <https://docs.docker.com/reference/cli/docker/>
- [26] JavaScript Object Notation, https://it.wikipedia.org/wiki/JavaScript_Object_Notation
- [27] An Unprecedented Edge AI and Robotics Platform, <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/>
- [28] Nvidia boosts safety critical design with Jetson Orin AGX module, <https://www.eenewseurope.com/en/nvidia-boosts-safety-critical-design-with-jetson-orin-agx-module/>, 21 Novembre 2021
- [29] CUDA Toolkit, <https://developer.nvidia.com/cuda-toolkit>
- [30] ROCK 5 Model B, <https://wiki.radxa.com/Rock5/5B>
- [31] Adding bootloader support for USB 2.0 Host for Radxa ROCK 5B RK3588, <https://www.collabora.com/news-and-blog/blog/2023/04/27/radxa-rock-5b-usb20-host-in-u-boot/>, 27 Aprile 2023
- [32] ROCK 5B, <https://wiki.radxa.com/Rock5/hardware/5b>
- [33] PC NUC, cos'è e a chi serve, <https://www.fastweb.it/fastweb-plus/digital-magazine/pc-nuc-quali-sono/#:~:text=NUC%20%C3%A8%20un%20marchio%20registrato,con%20alcune%20componenti%20non%20includere>
- [34] Shelly ITALIA, https://www.shellyitalia.com/shelly-pro-2pm/?gad_source=1&gclid=CjwKCAiAivGuBhBEEiwAWiFmYbPskBPN5h7Mg66can9W1kCpxUn0sHxJ5gc9sjxL3kElvsoo3iM7xRoCZKQQAvD_BwE
- [35] Yolov5, <https://github.com/ultralytics/yolov5>, 25 Febbraio 2024
- [36] Getting Started with Jetson AGX Orin Developer Kit, <https://developer.nvidia.com/embedded/learn/get-started-jetson-agx-orin-devkit>
- [37] NVIDIA L4T PyTorch, <https://catalog.ngc.nvidia.com/orgs/nvidia/containers/l4t-pytorch>, 4 Febbraio 2024
- [38] Compiling OpenCV from Source, https://developer.ridgerun.com/wiki/index.php/Compiling_OpenCV_from_Source
- [39] Run RKNN2 Demo on ROCK 5B, <https://forum.radxa.com/t/run-rknn2-demo-on-rock-5b/10914>

[40] RK3588_Linux, https://gitlab.com/rk3588_linux/linux/debian/-/tree/master/packages/arm64/rga2?ref_type=heads

[41] Deep learning Accellerator (DLA), <https://developer.nvidia.com/deep-learning-accelerator>

8. Ringraziamenti

Con il cuore colmo di gratitudine, desidero porgere i miei più sentiti ringraziamenti a tutti coloro che hanno contribuito al raggiungimento di questo traguardo.

Al Professor Marco Guazzone, la mia profonda riconoscenza per la sua costante disponibilità, la sua preziosa guida e i suoi puntuali chiarimenti. Il suo supporto è stato fondamentale per accompagnarmi al termine di questo percorso universitario, costellato di sfide e tempistiche ristrette.

Al Dottor Giorgio Bernardi, esprimo la mia sincera gratitudine per la fiducia riposta in me e per l'imperdibile opportunità di crescita professionale che mi ha offerto presso l'azienda INRETE S.R.L. Ringrazio inoltre il Dottor Gigi, Alessandro, Luca, Marco, Riccardo, Eugenio, Stefano, Barbara e Mauro per la loro costante collaborazione, il loro sostegno e i piacevoli momenti condivisi.

Ai cari amici di Cocconato e dintorni, un caloroso abbraccio per avermi accolto a braccia aperte fin dal mio arrivo in Piemonte. Il loro affetto silenzioso e la loro contagiosa allegria hanno allietato le mie giornate e alleviato lo stress quotidiano.

Ai ragazzi della "Iettiti A Mari" e a Michele, la mia più sincera riconoscenza per la loro incrollabile amicizia. La loro presenza costante e il loro supporto incondizionato hanno rappresentato una parte fondamentale della mia crescita personale.

A tutti i miei compagni dell'università, che si sono rivelati amici preziosi e sempre disponibili, anche al di fuori dell'ambiente scolastico. Il loro sostegno e la loro complicità hanno reso questo percorso ancora più speciale.

Alla mia famiglia, il pilastro su cui ho sempre potuto contare. Ai miei nonni: Antonino, Matilde, Pietro e Santina, per i sani principi che mi hanno trasmesso e per la loro incrollabile fiducia in me. Ai miei genitori, che con infinita pazienza e amore incondizionato mi hanno insegnato a diventare la persona che sono oggi, incoraggiandomi a pensare con la mia testa e a prendere decisioni sempre più importanti.

A mio fratello Pietro, con cui ho condiviso innumerevoli momenti in palestra, che hanno alimentato la mia passione e mi hanno dato la forza di superare ogni ostacolo e raggiungere questo obiettivo.

Al mio migliore amico Giorgio, un compagno di avventure rocambolesche, il mio più grande sostegno e la mia spalla su cui piangere. La sua costante presenza, la sua fiducia in me e la sua onestà hanno reso questo percorso e tutta la mia vita un viaggio indimenticabile.

Infine, a Cecilia, la donna che ha dato un nuovo senso alle mie giornate. Il tempo trascorso con lei all'università volava via, e ogni suo piccolo gesto alimentava il mio amore per lei. Insieme abbiamo imparato ad amarci, ad amare e ad esprimere la nostra individualità.

Grazie a tutti voi per aver reso questo traguardo un successo condiviso. Il vostro sostegno è stato il mio faro nella tempesta e la vostra fiducia la mia forza motrice. Vi porterò sempre nel mio cuore.