
Ph.D. in Elettronica Applicata

XXXVII Ciclo

*Ph.D. Tesi*

---

# Progettazione e sviluppo di algoritmi e tecniche di machine learning e deep learning per l'identificazione e classificazione di obiettivi spaziali attraverso l'elaborazione di segnali radar

---

Candidata: Federica Massimi

Relatore: prof. Francesco Benedetto

Co-Relatore: ing. Pasquale Ferrara

# Contents

# List of Figures

# List of Tables

# Introduzione

Negli ultimi anni, l'evoluzione dell'Intelligenza Artificiale ha trasformato radicalmente il modo in cui affrontiamo problemi complessi. La crescente capacità dei sistemi intelligenti di elaborare grandi quantità di dati, di adattarsi a contesti incerti e di operare in tempo reale ha aperto nuove frontiere in molteplici settori applicativi, in particolare quello del settore Spaziale.

Questo progetto di dottorato nasce con l'obiettivo di contribuire alla protezione di infrastrutture strategiche, alla comprensione dell'ambiente spaziale e all'identificazione precoce di minacce, sviluppando nuove tecniche basate su Machine Learning e Deep Learning per l'analisi dei segnali radar. L'intento principale è stato quello di indagare la possibilità di identificare e classificare in modo automatico i bersagli spaziali, sfruttando le potenzialità dell'IA per superare i limiti dei metodi radaristici tradizionali.

L'approccio seguito nel lavoro ha integrato tecniche classiche, come il rilevamento CFAR (Constant False Alarm Rate), con modelli neurali avanzati: tra questi, architetture YOLO, reti dotate di meccanismi di self-attention e tecniche di regressione e classificazione multi-head, in grado di stimare simultaneamente parametri fisici del target. Le soluzioni proposte sono state validate tramite simulazioni e dataset appositamente generati, permettendo un confronto dettagliato con le metriche consolidate nella letteratura. Questo ha consentito di valutare l'efficacia, l'accuratezza e la generalizzabilità dei modelli in scenari operativi complessi.

Tuttavia, il dominio spaziale non ha rappresentato l'unico ambito di indagine della ricerca. L'interesse per l'Intelligenza Artificiale è stato esteso ad altri contesti di rilievo: il primo è stato quello della sicurezza informatica, in particolare con lo studio e la rilevazione di comunicazioni nascoste (covert channels) all'interno dei sistemi Internet of Medical Things (IoMT) e reti wireless; il secondo ha riguardato l'applicazione dell'IA al Ground Penetrating Radar (GPR) per il rilevamento non invasivo delle radici arboree, con l'obiettivo di dimostrare la versatilità degli approcci neurali anche in ambito geotecnico.

Il lavoro presentato in questa tesi si è svolto nell'ambito della Convenzione tra l'Università Roma Tre e Leonardo SpA, e cofinanziato dalla Regione Lazio, dal titolo "Space Situational Awareness, progettazione e sviluppo di algoritmi e tecniche di machine learning per l'identificazione e classificazione di target spaziali tramite elaborazione di segnali radar".

# 1

## 1.1 Motivazioni e contesto della ricerca

## Premessa

La crescente complessità dei sistemi tecnologici ha reso necessario lo sviluppo di strumenti avanzati per il monitoraggio, il controllo e la protezione. L'Intelligenza Artificiale (IA) si è dimostrata particolarmente adatta a questi compiti, grazie alla capacità di elaborare grandi quantità di dati e di adattarsi a scenari dinamici e incerti. In questo contesto, l'IA viene utilizzata per monitorare il comportamento di sistemi complessi come infrastrutture critiche, reti di comunicazione o piattaforme aerospaziali rilevando in tempo reale eventuali anomalie o segnali di malfunzionamento.

Non si tratta solo di osservare, ma anche di intervenire: l'IA consente di controllare le reazioni del sistema, adattandole alle condizioni operative per garantire stabilità ed efficienza. Parallelamente, svolge un ruolo centrale nella protezione dei sistemi stessi, intervenendo per prevenire attacchi, preservare l'integrità dei dati e difendere risorse fisiche e digitali da minacce esterne. In applicazioni civili e militari, tutto questo è fondamentale per garantire affidabilità e continuità operativa. Inizialmente concepita come strumento di automazione, l'IA si è evoluta fino a diventare una tecnologia chiave per la sicurezza e gestione dei sistemi complessi.

Questa ricerca esplora come l'IA possa contribuire concretamente al rilevamento di anomalie, alla classificazione automatica di eventi e comportamenti, e alla protezione attiva dei sistemi, riducendo il rischio di malfunzionamenti, attacchi o eventi imprevisti. L'IA si è evoluta in modo significativo, diventando essenziale non solo per l'automazione, ma anche per la gestione di flussi di dati complessi.

Gli investimenti nell'IA, in continuo aumento negli ultimi anni, riflettono l'importanza crescente di questa tecnologia. Come mostrato in Figura 1.1, il mercato globale dell'intelligenza artificiale sta vivendo una fase di espansione senza precedenti. Le stime più recenti indicano che il valore complessivo del settore, pari a circa 638,23 miliardi di dollari nel 2024, raggiungerà i 757,58 miliardi già nel 2025, per poi proseguire lungo una traiettoria di crescita esponenziale fino a toccare i 3.680,47 miliardi di dollari entro il 2034.

Questo andamento riflette un cambiamento profondo nella centralità strategica dell'IA all'interno dei processi produttivi, decisionali e infrastrutturali delle società contemporanee.

Nel 2024, il Nord America ha generato oltre il 36,92% della quota di mercato, confermandosi come l'area più sviluppata nell'uso dell'Intelligenza Artificiale, grazie alla presenza di grandi aziende tecnologiche e ottime infrastrutture. Allo stesso tempo, si prevede che il mercato dell'area Asia-Pacifico crescerà al ritmo più veloce, con un tasso annuo del 19,8% tra il 2025 e il 2034, spinto dagli investimenti di Paesi come Cina, Corea del Sud e Giappone.

Tra le tecnologie, il deep learning ha conquistato una quota di mercato del 37,4% nel 2024. Per quanto riguarda le soluzioni adottate dalle aziende, il settore dei servizi ha rappresentato una quota di mercato superiore al 39,2%, segno che molte imprese preferiscono affidarsi a esperti esterni piuttosto che sviluppare l'IA internamente. Infine, guardando agli utenti finali, il settore BFSI (banche, servizi finanziari e assicurazioni) ha rappresentato il 17,4% della quota di mercato nel 2024, confermando quanto l'IA sia utile per gestire operazioni, rischi e sicurezza in ambito economico.

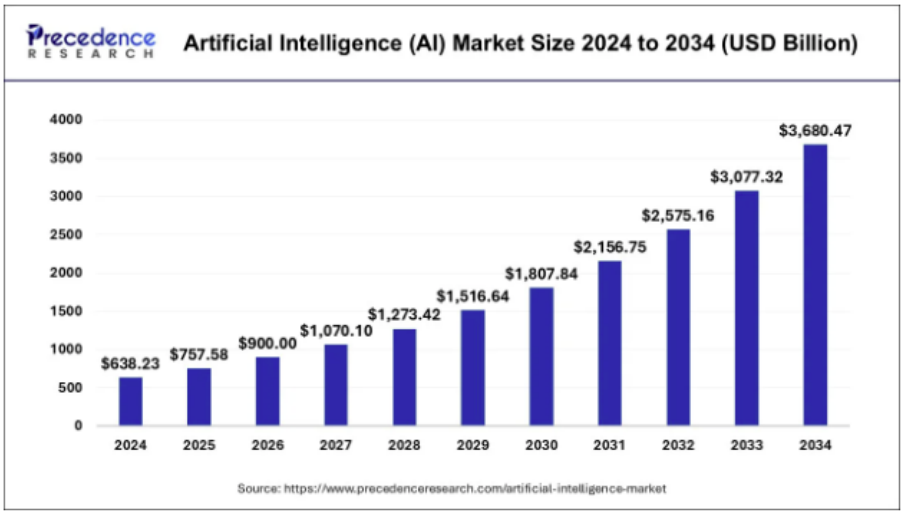



**Figure 1.1** Crescita prevista del mercato globale dell'intelligenza artificiale tra il 2024 e il 2034 [1]

Per comprendere pienamente le dinamiche che guidano la crescita complessiva del mercato dell'intelligenza artificiale , è necessario analizzare la composizione interna di tale

mercato, ossia quali tecnologie contribuiscono maggiormente al suo sviluppo. Il grafico 1.2 mostra l'evoluzione, nel periodo compreso tra il 2020 e il 2030, delle principali componenti tecnologiche che costituiscono l'ecosistema dell'IA.

Tra tutte le tecnologie rappresentate, il Machine Learning si conferma come la colonna portante del settore, detenendo la quota maggiore per l'intero decennio. Le tecniche di apprendimento automatico costituiscono il fondamento su cui si basano molte delle applicazioni IA oggi in uso. La loro capacità di adattarsi ai dati e di migliorare le prestazioni nel tempo li rende strumenti particolarmente versatili e strategici in un'ampia gamma di contesti applicativi (alcuni dei quali riportati in questo lavoro).

Un altro settore in costante e significativa crescita è quello del Natural Language Processing (NLP). L'interesse verso questa tecnologia è cresciuto esponenzialmente con l'affermazione di assistenti virtuali, chatbot, motori di ricerca semantici e traduttori automatici. L'evoluzione degli LLM (Large Language Models), come i transformer di nuova generazione, ha ulteriormente ampliato le possibilità del NLP, portando la comprensione e la generazione del linguaggio naturale a livelli fino a pochi anni fa impensabili. In prospettiva, il NLP rappresenta uno dei pilastri principali dell'interazione naturale uomo-macchina e della trasformazione digitale in settori come sanità, giustizia, educazione e servizi pubblici. Seguono, per dimensione di mercato, la Computer Vision, che permette ai sistemi di elaborare e comprendere immagini e video, e che trova applicazioni in ambiti quali la sorveglianza, il riconoscimento facciale, la diagnostica medica per immagini e la robotica. La Robotics AI e le tecnologie autonome basate su sensori (Autonomous & Sensor Technology) completano il quadro, con una crescita più graduale ma significativa, alimentata soprattutto dalla diffusione dell'automazione industriale, dei veicoli autonomi e della sensoristica intelligente in ambito IoT (Internet of Things).

## 1.2 Obiettivi della tesi

L'obiettivo principale di questa tesi è stato lo studio, lo sviluppo e la validazione di metodologie basate sull'Intelligenza Artificiale, con particolare attenzione all'ambito aereospaziale e dei detriti spaziali, la cui crescente presenza rappresenta oggi una seria minaccia per la sicurezza delle missioni spaziali. Questi oggetti, spesso di dimensioni ridotte ma ad altissima velocità, possono causare collisioni con satelliti operativi, compromettendo infrastrutture fondamentali. Questa tesi si propone dunque di sviluppare metodi basati su intelligenza artificiale per il rilevamento, la classificazione e il tracciamento di detriti spaziali, confrontando l'efficacia rispetto alle tecniche radar tradizionali.

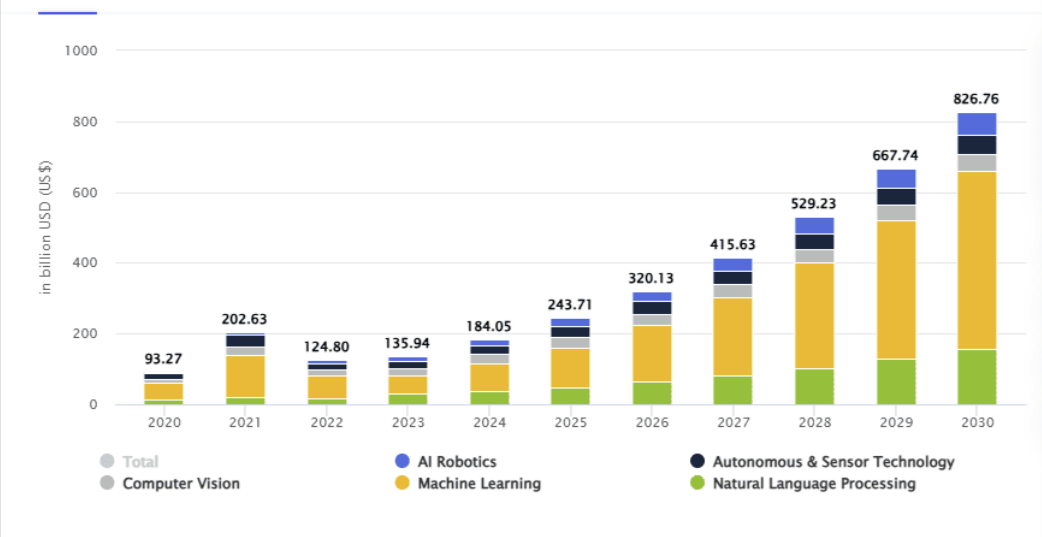



**Figure 1.2** Suddivisione del mercato globale dell'intelligenza artificiale [2]

Nel corso dei tre anni di ricerca, le tecniche di Intelligenza Artificiale, con particolare riferimento al Machine Learning (ML) e al Deep Learning (DL), sono state inizialmente applicate in due ambiti distinti con l'obiettivo di valutarne le prestazioni e ottimizzarne l'efficacia. Il primo ambito ha riguardato i sistemi CTC (Covert Timing Channel), con lo scopo di individuare potenziali minacce alla sicurezza delle comunicazioni attraverso l'analisi di canali nascosti. Il secondo ha invece coinvolto l'impiego del radar GPR (Ground Penetrating Radar) per l'analisi non invasiva delle radici degli alberi, integrando soluzioni basate su IA per migliorarne l'accuratezza e la robustezza.

Le competenze sviluppate in questi contesti hanno poi costituito la base metodologica per affrontare lo scenario dei detriti spaziali.

Le soluzioni proposte sono state confrontate con approcci radaristici tradizionali e analisi classiche dei dati radar, al fine di valutarne l'efficacia e l'affidabilità in scenari sia simulati che reali.

## 1.3 Struttura della tesi

La tesi è articolata in più capitoli che riflettono la progressiva evoluzione della ricerca.

- Il Capitolo 2 introduce i concetti fondamentali dell'Intelligenza Artificiale e del Deep Learning, presentando una panoramica delle principali reti neurali artificiali (2.1) e delle architetture avanzate attualmente più diffuse, tra cui CNN, YOLO, Transformers e GAN (2.2). Vengono inoltre discusse le principali tecniche di addestramento,

validazione e ottimizzazione dei modelli (2.3). Una parte significativa del capitolo è dedicata alle applicazioni pratiche degli algoritmi di Machine Learning e Deep Learning nell'analisi delle immagini (2.4), con particolare attenzione agli impieghi nei sistemi Georadar (2.4.1) e nella sicurezza dei sistemi complessi, come il CTC (2.4.2).

- Il Capitolo 3 è dedicato al tema della Space Situational Awareness (SSA) e al contributo dell'Intelligenza Artificiale nel monitoraggio dello spazio. Dopo una panoramica sul ruolo strategico dello spazio a livello economico e geopolitico (3.1), vengono esaminate le principali sfide nel tracciamento dei detriti spaziali. Il capitolo prosegue con una rassegna dello stato dell'arte nell'elaborazione dei segnali radar per il rilevamento degli oggetti orbitali (3.2) e delle tecniche di deep learning per la classificazione e il tracciamento di oggetti spaziali (3.3), evidenziando il crescente ruolo dell'IA in questo ambito. Infine, si presenta un'analisi del gap di conoscenza che la presente ricerca intende colmare (3.4).
- Il Capitolo 4 descrive nel dettaglio i metodi proposti. Dopo una panoramica sulla struttura generale del capitolo e l'implementazione dell'ambiente di test TIRA (4.1), si presenta inizialmente la tecnica CFAR (Constant False Alarm Rate), come approccio tradizionale al rilevamento (4.2). Successivamente, vengono introdotte le soluzioni innovative basate su Deep Learning sviluppate per il tracciamento dei detriti spaziali (4.3), tra cui una tecnica di regressione innovativa specificamente progettata per il contesto orbitale (4.4).
- Il Capitolo 5 riporta i risultati sperimentali ottenuti dalle simulazioni e dalle analisi. Vengono mostrati i risultati relativi al tracciamento degli oggetti spaziali tramite le tecniche proposte (5.1) e quelli riguardanti la prestazione della tecnica di regressione (5.2), evidenziando i vantaggi rispetto ai metodi tradizionali in termini di accuratezza, robustezza e capacità di generalizzazione.

## 1.4 Pubblicazioni in tre anni di dottorato

Durante i tre anni di attività di ricerca, il lavoro è stato costantemente condiviso e validato con la comunità scientifica internazionale attraverso pubblicazioni su riviste e conferenze di settore. Queste pubblicazioni hanno contribuito alla diffusione dei risultati e hanno permesso un confronto proficuo con altri ricercatori nel campo.

### 1.4.1 Pubblicazioni su rivista

- Massimi, F., Ferrara, P., Benedetto, F., *Deep Learning-Based Space Debris Detection for Space Situational Awareness: A Feasibility Study Applied to the Radar Processing*, *IET Radar, Sonar & Navigation*, vol. 18, n. 4, pp. 353–360, 2024. DOI: 10.1049/rsn2.12547.
- Cuomo, M., Massimi, F., *Detecting CTC Attack in IoMT Communications using Deep Learning Approach*, *Advances in Science Technology and Engineering Systems Journal*, vol. 8, n. 2, pp. 130–138, 2023. DOI: 10.25046/aj080215.
- Massimi, F., Ferrara, P., Benedetto, F., *Deep Learning Methods for Space Situational Awareness in Mega-Constellations Satellite-Based Internet of Things Networks*, *Sensors*, vol. 23, n. 1, art. 124, 2023. DOI: 10.3390/s23010124.

Le pubblicazioni su rivista si concentrano su aspetti metodologici e pratici dell'intelligenza artificiale applicata alla classificazione e rilevamento, nonché sull'analisi della sicurezza in contesti specifici come Space Situational Awarness o Covert Timing Channel.

### 1.4.2 Pubblicazioni in conferenza

- Massimi, F., Benedetto, F., *Security Analysis of Temporary Convolution Networks Based Side Channel Attacks*, *IEEE Open Journal of the Communications Society – IEEE Networking Letters section*, submitted, 2025.
- Massimi, F., Benedetto, F., *Performance Improvements of Covert Timing Channel Detection in the Era of Artificial Intelligence*, in: *Advances in Distributed Computing and Machine Learning*, *Lecture Notes in Networks and Systems*, vol. 764, pp. 399–410, 2024. DOI: 10.1007/978-981-97-1841-2_30.
- Massimi, F., Benedetto, F., *Artificial Intelligence-based Hidden Communications Detection in Wireless Networks*, *46th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 1–6, 2023. DOI: 10.1109/TSP59544.2023.10197831.
- Lantini, L., Massimi, F., Tosti, F., Alani, A. M., Benedetto, F., *Advancements in Using Deep Learning Methods for GPR Detection of Tree Roots*, *12th International Workshop on Advanced Ground Penetrating Radar (IWAGPR)*, pp. 1–6, 2023. DOI: 10.1109/IWAGPR57138.2023.10329164.

- Massimi, F., Benedetto, F., *A Deep Learning Approach for Tree Root Detection using GPR Spectrogram Imagery*, *45th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 391–394, 2022. DOI: 10.1109/TSP55681.2022.9851337.
- Massimi, F., Benedetto, F., *Deep Learning-based Detection Methods for Covert Communications in E-Health Transmissions*, *45th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 11–16, 2022. DOI: 10.1109/TSP55681.2022.9851366.

Le pubblicazioni in conferenza hanno permesso di presentare risultati preliminari, scambiare feedback con la comunità di riferimento e esplorare possibili estensioni e applicazioni delle soluzioni sviluppate.

# 2 Applicazioni dell'Intelligenza Artificiale nel Rilevamento, Classificazione e Sicurezza dei Sistemi Complessi

## 2.1 Reti Neurali Artificiali: Principi di Base

L'idea alla base delle **reti neurali artificiali** (Artificial Neural Networks, ANN) nasce dall'osservazione del funzionamento del cervello umano. Il cervello è composto da circa $10^{11}$ neuroni biologici, ciascuno dei quali è collegato a migliaia di altri tramite sinapsi in un fitto reticolo di connessioni che costituisce la base del comportamento intelligente . Quando un neurone riceve un segnale elettrico sufficiente attraverso queste connessioni, genera un impulso (detto *potenziale d'azione*) che viene trasmesso ai neuroni vicini innescando complesse catene di attivazione [3].

Questa struttura naturale ha ispirato fin dagli anni '40 tentativi di riprodurre computazionalmente il comportamento delle reti neuronali biologiche. Uno dei primi modelli teorici fu proposto da McCulloch e Pitts (1943), che descrissero un neurone in termini logico-matematici, mostrando come una rete di unità semplici potesse, in linea di principio, calcolare qualsiasi funzione computabile [4].

L'interesse per le reti neurali ebbe nuovi sviluppi con il lavoro di Rosenblatt (1958), che introdusse il *perceptron*, una delle prime implementazioni pratiche di un neurone artificiale. Sebbene inizialmente limitato, questo modello rappresentò un passo fondamentale nella storia dell'intelligenza artificiale. Solo decenni dopo, con l'avvento di maggiori potenze di calcolo e l'introduzione dell'algoritmo di backpropagation [5], si riuscì ad addestrare reti multilivello in modo efficace, inaugurando l'era del deep learning.

### 2.1.1 Dal neurone biologico al neurone artificiale

Il *neurone artificiale* è una funzione matematica che riceve in input un vettore $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, i cui elementi rappresentano le informazioni in ingresso. Ogni input è associato a un *peso sinaptico* $w_i$, e il neurone calcola una combinazione lineare:

$$z = \sum_{i=1}^{n} w_i x_i + b \tag{2.1}$$

dove $b$ è un termine di bias. Questo valore $z$ viene poi trasformato da una funzione di attivazione non lineare $\phi(z)$, come la funzione sigmoide, tangente iperbolica o ReLU:

$$a = \phi(z) \tag{2.2}$$

Questo semplice schema consente al neurone artificiale di apprendere relazioni complesse tra input e output, soprattutto quando è parte di una rete.

### 2.1.2 Architettura di una rete neurale

Una rete neurale artificiale è formata da un insieme di neuroni organizzati in strati (layers): uno strato di input, uno o più strati nascosti e uno strato di output come illustrato nella figura 2.1. Ogni neurone in uno strato è tipicamente connesso a tutti i neuroni dello strato successivo (architettura *fully connected*). Le reti più semplici sono dette *feedforward*, poiché il flusso dell'informazione procede in una sola direzione: dagli input agli output.

### 2.1.3 Diversi tipi di apprendimento

Le reti neurali possono essere addestrate attraverso diversi paradigmi di apprendimento, a seconda della natura dei dati disponibili e del tipo di problema da risolvere.

**1. Apprendimento supervisionato:** ogni esempio del dataset è accompagnato da un'etichetta (label) che rappresenta la risposta corretta. L'obiettivo della rete neurale è imparare una funzione che mappi correttamente gli input sugli output desiderati. È il paradigma più comune e viene impiegato in problemi come la classificazione delle immagini, la previsione di serie temporali o la diagnosi medica [7].

Esempi pratici:

- Riconoscimento facciale (label = identità della persona)
- Diagnosi di malattie da immagini mediche (label = presenza o assenza di patologia)

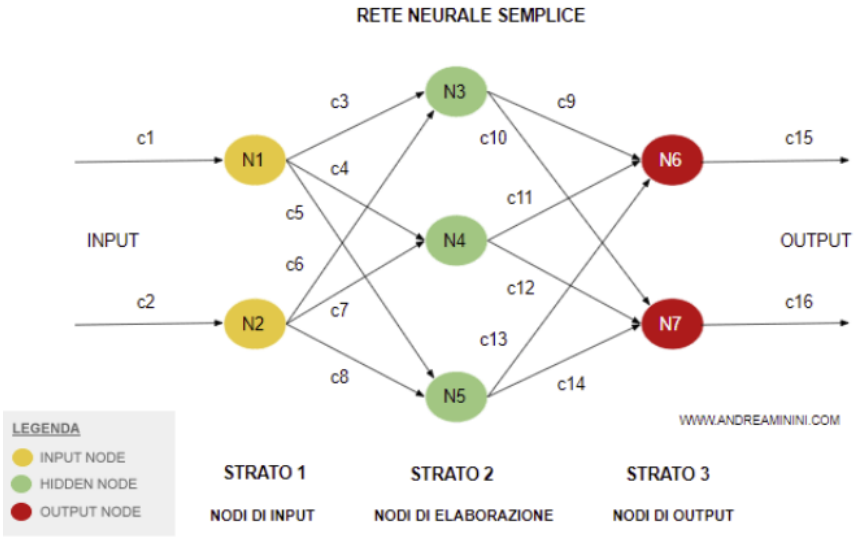



**Figure 2.1** Architettura rete neurale
[6]

**2. Apprendimento non supervisionato:** i dati non hanno etichette. L'obiettivo è scoprire strutture nascoste o rappresentazioni significative nei dati. Questo tipo di apprendimento è utile per compiti come il clustering, la riduzione dimensionale e il rilevamento di anomalie [8].

Esempi pratici:

- Raggruppamento automatico di clienti in base a comportamenti di acquisto
- Compressione di immagini tramite autoencoder

**3. Apprendimento semi-supervisionato:** si colloca tra i due estremi precedenti dove solo una parte dei dati ha etichette, mentre il resto è non etichettato. Questo approccio è particolarmente utile quando etichettare tutti i dati è costoso o impraticabile. Le reti neurali possono apprendere dai dati etichettati e al contempo sfruttare la struttura dei dati non etichettati [9].

Esempi pratici:

- Analisi di grandi set di documenti testuali dove solo alcuni sono stati classificati manualmente

**4. Apprendimento auto-supervisionato:** è un sottocaso dell'apprendimento non supervisionato, dove le etichette vengono generate automaticamente dai dati stessi. Questo

paradigma è diventato centrale nel campo del *self-supervised learning* in NLP e visione artificiale. Si sfruttano compiti ausiliari (pretext tasks), come la predizione della prossima parola o il riordinamento di frammenti di un'immagine, per apprendere rappresentazioni utili [10].

Esempi pratici:

- BERT (Bidirectional Encoder Representations from Transformers) addestra un modello a prevedere parole mancanti [11]
- SimCLR e MoCo per la visione artificiale [12, 13]

**5. Apprendimento per rinforzo (Reinforcement Learning):** l'agente (la rete neurale) interagisce con un ambiente e apprende una politica ottimale per massimizzare una ricompensa cumulativa. A differenza dell'apprendimento supervisionato, qui non ci sono etichette esplicite per ogni azione, ma solo un segnale di ricompensa. È ampiamente usato in robotica, giochi e controllo autonomo [14].

Esempi pratici:

- AlphaGo, il sistema che ha battuto i campioni umani nel gioco del Go [15]
- Sistemi di guida autonoma

### 2.1.4 Dal modello biologico al deep learning

L'evoluzione delle reti neurali artificiali ha seguito una traiettoria che, partendo da una semplice ispirazione al funzionamento dei neuroni biologici, ha condotto allo sviluppo di sofisticati modelli computazionali noti come reti di deep learning. Inizialmente, i modelli di neuroni artificiali erano in grado di simulare solo funzioni logiche elementari. Il perceptron di Rosenblatt rappresentò un passo avanti significativo, ma rimase limitato alla classificazione di dati linearmente separabili [16].

Il punto di svolta si è verificato negli anni '80 con l'introduzione dell'algoritmo di backpropagation, che ha permesso l'addestramento efficace di reti neurali multilivello . Da quel momento in poi, con l'aumento della potenza computazionale e la disponibilità di grandi quantità di dati, le reti neurali hanno subito un'accelerazione senza precedenti. È così nata l'era del deep learning, caratterizzata da modelli a molti strati in grado di apprendere rappresentazioni gerarchiche dei dati. Le reti profonde (deep networks), spesso composte da decine o centinaia di layer, sono in grado di catturare strutture complesse, non solo nei dati visivi ma anche in quelli testuali, audio e temporali.

Questa transizione dal modello ispirato biologicamente a sistemi altamente ingegnerizzati ha reso possibile affrontare problemi complessi di classificazione, regressione, segmentazione e previsione, aprendo la strada a nuove applicazioni in settori come la visione artificiale, il linguaggio naturale, la robotica e l'analisi dei segnali [17]. Sebbene le reti neurali profonde siano ancora lontane dal replicare pienamente l'intelligenza del cervello umano, esse rappresentano il tentativo attualmente più avanzato di modellare l'apprendimento in maniera automatica e scalabile.

## 2.2 Architetture avanzate: CNN, YOLO, Transformers, GAN

Nel contesto delle reti neurali artificiali, le architetture avanzate hanno rivoluzionato il deep learning, consentendo progressi significativi in vari ambiti, come la visione artificiale, il trattamento del linguaggio naturale e la generazione di contenuti. Tra le architetture più rilevanti spiccano le Convolutional Neural Networks (CNN), il modello YOLO (You Only Look Once), i Transformers e le Generative Adversarial Networks (GAN). Ognuna di queste ha trovato applicazioni in numerosi settori, spingendo i limiti delle capacità del deep learning.

### 2.2.1 CNN (Le Convolutional Neural Networks (CNN))

Le Convolutional Neural Networks (CNN) sono una classe di reti neurali profondamente radicate nell'elaborazione di immagini e segnali visivi. La loro struttura è pensata per trattare dati con una struttura spaziale definita, come immagini, video, ma anche segnali radar o spettrogrammi. A differenza delle reti neurali dense (fully connected), le CNN sfruttano il concetto di convoluzione, cioè un'operazione matematica che permette di rilevare schemi locali all'interno di un'immagine, come bordi, texture o forme semplici.

Il cuore di una CNN è infatti costituito dagli strati convoluzionali, che applicano una serie di filtri (o kernel) su piccole porzioni dell'immagine. Questi filtri vengono appresi durante il processo di training e sono in grado di catturare caratteristiche visive importanti. Nei primi strati, si imparano pattern molto semplici come linee orizzontali o verticali, mentre nei livelli più profondi si arriva a riconoscere strutture complesse, come occhi, volti, veicoli o anomalie specifiche nei dati [18].

A valle di questi strati si trovano in genere operazioni di pooling, in particolare max pooling, che riducono la dimensione delle feature map mantenendo le informazioni più salienti. Questo consente sia una maggiore efficienza computazionale sia una maggiore robustezza del modello a piccole traslazioni o distorsioni dell'immagine.

I modelli storici più noti che hanno fatto scuola nel campo delle CNN includono AlexNet [18], VGGNet [19], ResNet [20] e Inception [21]. In particolare, ResNet ha introdotto le cosiddette residual connections, cioè scorciatoie che permettono di saltare uno o più strati nella rete, facilitando il flusso del gradiente e permettendo di allenare reti molto profonde senza incorrere nel problema della scomparsa del gradiente.

Negli ultimi anni (dal 2019 in poi), sono stati fatti passi avanti enormi. Uno dei modelli più significativi è EfficientNet [22]. Questo modello si basa su un concetto chiamato compound scaling, che consiste nello scalare in modo bilanciato la profondità, la larghezza e la risoluzione delle immagini. Nel 2021, gli stessi autori hanno introdotto EfficientNetV2 [23], che ha migliorato ulteriormente i tempi di training e la qualità della convergenza grazie all'introduzione del progressive learning.

Un'altra grande innovazione recente è l'introduzione dei meccanismi di attenzione all'interno delle CNN. Questi meccanismi permettono alla rete di dare maggiore importanza a certe parti dell'immagine o a certi canali delle feature map. Il modulo Squeeze-and-Excitation (SE) [24] agisce a livello di canale, mentre il più avanzato CBAM (Convolutional Block Attention Module) [25] combina attenzione spaziale e canale, migliorando la capacità della rete di concentrarsi sulle parti visivamente più rilevanti.

Nel 2021 è nato CoAtNet [26], un modello ibrido CNN–Transformer che unisce convoluzioni e self-attention per beneficiare della località spaziale e della modellazione globale. Questo ha aperto la strada a una nuova generazione di architetture ibride altamente performanti.

Negli ultimissimi anni, numerosi studi hanno proposto ulteriori ottimizzazioni:

CBAM-EfficientNetV2 ha mostrato prestazioni eccezionali in ambito medicale (classificazione istopatologica), integrando il modulo CBAM su EfficientNetV2 e migliorando drasticamente accuratezza e capacità discriminativa [27].

EfficientNetV2-RegNet è stato recentemente adottato per applicazioni di sicurezza in reti IoT, combinando robustezza e accuratezza con costi computazionali contenuti [28].

Per quanto riguarda l'ottimizzazione delle reti, OCNNA ha introdotto un metodo combinato di pruning e knowledge distillation per alleggerire le CNN senza perdere accuratezza, rendendole più adatte a dispositivi edge o mobile [29].

Infine, MobileNetV4 (2024) ha introdotto una serie di architetture universalmente scalabili per il mobile, migliorando ulteriormente i moduli Inverted Bottleneck e i meccanismi di attenzione con multi-query attention [30].

Queste architetture sono alla base di alcuni dei modelli sperimentati in questa tesi. Le CNN si sono infatti dimostrate efficaci nell'analisi automatica di segnali radar, dove la struttura spaziale dei dati consente di sfruttare appieno le capacità di estrazione di pattern

locali. Analogamente, nel contesto dell'elaborazione di immagini georadar (GPR), hanno permesso di migliorare la classificazione e l'individuazione di oggetti nascosti nel sottosuolo, contribuendo in modo significativo.

### 2.2.2 YOLO (You Only Look Once)

YOLO (You Only Look Once) è un approccio rivoluzionario nell'object detection, progettato per rilevare oggetti all'interno di immagini e video in modo rapido e accurato. A differenza di altre architetture di rilevamento oggetti che suddividono l'immagine in più fasi, YOLO applica una rete neurale convoluzionale (CNN) su tutta l'immagine simultaneamente, trattando il problema come una regressione diretta. In particolare, l'immagine viene suddivisa in una griglia e ogni cella di questa griglia è responsabile della predizione di un insieme di informazioni, tra cui le coordinate del bounding box, la probabilità di presenza di un oggetto e la classe a cui l'oggetto appartiene [31].

Il modello YOLO opera in un solo passaggio per ogni immagine. Durante il training, la rete apprende a classificare e localizzare oggetti direttamente, anziché estrarre manualmente caratteristiche da immagini, come avviene in altri approcci basati su region proposal. Ogni cella nella griglia prevede una serie di parametri che descrivono l'oggetto rilevato, inclusi: le coordinate del centro del bounding box, la larghezza e l'altezza, un punteggio di confidenza (che rappresenta la probabilità che il box contenga un oggetto) e la probabilità di ciascuna classe dell'oggetto. Questo approccio consente a YOLO di essere molto più veloce rispetto a metodi che richiedono molteplici passaggi di elaborazione [32].

La rete di YOLO è un'architettura completamente end-to-end, che significa che tutto il processo di rilevamento viene eseguito in una singola passata della rete. Dopo che la rete ha fatto le sue predizioni, un processo di Non-Maximum Suppression (NMS) è utilizzato per rimuovere i box che si sovrappongono e mantenere solo quelli con la maggiore confidenza. Ciò permette di ridurre al minimo le predizioni false e ottenere una localizzazione più precisa [33].

Le versioni più recenti di YOLO, come YOLOv5, YOLOv6, e fino alla versione attuale YOLOv12, hanno portato numerosi miglioramenti, tra cui ottimizzazioni nei modelli di backbone, miglioramenti nella gestione delle ancore (anchor boxes), l'introduzione di moduli di attenzione e l'adozione di approcci per gestire più efficacemente gli oggetti di piccole dimensioni. Inoltre, l'efficienza computazionale è stata migliorata ulteriormente, con l'adozione di tecniche come la distillazione della conoscenza e ottimizzazioni nei modelli per dispositivi mobile e edge, che permettono a YOLO di funzionare in tempo reale anche su hardware meno potente [34, 35].

Una delle innovazioni più recenti, presente in YOLOv12, è l'integrazione diretta del processo di selezione dei bounding box all'interno della rete, eliminando la necessità del tradizionale NMS. Questo ha portato a una maggiore velocità e precisione nella localizzazione degli oggetti. Inoltre, l'adozione dei Transformer-based architectures ha migliorato ulteriormente la capacità di YOLO di rilevare oggetti con caratteristiche complesse o distribuiti su grandi aree [35].

Con queste innovazioni, YOLO continua a essere uno dei modelli più utilizzati nell'object detection, adattandosi rapidamente a nuove sfide e applicazioni, dai veicoli autonomi alla videosorveglianza, fino alle applicazioni industriali e mediche.

### 2.2.3 Transformers

I Transformer rappresentano una delle innovazioni più significative nel campo dell'apprendimento profondo, originariamente introdotti per il trattamento di sequenze nel contesto del Natural Language Processing (NLP). La loro struttura si basa sull'autoattenzione (self-attention), che permette al modello di "guardare" tutte le parti di una sequenza simultaneamente e determinare l'importanza relativa di ogni elemento per gli altri, senza la necessità di passare attraverso le informazioni in ordine sequenziale, come avviene nelle reti ricorrenti (RNN). Questo approccio ha portato a miglioramenti significativi nelle prestazioni di modelli NLP, ma ha anche trovato applicazione in altri domini come la visione artificiale e l'object detection [36].

I Transformer sono costruiti attorno al concetto di attenzione multi-testa, dove ogni testa di attenzione apprende una diversa rappresentazione del contesto. Questo consente al modello di cogliere informazioni da diverse parti della sequenza o dell'immagine, e di combinare queste informazioni in modo da ottenere una comprensione più completa. L'adozione dei Transformer nella computer vision ha portato alla nascita di modelli come il Vision Transformer (ViT), che applica l'architettura dei Transformer alle immagini trattandole come sequenze di patch. Questo approccio ha superato le prestazioni delle CNN tradizionali su alcuni benchmark [37].

Nel campo dell'object detection, l'integrazione dei Transformer con modelli esistenti ha prodotto architetture ibride, come il DETR (DEtection TRansformer), che combina il potere della self-attention per migliorare la localizzazione e la classificazione degli oggetti. A differenza dei metodi tradizionali, che utilizzano region proposal networks (RPN) per generare proposte di oggetti, DETR applica il Transformer direttamente sui pixel dell'immagine per produrre le previsioni finali in un solo passaggio [38].

Le versioni più recenti di modelli Transformer-based per l'object detection hanno ulteriormente ottimizzato l'architettura per migliorare l'efficienza computazionale e le prestazioni

in scenari complessi. Ad esempio, nel 2023, sono stati introdotti i modelli Deformable DETR e Swin Transformer, che utilizzano meccanismi di attenzione locale e gerarchica per affrontare sfide come la gestione di oggetti di piccole dimensioni e la riduzione dei costi computazionali. Questi miglioramenti hanno permesso ai modelli basati su Transformer di diventare una scelta sempre più popolare per il rilevamento di oggetti, superando spesso le tradizionali CNN in termini di precisione e velocità [39, 40].

Infine, la continua evoluzione dei Transformer ha portato a modelli ibridi che combinano le loro capacità con quelle di altre architetture, come le CNN. Questi modelli, come il CoAtNet, combinano il meglio dei Transformer e delle CNN per approfittare sia della potenza della self-attention che dell'efficienza delle convoluzioni, migliorando ulteriormente le performance in scenari complessi di rilevamento e segmentazione .

### 2.2.4 Generative Adversarial Networks (GAN)

Le Generative Adversarial Networks (GAN) sono una delle innovazioni più potenti nel campo dell'apprendimento automatico, introdotte per la prima volta nel 2014 da Ian Goodfellow e dai suoi collaboratori. Questi modelli sono composti da due reti neurali che giocano un "gioco" competitivo: il Generatore e il Discriminatore. Il generatore è responsabile della creazione di dati sintetici, come immagini, mentre il discriminatore deve distinguere tra i dati reali e quelli generati. Entrambi i modelli migliorano progressivamente grazie a questo processo di competizione. Il generatore cerca di ingannare il discriminatore creando dati che sembrano sempre più realistici, mentre il discriminatore perfeziona la sua capacità di rilevare le differenze tra dati veri e falsi. Questo processo di ottimizzazione ad alta competizione spinge entrambe le reti a evolversi, con l'obiettivo finale di ottenere dati generati che siano indistinguibili da quelli reali [41].

Le GAN sono particolarmente potenti nella generazione di contenuti. Ad esempio, possono essere utilizzate per creare immagini realistiche di volti umani, scene di paesaggi e perfino opere d'arte. Il Generatore parte da un input casuale, spesso un vettore di rumore, e attraverso un processo di apprendimento genera dati che emulano la distribuzione dei dati reali. Il Discriminatore, invece, riceve sia dati reali che sintetici e ha il compito di classificare correttamente l'origine dei dati. Questo approccio di addestramento competitivo è ciò che permette alle GAN di produrre risultati sempre più sofisticati nel tempo.

Nel corso degli anni, sono emerse varie varianti delle GAN, ciascuna con l'obiettivo di risolvere specifici problemi e migliorare le prestazioni. Ad esempio, le DCGAN (Deep Convolutional GAN) utilizzano architetture convoluzionali, ottimizzando la generazione di immagini, specialmente per compiti ad alta risoluzione [42]. Un altro esempio significativo è il WGAN (Wasserstein GAN), che introduce una nuova funzione di perdita basata sulla

distanza di Wasserstein. Questo approccio ha migliorato la stabilità dell'addestramento delle GAN, risolvendo il problema del "mode collapse", una situazione in cui il generatore produce solo una varietà limitata di output [43]. Inoltre, CycleGAN è una variante che ha portato innovazioni nell'ambito della traduzione di immagini da un dominio a un altro senza la necessità di coppie di immagini corrispondenti, come nel caso del trasferimento di stile tra immagini [44].

Uno degli sviluppi più recenti nelle GAN è rappresentato da StyleGAN, che ha introdotto un modo sofisticato di controllare e modificare il "stile" delle immagini generate. Questo ha portato alla creazione di immagini straordinariamente realistiche, come i volti umani generati da modelli StyleGAN, che sono difficili da distinguere da fotografie reali [45]. Grazie alla possibilità di manipolare dettagli come l'angolazione della faccia o l'espressione, le applicazioni di StyleGAN si estendono dalla generazione di volti alla creazione di opere d'arte e oggetti virtuali altamente realistici.

Oltre alla generazione di immagini, le GAN sono state applicate anche in contesti come la super-risoluzione, dove modelli GAN migliorano la qualità delle immagini a bassa risoluzione generandone versioni più dettagliate [46]. Altre applicazioni interessanti delle GAN includono la generazione di video, dove i modelli possono produrre sequenze video realiste partendo da descrizioni testuali o immagini statiche [47], e la generazione di dati sintetici per l'addestramento di altri modelli di machine learning, in scenari in cui i dati reali sono scarsi o difficili da raccogliere [48].

## 2.3 Tecniche di addestramento, validazione e ottimizzazione

Durante lo sviluppo di un modello di apprendimento automatico, è fondamentale comprendere le principali tecniche impiegate per addestrare, validare e ottimizzare il modello stesso. In questa sezione vengono descritte le fasi più importanti di questo processo.

- **L'addestramento:** è il processo in cui un modello di machine learning apprende dai dati. Durante questa fase, il modello viene esposto ai dati di addestramento e cerca di trovare un insieme di parametri (pesi) che minimizzano l'errore rispetto ai dati stessi. Il cuore dell'addestramento è l'ottimizzazione di una funzione di perdita, che misura quanto il modello è lontano dalla risposta corretta. Durante l'addestramento, il modello aggiorna iterativamente i suoi parametri per ridurre il valore della funzione di perdita.

  Il più comune algoritmo di ottimizzazione utilizzato nell'addestramento delle reti neurali è la discesa del gradiente. In sostanza, questo metodo calcola il gradiente (cioè

la derivata) della funzione di perdita rispetto ai pesi e modifica i pesi nella direzione opposta al gradiente per ridurre l'errore. La discesa del gradiente stocastico (SGD) è una versione che utilizza solo un singolo esempio di dati alla volta per aggiornare i pesi, rendendo l'addestramento più veloce e meno costoso computazionalmente. Tuttavia, l'SGD può essere più rumoroso, quindi spesso vengono utilizzati metodi più avanzati, come Adam, che combinano gli aggiornamenti adattivi del tasso di apprendimento con la discesa del gradiente [49].

- **La validazione:** è un passaggio cruciale nel ciclo di vita del modello, in cui si valuta la capacità di generalizzazione del modello. Il modello viene addestrato sui dati di addestramento, ma il set di dati di validazione è separato da quelli di addestramento e viene utilizzato per monitorare le prestazioni del modello durante l'addestramento. Questo aiuta a evitare l'overfitting, un problema in cui il modello diventa troppo specifico ai dati di addestramento, imparando anche il rumore e le anomalie, invece di generalizzare bene a nuovi dati.

  La k-fold cross-validation è una tecnica avanzata di validazione che suddivide i dati in "k" sottoinsiemi o fold. Il modello viene addestrato su k-1 fold e testato sul fold rimanente, ripetendo questo processo per tutti i fold. Questo approccio fornisce una stima più affidabile delle prestazioni del modello, riducendo la dipendenza dalla divisione casuale dei dati [50].

- **L'ottimizzazione:** è il processo di miglioramento delle prestazioni di un modello attraverso l'aggiornamento iterativo dei suoi parametri. La discesa del gradiente è la tecnica di base, ma esistono algoritmi più avanzati, come Adam (Adaptive Moment Estimation) [51], che calcolano una stima adattiva dei tassi di apprendimento per ciascun parametro. Questo migliora la velocità di convergenza, particolarmente in presenza di dati complessi o grandi dataset, e consente di evitare il problema del tasso di apprendimento fisso che può essere troppo alto o troppo basso per alcune situazioni.

  RMSprop e Adagrad sono altri esempi di algoritmi che ottimizzano la gestione del tasso di apprendimento. Questi algoritmi utilizzano informazioni sulle medie mobili dei gradienti passati per adattare il tasso di apprendimento in modo che cambi dinamicamente durante l'addestramento, il che può essere particolarmente utile quando i dati contengono caratteristiche che cambiano nel tempo.

- **Regolarizzazione:** è una tecnica usata per prevenire l' overfitting del modello, che si verifica quando il modello diventa troppo complesso e perde la capacità di generalizzare ai dati nuovi. Una delle tecniche di regolarizzazione più comuni è il dropout [52], che disattiva casualmente una parte dei neuroni durante l'addestramento. Questo costringe la rete a non fare affidamento su singoli neuroni e a distribuire l'apprendimento tra più

unità, migliorando la robustezza del modello e riducendo il rischio di overfitting. Il decadimento del peso L2 (o penalizzazione L2) è un'altra tecnica di regolarizzazione che penalizza i pesi di grande valore. In pratica, aggiunge un termine alla funzione di perdita che cresce quadratamente con la grandezza dei pesi. Questo incoraggia il modello a mantenere i pesi piccoli e a evitare di sviluppare complessità inutili nei suoi parametri. Questo tipo di regolarizzazione è particolarmente utile quando i dati sono rumorosi e si vuole evitare che il modello si adatti troppo alle piccole fluttuazioni nei dati di addestramento.

- **Il early stopping:** è una strategia per fermare l'addestramento prima che il modello inizi a sovraccaricarsi. Durante l'addestramento, se le prestazioni sul set di validazione iniziano a peggiorare (indicato da un aumento della funzione di perdita sul set di validazione), il modello viene fermato prima che si adatti troppo ai dati di addestramento. Questo è un modo semplice ma efficace per migliorare la generalizzazione.

## 2.4 Applicazioni di algoritmi di DL e ML all'analisi delle immagini

Gli algoritmi di Machine Learning (ML) e Deep Learning (DL) hanno rivoluzionato l'analisi dei dati visivi, con applicazioni che spaziano dalla visione artificiale alla diagnostica medica, fino alla sorveglianza industriale. Un ambito particolarmente promettente è l'analisi automatica degli spettrogrammi, rappresentazioni tempo-frequenza di segnali temporali che rendono visibili pattern altrimenti difficili da cogliere nei dati grezzi.

Lo spettrogramma è una rappresentazione bidimensionale in cui il tempo è sull'asse delle ascisse, la frequenza su quello delle ordinate, e l'intensità (o energia) del segnale è codificata mediante colori o livelli di grigio. Viene calcolato tramite tecniche come la trasformata di Fourier a breve termine (STFT) o la trasformata wavelet [53]. Questa trasformazione fornisce una sorta di "fotografia" spettrale del segnale, utile per individuare come varia la sua composizione frequenziale nel tempo [54].

L'utilizzo dello spettrogramma è essenziale per analizzare segnali complessi — ad esempio audio, EEG o vibrazioni meccaniche — che, se considerati come semplici serie temporali, risulterebbero poco interpretabili. La STFT, tuttavia, presenta dei limiti legati alla risoluzione tempo-frequenza, che dipende dalla scelta della finestra. La trasformata wavelet può offrire una miglior localizzazione in frequenza, soprattutto per segnali che contengono componenti sia lente che rapide [53].

Una volta convertiti in immagini, gli spettrogrammi possono essere analizzati con algoritmi di computer vision. In particolare, le reti neurali convoluzionali (CNN) risultano particolarmente efficaci grazie alla loro capacità di individuare pattern locali, invarianze spaziali e strutture gerarchiche [55]. Anche se lo spettrogramma deriva da dati sequenziali, la sua struttura visiva lo rende adatto all'analisi tramite CNN, che trattano queste rappresentazioni in modo analogo alle immagini naturali [56].

Le CNN, progettate originariamente per l'elaborazione di immagini, applicano filtri convoluzionali che identificano bordi, contorni e forme in modo automatico. Questa architettura si adatta perfettamente alla lettura di pattern temporali e frequenziali presenti negli spettrogrammi [57, 58]. Mentre i livelli più bassi di una CNN rilevano caratteristiche semplici, quelli superiori sono in grado di cogliere strutture più complesse e semantiche, come sequenze o periodicità [59].

Ad esempio, in un segnale audio, le CNN possono riconoscere variazioni di frequenza che corrispondono a cambiamenti tonali, come le note di un brano musicale. In ambito medico, possono identificare anomalie nei segnali fisiologici, come picchi anomali in uno spettro EEG, oppure, in contesti industriali, rilevare malfunzionamenti in macchinari analizzando i cambiamenti nelle vibrazioni [60–63].

### 2.4.1 Applicazioni Georadar

In questo contesto ci concentriamo sul Ground Penetrating Radar (GPR). Esso è una tecnologia di telerilevamento terrestre che si distingue dalle tecniche tradizionali perché non si limita a osservare la superficie del suolo, ma ne esplora anche il sottosuolo. A differenza del telerilevamento classico, che si basa su immagini satellitari o aeree, il GPR penetra nel terreno grazie all'emissione di impulsi elettromagnetici.

Questi impulsi vengono trasmessi tramite un'antenna verso il suolo: quando incontrano materiali diversi come rocce, acqua, radici o cavità, parte dell'energia viene riflessa e torna indietro al dispositivo. Il GPR misura il tempo impiegato dal segnale per tornare e la sua intensità. In questo modo, è possibile ottenere una "mappa" del sottosuolo, simile a un'ecografia del terreno.

Il GPR trova applicazione in numerosi settori:

- Ingegneria civile: per ispezionare infrastrutture come ponti, strade, dighe e marciapiedi senza interrompere il loro utilizzo. Viene usato anche per localizzare reti sotterranee (tubi, cavi, condotti) e valutare lo stato di fondazioni e strutture in calcestruzzo.
- Archeologia: permette di individuare strutture sepolte (tombe, muri antichi, edifici, reperti) senza dover scavare, aiutando così a preservare i siti storici.

- Geologia: per analizzare la composizione e la stratificazione del terreno e delle rocce, individuare falde acquifere o studiare i ghiacciai.
- Esplorazione spaziale: utilizzato nelle missioni su Marte o sulla Luna per studiare il sottosuolo e identificare risorse come l'acqua ghiacciata, utili per missioni future.

Le radici degli alberi svolgono un ruolo cruciale nell'ecosistema: forniscono stabilità alla pianta, assorbono acqua e nutrienti dal suolo, fungono da riserva di risorse e producono ormoni fondamentali per la crescita. La salute delle radici è strettamente connessa alla salute complessiva dell'albero, ed è quindi fondamentale per comprenderne le condizioni.

Comprendere lo stato delle radici significa poter valutare la salute complessiva dell'albero. Le tecniche tradizionali per studiare le radici, tuttavia, risultano spesso invasive:

- **Scavo manuale del terreno intorno alle radici:** è uno dei metodi più semplici e diretti, ma anche tra i più laboriosi. Richiede tempo, attrezzature manuali e una grande attenzione per non danneggiare le radici durante l'asportazione del terreno. Inoltre, rimuovere grandi volumi di suolo può compromettere la stabilità dell'albero, disturbare l'ecosistema circostante (microfauna, flora, microrganismi) e generare un impatto ambientale significativo, soprattutto in contesti naturali o urbani sensibili.
- **Estirpazione dell'albero:** questa tecnica garantisce una visione completa dell'apparato radicale, ma comporta la perdita irreversibile dell'albero. È spesso usata in ambito accademico o per casi molto specifici, quando l'indagine è finalizzata allo studio completo della morfologia radicale. Tuttavia, non è applicabile su larga scala né compatibile con strategie di conservazione ambientale o gestione del verde urbano.
- **Potatura radicale e prelievo in laboratorio:** consiste nel tagliare una porzione delle radici, che viene poi analizzata in laboratorio per valutare caratteristiche fisiologiche, patologiche o chimiche. Anche se meno distruttiva dell'estirpazione, questa tecnica può compromettere la salute dell'albero, ridurne la stabilità e aumentare il rischio di malattie o stress idrico. In alcuni casi può anche innescare processi di deperimento a lungo termine.
- **Trattamenti chimici per sciogliere il terreno:** soluzioni chimiche (come acidi deboli) per rimuovere il terreno e mettere in evidenza la rete radicale. Questo metodo può essere utile per piccoli campioni, ma presenta gravi rischi ecologici: alterazione del pH del suolo, contaminazione delle acque sotterranee e danni alle radici stesse. Inoltre, è spesso poco praticabile in ambienti naturali o agricoli.

Inoltre, l'analisi del suolo presenta numerose sfide:

- **Alta variabilità della composizione del terreno** I terreni possono variare notevolmente anche su distanze molto brevi: in profondità, densità, umidità, presenza di argilla, sabbia, limo, sostanze organiche, minerali, ecc. Questa variabilità rende difficile ottenere dati coerenti e confrontabili. Inoltre, la stessa specie vegetale può sviluppare un apparato radicale diverso in base al tipo di suolo, rendendo complicata la modellizzazione.
- **Possibilità di errore nelle misurazioni** Le indagini manuali o strumentali possono essere soggette a errori umani (es. profondità sbagliata, identificazione errata delle radici) o tecnici (sensori non tarati, strumenti sensibili a interferenze ambientali). Anche piccole imprecisioni possono compromettere la qualità del dataset, rendendo inaffidabili le analisi successive.
- **Complessità nella gestione di grandi quantità di dati** I dati raccolti durante studi sul suolo e sulle radici (immagini, misure fisico-chimiche, dati temporali, profondità, localizzazione) sono numerosi e spesso eterogenei. La loro elaborazione richiede sistemi avanzati di archiviazione, pulizia, analisi e visualizzazione. Inoltre, l'integrazione con modelli predittivi o di apprendimento automatico richiede competenze multidisciplinari e software specifici.

E proprio a causa della natura fortemente invasiva delle tecniche tradizionali sopra descritte che spesso comportano danni irreversibili all'apparato radicale, alterazioni dell'ambiente circostante o la compromissione della salute dell'albero – il presente lavoro si è focalizzato sull'esplorazione del GPR radar come metodo non distruttivo.

Nel caso specifico analizzato, le scansioni sono state effettuate nel Glasboury Park (Figura 2.2). Ogni singola scansione ha richiesto un tempo di acquisizione pari a 80 nanosecondi un intervallo molto breve che consente di ottenere misure dettagliate. Complessivamente sono state eseguite 36 scansioni, distribuite in modo semicircolare attorno all'albero, coprendo una superficie totale di 218,04 m². Ogni scansione è stata effettuata a 30 cm di distanza dalla precedente, per assicurare una copertura sistematica dell'area analizzata. Le misure nel Glasboury Park non sono state raccolte direttamente, ma sono state fornite dal gruppo di ricerca dell' *University of West London*, che ne detiene la proprietà e la responsabilità.

**Figure 2.2** Glasboury Park, Londra

I dati acquisiti sono stati successivamente elaborati attraverso diverse fasi di pre-processing:

- **Zero Offset Removal:** per allineare il punto iniziale delle misurazioni.
- **Zero Time Correction:** per sincronizzare temporalmente i segnali.
- **T**ime-Varying Gain: per amplificare i segnali più deboli.
- **Filtro SVD (Singular Value Decomposition):** per rimuovere il rumore e isolare i segnali utili.
- **Bandpass Filtering:** per eliminare frequenze indesiderate.

Infine, i dati GPR sono stati trasformati in spettrogrammi bidimensionali mediante la tecnica della Short-Time Fourier Transform (STFT). Nelle figure 2.3, 2.4,2.5,2.6 possiamo vedere esempi di spettrogrammi generati.

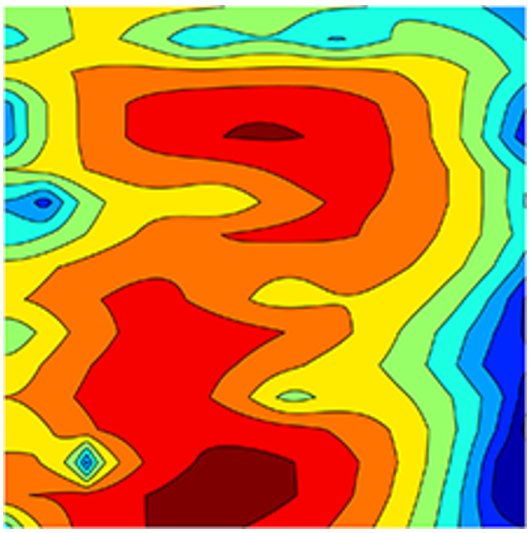


**Figure 2.3** Radice

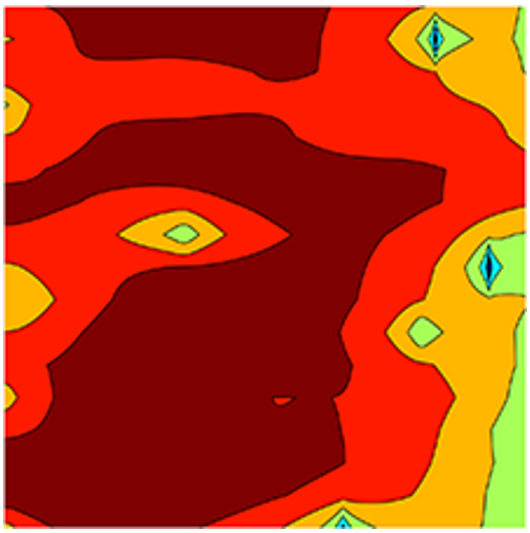


**Figure 2.4** Clustering di radici

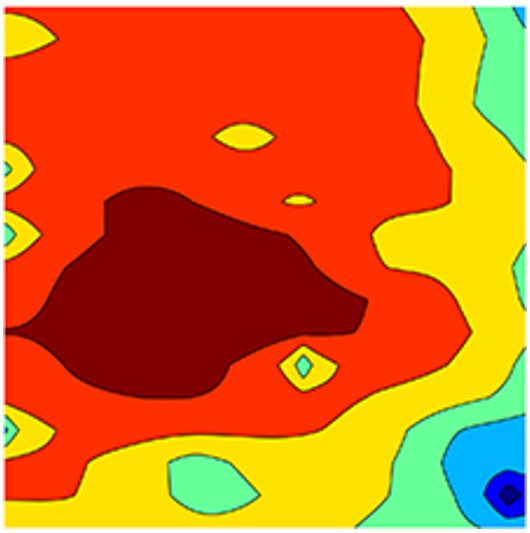


**Figure 2.5** Pietre

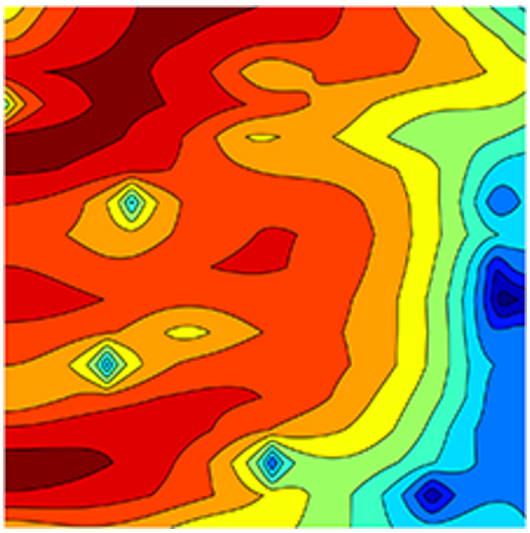


**Figure 2.6** Nessun oggetto

Nel trattamento dei dati reali, però, i segnali GPR possono essere affetti da rumore dovuto a vari fattori, come interferenze elettromagnetiche, errori strumentali o disturbi ambientali. Per rendere il sistema più robusto, durante il processo di data augmentation viene introdotto rumore gaussiano simulato, che permette al modello di apprendere come distinguere i segnali significativi anche in condizioni non ideali. Le feature estratte come la deviazione standard, la mediana e la densità integrata sono misure statistiche che descrivono l'intensità e la distribuzione del segnale. Queste informazioni risultano spesso rappresentative delle caratteristiche del sottosuolo, e sono quindi fondamentali per un'analisi accurata e affidabile fig 2.7.

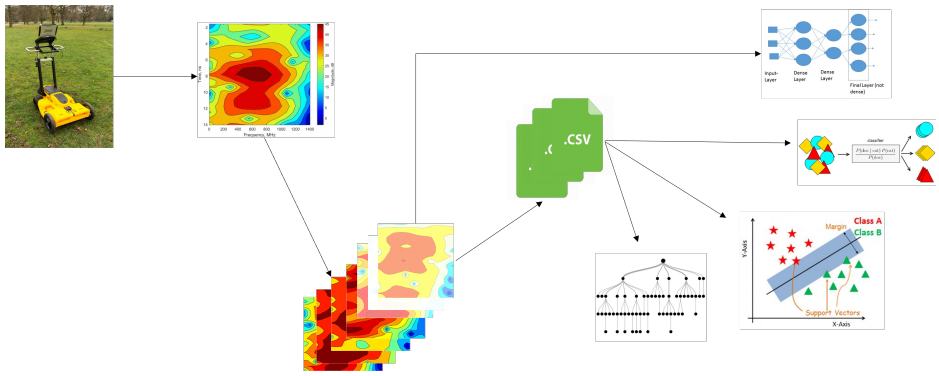



**Figure 2.7** Workflow dalle scansioni GPR all'elaborazione e classificazione tramite algoritmi di machine learning

Le Convolutional Neural Networks (CNN) sono ispirate alla corteccia visiva degli animali e si sono affermate come uno strumento fondamentale nell'elaborazione delle immagini, come già ampliamente illustrato e sono state utilizzate anche in questo progetto. Tra le architetture più rappresentative troviamo SqueezeNet e VGG-16, ciascuna con caratteristiche specifiche.

**SqueezeNet:** è una CNN leggera progettata per ottenere elevate prestazioni con un numero ridotto di parametri. L'architettura si basa sull'utilizzo dei *fire modules*, composti da:

- **Squeeze layers**: che concentrano l'attenzione sulle caratteristiche più rilevanti dell'immagine;
- **Expand layers**: che ampliano la rete per migliorarne la capacità rappresentativa.

Questa struttura rende SqueezeNet particolarmente efficiente per dispositivi con risorse computazionali limitate, come smartphone o sistemi embedded.

**VGG-16**, invece, è un'architettura più profonda composta da 16 strati, principalmente convoluzionali, seguiti da strati completamente connessi. Fa uso di filtri di piccole dimensioni ($3\times3$), impilati in successione per estrarre in modo sempre più dettagliato le caratteristiche dell'immagine.

- È molto precisa e affidabile;
- Tuttavia, risulta pesante dal punto di vista computazionale e richiede una notevole quantità di memoria;
- È adatta a contesti di ricerca o ad applicazioni in cui non vi siano particolari vincoli hardware.

Per garantire condizioni di parità tra tutte le reti, sono state definite specifiche comuni per l'addestramento. Tutte le reti neurali sono state addestrate per 50 epoche con una dimensione

del batch pari a 50. Per contrastare i problemi legati all'overfitting, sono state aggiunte funzioni di regolarizzazione e tecniche di validazione. Inoltre, è stato utilizzato un pool parallelo per velocizzare l'addestramento.

L'ottimizzazione è stata effettuata tramite l'algoritmo Stochastic Gradient Descent with Momentum (SGDM), con una velocità iniziale di apprendimento impostata a 0.0003. Durante l'addestramento, la validazione è stata eseguita ogni 30 iterazioni. Il dataset è stato suddiviso nel seguente modo: 70% per l'addestramento, 10% per la validazione e 20% per il test: In figura 2.8 vengono mostrati i risultati espressi in percentuale.

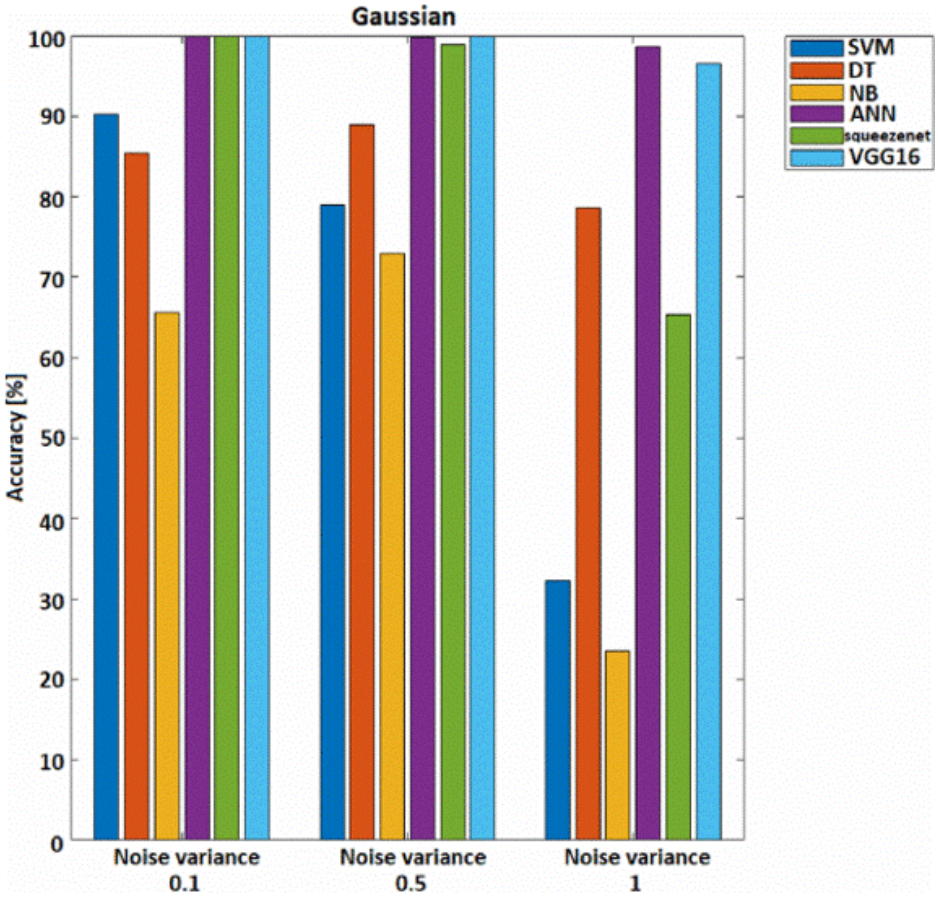



**Figure 2.8** Risultati in termini di accuracy con varianza variabile

I metodi SVM e Naive Bayes (NB) si sono rivelati poco adatti a questo tipo di compito, mostrando un'accuratezza compresa rispettivamente tra 45% e 65% e tra 37% e 50%.

La rete VGG-16 ha mostrato risultati più accurati, in linea con le aspettative, grazie alla sua struttura profonda e alla capacità di apprendere rappresentazioni complesse. Tuttavia, all'aumentare del rumore, si osserva un calo generalizzato delle prestazioni: in particolare, le reti basate su Naive Bayes e SVM risultano essere le più sensibili al disturbo. Il Decision Tree riesce invece a mantenere una certa robustezza. Anche SqueezeNet mostra una riduzione

nelle prestazioni, coerentemente con il suo design: è infatti pensata per essere leggera ed efficiente, con un numero di parametri ridotto rispetto ad altre CNN. Questo comporta però una minore capacità di apprendere caratteristiche complesse e, di conseguenza, una più bassa tolleranza al rumore.Tuttavia nonostante la varianza di rumore più alta entrambe le reti riescono a mantenere risultati accettabili. Nelle tabelle sottostanti 2.1, 2.2 vengono ripostate anche la precision , la recall e la F-measure.

**Table 2.1** Risultati dei classificatori con varianza di rumore pari a 0.5

| Modello | Precision | Recall | F-measure |
|---|---|---|---|
| SVM | 0.818 | 0.793 | 0.783 |
| DT | 0.893 | 0.889 | 0.886 |
| NB | 0.740 | 0.732 | 0.718 |
| Squeeze-net | 0.987 | 0.990 | 0.987 |
| VGG-16 | 1.000 | 1.000 | 1.000 |

**Table 2.2** Risultati dei classificatori con varianza di rumore pari a 1

| Modello | Precision | Recall | F-measure |
|---|---|---|---|
| SVM | 0.326 | 0.323 | 0.0324 |
| DT | 0.812 | 0.786 | 0.777 |
| NB | 0.444 | 0.235 | 0.110 |
| Squeeze-net | 0.617 | 0.695 | 0.600 |
| VGG-16 | 0.962 | 0.962 | 0.962 |

Visti gli ottimi risultati ottenuti dalle reti neurali, abbiamo deciso di proseguire in questa direzione esplorando ulteriori architetture.

Il Ground Penetrating Radar (GPR) genera segnali complessi e immagini caratterizzate da pattern non lineari, difficili da interpretare con i metodi tradizionali, che spesso richiedono un'intensa fase di pre-processing. Il deep learning semplifica questo processo, in quanto è in grado di apprendere direttamente dai dati grezzi, eliminando la necessità di un'estrazione manuale delle caratteristiche e migliorando la precisione dei risultati.

Inoltre, il deep learning risulta particolarmente efficace quando si lavora con grandi quantità di dati, poiché è in grado di individuare rappresentazioni profonde e dettagliate a partire da dati etichettati. Questa scalabilità lo rende particolarmente adatto alle applicazioni GPR, dove solitamente si raccolgono volumi consistenti di informazioni (Figura 2.9).

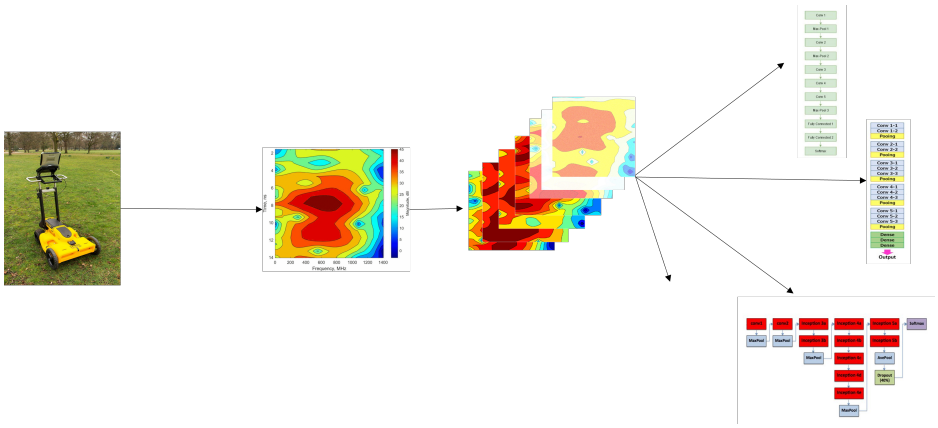


**Figure 2.9** Workflow dalle scansioni GPR all'elaborazione e classificazione tramite reti neurali convoluzionali

Vedendo come le reti neurali abbiano performato bene , si è voluto espandere il progetto sperimentando altre tipologie di reti.

GoogLeNet è nota per la sua efficienza e per il design innovativo, grazie all'introduzione del modulo Inception.

- **Moduli Inception:** combinano convoluzioni di diverse dimensioni ($1 \times 1$, $3 \times 3$, $5 \times 5$) in parallelo, permettendo di catturare dettagli su più scale.
- **Struttura profonda:** composta da 22 strati, riesce a ridurre il numero di parametri rispetto a reti precedenti (come AlexNet), grazie all'uso efficiente delle convoluzioni $1 \times 1$.
- **Efficienza:** il numero di parametri è inferiore rispetto ad altre reti simili, rendendola più leggera da addestrare.

Nonostante i vantaggi descritti, GoogLeNet presenta alcune limitazioni, tra cui:

- **Profondità e complessità:** GoogLeNet utilizza moduli Inception con convoluzioni parallele di varie dimensioni. Sebbene questo approccio consenta di estrarre molte caratteristiche, aumenta la complessità della rete.
- **Maggiore sensibilità al rumore:** la profondità e le connessioni parallele possono amplificare il rumore presente nei dati in ingresso, causando un degrado delle prestazioni in presenza di perturbazioni.
- **Sensibilità alla qualità del dataset:** se il dataset non è ben bilanciato o contiene molto rumore, GoogLeNet può risultare meno efficace rispetto a reti più compatte come SqueezeNet o più robuste come VGG-16.

AlexNet d'altra parte, ha rappresentato una svolta nell'ambito delle reti neurali profonde, introducendo alcune tecniche chiave:

- **8 strati:** di cui 5 convoluzionali e 3 completamente connessi.
- **Attivazioni ReLU:** utilizza la funzione ReLU per velocizzare l'addestramento.
- **Data Augmentation:** applicato per aumentare la varietà del dataset e migliorare le performance.
- **Dropout:** impiegato per ridurre l'overfitting.

In 2.10 vengono messi a confronto i risultati delle 4 reti in termini di accuracy

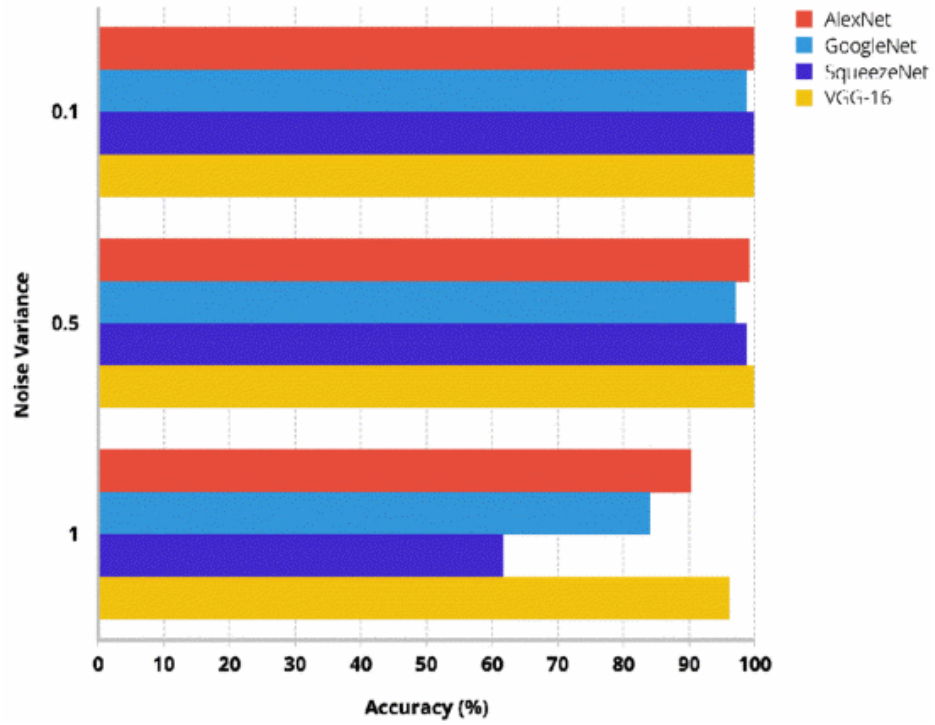



**Figure 2.10** Risultati in termini di accuracy con varianza variabile delle reti convoluzionali

Si osserva che, con una varianza di rumore molto bassa (0.1), tutte le reti mantengono prestazioni eccellenti, con accuratezze superiori al 90%. Questo significa che un livello minimo di disturbo non compromette significativamente la capacità delle reti di riconoscere correttamente le immagini.

Quando la varianza del rumore aumenta a 0.5, iniziano a manifestarsi le prime differenze: AlexNet, pur mantenendo un buon livello di accuratezza, mostra un leggero calo più evidente rispetto alle altre reti. GoogleNet e SqueezeNet rimangono abbastanza stabili, mentre VGG-16 continua a distinguersi per la sua robustezza.

Infine, al livello di rumore più elevato (varianza pari a 1), le differenze si amplificano. SqueezeNet risente maggiormente del disturbo, scendendo sotto il 70% di accuratezza. AlexNet e GoogleNet subiscono anch'esse un calo, anche se in misura minore, mentre VGG-16 dimostra una resistenza notevole, mantenendo un'accuratezza superiore al 90%.

Anche l'orientamento delle radici degli alberi può influenzare in modo significativo l'accuratezza e l'interpretazione dei dati acquisiti tramite Ground Penetrating Radar (GPR).

Le radici orientate perpendicolarmente al sensore GPR generano una riflessione più intensa del segnale radar, risultando così più facili da rilevare. Al contrario, quando l'inclinazione delle radici è bassa, ovvero quando sono più parallele alla superficie di scansione, la riflessione delle onde radar risulta debole. Questa debole risposta può compromettere la chiarezza del rilevamento, aumentando la possibilità di confondere le radici con altri oggetti sotterranei o con rumore di fondo.

In questa parte di progetto lo studio è stato condotto su nove alberi appartenenti a specie diverse, situati presso il Walpole Park di Londra (UK).

Le radici esposte sono state analizzate con il radar GPR posizionato a diversi angoli d'incidenza: 0°, 22.5°, 45° e 90°. L'analisi è stata poi ripetuta anche sulla prosecuzione sotterranea delle radici, mantenendo gli stessi angoli d'incidenza, al fine di valutare l'effetto dell'orientamento anche nel sottosuolo. Nelle fig. 2.11, 2.12 i risultati.

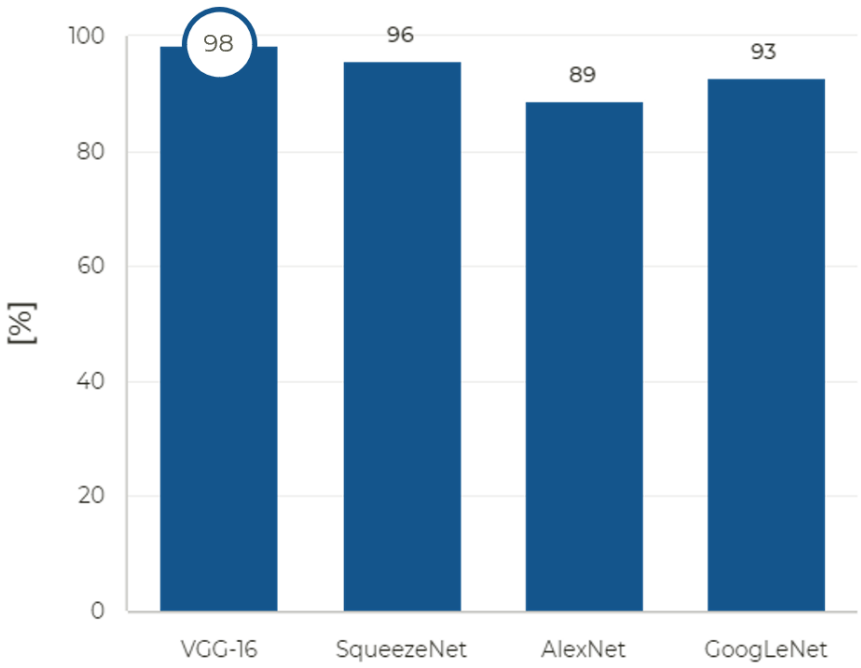



**Figure 2.11** Risultati in termini di accuracy delle reti convoluzionali nel caso di radici esposte

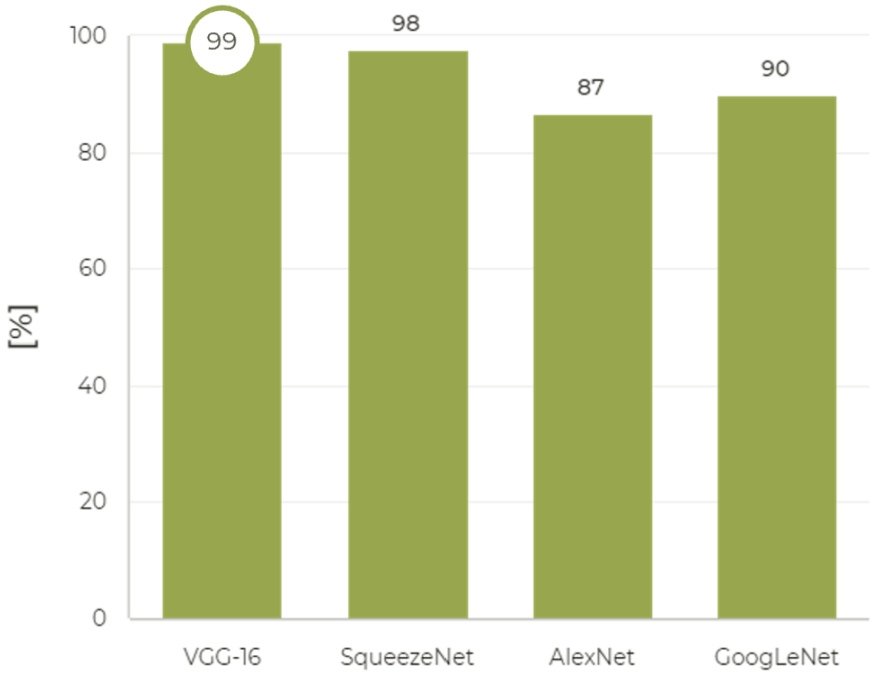



**Figure 2.12** Risultati in termini di accuracy delle reti convoluzionali nel caso di radici non esposte

Nella prima immagine, vediamo che il modello VGG-16 ottiene il risultato migliore tra tutti quelli considerati, raggiungendo un'accuratezza del 98%, valore che viene anche evidenziato graficamente con un cerchio sopra la barra. Subito dopo si posiziona SqueezeNet, che ottiene comunque un ottimo valore di 96%, mostrando una performance molto vicina a quella di VGG-16. Più distanziati troviamo invece GoogLeNet, con un'accuratezza del 93%, e infine AlexNet, che raggiunge solo l'89%, risultando il modello con le prestazioni peggiori tra quelli mostrati nella prima immagine.

Passando alla seconda immagine, notiamo che tutti i valori cambiano leggermente. In questo caso, VGG-16 migliora ulteriormente la propria prestazione, arrivando al 99%, confermandosi come il miglior modello anche in questo secondo contesto. Anche SqueezeNet mostra un miglioramento, passando a un'accuratezza del 98%, diventando ancora più vicino a VGG-16 e riducendo così la distanza tra i due. Al contrario, sia AlexNet sia GoogLeNet peggiorano: AlexNet scende dal precedente 89% all'87%, mentre GoogLeNet passa dal 93% al 90%.

Se osserviamo il confronto tra le due immagini nel loro insieme, si può dire che i modelli più performanti (VGG-16 e SqueezeNet) migliorano ulteriormente nella seconda immagine, mentre quelli con prestazioni più basse (AlexNet e GoogLeNet) peggiorano. In particolare, VGG-16 mantiene sempre il primo posto, consolidando la sua superiorità sia nella prima sia nella seconda analisi. SqueezeNet, invece, mostra un progresso importante. AlexNet, già il

peggiore nella prima immagine, peggiora ancora, mentre GoogLeNet, pur essendo superiore ad AlexNet, subisce anche lui un leggero calo.

In conclusione, l'analisi condotta mostra chiaramente come le diverse architetture di reti neurali convoluzionali reagiscano in maniera differente alla presenza di rumore e all'orientamento delle radici. VGG-16 si conferma il modello più robusto e accurato in tutti gli scenari, seguito da SqueezeNet, che pur essendo più leggero riesce a garantire prestazioni comparabili. GoogLeNet e AlexNet, invece, risultano più sensibili sia al rumore sia alle variazioni di orientamento, con un calo più marcato nelle prestazioni.

### 2.4.2 Applicazioni sicurezza: Covert Timing Channel

In questo progetto ci siamo focalizzati sull'analisi e creazione di Covert Timing Channel (CTC), una particolare tipologia di Covert Channel che trasmette messaggi alterando i ritardi tra i pacchetti (inter-arrival time) nelle comunicazioni TCP tra client e server.Questa tecnologia può essere sfruttata in modo malevolo, ad esempio per esfiltrare dati sensibili o lanciare attacchi informatici.

Un settore particolarmente vulnerabile è quello dell'Internet of Medical Things (IoMT), dove dispositivi connessi gestiscono informazioni sanitarie critiche e la compromissione della rete può mettere in pericolo la salute dei pazienti.

Successivamente, l'attenzione si è spostata sul contesto delle reti wireless, ormai ubiquitarie nella vita quotidiana. Anche in questo caso, sono stati sviluppati canali di comunicazione nascosti attraverso la modulazione dei ritardi tra pacchetti, ed è stata proposta una serie di modelli DL in grado di distinguere il traffico lecito da quello malevolo. I risultati ottenuti dimostrano la flessibilità e l'efficacia delle tecniche DL rispetto agli approcci ML tradizionali per il rilevamento di comunicazioni occulte in reti wireless.

Un **Covert Timing Channel (CTC)** è un canale nascosto che permette a un attaccante di trasmettere informazioni segrete sfruttando i *tempi di arrivo* dei pacchetti in una normale comunicazione di rete.

Il CTC dunque non modifica i dati dei pacchetti, ma trasmette un messaggio codificando le informazioni nei ritardi temporali tra un pacchetto e l'altro. Questo lo rende molto difficile da rilevare, perché dall'esterno il traffico può sembrare del tutto lecito 2.13.

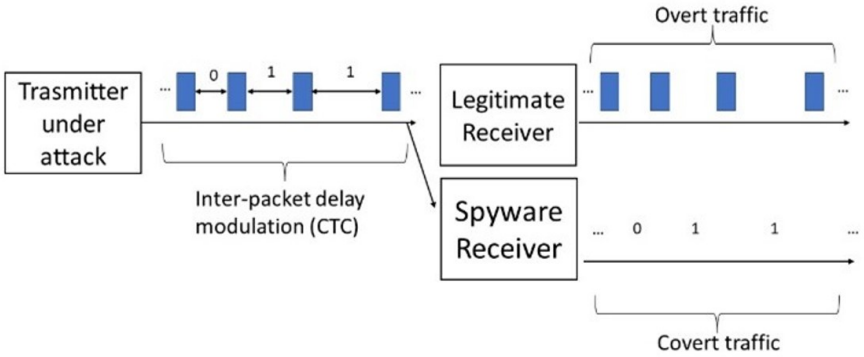



**Figure 2.13** Schema a blocchi che descrive il funzionamento di un CTC

Il CTC descritto è stato costruito su una connessione TCP (tipica di molte comunicazioni di rete, inclusi i sistemi e-health), con le seguenti caratteristiche:

1. **Codifica ON-OFF (modulazione binaria):**
   - Per inviare un bit 1 (*ON*), viene trasmesso un pacchetto sulla connessione TCP.
   - Per inviare un bit 0 (*OFF*), non viene inviato alcun pacchetto (cioè silenzio).

   Questo pattern di invio/silenzio crea un codice temporale interpretabile solo da chi conosce il trucco.
2. **Pattern di sincronizzazione:** Ogni 3 byte di messaggio covert vengono inseriti 4 bit aggiuntivi di sincronizzazione. Questo aiuta a riagganciare la sincronizzazione tra mittente e destinatario nel caso qualche pacchetto venga perso.
3. **Manchester Coding (codifica di linea):** Utilizzata per evitare lunghe sequenze di 0 o 1 che potrebbero causare perdita di sincronizzazione.
   - Il bit 0 è rappresentato come *basso-alto* (01).
   - Il bit 1 è rappresentato come *alto-basso* (10).
   - Le combinazioni 00 e 11 sono considerate non valide e quindi ignorate.

   Questa codifica raddoppia il numero di bit da inviare, ma aumenta l'affidabilità.
4. **Codice di Hamming (12,8):** Serve per correggere errori nei dati trasmessi. Per ogni 8 bit di messaggio, vengono aggiunti 4 bit di ridondanza, formando parole da 12 bit. Questo schema permette di:
   - Correggere 1 bit errato
   - Rilevare fino a 2 bit errati

   Si ha un aumento del 50% della lunghezza del messaggio, ma si migliora la robustezza della comunicazione.

5. **Strumenti utilizzati:** Wireshark viene usato per monitorare e catturare i flussi di traffico (sia legittimi che contenenti dati nascosti).

Questo sistema può trasportare dati sensibili (ad esempio cartelle cliniche, identità) senza essere rilevato facilmente. È un rischio serio per la sicurezza dei sistemi , dove la confidenzialità e l'integrità dei dati sono cruciali.

Seguendo l'approccio descritto, è stato proseguito il lavoro convertendo le sequenze dei ritardi tra pacchetti (*inter-packet delays*) di un canale CTC in spettrogrammi, applicando la Trasformata di Fourier a breve termine (STFT). L'intervallo di frequenze nello spettrogramma viene normalizzato tra 0 e 1. Grazie all'impiego della funzione *chirp*, è stato possibile generare dei campioni di un segnale cosenoidale a frequenza variabile (swept-frequency) in corrispondenza degli istanti temporali definiti dall'array dei ritardi tra pacchetti. La frequenza istantanea al tempo 0 è $f_0$, mentre la frequenza istantanea al tempo finale è $f_f$.

Abbiamo deciso di considerare quattro tipi di informazioni malevole, basandoci sulla lunghezza dei bit nascosti (*covert bits*) incorporati, come riportato nella Tabella 2.3.

| **Covert bits length** | **Classification** |
|---|---|
| $< 64$ | Very short |
| $64 \leq$ length $< 128$ | Short |
| $128 \leq$ length $\leq 150$ | Medium |
| $> 150$ | Long |

**Table 2.3** Tipologie di messaggi nascosti in base alla loro lunghezza

La suddivisione dei messaggi nascosti in quattro classi è stata introdotta per evidenziare come la lunghezza della sequenza di bit covert influenzi le prestazioni del classificatore. Infatti, sequenze molto brevi sono più difficili da distinguere da traffico lecito, mentre sequenze più lunghe presentano pattern temporali più riconoscibili e quindi più facili da rilevare.

Una volta raccolti i ritardi tra pacchetti per ciascuna delle quattro classi, queste sequenze vengono trasformate, tramite la STFT, in rappresentazioni bidimensionali (2-D), ovvero in spettrogrammi. Ogni colonna di queste rappresentazioni 2-D contiene una stima del contenuto in frequenza localizzato nel tempo della sequenza dei ritardi tra pacchetti 2.14.

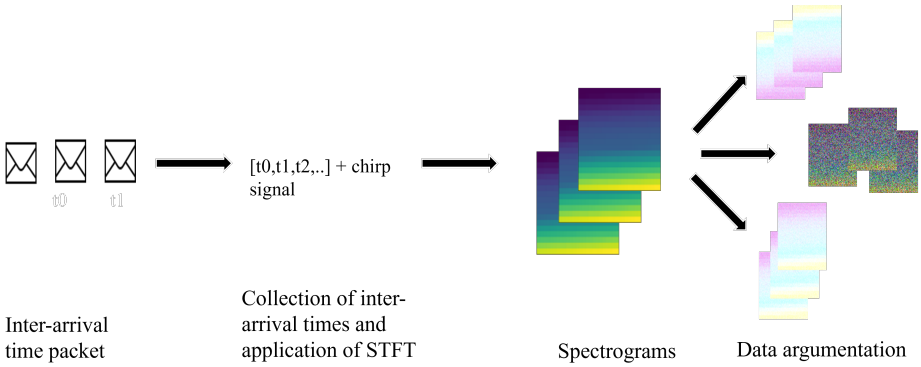



**Figure 2.14** Processo di generazione del dataset per l'analisi del CTC

Poiché gli input della fase di rilevamento sono rappresentati da immagini bidimensionali, abbiamo impiegato framework di reti convoluzionali progettati specificamente per gestire in modo efficiente ed efficace questo tipo di input. Sono state aggiunte due nuove reti per le analisi:

La **ResNet-50:** è una variante della famiglia ResNet che utilizza un design chiamato *bottleneck* nei suoi blocchi residui. Questo approccio prevede l'uso di convoluzioni $1 \times 1$, che servono a ridurre il numero di parametri e le operazioni di moltiplicazione tra matrici. In questo modo si velocizza notevolmente l'addestramento dei singoli strati.

A differenza delle versioni più semplici, la ResNet-50 impiega una struttura a tre strati nei blocchi residui (invece dei classici due), migliorando così l'efficienza e l'efficacia complessiva del modello.

La struttura della ResNet-50 prevede:

- Una convoluzione iniziale $7 \times 7$ seguita da uno strato di *max pooling*.
- Una serie di blocchi residui con combinazioni di convoluzioni $1 \times 1$, $3 \times 3$ e ancora $1 \times 1$.
- Un livello di *average pooling*.
- Un livello completamente connesso (*fully connected*) con 1000 nodi per la classificazione in 1000 categorie (es. ImageNet), con attivazione *SoftMax*.

Grazie alle connessioni *shortcut* e al design a *bottleneck*, ResNet-50 è una rete neurale profonda molto efficiente, in grado di risolvere problemi complessi di visione artificiale con ottime prestazioni.

La **ResNet-18** è un'altra rete della famiglia ResNet, più leggera della ResNet-50, progettata per l'elaborazione di immagini con dimensione standard di $224 \times 224$ pixel e 3 canali (rosso, verde e blu).

La struttura di base include:

- Uno strato di convoluzione iniziale $7 \times 7$, seguito da normalizzazione (*batch normalization*) e funzione di attivazione *ReLU*.
- Quattro blocchi residui, ciascuno composto da due strati di convoluzione, ciascuno seguito da *batch normalization* e *ReLU*.
- Connessioni shortcut che permettono al flusso di dati di saltare uno o più strati all'interno dei blocchi, facilitando l'apprendimento e risolvendo il problema della scomparsa del gradiente.
- Strati di *max pooling* tra i blocchi per ridurre progressivamente le dimensioni delle mappe di attivazione.
- Uno strato *fully connected* finale con 1000 nodi per la classificazione su 1000 categorie (come ImageNet), con attivazione *SoftMax*.

La ResNet-18 è una rete più semplice rispetto alla ResNet-50, ma rimane molto efficace, soprattutto per applicazioni dove è richiesto un buon compromesso tra accuratezza e velocità di esecuzione.

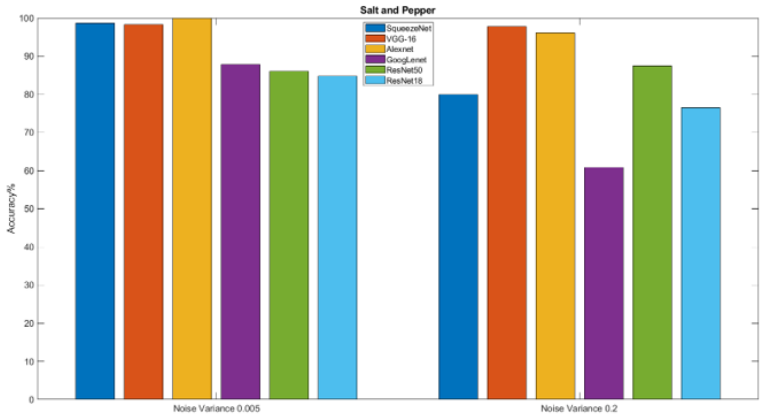



**Figure 2.15** Risultati in termini di accuracy per immagini affette da rumore sale e pepe

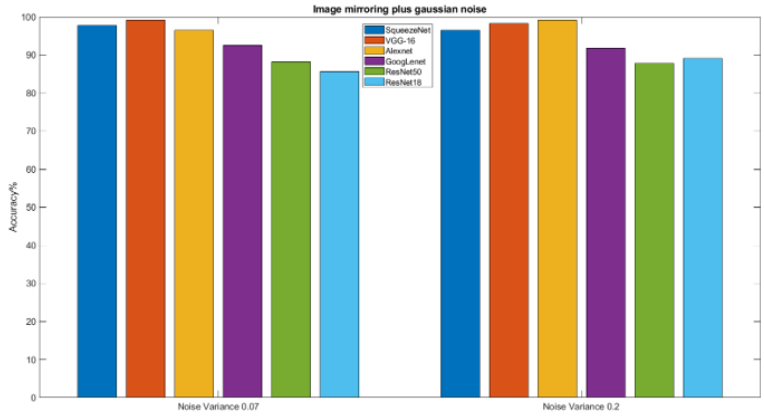



**Figure 2.16** Risultati in termini di accuracy per immagini affette da rumore gaussiano e immagini specchiate

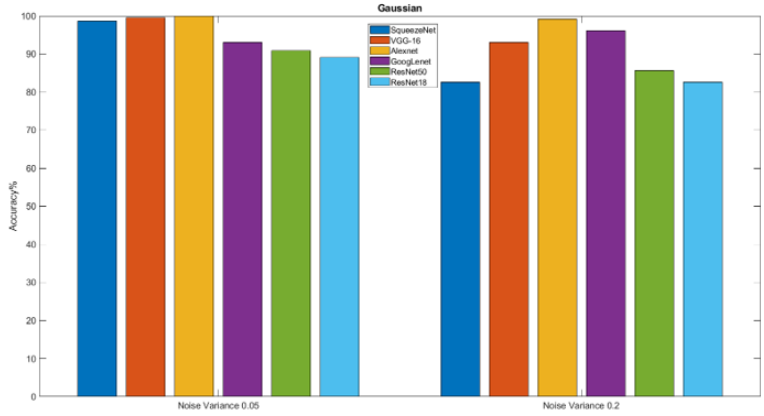



**Figure 2.17** Risultati in termini di accuracy per immagini affette da rumore gaussiano

Nel nostro dataset, abbiamo introdotto corruzione in tutte le immagini utilizzando due tipi distinti di rumore: *salt and pepper* e *rumore Gaussiano*. Il rumore Gaussiano, è stato utilizzato per simulare il rumore tipico delle comunicazioni. Al contrario, il rumore salt and pepper è stato impiegato come stress test per valutare la robustezza delle reti neurali in condizioni estreme.

Inoltre, alle immagini è stato applicato anche un *mirroring*, al fine di ampliare il dataset e rendere più completa la valutazione dell'efficacia degli approcci considerati.

Il dataset complessivo è composto da 800 immagini, utilizzate come segue: 160 immagini (pari al 20% del dataset) sono state impiegate per il test delle reti neurali, mentre 640 immagini, sono state utilizzate nella fase di addestramento.

L'ottimizzazione è stata effettuata tramite l'algoritmo stochastic gradient descent with momentum (SGDM), utilizzando 50 epoche, un batch size di 50 e un learning rate pari a 0.0003.

Nelle figure 2.15, 2.16 e 2.17 mostriamo i risultati in termini di accuratezza delle reti neurali, al variare della varianza del rumore: tra 0.005 e 0.2 per il rumore salt and pepper, e tra 0.05 e 0.2 per il rumore Gaussiano. L'accuracy è stata calcolata nel contesto di una classificazione multiclasse, comprendente le quattro categorie di messaggi nascosti. Essa rappresenta quindi il rapporto tra il numero totale di etichette correttamente predette e il numero complessivo di campioni.

I risultati ottenuti sono generalmente buoni per tutte le reti e per entrambi i tipi di rumore, con valori di accuratezza che oscillano tra il 75% e il 99%. Tuttavia, si osserva un calo significativo delle prestazioni per la rete GoogleNet , la cui accuratezza scende drasticamente fino al 60%.

Le reti che hanno ottenuto le migliori prestazioni si confermano essere le reti convoluzionali, in particolare VGG-16 e AlexNet, con valori che superano il 90% anche in presenza di forti variazioni del rumore. In generale, la robustezza della VGG-16, dovuta ai suoi filtri convoluzionali più piccoli, e quella della AlexNet, legata alle tecniche di DropOut adottate, rendono queste architetture particolarmente efficaci nel campo della computer vision, con prestazioni influenzate positivamente sia dalla struttura interna sia dalla configurazione degli iperparametri utilizzata.

Le tecniche sperimentate in questo capitolo, pongono le basi per le successive applicazioni in ambito radar. I principi sviluppati saranno infatti adattati e riutilizzati come prime sperimentazioni nell'ambito dell'elaborazione di segnali radar e della classificazione di oggetti spaziali, dove l'analisi di spettrogrammi e mappe range-Doppler richiede strategie analoghe per l'individuazione di pattern rilevanti.

# 3 Metodi di intelligenza artificiale per il monitoraggio dei detriti spaziali

## 3.1 Introduzione

L'esplorazione e l'utilizzo dell'orbita terrestre sono entrati in una nuova fase, spesso definita New Space Economy. La crescente accessibilità allo spazio ha condotto a un incremento esponenziale del numero di satelliti operativi, accompagnato da una proliferazione di detriti spaziali, al punto che oggi si parla apertamente di "sindrome di Kessler" [64] che descrive un fenomeno in cui la crescente densità di detriti spaziali porta a collisioni tra oggetti in orbita, generando ulteriori frammenti in un effetto palla di neve. Secondo i dati dell'Agenzia Spaziale Europea (ESA, 2024), oltre 36.500 oggetti superiori a 10 cm orbitano attorno alla Terra, senza contare milioni di frammenti più piccoli [65]. La capacità di monitorare, prevedere e mitigare i rischi associati a questa congestione orbitale è oggi un elemento essenziale per garantire la sostenibilità delle operazioni spaziali [66]. Nonostante i progressi nei sistemi radar convenzionali, le soluzioni attuali risultano limitate nella rilevazione di oggetti di piccole dimensioni a bassa SNR. Questa tesi propone approcci AI per colmare tale gap.

In questo scenario, l'Intelligenza Artificiale si configura non solo come un'opportunità tecnologica, ma come una necessità strategica [67]. Dalla predizione delle collisioni all'automazione delle decisioni operative, l'AI promette di superare i limiti dei metodi tradizionali, offrendo maggiore reattività, scalabilità e robustezza [68]. Tuttavia, l'impiego crescente di sistemi autonomi nello spazio pone anche nuove sfide etiche, giuridiche e di governance.

Guardando due immagini esemplificative, emerge chiaramente quanto il tema dell'attività spaziale sia oggi estremamente complesso, fatto tanto di incredibili opportunità quanto di rischi altrettanto concreti.

La prima figura 3.1 mostra un grafico che racconta l'evoluzione degli oggetti in orbita terrestre dagli anni '60 fino ai giorni nostri. All'inizio la crescita è lenta, ma poi, col passare dei decenni, esplode. Negli ultimi vent'anni, il numero di oggetti in orbita cresce a ritmi impressionanti, fino a superare le decine di migliaia. Non si tratta solo di satelliti funzionanti: la maggior parte degli oggetti in orbita sono frammenti, detriti, resti di razzi e missioni passate, molti dei quali addirittura non identificati [69]. È un dato preoccupante: più aumentano gli oggetti, più cresce il rischio di collisioni, e con esso la possibilità di un effetto domino incontrollabile, innescando una cascata di frammenti che potrebbe rendere alcune orbite inutilizzabili per decenni [70].

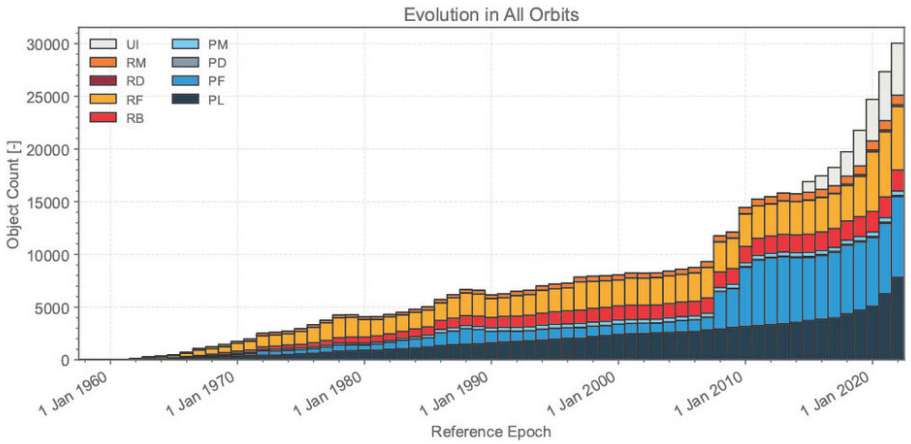



**Figure 3.1** Evoluzione degli oggetti in orbita terrestre [71]

La seconda figura 3.2 amplia ulteriormente la prospettiva: si confrontano le straordinarie opportunità offerte dallo spazio come innovazione tecnologica, osservazione della Terra, telecomunicazioni, esplorazione scientifica con i rischi insiti nello sviluppo accelerato del settore spaziale. Lo spazio è una risorsa straordinaria: consente di proteggere il nostro pianeta, migliorare la vita quotidiana tramite sistemi GPS, comunicazioni globali e monitoraggi climatici avanzati [72]. Tuttavia, la crescente accessibilità ha trasformato l'ambiente orbitale in un territorio sempre più congestionato e competitivo.

Non si tratta solo di rischi tecnici, malfunzionamenti, errori di progettazione, gestione inadeguata dei detriti ma anche di problemi più sottili: la mancanza di regolamentazioni internazionali condivise, l'assenza di un quadro chiaro di governance spaziale, e la difficoltà di attribuire responsabilità in caso di incidenti [74]. In questo quadro, l'impiego di intelligenza artificiale aggiunge ulteriori livelli di complessità: sistemi autonomi in orbita saranno sempre più capaci di prendere decisioni in frazioni di secondo, spesso lontani dal diretto

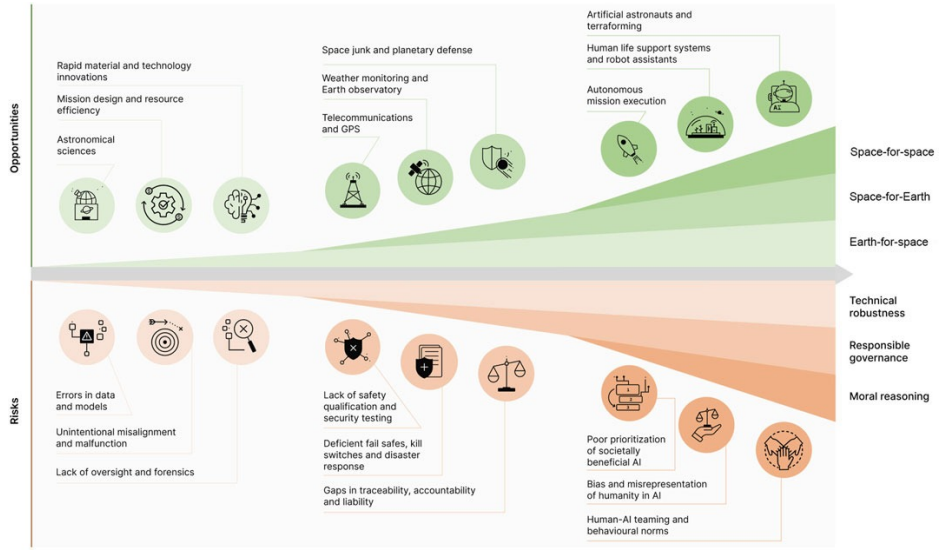



**Figure 3.2** Analisi delle opportunità e dei rischi nella SSA [73]

controllo umano. Se non si stabiliscono limiti etici chiari e criteri trasparenti di responsabilità, rischiamo che le scelte vengano affidate a macchine senza garanzie di equità o sicurezza [75].

Osservando questi sviluppi, si comprende che la conquista dello spazio non rappresenta un semplice percorso di progresso lineare. È, invece, una sfida continua tra la volontà di spingersi oltre e la necessità di farlo in modo responsabile.

Un'analisi quantitativa delle applicazioni dell'AI nello spazio è proposta da Emergen Research (2024), che suddivide il mercato per settori applicativi.

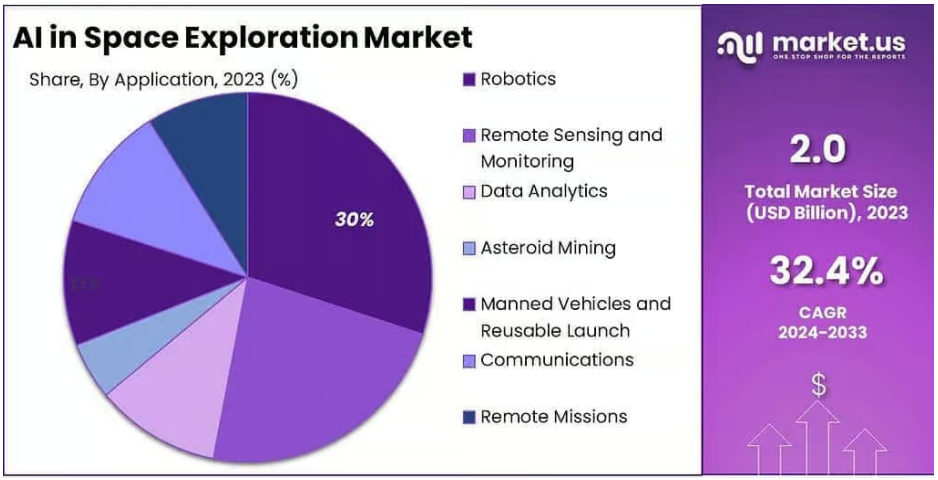



**Figure 3.3** Distribuzione del mercato dell'Intelligenza Artificiale nelle applicazioni spaziali per settore (2023). Fonte: Emergen Research (2024).

[76]

Come illustrato nella Figura 3.3, il comparto della robotica spaziale rappresenta circa il 30% del mercato. Ciò riflette l'importanza attribuita alle tecnologie autonome capaci di svolgere operazioni complesse senza intervento umano diretto, come la manutenzione in orbita o l'assemblaggio modulare.

Il *Remote Sensing* (27%) emerge come il secondo settore, evidenziando il ruolo chiave dell'AI nell'elaborazione rapida di immagini satellitari per applicazioni terrestri e spaziali. Allo stesso modo, il *Data Analytics* (23%) assume crescente importanza nella gestione di enormi quantità di dati di sorveglianza, mentre l'*Asteroid Mining* (15%) rappresenta un settore emergente, ancora in fase di esplorazione tecnologica.

Va osservato criticamente che la suddivisione proposta presenta alcune limitazioni: ad esempio, le categorie non sono sempre mutuamente esclusive e possono sovrapporsi (es. robotica e remote sensing integrati in missioni autonome), suggerendo che il mercato reale potrebbe essere più complesso di quanto rappresentato.

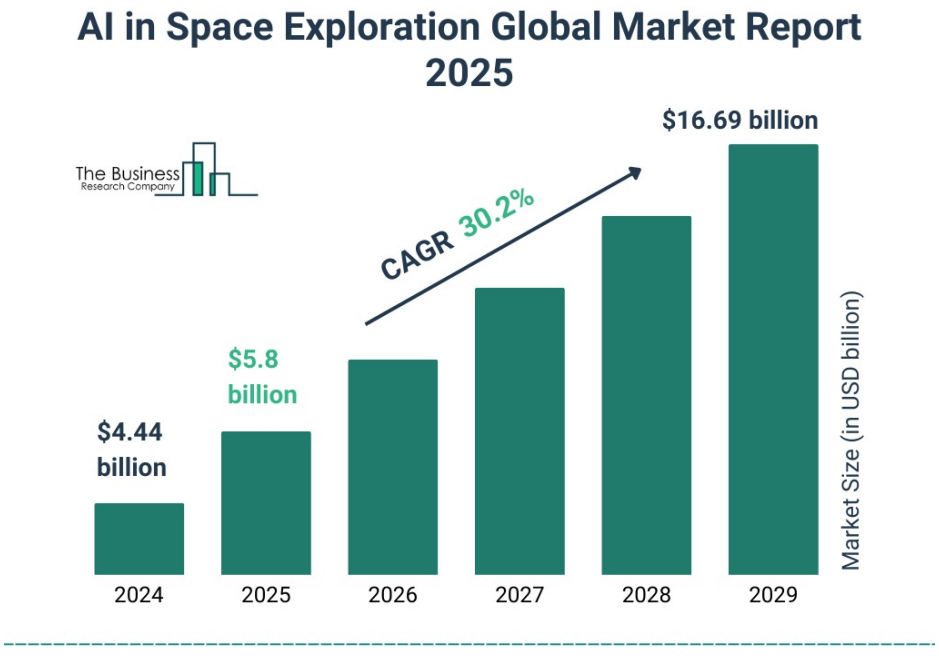



**Figure 3.4** Andamento futuro del mercato AI
[77]

La Figura 3.4 evidenzia una crescita estremamente rapida, con un CAGR del 30,2% e un'espansione da 4,44 miliardi USD nel 2024 a 16,69 miliardi USD nel 2029. Questo trend suggerisce che l'adozione di tecnologie AI sarà probabilmente un fattore critico di successo per gli operatori spaziali, sia governativi che privati.

Tuttavia, tale evoluzione comporta anche rischi non trascurabili. L'automazione crescente, se non accompagnata da adeguati sistemi di validazione e certificazione, potrebbe introdurre vulnerabilità nuove, come decisioni subottimali in scenari complessi o, peggio, reazioni imprevedibili a eventi anomali. Inoltre, la crescente dipendenza da modelli AI pone sfide anche in termini di cyber-sicurezza orbitale, tema ancora oggi relativamente poco regolamentato a livello internazionale .

## 3.2 Elaborazione di segnali radar per il rilevamento di detriti

I radar svolgono un ruolo cruciale nella rilevazione e nel tracciamento dei detriti spaziali. I sistemi radar si basano sul semplice principio dell'eco: un segnale viene inviato verso un bersaglio specifico, che ne riflette una parte, e l'eco ritorna alla sorgente. I sistemi radar possono determinare la distanza, la posizione e la velocità di un oggetto, rimbalzando onde radio contro questi oggetti e misurando il tempo impiegato dalle onde per ritornare. Queste informazioni sono fondamentali per mantenere la sicurezza e la sostenibilità delle operazioni spaziali, poiché permettono agli operatori di monitorare e prevedere il comportamento degli oggetti nello spazio, inclusi potenziali collisioni [78].

Esistono diversi tipi di sistemi radar che possono essere utilizzati per il rilevamento e il tracciamento dei detriti spaziali, tra cui:

1. **Phased Array Radar:** questo tipo di radar utilizza più antenne che possono orientare il fascio in diverse direzioni senza la necessità di muovere fisicamente l'antenna stessa. Ciò consente una scansione più rapida del cielo e il rilevamento di oggetti in più direzioni simultaneamente.
2. **Radar Monostatico e Bistatico:** utilizza un'unica antenna sia per la trasmissione sia per la ricezione dei segnali, mentre il radar bistatico utilizza antenne separate. Il radar bistatico può offrire prestazioni migliori in termini di rapporto segnale-rumore (SNR) e sensibilità, ma richiede una maggiore quantità di hardware.
3. **Radar Multistatico:** coinvolge più antenne sia per la trasmissione che per la ricezione dei segnali. Le antenne sono distribuite in posizioni diverse, creando una rete di nodi radar che lavorano insieme per rilevare e tracciare i bersagli. Il radar multistatico offre una copertura migliorata, una maggiore ridondanza del sistema e migliori capacità di rilevamento dei bersagli.

4. **Radar ad Apertura Sintetica (SAR):** utilizza una grande antenna o più antenne combinate per creare un'antenna virtuale molto più grande di quella fisica. Questo consente di ottenere immagini ad alta risoluzione degli oggetti nello spazio. I sistemi SAR utilizzano orbite flessibili, frequente acquisizione di dati e regolazioni dell'effetto Doppler per inseguire bersagli estremamente manovrabili. Alcuni sistemi impiegano metodi sofisticati come l'interferometria multi-passaggio e il tracciamento di bersagli in movimento insieme a dispositivi di tracciamento per inseguire questi tipi di bersagli. L'identificazione accurata dei bersagli è migliorata grazie all'integrazione con sensori aggiuntivi, ed è essenziale l'elaborazione in tempo reale per applicazioni che richiedono decisioni rapide.

Per quanto riguarda la situazione europea, un importante sistema disponibile 24 ore su 24 per le osservazioni di sorveglianza è il Grand Réseau Adapté à la Veille Spatiale (GRAVES). Si tratta di un radar bistatico a onda continua militare, operante a 143,05 MHz, situato a Digione, in Francia [79].

In [80], viene presentato il sistema radar GESTRA (German Experimental Space Surveillance and Tracking Radar), un esempio significativo dell'uso della tecnologia radar a rete fase per la sorveglianza e il tracciamento spaziale. Sviluppato dal Fraunhofer Institute (FHR), è progettato per rilevare e tracciare oggetti in orbita terrestre bassa (LEO) ad altitudini comprese tra 300 e 3000 km.

I sistemi radar italiani sono continuamente in fase di aggiornamento, con l'obiettivo di migliorarne le capacità di tracciamento dei detriti spaziali in orbita bassa terrestre .

L'obiettivo dell'articolo [81] è fornire una descrizione dettagliata dei sistemi radar BIRALES (BIstatic RAdar for LEO Survey) e BIRALET (BIstatic RAdar for LEO Tracking), che rappresentano il contributo italiano al monitoraggio dei detriti spaziali in LEO, inclusi i componenti hardware e software.

Il lavoro [82] propone l'EISCAT 3D, un sistema radar di nuova generazione attualmente in fase di sviluppo nell'Europa artica e progettato per tecniche di misurazione avanzate. Il sistema sarà composto da tre siti situati nella Scandinavia settentrionale, con ciascun sito costituito da circa 10.000 antenne alimentate da un potente trasmettitore da 5 MW a Skibotn e da un ricevitore in ciascuno dei tre siti. EISCAT 3D è progettato per fornire imaging volumetrico, sintesi di apertura e imaging multistatico, tracciamento e esperimenti adattivi, insieme a operazioni continue. Questa versatilità unica consentirà il tracciamento di bersagli difficili come detriti spaziali, oggetti vicini alla Terra (NEO) ed echi meteorici in parallelo con esperimenti radar per risolvere questioni fondamentali riguardanti l'accoppiamento tra strati atmosferici, le interazioni Sole-Terra e la turbolenza del plasma.

Un esempio di radar SAR europeo si trova in [83]. COSMO-SkyMed è infatti una costellazione di quattro satelliti SAR: COSMO-SkyMed 1, 2, 3 e 4. Dotati di sensori SAR avanzati, questi satelliti sono in grado di catturare immagini ad alta risoluzione della superficie terrestre in qualsiasi condizione atmosferica. Sfruttando le sue capacità di imaging SAR, COSMO-SkyMed può catturare immagini degli oggetti di detriti spaziali e tracciare i loro movimenti nel tempo. Inoltre, l'imaging SAR ad alta risoluzione di COSMO-SkyMed assiste nell'identificazione e classificazione degli oggetti di detriti spaziali, contribuendo alla caratterizzazione della popolazione di detriti. Queste informazioni sono fondamentali per comprendere la distribuzione, la densità e il comportamento dei detriti spaziali in orbita.

L'E-SAR [84] è un sistema SAR aerotrasportato progettato specificamente per applicazioni avanzate di telerilevamento. Opera in diverse bande di frequenza, tra cui la banda X e la banda P, consentendo modalità di imaging e capacità flessibili. Il sistema è dotato di algoritmi di imaging avanzati e tecniche di elaborazione, consentendo l'acquisizione di dati SAR ad alta risoluzione su ampie aree. Per il monitoraggio dei detriti spaziali, E-SAR acquisisce immagini SAR della popolazione di detriti, consentendo ai ricercatori di tracciare le loro traiettorie e misurarne le caratteristiche per evitare collisioni.

Una tendenza emergente per la Consapevolezza della Situazione Spaziale (SSA) è rappresentata dai sensori radar spaziali (SBR), che completano la sorveglianza terrestre. Un confronto dei limiti e dei vincoli dei SBR e dei radar terrestri è presentato in [85] da una prospettiva ontologica.

Recentemente, in [86] definiscono l'architettura di un radar a impulsi Doppler basato su monopulse nella banda Ka, dotato di un'antenna a scansione elettronica attiva, elaborazione tipo CFAR e un tracciatore bayesiano per il rilevamento e il tracciamento dei detriti. Il proof-of-concept è stato esteso a configurazioni multiple-input-multiple-output (MIMO) al fine di migliorare le capacità di sorveglianza [87].

Uno dei principali vantaggi del radar per il tracciamento dei detriti spaziali è la capacità di operare in tutte le condizioni atmosferiche e a qualsiasi ora del giorno. A differenza di componenti come i telescopi ottici, che richiedono cieli sereni e luce solare, il radar può penetrare attraverso nuvole, polvere e detriti, rendendolo uno strumento prezioso per il monitoraggio della popolazione di detriti. Tuttavia, il radar presenta alcune limitazioni, tra cui la sensibilità alla dimensione e alla forma dell'oggetto tracciato, l'incapacità di distinguere tra oggetti vicini tra loro e gli effetti del mezzo plasma con scintillazione debole [88].

## 3.3 Tecniche di deep learning per la classificazione e il tracciamento di oggetti spaziali

L'automazione tramite l'Intelligenza Artificiale (AI) può risultare molto utile per ottimizzare l'enorme quantità di dati raccolti da missioni scientifiche come sonde spaziali, veicoli per l'osservazione della Terra e rover. Aiuta anche nella valutazione dei dati e nella diffusione dei risultati agli utenti finali. Impiegando l'AI a bordo delle navette spaziali, sarà possibile costruire dataset e mappe in modo completamente autonomo. La tecnologia AI non solo è capace di elaborare grandi quantità di informazioni, ma può anche ridurre o eliminare i dati ridondanti (ad esempio, mediante compressione), rendendo così le reti più efficienti. L'Intelligenza Artificiale permette anche una varietà di compiti di monitoraggio grazie all'uso pervasivo delle immagini satellitari nello spazio. Il Deep Learning (DL), come sottoinsieme dell'Intelligenza Artificiale, consente un controllo preciso e automatizzato, facilitando attività a bordo come il docking o la navigazione [89–91].

La ragione dietro il crescente entusiasmo per il DL risiede nelle limitazioni delle tradizionali tecniche di Machine Learning (ML) nell'affrontare le crescenti esigenze analitiche dei sistemi IoT. I modelli DL (soprattutto le reti neurali convoluzionali, CNN) sono superiori alle tecniche ML tradizionali sotto diversi aspetti: innanzitutto, rispetto alle tecniche ML, richiedono pochi o nessun passaggio di pre-elaborazione, eliminando il prerequisito di utilizzare dati etichettati per l'addestramento. In questo modo, le caratteristiche che potrebbero non essere identificabili da un essere umano possono essere estratte in modo efficiente con gli approcci DL. Inoltre, questi approcci superano le tecniche tradizionali in termini di accuratezza. Infine, le architetture DL sono adatte a modellare comportamenti complessi di dataset multimodali convenzionali. In sintesi, i modelli DL possono lavorare su qualsiasi tipo di dato, che sia strutturato, non strutturato o semi-strutturato [92].

Le reti neurali convoluzionali (CNN) hanno dimostrato la loro efficacia nelle aree di riconoscimento e classificazione delle immagini. Esse sono classificate come algoritmi di deep learning (DL), che utilizzano segnali digitali come audio, immagini e video come input. Hanno raggiunto risultati eccezionali nel campo della classificazione delle immagini. Le immagini attraversano strati convoluzionali discreti che mirano ad estrarre diversi aspetti o caratteristiche dell'immagine e ad assegnare pesi e bias per classificarle. Sono state implementate diverse soluzioni per il tracciamento e la classificazione degli oggetti. In [93], gli autori propongono un metodo per rilevare la salienza dei detriti spaziali basato su una rete completamente convoluzionale (FCN) per la piattaforma di sorveglianza spaziale. Allo stesso tempo, la rete apprende direttamente la relazione interna tra due fotogrammi, evitando così il costoso calcolo del flusso ottico. Tuttavia, questo metodo non è adatto per

il rilevamento di piccoli obiettivi a causa delle caratteristiche limitate estratte dai piccoli detriti spaziali. In [94], viene esplorato un nuovo approccio basato su una rete neurale profonda U-Net per la segmentazione delle immagini, finalizzata all'estrazione in tempo reale dei tracklet dalle acquisizioni ottiche. Immagini etichettate artificialmente e filmati sintetici del cielo notturno di oggetti che passano sopra una località specificata sono stati generati artificialmente, producendo le etichette corrispondenti: le stelle appaiono come punti nel campo visivo (FoV), mentre i detriti appaiono come strisce (di solito chiamate "tracklet"), risultando in una striscia luminosa di pixel nell'immagine. Dopo una fase di pre-elaborazione, le immagini reali sono state ottimizzate per la riduzione della vignettatura e l'uniformità della luminosità di fondo. Nel campo del riconoscimento e della classificazione degli oggetti, le reti neurali convoluzionali superano non solo le tecniche sopra menzionate, ma anche qualsiasi tecnica di machine learning, come dimostrato nel lavoro di [95]. Qui, gli autori ottengono la migliore accuratezza degli oggetti rilevati nell'immagine utilizzando telecamere di sorveglianza intelligenti. L'uso di modelli MCNNT (Modified Convolutional Neural Networks Techniques) a gerarchia nella modifica dei dati migliora le prestazioni della classificazione CNN, raggiungendo un'accuratezza del 96% rispetto ai modelli esistenti di Support Vector Machine (SVM).

Un nuovo tipo di rete convoluzionale viene presentato in [96]. PSNet è una rete sensibile alla prospettiva per rilevare oggetti da diverse angolazioni di vista. Le caratteristiche vengono mappate sugli spazi multi-prospettici preimpostati per ottenere la caratteristica semantica specifica dell'oggetto decoupled dall'angolo di vista. Grazie a PSNet, la separabilità complessiva delle funzionalità è migliorata, poiché gli oggetti tra le classi vengono proiettati su segmenti diversi, e gli oggetti tra le classi con angoli di vista differenti vengono proiettati sullo stesso segmento. Il lavoro di [97] propone una rete di segmentazione delle immagini delle navette spaziali end-to-end, utilizzando la rete di segmentazione semantica DeepLabv3+ come framework di base. Viene sviluppata una rete neurale multi-scala basata su convoluzioni dilatate (SISnet). L'innovazione di questo metodo è duplice: in primo luogo, la capacità di estrarre caratteristiche è migliorata dalla rete convoluzionale dilatata, e in secondo luogo, il meccanismo di attenzione sui canali è introdotto nella rete per ricalibrare le risposte funzionali. Infine, una struttura di pooling spaziale piramidale parallela (ASPP) è progettata per migliorare le informazioni contestuali della rete. Per verificare l'efficacia del metodo, sono stati condotti esperimenti su un dataset di segmentazione delle navette spaziali. I risultati sperimentali mostrano che la struttura encoder + attenzione + decoder proposta in questo lavoro può ottenere maschere chiare e complete degli obiettivi spaziali con alta accuratezza di segmentazione.

Il concetto di RNN è diverso da quello di CNN. Il livello nascosto di una RNN è legato al tempo. Per i dati delle serie temporali, l'input viene elaborato in modo sequenziale da diversi livelli. Originariamente, le RNN avevano il problema del gradiente che scompare, il che significa che il gradiente diventava estremamente piccolo con il numero di livelli di propagazione. Questo implica che i pesi della rete potrebbero non essere aggiornati durante il processo di apprendimento. Per risolvere questo problema, è stato proposto il Long-Short-Term Memory (LSTM) [98], che può essere visto come un miglioramento per risolvere il problema originale delle RNN. Sia le CNN che le RNN mostrano risultati promettenti rispetto all'algoritmo convenzionale k-NN-DTW (k-nearest neighbor combinato con Dynamic Time Warping). Anche le CNN bidimensionali mostrano risultati promettenti, fornendo un'alternativa per l'utilizzo diretto dei dati delle serie temporali.

L'algoritmo YOLO utilizza reti neurali per fornire rilevamento di oggetti in tempo reale, trattando il rilevamento come un problema di regressione. Grazie alla sua precisione e velocità, YOLO è utilizzato in molte applicazioni di rilevamento di oggetti. In [99], viene proposto un framework semplice per il riconoscimento di oggetti in movimento. Il framework consiste in due passaggi che vengono applicati sequenzialmente a ciascun fotogramma video. Nel primo passaggio, utilizzando una semplice sottrazione dello sfondo, vengono trovate le regioni di interesse (ROI) in un fotogramma video. Nel secondo passaggio, una rete YOLO (ovvero YOLOv2) [100] viene utilizzata per classificare le ROI rilevate in base a un insieme di criteri predefiniti. Il framework offre il vantaggio di riutilizzare la sottrazione dello sfondo. Infatti, il tempo di elaborazione reale dedicato esclusivamente al rilevamento degli oggetti nel framework è il tempo di inferenza della CNN. YOLO è anche utilizzato per il rilevamento di target di visione artificiale su veicoli aerei senza pilota (UAV). YOLOv3-Tiny ottimizza la struttura della rete YOLOv3 e riduce l'output di uno scale. In questo lavoro, le immagini umane sono utilizzate per addestrare YOLOv3-Tiny a riconoscere gli esseri umani come target di tracciamento. Successivamente, YOLOv3-Tiny analizza le immagini raccolte dall'UAV per il tracciamento dei target. Infine, il segnale di controllo del sistema di tracciamento viene inviato al controller di volo dell'UAV [101].

In [102], viene studiato un nuovo modello per il riconoscimento delle azioni zero-click, che cattura congiuntamente le relazioni tra gli oggetti di un fotogramma statico e modella i pattern temporali del movimento tra fotogrammi adiacenti. Il rilevatore di oggetti prima rileva ed estrae le caratteristiche dell'oggetto. Successivamente, vengono effettuate le convoluzioni del grafo per sfruttare efficacemente le relazioni tra gli oggetti. Il lavoro presentato in [103] fornisce un metodo per la pianificazione dell'uso dell'energia solare. Previsioni accurate della produzione sono essenziali per i progetti fotovoltaici al fine di ottenere una potenza stabile. Le previsioni tradizionali basate su serie temporali di osservazioni del suolo sono spesso

influenzate dal problema del cambiamento di fase dovuto alla considerazione incompleta degli impatti del movimento delle nuvole. Questo lavoro sviluppa un framework innovativo che integra le osservazioni terrestri e satellitari tramite deep learning (DL) per migliorare le previsioni della produzione fotovoltaica. Il framework si basa su un modello di previsione ibrido per tenere conto dell'impatto del movimento delle nuvole sui successivi cambiamenti nella radiazione solare, su un calcolatore geometrico per simulare la posizione relativa tra il Sole e i pannelli fotovoltaici, e su un stimatore fotovoltaico per modellare la performance dei moduli fotovoltaici durante le conversioni fotoelettriche.

Nel corso degli anni, sono state ideate soluzioni che non coinvolgono l'uso di una rete neurale convoluzionale. La maggior parte di questi lavori ha sfruttato le reti neurali profonde in framework di Apprendimento per Rinforzo (RL) applicati al dominio SSA. L'Apprendimento per Rinforzo è un'area dell'IA, in cui gli agenti sono addestrati a prendere decisioni o azioni interagendo con l'ambiente per massimizzare un ritorno [104]. Ad esempio, in [105], gli autori hanno simulato un telescopio terrestre controllabile che osserva satelliti in LEO in un ambiente RL (Double Deep Q-Learning). Una volta che il numero di satelliti osservati in un periodo è stato massimizzato e il numero di misurazioni riuscite è aumentato, è stato utilizzato un filtro di Kalman esteso per ridurre significativamente le incertezze delle misurazioni di posizione e velocità per i satelliti osservati. Questo lavoro forma la base di un framework basato su Apprendimento per Rinforzo che sarà esplorato in future ricerche nel campo della consapevolezza situazionale spaziale. Dispositivi come i Manipolatori Spaziali Fluttuanti (FFSM) sono sempre più utilizzati in varie attività spaziali, e il Tracciamento Attivo degli Oggetti (AOT) è la base di molte missioni spaziali. L'AOT dei sistemi FFSM presenta due sfide principali: la modellazione e il controllo dei sistemi FFSM e la pianificazione del movimento di tracciamento dei manipolatori spaziali. Per affrontare queste problematiche, il lavoro [106] ha presentato una strategia per il tracciamento attivo degli oggetti dei sistemi FFSM utilizzando l'Apprendimento per Rinforzo Profondo (DRL), l'Ottimizzazione della Politica Prossimale (PPO) e una rete neurale fuzzy per affrontare queste sfide (FNN). Il paradigma DRL offre una nuova prospettiva su come affrontare missioni spaziali impegnative e funziona bene in combinazione con approcci più convenzionali. Il processo di addestramento dell'algoritmo è stato completato in un ambiente di simulazione il più realistico possibile, dove l'algoritmo è stato poi applicato direttamente a un problema fisico reale. Il lavoro [107] fornisce i primi risultati per risolvere il problema dell'assegnazione dei sensori e della gestione dei sensori (SM) per la consapevolezza situazionale spaziale (SSA) utilizzando il metodo di vantaggio asincrono Actor-Critic (A3C), cioè un approccio di Apprendimento per Rinforzo (RL). Esso può imparare a prendere decisioni ottimali interagendo con l'ambiente. Il metodo A3C combina i benefici di entrambi i principali tipi di approcci RL—approcci

basati su politiche e basati su valori. L'approccio A3C ha anche il vantaggio di fornire stime più accurate del gradiente politico con meno rumore, il che può migliorare la convergenza. In questo studio, sono stati presentati i risultati preliminari per due casi di simulazione. Il primo caso ha coinvolto 100 SO, e il secondo caso ha coinvolto 300 SO. Tuttavia, i risultati necessitano di essere esplorati ulteriormente, poiché l'ottimizzazione della rete non è stata esplorata. Infine, concludiamo la sezione con [108], dove viene proposto un metodo di rilevamento veloce dei detriti spaziali con apprendimento basato su griglia. L'immagine viene divisa in griglie 14x14, quindi viene utilizzata la rete neurale veloce basata su griglia (FGBNN) per individuare la posizione dei detriti spaziali nelle griglie. Il metodo proposto dimostra eccellenti prestazioni e alta velocità di rilevamento. FGBNN può elaborare 430 immagini con una dimensione di immagine di 224x224 al secondo (2,3 ms/immagine) basato sul framework avanzato a passaggio singolo con pochi parametri. I risultati sperimentali mostrano che questo metodo, caratterizzato da un'alta percentuale di rilevamento (circa il 98,8% di accuratezza) e un basso carico computazionale, può aiutare a rilevare i detriti spaziali in applicazioni con ritardi brevi e basso consumo energetico. Alcuni di questi modelli descritti in questa sezione sono stati adattati e applicati nei capitoli successivi su dati reali e simulati radar

### 3.3.1 Caso Studio

In questa sotto-sezione, mostriamo i risultati preliminari e i confronti ottenuti applicando metodi di deep learning per le applicazioni di SSA (Space Situational Awareness) in un ambiente simulato. Sebbene il deep learning possa essere utilizzato per diversi compiti e in varie fasi di una tipica catena di elaborazione, nei nostri test ci concentreremo su un caso d'uso ben definito e paradigmatico, ovvero la rilevazione di piccoli oggetti in movimento in orbita bassa (LEO) a partire dai segnali radar. Il presente caso studio è stato realizzato partendo da una simulazione radar parametrizzata ad hoc. Per fare ciò, simuliamo la catena di elaborazione standard di un radar monostatico a impulsi Doppler, che rileva la velocità radiale dei target in movimento a determinate distanze. L'output del radar viene utilizzato come input per un'architettura di rete neurale che fornisce il numero di target rilevati, come mostrato nella figura 3.5.

Nel ricevitore, gli echi elettromagnetici riflessi da un target vengono separati nelle loro componenti I e Q (in-phase e quadrature) attraverso una demodulazione coerente. Come modello di propagazione, abbiamo adottato un modello di riflessione a due raggi dal suolo, mentre i target sono stati caratterizzati in termini di distanza (ovvero range) dal radar, velocità e Sezione Trasversale Radar (RCS). Una volta estratte le componenti I e Q, viene effettuato il campionamento digitale dopo aver applicato una finestratura del segnale per controllare i

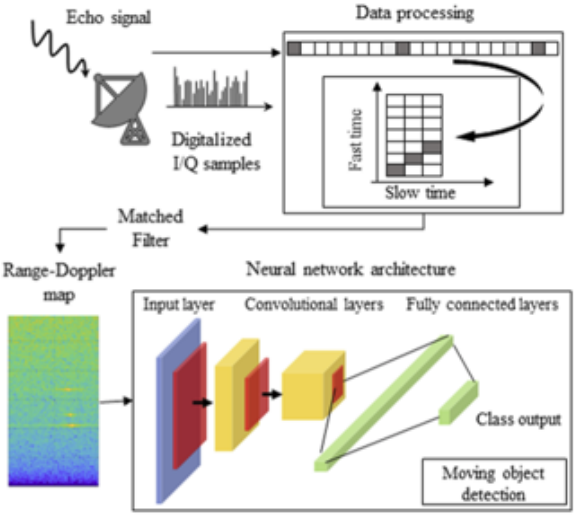



**Figure 3.5** Catena di elaborazione standard di un radar monostatico con rete neurale

lobi laterali generati dal campionamento. Successivamente, l'elaborazione del segnale pulse-Doppler separa i segnali riflessi in canali mediante un set di filtri per ogni range ambiguo. Il massimo range non ambiguo è legato all'inverso della frequenza di ripetizione degli impulsi (PRF) del radar.

Più nel dettaglio, i segnali I e Q vengono filtrati nel seguente modo: i campioni vengono organizzati in una matrice nel dominio del tempo; le colonne corrispondono ai campioni di range (tempo veloce), mentre le righe corrispondono agli intervalli di impulsi (tempo lento). Convoluta con il filtro matched tramite una trasformata di Fourier veloce, la matrice di output fornisce una stima della densità spettrale di potenza del segnale ritornato in funzione del range e della frequenza Doppler. Questa matrice è anche conosciuta come mappa range-Doppler. In una tipica catena di elaborazione radar pulse-Doppler, la stima della posizione e della velocità del target avviene mediante thresholding della mappa range-Doppler, individuando i bin di range e Doppler in cui l'energia supera una soglia predeterminata. Nei nostri esperimenti, queste mappe costituiscono l'input per le architetture di rete neurale che vogliamo confrontare. Le mappe range-Doppler vengono quindi utilizzate come input per i framework di deep learning che testiamo nel nostro confronto. Esaminiamo le reti neurali convoluzionali per verificarne il corretto funzionamento ed efficienza con input immagine 2D. Le reti selezionate sono SqueezeNet, VGG-16, AlexNet e GoogLeNet, per le quali abbiamo regolato parametri e livelli specifici per questo caso d'uso e delle quali abbiamo parlato nei precedenti capitoli.

Per svolgere i nostri esperimenti, abbiamo costruito un dataset strutturato come segue: le mappe range-Doppler sono state generate mentre la posizione e la velocità variavano, e a ciascuno spettro è stato assegnato uno dei quattro possibili etichette, ognuna delle quali rappresentava il numero di obiettivi rilevati (zero se non ci sono oggetti, uno se c'è un solo

obiettivo e così via). Il numero massimo di target considerati in ogni scenario simulato è pari a tre. Una volta generate le immagini, sono state fornite come input alle reti neurali per la classificazione. Le reti, quindi, classificano se l'immagine, quando la posizione dell'oggetto e la velocità con cui si muove variano, appartiene al primo, secondo, terzo o quarto caso menzionato. Le posizioni di ciascun obiettivo variano in un intervallo da 1 a 3000 m, mentre le velocità dei target sono state generate secondo una distribuzione uniforme, variando da 0 fino a una velocità massima di 225 m/s.

L'intero dataset è composto da 800 immagini, ed è stato utilizzato come segue: 640 immagini (80% del dataset) sono state utilizzate per la fase di addestramento, mentre 160 immagini (20% del dataset) sono state utilizzate per testare le reti. Inoltre, abbiamo utilizzato l'ottimizzatore "stochastic gradient descent with momentum" (SGDM), con un tasso di apprendimento di 0,0003, 50 epoche e una dimensione del batch di 50. Gli iperparametri sono stati scelti basandosi sulle esperienze e sulle migliori pratiche presenti nella letteratura. Nella fig 3.6 vengono riportati i risultati.

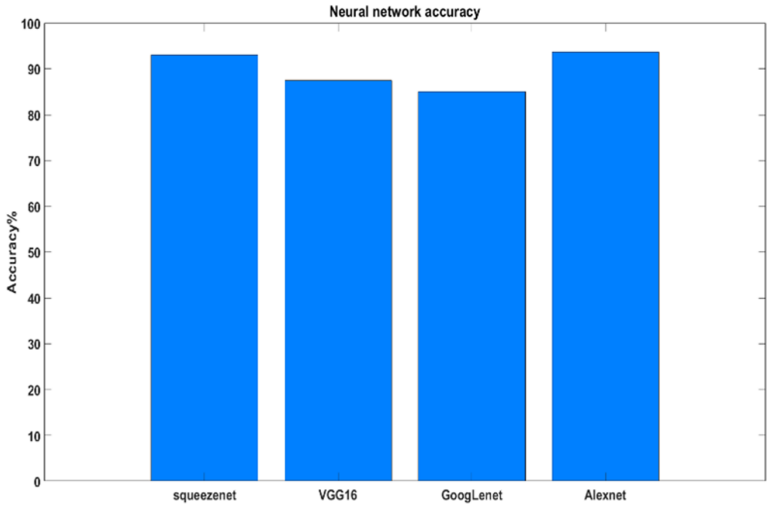



**Figure 3.6** Risultati della classificazione dei detriti in termini di accuracy

Per comprendere le prestazioni dei classificatori deep learning considerati, abbiamo utilizzato le seguenti metriche:

$$\text{Accuratezza} = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.1}$$

$$\text{Precisione} = \frac{TP}{TP+FP} \tag{3.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.3}$$

$$\text{F-Measure} = \frac{2 \times \text{Precisione} \times \text{Recall}}{\text{Precisione} + \text{Recall}} \tag{3.4}$$

I veri positivi (TP) sono il numero di campioni classificati correttamente come appartenenti alla loro classe vera, mentre i veri negativi (TN) sono il numero di campioni classificati correttamente come non appartenenti a una data classe. I falsi positivi (FP) sono il numero di campioni classificati erroneamente come provenienti da una data classe, ma che appartengono a un'altra classe. Al contrario, i falsi negativi (FN) sono il numero di campioni classificati come appartenenti a una classe che non corrisponde alla classe vera. L'accuratezza, come mostrato in 3.1, è una funzione di queste quantità e fornisce un modo intuitivo per determinare quale modello è migliore nel classificare i dati di input. Inoltre, più un modello è in grado di generalizzare i dati "non visti", migliori saranno le previsioni e le informazioni che può produrre. La precisione, 3.2, quantifica il numero di previsioni della classe positiva che appartengono effettivamente alla classe positiva, mentre il recall, come indicato in , è il numero di previsioni della classe positiva ottenute da tutti gli esempi positivi nel dataset. Il vantaggio di utilizzare precisione e recall è che possono descrivere correttamente le prestazioni della classificazione anche in presenza di dataset sbilanciati. Infine, la F-Measure in 3.4 fornisce un singolo punteggio che bilancia entrambe le preoccupazioni di precisione e recall in un unico numero.

I risultati ottenuti sono molto buoni, e tutte le reti oscillano attorno all'85–95%. Nella Tabella 3.1, mostriamo i risultati in termini di precisione, recall e F-measure per ciascuna rete considerata.

SqueezeNet è stata trovata essere la rete con le migliori prestazioni in termini di accuratezza complessiva. Oltre ad essere una rete "compressa", quindi con molti meno parametri, è stata anche la più veloce da allenare. La rete AlexNet segue in termini di prestazioni. Usare una CNN con meno strati ha il vantaggio di richiedere meno risorse hardware e tempi di allenamento più brevi rispetto a VGG-16 e GoogLeNet. Infatti, tempi di allenamento più brevi consentono di testare più iperparametri. Questo rende l'intero processo di allenamento più semplice. Una delle scelte progettuali chiave della rete VGG-16 è stata quella di utilizzare filtri convoluzionali più piccoli $(3 \times 3)$ rispetto alle reti precedenti. Questo consente un riconoscimento accurato delle immagini (con un'accuratezza complessiva dell'87,50%, come possiamo vedere nella Figura 9), a costo di aumentare drammaticamente la profondità della rete. Il modello VGG-16 è una rete estremamente pesante (anche grazie alla sua profondità e al numero di strati completamente connessi), caratterizzata dai tempi di allenamento più lenti.

La rete GoogLeNet, sebbene abbia raggiunto una buona accuratezza (85%), si è rivelata la meno performante, ma comunque più veloce di VGG-16 (che supera solo del 2% in accuratezza).

| Rete | Precisione | Recall | F-Measure |
|---|---|---|---|
| SqueezeNet | 0.92 | 0.93 | 0.92 |
| AlexNet | 0.89 | 0.90 | 0.89 |
| VGG-16 | 0.87 | 0.88 | 0.87 |
| GoogLeNet | 0.84 | 0.85 | 0.84 |

**Table 3.1** Risultati in termini di Precisione, Recall e F-Measure per ciascuna rete considerata.

## 3.4 Gap di ricerca e motivazione dei metodi proposti

Nonostante l'uso consolidato di tecniche radar tradizionali, permangono limiti nella capacità di rilevare e tracciare oggetti orbitali in condizioni di forte rumore, segnali deboli e ambiguità Doppler.

I primi esperimenti di classificazione, condotti in questo capitolo hanno mostrato risultati promettenti nel riconoscimento di pattern complessi, rafforzando l'idea che l'IA possa migliorare significativamente anche nei problemi di tracking.

Per questo motivo, nei capitoli successivi si proporrà un framework che confronta metodi tradizionali con modelli neurali avanzati, applicati a un ambiente radar simulato.

In particolare, i capitoli 4 e 5 proporranno uno scenario realistico di tracciamento basato su due simulazioni radar differenti, dove verranno confrontati approcci CFAR e modelli neurali quali YOLO e reti multi-head con Self-Attention che consentono di apprendere rappresentazioni temporali efficaci per il rilevamento di oggetti orbitali.

# 4 Metodi Proposti

In questa sezione descriviamo il metodo di riferimento adottato per dimostrare la fattibilità dell'utilizzo di tecniche di Deep Learning (DL) nel dominio della Space Situational Awareness (SSA). A partire dalla simulazione radar, abbiamo confrontato i nostri metodi basati su Deep Learning con l'algoritmo tradizionale CFAR (Constant False Alarm Rate).

## 4.1 Simulazione radar TIRA

Nel nostro ambiente simulato, abbiamo modellato un radar terrestre in grado di rilevare la presenza di piccoli oggetti (ossia target) in orbita LEO, a distanze comprese tra 300 e 2000 km. I target sono assunti come oggetti sferici metallici, caratterizzati da proprie distanze, velocità e Radar Cross Section (RCS). Abbiamo condotto i nostri esperimenti con oggetti aventi diametri compresi tra 0,5 e 8 cm.

La simulazione è stata progettata da zero, modellando uno scenario radar realistico ispirato al sistema TIRA (Tracking and Imaging Radar) e generando mappe Range-Doppler sintetiche per l'addestramento e la valutazione dei modelli di Intelligenza Artificiale. Il dataset risultante comprende tre sottoinsiemi principali. Il training set include 4.000 mappe: 1.000 mappe con 1 target, 1.000 mappe con 2 target, 1.000 mappe con 3 target e 1.000 mappe senza target, utilizzate per rappresentare lo sfondo. Il validation set comprende 600 mappe: 200 con 1 target, 200 con 2 target e 200 con 3 target.

Il test set è costituito da 1.000 mappe, ciascuna contenente un singolo target con un valore specifico di RCS, in modo da valutare in maniera controllata l'effetto della Radar Cross Section sulle prestazioni dei modelli.

I dati di input sono rappresentati come matrice numerica (.txt). Per verificare la robustezza dei modelli, le mappe sono state generate con diversi livelli di rumore additivo, sia gaussiano bianco (AWGN) sia rumore rosa, simulando condizioni operative realistiche.

Le corrispondenti RCS sono riportate nella Tabella 4.1 e sono ottenute tramite l'approssimazione con serie di Mie [109, 110] come segue:

| Diametro oggetto (cm) | RCS ($m^2$) |
|---|---|
| 0.5 | 0.000078 |
| 1 | 0.000314 |
| 2 | 0.001256 |
| 4 | 0.005026 |
| 8 | 0.020106 |

**Table 4.1** Radar Cross Section (RCS) in funzione del diametro dell'oggetto.

$$\sigma_{\text{rcs}} = \pi r^2 \left| \frac{1}{kr} \sum_{n=1}^{\infty} (-1)^n (2n+1) \left[ \frac{krJ_n(kr) - nJ_n(kr)}{krH_{n-1}^{(1)}(kr) - nH_n^{(1)}(kr)} - \frac{J_n(kr)}{H_n^{(1)}(kr)} \right] \right|^2 \tag{4.1}$$

Dove $r$ è il raggio della sfera, $k = \frac{2\pi}{\lambda}$ è il numero d'onda, $\lambda$ è la lunghezza d'onda, $J_n$ è la funzione di Bessel sferica del primo tipo di ordine $n$ e $H_n^{(1)}$ è la funzione di Hankel del primo tipo di ordine $n$.

Come sistema radar di riferimento, prendiamo in considerazione il TIRA (Tracking and Imaging Radar), un importante sistema radar europeo situato a Wachtberg, in Germania. Esso opera in due bande di frequenza: la banda L e la banda Ku, che permettono rispettivamente il tracciamento narrowband e l'imaging ad alta risoluzione.

Il trasmettitore in banda L emette impulsi ad alta frequenza con potenze di picco comprese tra 1 e 2 MW, utilizzando una polarizzazione circolare destrorsa. In quanto radar monostatico, il TIRA è in grado di rilevare oggetti di dimensioni fino a 2 cm a una distanza di 1000 km.

Il sistema TIRA è composto da tre sottosistemi principali, che ne garantiscono prestazioni eccellenti:

- Il primo è un'antenna parabolica da 34 metri, montata su una base azimut-elevazione controllata da computer, che consente il posizionamento preciso del radar.
- Il secondo sottosistema è costituito da un radar di tracciamento in banda L ad alta potenza con tecnologia monopulse a quattro corni. Questo radar opera a una frequenza di 1,33 GHz e consente un tracciamento accurato dei target.
- Il terzo sottosistema è un radar di imaging in banda Ku a larga banda, che opera a una frequenza di 16,7 GHz e genera immagini ad alta risoluzione di tipo Inverse Synthetic Aperture Radar (ISAR). Con una risoluzione massima di 6,3 cm, permette l'imaging dettagliato degli oggetti.

Il fascio radar è caratterizzato da una stretta larghezza a 3 dB pari a 0,031°, migliorando così la precisione nel rilevamento delle immagini.

Per soddisfare i requisiti di tracciamento degli oggetti spaziali (SO), che spesso viaggiano a elevate velocità (circa 8 km/s) in orbita LEO, il sistema di movimento dell'antenna TIRA è stato progettato per essere estremamente reattivo.

Con una massa mobile di circa 240 tonnellate, il sistema è in grado di eseguire movimenti rapidi e precisi, raggiungendo una velocità angolare massima di 24°/s in azimut e 6°/s in elevazione, con un'accelerazione angolare massima di 6°/s$^2$ in azimut e 1,5°/s$^2$ in elevazione. Queste caratteristiche dinamiche assicurano un tracciamento efficace dei target senza perdita del segnale, anche durante i loro movimenti rapidi nelle direzioni di azimut ed elevazione[111, 112]. Nella tabella 4.2 abbiamo i parametri utilizzati nel nostro codice. Il simultatore è stato implementato in linguaggio Matlab del quale andiamo ad illustrare e descrivere lo pseudocodice.

| **Parametro** | **Valore** |
|---|---|
| Velocità di propagazione | 100 m/s (*) |
| Numero di impulsi integrati | 5 kHz (*) |
| Frequenza di campionamento | 3000 km |
| Raggio massimo | 1.33 GHz (L-band) |
| Frequenza operativa | 1.5 × 10 W |
| Potenza di picco | 49.7 dB |
| Guadagno dell'antenna | 6 |
| Larghezza dell'impulso | 1 millisecondo |
| Forma dell'impulso | Rettangolare |
| PRF | 25 Hz |
| Frequenza dell'antenna | 1–2 GHz |

**Table 4.2** Parametri del radar TIRA

```
% Antenna
antenna = phased.IsotropicAntennaElement('FrequencyRange',
frequency_range);

% Posizione dell'antenna
txloc = [0; 0; 0];
antennaplatform = phased.Platform('InitialPosition', txloc);

% Canale di Propagazione
channel = phased.FreeSpace('PropagationSpeed', c, ...
    'MaximumDistanceSource', 'Property', ...
```

```
    'MaximumDistance', max_distance, ...
    'OperatingFrequency', carrier, 'TwoWayPropagation',
    true, 'SampleRate', sample_rate);

% Collector (Sensore di ricezione)
collector = phased.Collector('PropagationSpeed', c, ...
    'OperatingFrequency', carrier, 'Sensor', antenna);

% Radiatore (Sensore di trasmissione)
radiator = phased.Radiator('PropagationSpeed', c, ...
    'OperatingFrequency', carrier, 'Sensor', antenna);

% Sensore di Movimento
sensormotion = phased.Platform("MotionModel", "Velocity", ...
    "InitialPosition", [0; 0; 0], ...
    "VelocitySource", "Property", ...
    "Velocity", [0; 0; 0], ...
    "ScanMode", "None", ...
    "InitialOrientationAxes", eye(3), ...
    "OrientationAxesOutputPort", false);

% Trasmettitore
transmitter = phased.Transmitter('PeakPower', peak_power, ...
    'Gain', antenna_gain, 'InUseOutputPort', true);

% Onda (Forma d'onda rettangolare)
waveform = phased.RectangularWaveform('SampleRate', sample_rate, ...
    'PulseWidth', pulse_width, 'OutputFormat', 'Pulses', ...
    'PRF', prf, 'NumPulses', 1);

% Ricevitore
receiver = phased.ReceiverPreamp('NoiseFigure', noise_figure, ...
    'EnableInputPort', true, 'SeedSource', 'Property',
    'Seed', 2009, ...
    'SampleRate', sample_rate, 'Gain', antenna_gain);

% Griglie di tempo
fast_time_grid = unigrid(0, 1/sample_rate, 1/prf, '[)');
slow_time_grid = (0:num_pulse_int-1)/prf;
```

```
51
52 % Pre-allocazione dell'array
53 rxpulses = zeros(numel(fast_time_grid), num_pulse_int);
```

Il radar è stato implementato nel seguente modo:

1. **Antenna**
   Viene definita un'antenna isotropica, utilizzando la funzione `phased.IsotropicAntennaElement`. L'antenna è configurata per operare su un intervallo di frequenze definito dalla variabile `frequency_range`.
2. **Posizione dell'antenna**
   Viene definita la posizione iniziale dell'antenna come il vettore `txloc = [0; 0; 0]`, che posiziona l'antenna all'origine del sistema di coordinate. Inoltre, viene creata una piattaforma che rappresenta l'antenna utilizzando la funzione `phased.Platform`. La piattaforma è inizializzata con la posizione `txloc`.
3. **Canale di Propagazione**
   Qui viene definito un canale di propagazione libero (senza ostacoli) utilizzando la funzione `phased.FreeSpace`. In questa definizione, vengono specificati vari parametri come:
   - La velocità di propagazione (`PropagationSpeed`, definita dalla variabile `c`).
   - La distanza massima di propagazione (`MaximumDistance`), che è definita dalla variabile `max_distance`.
   - La frequenza operativa del canale (`OperatingFrequency`, definita dalla variabile `carrier`).
   - Il campionamento (`SampleRate`), che definisce la frequenza di campionamento dei segnali.
4. **Collector (Sensore di ricezione)**
   Viene creato un oggetto `phased.Collector`, che rappresenta il sensore di ricezione. La funzione è configurata per operare con la stessa velocità di propagazione e frequenza del canale di propagazione. Inoltre, viene associato al sensore l'antenna definita precedentemente.
5. **Radiatore (Sensore di trasmissione)**
   La definizione del radiatore è simile a quella del Collector, ma in questo caso l'oggetto `phased.Radiator` è utilizzato per rappresentare il sensore di trasmissione. Anche in questo caso, la velocità di propagazione e la frequenza operativa sono configurate, e l'antenna è associata al radiatore.

6. **Sensore di Movimento**
   Viene definito un sensore di movimento tramite l'oggetto `phased.Platform`. Questo sensore è configurato per seguire un modello di movimento basato sulla velocità, con una posizione iniziale definita da `[0; 0; 0]` (all'origine del sistema di coordinate) e una velocità iniziale di `[0; 0; 0]`. Inoltre, viene specificato che la piattaforma non ha una modalità di scansione (`ScanMode`, `"None"`).

7. **Trasmettitore**
   Il trasmettitore è definito con l'oggetto `phased.Transmitter`. In questa configurazione, vengono definiti:
   - La potenza di picco (`PeakPower`), che rappresenta la potenza del segnale trasmesso.
   - Il guadagno dell'antenna (`Gain`), che determina l'efficienza di trasmissione dell'antenna.
   - La porta di uscita per indicare quando il trasmettitore è in uso (`InUseOutputPort`).

8. **Onda (Forma d'onda rettangolare)**
   Viene definita una forma d'onda rettangolare tramite l'oggetto `phased.RectangularWaveform`. In questa configurazione:
   - La frequenza di campionamento (`SampleRate`).
   - La durata dell'impulso (`PulseWidth`).
   - Il formato di uscita come "Pulses".
   - La frequenza di ripetizione degli impulsi (`PRF`), che definisce la distanza tra gli impulsi.
   - Il numero di impulsi (`NumPulses`), impostato a 1 in questo caso.

9. **Ricevitore**
   Il ricevitore viene creato con l'oggetto `phased.ReceiverPreamp`, che simula un ricevitore radar con amplificazione del segnale. I parametri configurati sono:
   - Il numero di figure di rumore (`NoiseFigure`), che rappresenta l'aggiunta di rumore nel sistema.
   - Il guadagno dell'antenna per il ricevitore (`Gain`).
   - La porta di abilitazione per il ricevitore (`EnableInputPort`), che consente di attivare o disattivare il ricevitore.

10. **Griglie di tempo**
    Due griglie di tempo vengono definite per il calcolo della simulazione:

- La griglia di tempo veloce (`fast_time_grid`), che dipende dalla frequenza di campionamento e dalla frequenza di ripetizione degli impulsi.
- La griglia di tempo lenta (`slow_time_grid`), che è utilizzata per definire il numero di impulsi.

11. **Pre-allocazione dell'array**
    Infine, viene pre-allocato un array (`rxpulses`) per immagazzinare i dati dei segnali ricevuti. L'array è dimensionato in base alla griglia di tempo veloce e al numero di impulsi.

Successivamente siamo andati ad implementare il ciclo `while` per generare al massimo un numero di mappe, definite dalla variabile `N_maps`. Ogni iterazione del ciclo genera informazioni relative ai target, come la loro Radar Cross Section (RCS), posizione e velocità. Inizialmente, vengono creati tre vettori vuoti per memorizzare i dati relativi ai target: `tgtrcs` per l'RCS, `tgtloc` per la posizione e `tgtvel` per la velocità.

Successivamente, all'interno di un ciclo `for` che itera per il numero di target (`N_targets`), vengono generati i dati relativi a ciascun target. Per ogni target, il codice esegue una permutazione casuale dell'elenco di RCS, selezionando il primo valore disponibile. Allo stesso modo, vengono generate posizioni e velocità casuali per ciascun target, con le coordinate generate all'interno di un intervallo definito.

Una volta che i dati relativi ai target sono stati generati, viene creato un oggetto `RadarTarget` che rappresenta il target radar, configurato con un modello senza fluttuazioni. La frequenza operativa è definita dalla variabile `carrier`, mentre l'RCS è il valore generato precedentemente.

In seguito, viene creata la piattaforma del target, un oggetto `Platform` che tiene traccia della posizione e della velocità iniziale del target. Infine, la distanza e l'angolo tra la posizione del target e quella dell'antenna vengono calcolati utilizzando la funzione `rangeangle`.

Il ciclo `while` continua fino a quando non vengono generate tutte le mappe richieste. In ogni iterazione, vengono aggiornati i dati relativi ai target, come la posizione e la velocità, e viene calcolata la distanza e l'angolo rispetto alla piattaforma dell'antenna.

```
% Ciclo while per generare al massimo N_maps
i = 1

while i <= N_maps do
    % Inizializzazione delle variabili per i target
    tgtrcs = []   % RCS (Radar Cross Section) del target
    tgtloc = []   % Posizione del target
    tgtvel = []   % Velocita del target
```

```
9
10      % Ciclo per la generazione di N_targets
11      for j = 1 to N_targets do
12          % Generazione di un RCS casuale da un set discreto
13          RCSs = random permutation of RCSs
14          tgtrcs = append(tgtrcs, RCSs[1])
% Selezione di un RCS casuale
15          tgtloc = append(tgtloc, [random(min_x, max_x),
16          random(min_y, max_y), random(min_y, max_y)])
17          % Posizione casuale
18          tgtvel = append(tgtvel,
19          [random(0, max_target_velocity),
20
21          random(0, max_target_velocity),
22          random(0, max_target_velocity)]) % Velocita casuale
23
24      end for
25
26      % Creazione del target radar
27      target = create RadarTarget with Model = 'Nonfluctuating',
28      MeanRCS = tgtrcs, OperatingFrequency = carrier
29
30
31      % Creazione della piattaforma del target
32      targetplatform = create Platform
33      with InitialPosition = tgtloc, Velocity = tgtvel
34
35      % Calcolo della distanza e dell'angolo tra il target e l'antenna
36      [tgtrng, tgtang] = calculate rangeangle
37      (targetplatform.InitialPosition,antennaplatform.InitialPosition)
38
39
40  end while
```

```
1  % Ciclo for per l'integrazione dei pulesi
2  for m = 1 to num_pulse_int do
3      % Aggiornamento delle posizioni del sensore e del target
4      [sensorpos, sensorvel] = sensormotion(1/prf)
5      [tgtloc, tgtvel] = targetplatform(1/prf)
```

```
6
7       % Calcolo degli angoli del target visti dal sensore
8       [tgtrng, tgtang] = rangeangle(tgtloc, sensorpos)
9
10      % Simulazione della propagazione del pulesi in direzione del target
11      pulse = waveform()
12      [txsig, txstatus] = transmitter(pulse)
13      txsig = radiator(txsig, tgtang)
14      txsig = channel(txsig, sensorpos, tgtloc, sensorvel, tgtvel)
15
16      % Riflesso del pulesi dai target
17      tgtsig = target(txsig)
18
19      % Ricezione dei ritorni del target al sensore
20      rxsig = collector(tgtsig, tgtang)
21      rxpulses(:,m) = receiver(rxsig, ~(txstatus > 0))
22  end for
23
24  % Applicazione del filtro abbinato
25  matchingcoeff = getMatchedFilter(waveform)
26  matchedfilter = phased.MatchedFilter(
27      'Coefficients', matchingcoeff,
28      'GainOutputPort', true)
29  [rxpulses, mfgain] = matchedfilter(rxpulses)
30
31  % Compensazione del ritardo del filtro abbinato
32  matchingdelay = size(matchingcoeff,1) - 1
33  rxpulses = buffer(rxpulses(matchingdelay + 1:end), size(rxpulses,1))
34
35  % Applicazione del guadagno variabile nel tempo
36  %per compensare la perdita dipendente dalla distanza
37  range_gates = c * fast_time_grid / 2
38  lambda = c / carrier
39
40  tvg = phased.TimeVaryingGain(
41      'RangeLoss', 2 * fspl(range_gates, lambda),
42      'ReferenceLoss', 2 * fspl(c / (prf * 2), lambda))
43
44  % rxpulses = tvg(rxpulses)
```

```
45
46 % Analisi Range-Doppler
47 rangedoppler = phased.RangeDopplerResponse(
48     'RangeMethod', 'Matched Filter',
49     'SampleRate', sample_rate,
50     'PropagationSpeed', c,
51     'DopplerOutput', 'Speed',
52     'OperatingFrequency', carrier)
53
54 plotResponse(rangedoppler, rxpulses, matchingcoeff)
55
56 % Calcolo della posizione del target nella mappa Range-Doppler
57 Rvel = radialspeed(tgtloc, tgtvel)
58
59 for id = 1 to length(Rvel) do
60     if Rvel(id) >= 0 then
61         if abs(Rvel(id)) < (velMax / 2) then
62             rdvel(id) = Rvel(id)
63         else
64             rdvel(id) = -((velMax / 2) - mod(Rvel(id), (velMax/2)))
65         end if
66     else
67         if abs(Rvel(id)) < (velMax / 2) then
68             rdvel(id) = Rvel(id)
69         else
70             rdvel(id) = mod(Rvel(id), (velMax / 2))
71         end if
72     end if
73 end for
74
75 % Plot della risposta Range-Doppler
76 h1 = plotResponse(rangedoppler, rxpulses, matchingcoeff)
77
78 % Calcolo della risposta Range-Doppler
79 [resp, rng_grid, dop_grid] = rangedoppler(rxpulses, matchingcoeff)
80
81 % Impostazione dei valori di risposta uguali
82 %a zero a un valore molto piccolo (eps)
83 resp(resp == 0) = eps
```

```
84
85 % Conversione della risposta in decibel
86 rdmap = mag2db(abs(resp))
87
88 % Salvataggio dei dati
89 % save('Dati300Km.mat', 'dop_grid', 'rng_grid')
```

Il codice prosegue con un ciclo `for` che integra gli impulsi radar per un numero definito di iterazioni, indicato da `num_pulse_int`. Per ogni impulso, vengono aggiornate le posizioni e le velocità sia del sensore che del target. In seguito, vengono calcolati gli angoli del target rispetto al sensore utilizzando la funzione `rangeangle`.

L'impulso viene poi simulato in direzione del target, attraverso una serie di passaggi: la generazione del pulesi (`waveform`), la trasmissione del segnale (`transmitter`), la radiazione del segnale nella direzione del target (`radiator`), e la simulazione del canale (`channel`) che tiene conto delle condizioni relative al sensore e al target.

Successivamente, il segnale viene riflesso dai target, e i ritorni vengono raccolti dal sensore (`collector`). Il segnale ricevuto viene quindi memorizzato nella matrice `rxpulses`.

Il passo successivo applica il filtro abbinato al segnale ricevuto, usando i coefficienti generati dalla funzione `getMatchedFilter`. Viene anche compensato il ritardo introdotto dal filtro abbinato. Successivamente, viene applicato un guadagno variabile nel tempo per compensare la perdita di segnale dovuta alla distanza dal sensore.

Viene poi calcolata la risposta Range-Doppler utilizzando la funzione `RangeDopplerResponse`, e il risultato viene visualizzato attraverso la funzione `plotResponse`.

Nel calcolo della velocità radiale, il codice normalizza la velocità a una certa gamma massima, `velMax`, adattandola in base alla velocità misurata, e corregge il segno della velocità in base alla direzione del target. In seguito, vengono calcolati i valori di risposta nella mappa Range-Doppler e convertiti in decibel.

```
%% index within the range-Doppler maps

% save png
saveas(h1, sprintf([png_folder, '%d%s%d.png'], N_targets,
'_target', i));

% save map in a txt file
nomefile = sprintf([matrix_folder, '%d%s%d.txt'], N_targets,
'_target', i);
```

```
writematrix(rdmap, nomefile, 'Delimiter', ' ');

%% index within the range-Doppler maps for each target

nomefile = sprintf([label_folder, '%d%s%d.txt'], N_targets,
'_target', i);

% first open file in write mode, then save data
%in the loop in append mode
fileID = fopen(nomefile, 'w');
fclose(fileID);

for j = 1:N_targets

    rho = sqrt(sum(tgtloc(:,j).^2)); % range posizione
    idx_range = round((rho/Rmax) * range_bins); % posizione
    idx_vel = round((vel_bins/2) +
    (rdvel(j)/(velMax/2)) * (vel_bins/2)); % velocita

    % Bounding box limits
    he_min = max(idx_range - h_box, 1);
    he_max = min(idx_range + h_box, range_bins);
    we_min = max(idx_vel - w_box, 1);
    we_max = min(idx_vel + w_box, vel_bins);

    % Visualizza la porzione della mappa contenente il target
    figure;
    imagesc(rdmap(he_min:he_max, we_min:we_max));

    %% VALORI BOUNDING BOX NORMALIZZATI
    w_norm = (we_max - we_min + 1) / vel_bins;
    h_norm = (he_max - he_min + 1) / range_bins;

    idx_range_norm = idx_range / range_bins;
    idx_vel_norm = idx_vel / vel_bins;

    % YOLO label format
    bb = [0, idx_vel_norm, idx_range_norm, w_norm, h_norm];

```

```
49      fileID = fopen(nomefile, 'a');
50
51      if (N_targets - j) == 0
52          fmt = '%1d %.6f %.6f %.6f %.6f';
53      else
54          fmt = '%1d %.6f %.6f %.6f %.6f\n';
55      end
56
57      fprintf(fileID, fmt, bb);
58      fclose(fileID);
59
60  end
```

La porzione di codice MATLAB presentato sopra ha lo scopo di generare etichette nel formato YOLO, partendo da mappe range-Doppler contenenti uno o più bersagli. Questo processo è fondamentale poiché consente di creare la **ground truth** necessaria per addestrare reti neurali convoluzionali dedicate al rilevamento automatico di oggetti.

Si struttura in più fasi:

1. **Salvataggio della mappa range-Doppler**: viene salvata un'immagine `.png` e la corrispondente matrice in formato `.txt` per ciascun esempio. Il nome del file include il numero di target presenti e un indice progressivo.
2. **Creazione del file di etichette**: per ogni immagine, si genera un file `.txt` vuoto in cui saranno successivamente salvate le bounding box.
3. **Estrazione della posizione del target**: per ogni target presente nella scena, si calcola la distanza radiale $\rho$ e la velocità relativa (doppler shift). Tali valori vengono trasformati in indici sulla matrice range-Doppler per identificare la posizione del target.
4. **Definizione della bounding box**: attorno alla posizione identificata si definisce una finestra centrata, le cui dimensioni sono specificate tramite i parametri `h_box` e `w_box`, che determinano rispettivamente l'altezza e la larghezza del riquadro.
5. **Visualizzazione opzionale**: viene generata una figura con la porzione della mappa contenente il target, utile per il debugging o per la verifica visiva.
6. **Normalizzazione dei valori**: per rispettare il formato YOLO, i valori delle bounding box vengono normalizzati rispetto alla dimensione complessiva della matrice (`range_bins` e `vel_bins`). In particolare, le coordinate del centro della bounding box e le sue dimensioni vengono espresse in termini relativi.

7. **Scrittura del file etichetta**: si salva una riga per ogni bounding box secondo il formato YOLO:

   ```
   <class_id> <x_center> <y_center> <width> <height>
   ```

   Nel nostro caso, la `class_id` è sempre 0 poiché tutti i target appartengono alla stessa classe.

Nel nostro approccio abbiamo scelto di utilizzare direttamente le mappe range-Doppler come dominio di riferimento per la generazione delle etichette. In questo modo si assicura coerenza tra i dati forniti in input alla rete neurale e le etichette associate in output. Poiché la rete deve rilevare i target nello spazio range-Doppler, è fondamentale che le bounding box vengano definite in tale spazio, così da rispettare la discretizzazione e la risoluzione effettiva delle mappe.

La qualità di queste etichette influisce direttamente sulle prestazioni del modello in fase di test, rendendo essenziale una corretta e accurata generazione delle stesse. Nella fig 4.1 vengono riassunte in uno schema a blocchi le parti fondamentali del codice.

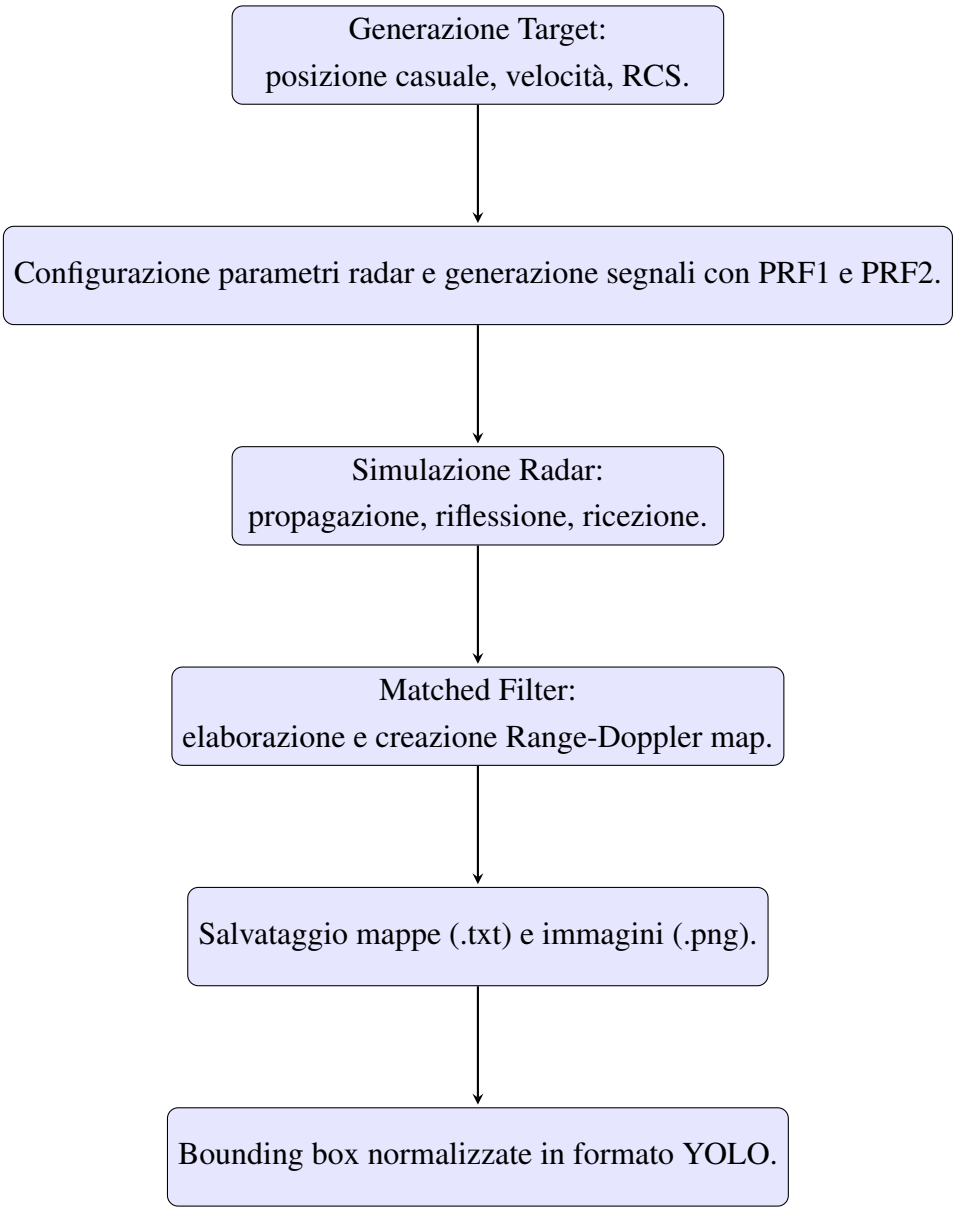



**Figure 4.1** Schema a blocchi del processo di generazione, simulazione e annotazione delle mappe radar.

Nella prossima sezione andiamo a mostrare come è avvenuta l'implementazione di due tipologie di CFAR

### 4.1.1 Metodi Tradizionali: CFAR

L'approccio più semplice per il rilevamento dei target nelle mappe range-Doppler è applicare una soglia a ciascuna cella della mappa. Utilizzando il criterio di Neyman-Pearson per il test delle ipotesi, la soglia può essere impostata fissando la probabilità di falso allarme a un

valore target [113]. La soglia può essere uguale per ogni cella o adattativa rispetto al rumore di fondo locale della mappa.

Questa seconda scelta, conosciuta come rilevamento a Tasso Costante di Falso Allarme (CFAR) [114], consiste nell'impostare adattivamente la soglia per ogni cella in funzione sia della probabilità di falso allarme attesa che delle statistiche del rumore. Queste statistiche vengono solitamente stimate in un'area di celle attorno alla Cellula sotto Test (CUT).

Fino ad oggi, CFAR rappresenta il criterio più utilizzato ed efficace per il rilevamento dei target nell'elaborazione dei segnali radar [115]. Nel corso degli anni, sono state proposte diverse varianti [116–118] con l'obiettivo di mitigare la perdita di prestazioni dovuta al numero finito di celle di addestramento [119] o dalla presenza di clutter che altera la distribuzione del rumore [120]. In questo articolo, ci riferiamo a tre tipi classici di rilevatori.

## 4.2 Detettore Neyman-Pearson a soglia fissa

Il rilevamento del target viene realizzato per ogni CUT (Cell Under Test) mediante un test ipotetico tra l'ipotesi nulla $H_0$ (solo rumore) e l'ipotesi alternativa $H_1$ (segnale più rumore):

$$p(x \mid H_1) \underset{H_0}{\overset{H_1}{\gtrless}} \tau \tag{4.2}$$

Si noti che fissiamo la soglia $\tau$ per ogni cella all'interno di una mappa e tra mappe diverse. Varia poi il valore della soglia per produrre le curve ROC (Receiver Operating Characteristic), considerando sia i veri che i falsi positivi. Un primo vantaggio nell'uso di questo metodo è la sua semplicità computazionale. Inoltre, non è influenzato da alcuna perdita di prestazioni, come nel caso dei metodi basati su CFAR, ed è in grado di rilevare target in qualsiasi intervallo o frequenza Doppler. D'altra parte, è sensibile alle variazioni globali e locali delle statistiche del rumore o all'instabilità del sistema.

## 4.3 Cell-Averaging (CA) CFAR 1D

In questo approccio, basato sulla configurazione *Cell-Averaging*, la potenza media del rumore viene stimata utilizzando le celle adiacenti, sia precedenti che successive rispetto alla cella in esame (CUT), lungo ciascun filtro Doppler della mappa. Indicando con $P_r$ la potenza media del rumore stimata a partire dagli $N$ vicini ovvero le *celle di riferimento* relativi alla cella d'intervallo $m$-esima, la soglia viene determinata secondo la seguente relazione:

$$\tau_r = \alpha \cdot P_r \tag{4.3}$$

dove il pedice $r$ è l'indice dell'intervallo. Il parametro $\alpha$ è scelto in funzione della probabilità desiderata di falso allarme $p_{fa}$. Assumendo i campioni di rumore I/Q come indipendenti e distribuiti secondo una legge gaussiana con un rilevatore a legge quadrata, si ha:

$$\alpha = N \cdot \left(p_{fa}^{-1}\right)^{\frac{1}{N-1}} \tag{4.4}$$

Poiché $P_r$ viene stimato utilizzando le celle adiacenti, le celle CUT che si trovano a una distanza inferiore a $N/2$ dagli estremi dell'intervallo (inizio o fine della mappa) dispongono di un numero di celle di riferimento inferiore a $N$, riducendo l'accuratezza della stima. Inoltre, poiché $P_r$ è una media campionaria che approssima la media della potenza vera, i rilevatori CFAR soffrono di una perdita di prestazioni rispetto all'approccio a soglia fissa. Tale perdita di prestazioni $L$ è inversamente proporzionale al numero di celle di addestramento:

$$L_{dB} = \frac{-\log_{10}\left(p_{fa}\right)}{N} \tag{4.5}$$

Questo comportamento verrà preso in considerazione nella progettazione degli esperimenti per evitare eventuali bias indesiderati nei risultati. Riportiamo nel dettaglio anche il relativo pseudocodice.

```
%% Parametri del rivelatore
N_guard = 4;
N_train = 16;
pfa = 1e-4;

% Inizializzazione dei contatori
TP = 0;
FP = 0;
N_target = 0;

%% Creazione del rivelatore CFAR 1D per ciascun filtro Doppler
cfar1D = phased.CFARDetector2D('Method', 'CA',
'ThresholdOutputPort', true);

cfar1D.GuardBandSize = [N_guard 0];
cfar1D.TrainingBandSize = [N_train 0];
cfar1D.ProbabilityFalseAlarm = pfa;

%% Lettura dei file di test (mappe in dB)
```

```
test_folder = './train_data/';
files = dir([test_folder 'images/test1cm/*.txt']);

%% Loop principale
for f = 1:length(files)
    disp(f);

    data = readmatrix([files(f).folder '/' files(f).name]);
    mag = db2mag(data);

    if f == 1
        rangeIndx = [N_guard + N_train + 1,
        size(mag,1)-(N_guard + N_train)];

        dopplerIndx = [1, size(mag,2)];
        [columnInds, rowInds] = meshgrid
                                (dopplerIndx(1):dopplerIndx(2), ...
                                rangeIndx(1):rangeIndx(2));
        CUTIdx = [rowInds(:) columnInds(:)]';
    end

    % Rilevamento CFAR
    detections = cfar1D(mag, CUTIdx);

    % Clustering delle rilevazioni
    M = clustering(CUTIdx, detections, mag);

    % Lettura delle label (bounding box)
    bb = readmatrix([test_folder, 'labels/test1cm/',
    files(f).name], 'Delimiter', ' ');

    yden = bb(:,2) * size(mag,2);
    xden = bb(:,3) * size(mag,1);
    Mv = [xden, yden];

    % Calcolo TP e FP
    N_target = N_target + size(Mv,1);
    ctp = 0;
    for ii = 1:size(Mv,1)
```

```
        V = Mv(ii,:);
        for jj = 1:size(M,1)
            u = M(jj,:);
            if abs(V(1)-u(1))<5 && abs(V(2)-u(2))<5
                ctp = ctp + 1;
            end
        end
    end
    TP = TP + ctp;
    FP = FP + (size(M,1) - ctp);
end

% Calcolo delle metriche di prestazione
FN = N_target - TP;
precision = TP / (TP + FP);
recall = TP / (TP + FN);
f_measure = 2 * (precision * recall) / (precision + recall);
detection_rate = TP / length(files);
false_alarm_rate = FP / ...
    (length(files)*(rangeIndx(2)-rangeIndx(1))
    *(dopplerIndx(2) - dopplerIndx(1)));

% Salvataggio dei risultati
save(out_file, 'TP', 'FP', 'FN', 'precision', 'recall',
                'f_measure','detection_rate', 'false_alarm_rate');
close all
```

Vengono impostati i parametri del rivelatore: il numero di celle di guardia, il numero di celle di addestramento e la probabilità di falso allarme desiderata. Vengono poi inizializzati i contatori per tenere traccia di true positive, false positive e numero totale di target.

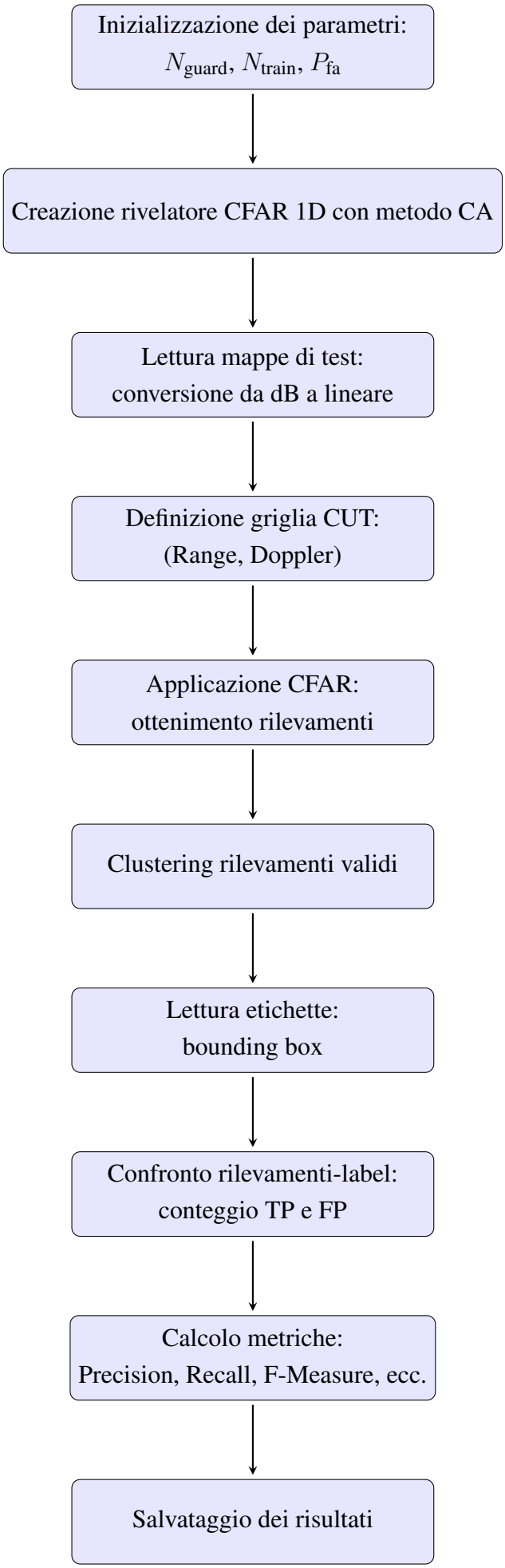
Successivamente viene creato un oggetto ‘CFARDetector2D‘ con configurazione 1D, in cui il rivelatore agisce solo lungo l’asse della distanza. Si configura questo rivelatore con i parametri stabiliti.

Il programma legge quindi una serie di file di test contenenti immagini in scala logaritmica (decibel), che vengono convertite in scala lineare. Durante la prima iterazione, vengono determinati gli indici delle celle sotto test (CUT), evitando i bordi per non includere celle incomplete nei calcoli.

Per ogni mappa, viene applicato il rilevamento CFAR. I risultati vengono poi raggruppati (clustering) per individuare i target rilevati come gruppi di celle vicine. Successivamente, vengono letti i file di etichette associati alla mappa, che contengono i ground truth in forma normalizzata; questi vengono convertiti in coordinate effettive rispetto alle dimensioni della mappa.

Il codice confronta quindi i target rilevati con quelli effettivi per valutare quante rilevazioni siano corrette (true positive) e quante siano errate (false positive). Una rilevazione è considerata corretta se la distanza in pixel tra un punto rilevato e uno reale è inferiore a una soglia prefissata (5 pixel in questo caso).

Alla fine del ciclo, si calcolano le metriche di prestazione: precision, recall, F-measure, tasso di rilevamento e tasso di falsi allarmi. Questi valori vengono salvati in un file. Viene riportato lo schema a blocchi dello pseudocodice 4.2


Inizializzazione dei parametri:
$N_{\text{guard}}$, $N_{\text{train}}$, $P_{\text{fa}}$
Creazione rivelatore CFAR 1D con metodo CA
Lettura mappe di test:
conversione da dB a lineare
Definizione griglia CUT:
(Range, Doppler)
Applicazione CFAR:
ottenimento rilevamenti
Clustering rilevamenti validi
Lettura etichette:
bounding box
Confronto rilevamenti-label:
conteggio TP e FP
Calcolo metriche:
Precision, Recall, F-Measure, ecc.
Salvataggio dei risultati

## 4.4 Cell-Averaging (CA) CFAR 2D

La potenza media del rumore viene stimata all'interno di una finestra bidimensionale centrata sulla CUT, che si estende lungo le dimensioni di range e Doppler. Rispetto alla versione monodimensionale (1D), la perdita di prestazioni è trascurabile grazie all'impiego di un numero significativamente maggiore di celle di addestramento.

Le stesse considerazioni sui CUT vicini ai bordi della mappa (cioè sia in range che in Doppler) si applicano anche a questo metodo. Nei nostri test, abbiamo progettato i metodi CA-CFAR assumendo 32 celle di addestramento e 8 celle di protezione (cioè 16 e 4 rispettivamente prima e dopo i CUT). Il fattore di soglia $\alpha$ è impostato come in Equazione (3). La probabilità teorica di falso allarme varia da $10^{-8}$ a $10^{-2}$.

Nel passaggio da un rilevatore CFAR 1D a uno 2D, le modifiche principali a livello implementativo si concentrano nella fase di inizializzazione del rilevatore e nella selezione degli indici delle celle sotto test (CUT). Di seguito, si evidenziano in pseudocodice solo le parti modificate rispetto alla versione 1D.

```
cfar1D = phased.CFARDetector2D('Method', 'CA','ThresholdOutputPort'
                                                              ,true);

cfar1D.GuardBandSize = [N_guard 0];
cfar1D.TrainingBandSize = [N_train 0];
cfar1D.ProbabilityFalseAlarm = pfa;

dopplerIndx = [1, size(mag,2)];
```

### CFAR 2D

```
cfar2D = phased.CFARDetector2D( ...
    'GuardBandSize' , N_guard, ...
    'TrainingBandSize', N_train, ...
    'ProbabilityFalseAlarm' , pfa);

dopplerIndx = [N_guard + N_train + 1, size(mag,2)-
                            (N_guard + N_train)];
```

Nel CFAR 2D, la soglia viene calcolata su entrambe le dimensioni (range e Doppler). Pertanto, sia le bande di addestramento che quelle di guardia sono applicate in entrambe le direzioni. Per questo motivo, l'indice Doppler minimo e massimo dei CUT è limitato proprio

da queste dimensioni, come si vede nella modifica di `dopplerIndx`, che ora considera un margine.

L'utilizzo del CFAR bidimensionale consente una soglia di rilevamento più robusta, in quanto si basa su una stima del rumore più completa, ottenuta considerando i livelli di potenza nelle celle circostanti la CUT sia lungo il range che lungo la frequenza Doppler. Questo intorno di celle, spesso definito ambiente locale, rappresenta il contesto statistico entro cui si valuta la presenza di un target, migliorando così l'adattività del rilevatore alle variazioni del rumore di fondo.

## 4.5 Clustering

I metodi di rilevamento descritti sopra classificano ogni CUT come occupato da un target o meno. Tuttavia, in scenari realistici, un singolo target probabilmente occupa più di una cella. Questo fenomeno è noto come "spill-over" del target. Per affrontare questo problema, ogni schema di rilevamento è seguito da un algoritmo di clustering che collega tra loro celle vicine (in intervallo e Doppler) rilevate come contenenti un target. Nel nostro studio, definiamo un metodo di clustering basato sulla teoria dei grafi. In particolare, costruiamo una matrice di adiacenza $A_{i,j}$ tra tutte le celle di intervallo-Doppler $c_{r,d}$, i cui pedici $(r,d) \in \mathbb{N}$ in $[0; R-1] \times [0; D-1]$ denotano le coordinate lungo, rispettivamente, le dimensioni di intervallo e Doppler di una mappa generica con $R \times D$ celle. Dopo la soglia, $c_{r,d} = 1$ se è stato rilevato un target, $0$ altrimenti. Definiamo ora il pedice generico $i$ (o $j$) come la versione linearizzata in ordine $C$ delle coordinate $(r,d)$, cioè $i = d \cdot R + r$ per ogni $(r,d)$, identificando ogni cella in una mappa. Si noti che la matrice di adiacenza $A_{i,j}$ è una matrice binaria con $(R \times D) \times (R \times D)$ voci, popolata come segue: per ogni $i \neq j$ e $c_i = c_j = 1$, allora $A_{i,j} = 1$ se le seguenti condizioni sono soddisfatte simultaneamente:

$$\|i-j\|_1 \bmod (R-1) \leq 1 \tag{4.6}$$

$$\left\lfloor \frac{i}{R} \right\rfloor - \left\lfloor \frac{j}{R} \right\rfloor \bmod (D-1) \leq 1 \tag{4.7}$$

La condizione (4.6) stabilisce che due celle i e j sono vicine in range se si trovano sulla stessa colonna (stesso Doppler) e distano al massimo una riga. L'operazione modulo gestisce correttamente i bordi superiori e inferiori della mappa.

La condizione (4.7) identifica celle vicine in Doppler se si trovano sulla stessa riga (stesso range) e distano al massimo una colonna. L'operazione modulo in entrambe le condizioni permette di gestire le ambiguità di intervallo e Doppler, mentre $\lfloor * \rfloor$ è l'operatore floor . Una

volta definito $A_{i,j}$ [121], i cluster di celle $\gamma_k$ vengono identificati trovando le componenti connesse del grafo non orientato definito da $A_{i,j}$. Infine, il range e il Doppler del target identificato dal cluster $\gamma_k$ sono stimati come segue:

$$(r^*, d^*) = \underset{c_{r,d} \in \gamma_k}{\arg\max}\, c_{r,d} \tag{4.8}$$

### 4.5.1 Tecniche innovative di DL per il tracciamento

YOLO (You Only Look Once) è un algoritmo di rilevamento oggetti introdotto per la prima volta da Joseph Redmon et al. nel 2016 [122]. L'idea chiave dietro YOLO è l'utilizzo di una singola rete neurale per prevedere contemporaneamente le probabilità di classe e le coordinate delle bounding box degli oggetti in un'immagine. A differenza degli algoritmi tradizionali di object detection, che si basano su più stadi per rilevare gli oggetti, YOLO adotta un approccio diverso elaborando l'intera immagine in un singolo passaggio. Questa scelta progettuale rende YOLO più veloce ed efficiente rispetto ad altri metodi.

Un concetto importante impiegato da YOLO è l'utilizzo delle *anchor box*, ovvero riquadri di riferimento con dimensioni e rapporti d'aspetto predefiniti. L'incorporazione delle anchor box migliora la capacità dell'algoritmo di rilevare con precisione oggetti di dimensioni e proporzioni diverse.

L'algoritmo YOLO opera attraverso le seguenti fasi:

1. **Input dell'immagine**: l'algoritmo riceve in input un'immagine, che può essere una foto o un fotogramma video.
2. **Suddivisione in griglia**: l'immagine viene suddivisa in una griglia di celle, ognuna delle quali è responsabile della rilevazione degli oggetti al proprio interno.
3. **Estrazione delle caratteristiche**: ogni cella viene elaborata da una CNN pre-addestrata che apprende caratteristiche rilevanti (contorni, texture, colori) da grandi dataset.
4. **Punteggio di presenza oggetto (*objectness score*)**: ogni cella assegna un punteggio basato su regressione logistica che indica la probabilità della presenza di un oggetto.
5. **Probabilità di classe**: se una cella rileva un oggetto, assegna anche una classe con la relativa probabilità tramite una funzione SoftMax.
6. **Bounding box**: per le celle che rilevano oggetti, YOLO prevede anche il riquadro (bounding box) che racchiude l'oggetto. Questo include le coordinate del centro (X, Y), la larghezza (W) e l'altezza (H), tutte relative alla dimensione della cella di griglia.

7. **Soppressione non massima (NMS)**: YOLO applica la soppressione non massima per eliminare le box ridondanti con punteggi più bassi, mantenendo solo quelle con maggiore confidenza.
8. **Output**: il risultato finale è un insieme di bounding box, ognuna associata a una classe e a un punteggio di confidenza, che indica quanto il modello è sicuro della rilevazione.

Per addestrare le reti YOLO, è stata definita una funzione di perdita avanzata che considera tre aspetti principali [122]:

- **Confidenza dell'oggetto**, che misura quanto bene il modello predice la presenza di un oggetto nella cella.
- **Confidenza dello sfondo**, che valuta quanto accuratamente il modello predice l'assenza di un oggetto.
- **Intersection over Union (IoU)**, che misura la sovrapposizione tra la bounding box prevista e quella reale.

La funzione di perdita complessiva è una somma pesata di queste metriche calcolata su tutte le celle e le scale della griglia.

Dalla sua introduzione, YOLO ha subito diversi aggiornamenti, ciascuno migliorando velocità, precisione e funzionalità. I ricercatori continuano a ottimizzare YOLO per prestazioni sempre migliori nel rilevamento oggetti in tempo reale [123, 124].

### Yolo v5

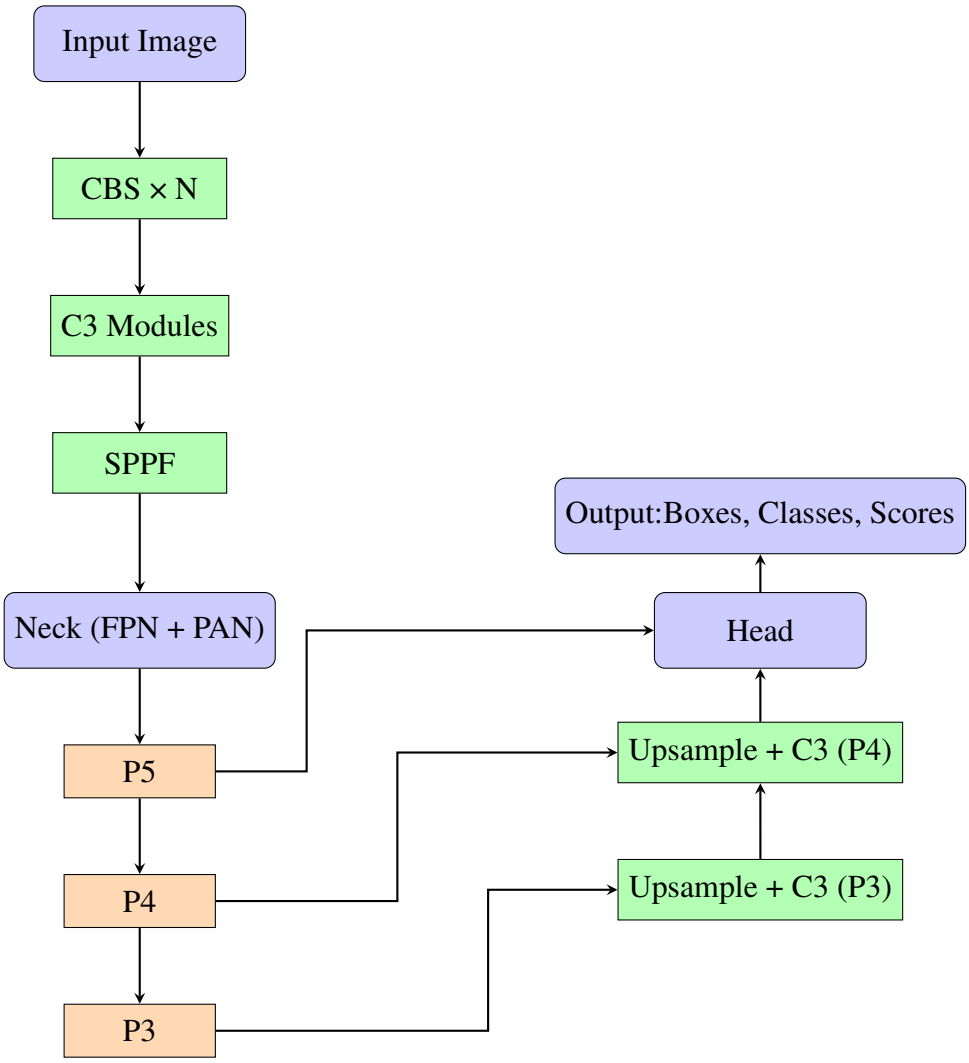
YOLOv5 rappresenta un notevole miglioramento in termini di velocità e precisione rispetto ai suoi predecessori. Questi progressi lo rendono un algoritmo eccezionale per compiti di rilevamento oggetti in tempo reale. L'architettura è composta da tre componenti principali: **backbone**, **neck** e **head**. In questo studio è stata utilizzata la versione più leggera della rete, comunemente indicata come *v5s*, composta da circa 7,2 milioni di parametri e con un'efficienza computazionale pari a 16,5 GFLOPs .

La rete **backbone** di YOLOv5 estrae informazioni delle feature dalle immagini di input utilizzando una combinazione di moduli CBS (Convolution + Batch Normalization + Sigmoid Linear Unit activation), il modulo C3 (triple convolution), e il modulo SPPF (Spatial Pyramid Pooling Fusion). Il modulo C3 consente di ridurre la complessità computazionale migliorando al tempo stesso la velocità di inferenza, grazie all'apprendimento di feature residue attraverso più blocchi bottleneck. Il modulo SPPF migliora l'accuratezza del modello fondendo informazioni multiscala mediante un kernel di pooling $5 \times 5$, rendendo l'elaborazione più rapida ed efficiente rispetto al precedente modulo SPP.

La rete **neck** di YOLOv5 si basa su una combinazione di Feature Pyramid Network (FPN) e Path Aggregation Network (PAN). L'FPN propaga l'informazione semantica in modo top-down, mentre il PAN facilita la propagazione delle informazioni a basso livello in modo bottom-up, migliorando la localizzazione. L'obiettivo principale di questa struttura è l'aggregazione di mappe di feature di dimensioni diverse, migliorando così la capacità di rilevare oggetti di varie dimensioni.

Durante la costruzione della **head** del modello, le mappe di feature associate a P4 e P3 vengono sottoposte a operazioni di upsampling per aumentare la risoluzione. Successivamente, queste mappe vengono concatenate con le rispettive mappe provenienti dal backbone, al fine di combinare informazioni a differenti scale spaziali. Le mappe concatenate attraversano poi i moduli C3, ottimizzati per raffinare le feature specifiche degli oggetti a varie scale. Questo processo viene ripetuto anche per le mappe associate a P4 e P5, tenendo in considerazione le dimensioni degli oggetti (piccoli, medi e grandi).

Infine, le mappe P3, P4 e P5 vengono utilizzate per localizzare gli oggetti.Gli anchor associati a ciascuna scala vengono utilizzati per effettuare previsioni riguardo alle classi degli oggetti, ai punteggi di confidenza e alle coordinate delle bounding box. Gli anchor sono regioni di interesse predefinite, con dimensioni e posizioni specifiche relative alle mappe di feature. Questi anchor aiutano il modello a generare proposte di bounding box per gli oggetti presenti nell'immagine. Riportiamo sottostante il diagramma di flusso della rete Yolo v5.




**Yolo v8**

YOLO v8 è una versione avanzata rispetto alle precedenti versioni di YOLO, con nuove funzionalità e miglioramenti. Innanzitutto, YOLO v8 presenta maggiore accuratezza e velocità rispetto ai suoi predecessori. Questo significa che è in grado di rilevare oggetti con maggiore precisione pur mantenendo capacità di elaborazione in tempo reale.

L'implementazione di YOLO v8 utilizzata in questo studio è la versione **nano** (indicata come *v8n* nella letteratura), che presenta 225 layer, 3.157.200 parametri e un'efficienza computazionale di 8,9 GFLOPs.

Per catturare efficacemente funzionalità di alto livello, YOLO v8 introduce un backbone aggiornato ispirato a EfficientNet. Questo consente al modello di comprendere schemi visivi complessi e di effettuare previsioni più accurate.

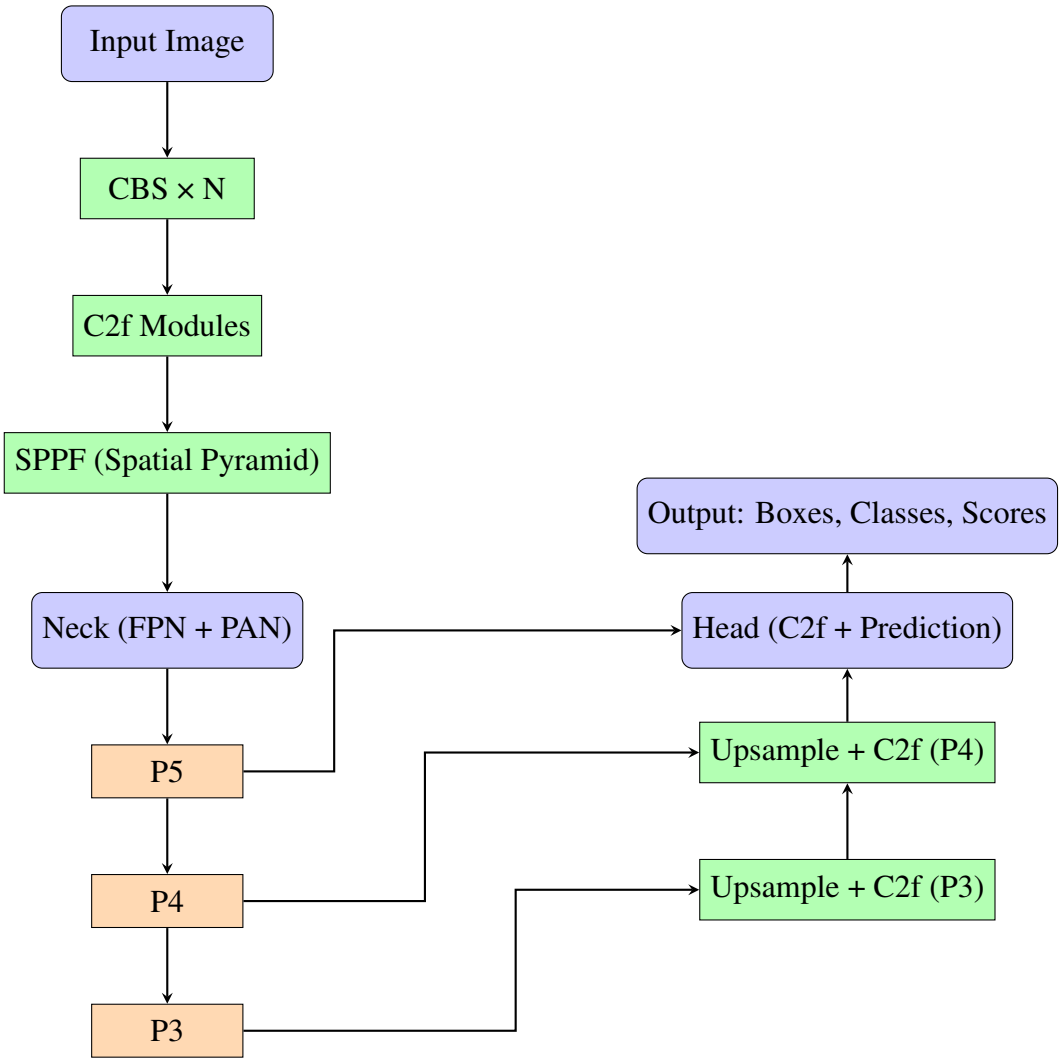
Un'altra novità introdotta da YOLO v8 è il modulo di fusione delle feature, che combina capacità informative a differenti scale, consentendo al modello di rilevare oggetti di dimensioni diverse e migliorare la precisione di localizzazione.

Per migliorare ulteriormente le prestazioni, YOLO v8 integra tecniche avanzate di data augmentation, come *MixUp* e *CutMix*. Queste tecniche ampliano i dati di addestramento combinando più immagini o mescolando porzioni di immagini, con il risultato di una migliore generalizzazione e robustezza.

Il backbone estrae informazioni dibasso e alto livello mediante una combinazione di strati convoluzionali e moduli *C2f*. Il modulo C2f è una versione migliorata del precedente C3 e rappresenta il principale modulo di apprendimento residuo. Esso riduce una convoluzione standard in modo da mantenere la leggerezza della rete e allo stesso tempo catturare maggiori informazioni sul flusso del gradiente.

Il modulo *SPPF* consente la creazione di un'astrazione a piramide spaziale. Le feature estratte vengono poi utilizzate dalla testa del modello per generare le previsioni. Come in YOLO v5, vengono effettuate operazioni di upsampling sulle feature map associate a P4 e P3 per aumentarne la risoluzione.

Successivamente, queste mappe vengono concatenate con le mappe corrispondenti del backbone. Dopo la concatenazione, le mappe combinate (P3, P4 e P5) vengono elaborate da moduli *C2f*, progettati per affinare le feature degli oggetti a diverse scale. Riportiamo sottostante il diagramma di flusso della rete Yolo v8.




Gli approcci convenzionali sono, in un certo senso, rilevatori basati sul valore: la decisione viene presa confrontando il valore di una determinata cella (CUT) con una soglia. Nel caso del CFAR, la soglia dipende sia dalla probabilità di falso allarme attesa sia da statistiche locali (di primo ordine, nel caso di cell averaging) del segnale. Una volta fissata la soglia, la decisione è presa confrontando solo quel valore specifico con la soglia.

Al contrario, i rilevatori basati su YOLO sfruttano non solo i valori del CUT (o del singolo pixel, nella terminologia della computer vision), ma anche la morfologia del segnale nelle celle circostanti. Questa caratteristica, ottenuta grazie all'uso di filtri convoluzionali su diverse scale, è utile per ridurre il numero di falsi allarmi, poiché la rilevazione non si basa su un singolo valore potenzialmente rumoroso.

Inoltre, l'utilizzo della confidenza dell'oggetto e dello sfondo all'interno della funzione di perdita consente all'architettura di apprendere modelli non lineari del rumore di fondo nelle mappe range-Doppler e del segnale.

In quest'ultimo caso, la presenza del target spill-over (ovvero la dispersione del segnale del bersaglio) non rappresenta un problema critico, come accade negli approcci classici, poiché YOLO lavora a livello di patch e non di singola cella. In figura 4.3 il workflow di tutto ciò che è stato descritto fino ad ora. I set di training e validazione vengono utilizzati per riaddestrare entrambe le architetture YOLO da zero. Per entrambi i modelli v8 e v5, il numero di campioni di training utilizzati in ogni fase di iterazione (dimensione del batch) era 32. I modelli sono stati inoltre addestrati con 100 epoche e Adam come ottimizzatore. A titolo di esempio, riportiamo il risultato dell'addestramento di YOLO v5 in termini di perdite di box loss e object loss 4.4.

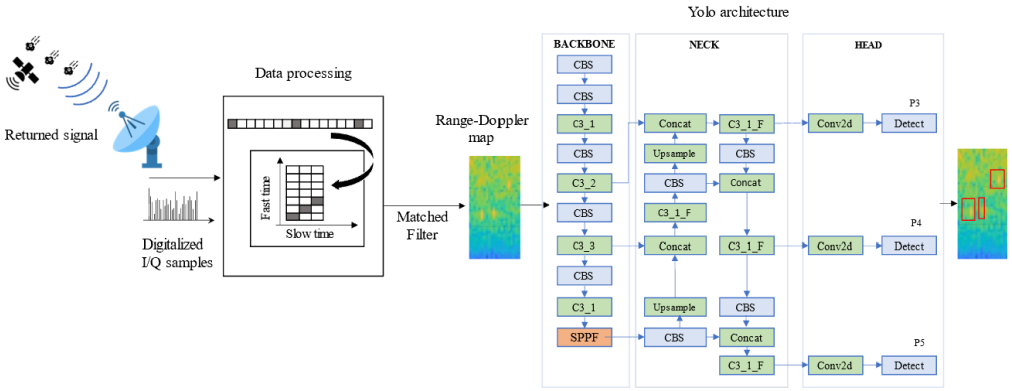



**Figure 4.3** Schema a blocchi semplificato del sistema radar Doppler a impulsi e conseguente elaborazione digitale, con l'introduzione di un rilevatore di bersaglio mobile basato su YOLO dopo il filtro adattato.

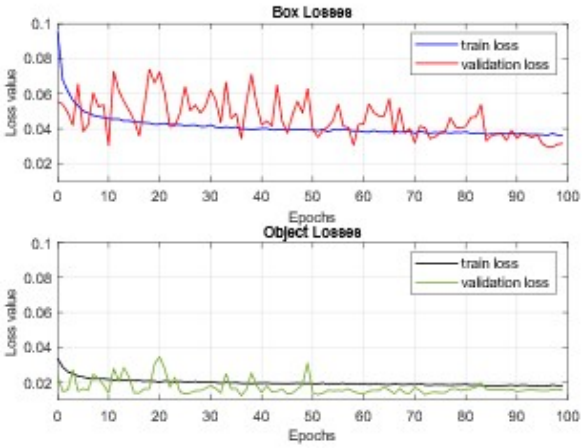



**Figure 4.4** Curve di apprendimento per yoloV5

### 4.5.2 Risultati e Discussioni

In questa sezione, mostriamo i risultati ottenuti nel nostro ambiente simulato. Per prima cosa, definiamo le metriche per misurare le prestazioni dei metodi e presentiamo i risultati quando è presente solo rumore AWGN al ricevitore. Successivamente, generalizziamo i nostri test alla presenza di rumore colorato.

Le metriche di prestazione che abbiamo utilizzato sono il *True Positive Rate* (TPR) e il *False Positive Rate* (FPR).

Il TPR misura la capacità dell'algoritmo di identificare e rilevare correttamente i bersagli reali. È definito come il rapporto tra il numero di bersagli reali correttamente rilevati dal sistema ($N_{\text{TP}}$) e il numero totale di bersagli presenti nel set di test ($N_{\text{targets}}$). Formalmente:

$$\text{TPR} = \frac{N_{\text{TP}}}{N_{\text{targets}}} \tag{4.9}$$

Una probabilità di rilevamento più alta indica un algoritmo più efficace nell'identificare e rilevare correttamente i bersagli.

L'FPR è definito come il rapporto tra il numero di bersagli rilevati erroneamente ($N_{\text{FP}}$) e il numero totale di celle testate nell'esperimento ($N_{\text{CUTs}}$). Formalmente:

$$\text{FPR} = \frac{N_{\text{FP}}}{N_{\text{CUTs}}} \tag{4.10}$$

È importante notare che, $N_{\text{TP}}$ e $N_{\text{FP}}$ rappresentano rispettivamente il numero di bersagli (veri e falsi) rilevati dopo il processo di clustering. Un bersaglio rilevato è classificato come vero positivo se la differenza tra la sua posizione range-Doppler stimata e la posizione reale è inferiore a una soglia di tolleranza che abbiamo fissato pari a 3. Se più rilevamenti ricadono all'interno di questa regione di accettazione, solo uno viene considerato vero positivo, mentre gli altri vengono conteggiati come falsi positivi. I bersagli rilevati al di fuori della regione di accettazione sono classificati come falsi positivi.

Per quanto riguarda le rilevazioni basate su YOLO, il processo di clustering è già integrato all'interno dell'architettura della rete.

Infine, per visualizzare tutte le prestazioni in un unico grafico, generiamo le curve *ROC* (Receiver Operating Characteristic), rappresentando il TPR in funzione del FPR, variando:

- la soglia di rilevamento $\tau$ per il metodo a soglia fissa;
- la probabilità attesa di falso allarme (PFA) per gli approcci CFAR, nell'intervallo da $10^{-8}$ a $10^{-2}$;

- il livello di confidenza in uscita degli approcci YOLO, nel set {0.001, 0.25, 0.50, 0.75, 0.90}.

Per una migliore visualizzazione, l'asse del FPR è rappresentato in scala logaritmica, come mostrato nella Figura 4.5 .

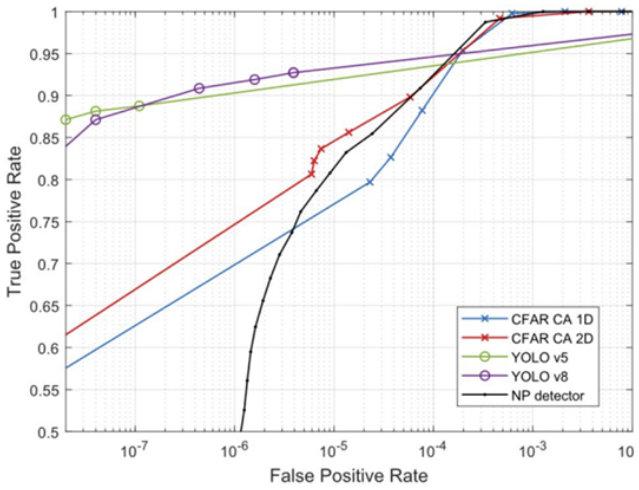



**Figure 4.5** Curve ROC dei rivelatori sul rumore gaussiano bianco additivo (AWGN) per rivelatori convenzionali e basati su YOLO.

I metodi CFAR con (*Cell Averaging*) mostrano un *False Positive Rate* (FPR) significativamente più elevato rispetto alla PFA teorica attesa. In teoria, l'FPR dovrebbe convergere alla PFA all'aumentare del numero di test. Tuttavia, nel caso del CFAR monodimensionale , si osserva un valore di FPR $\approx 9 \cdot 10^{-5}$ quando la PFA è fissata a $10^{-8}$. Questo comportamento si spiega considerando che il tasso di falsi positivi è calcolato su mappe contenenti un solo target. La presenza del target modifica le statistiche locali della potenza nella mappa, rendendo possibile la comparsa di falsi allarmi. Tali falsi rilevamenti possono essere causati, ad esempio, da effetti di spillover del segnale del target o da fenomeni di autocorrelazione.

Vale anche la pena notare che i valori di TPR (True Positive Rate) e FPR (False Positive Rate) sono calcolati in funzione del numero di target veri o falsi rilevati da ciascun sistema. Ciò implica che, per i rivelatori classici, queste metriche vengono valutate dopo aver effettuato il clustering delle celle individuate come potenzialmente contenenti un target. Di conseguenza, non sorprende che possano emergere effetti non lineari, come ad esempio una diminuzione del FPR all'aumentare del PFA, in quanto molte celle tendono a essere raggruppate in pochi cluster.

La figura 4.5 mostra come i metodi basati su CA-CFAR e il rivelatore di Neyman-Pearson presentino prestazioni simili in presenza di rumore AWGN nella regione con FPR $> 10^{-4}$.

Sotto questa soglia, si osserva una perdita di capacità di rilevamento del CA-CFAR 1D. Tale perdita è trascurabile nel CA-CFAR 2D, che utilizza un numero maggiore di celle di riferimento (32 in range e Doppler).

Al contrario, i metodi basati su YOLO offrono una migliore capacità di rilevamento nella regione FPR $< 10^{-4}$. YOLOv8 mostra un TPR leggermente superiore rispetto a YOLOv5 per FPR $> 10^{-7}$. Considerando lo stesso livello di confidenza, YOLOv8 rileva un numero maggiore di target rispetto a YOLOv5, ma con un incremento del numero di falsi positivi. Tale differenza può essere spiegata con l'architettura più complessa di YOLOv8, che comporta un rischio maggiore di overfitting.

In Figura 4.6 è mostrato il confronto tra i valori di TPR dei due modelli YOLO, al variare della soglia di confidenza, distinguendo l'analisi in base alla dimensione del target. Nel complesso, YOLOv8 ottiene un TPR superiore a parità di dimensione del target e livello di soglia, ma a scapito di un FPR maggiore.

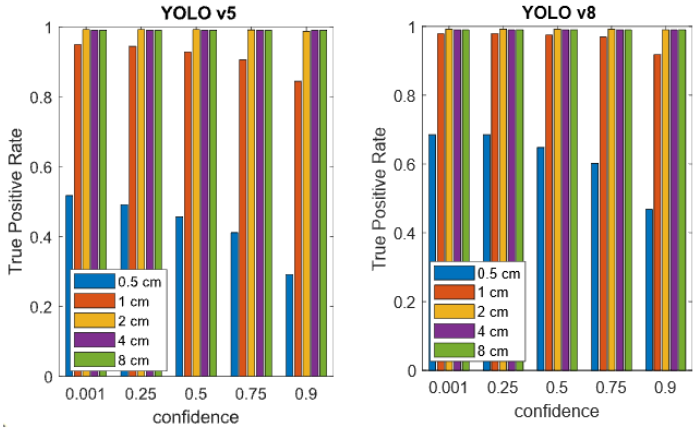



**Figure 4.6** Confronto tra veri positivi dei metodi YOLO v5 e YOLO v8 variando la soglia di confidenza e la dimensione del target in caso di rumore gaussiano bianco additivo.

### Risultati con rumore colorato

```
%%%  ADD Noise
% signal, noise type, power in dB
rxpulses = addColoredNoise(rxpulses, 1, -100);

%%%% prima del matched filter

%%% fine codice

function xout = addColoredNoise(x, shape, pow)
```

```
10
11      pinkNoise = dsp.ColoredNoise(shape ,size(x,2),2);
12      xout = zeros(size(x));
13
14      for i=1:size(x,1)
15          n = pinkNoise();
16          n = n(:,1) + 1i*n(:,2); % create complex noise
17          % normalizzazione per avere rumore a pow dB
18          n = (n./rms(n))*10^(pow/20)./size(x,1);
19          xout(i,:) = x(i,:) + n';
20      end
21
22  end
```

Per simulare condizioni di ricezione più realistiche, si è introdotto del rumore colorato nel segnale ricevuto. A tale scopo, è stata implementata una funzione MATLAB chiamata `addColoredNoise`, la quale permette di sommare al segnale complesso in ingresso un rumore di tipo colorato, come ad esempio il *rumore rosa*.

La funzione prende in ingresso tre parametri principali: il segnale complesso `x`, il parametro `shape` che definisce la colorazione del rumore (nel nostro caso, `shape = 1` corrisponde al rumore rosa) e la potenza del rumore desiderata, espressa in decibel (`pow`, ad esempio `-100 dB`).

Il rumore viene generato mediante l'oggetto `dsp.ColoredNoise`, che consente di produrre rumore colorato con specifiche statistiche di potenza. Per ciascuna riga della matrice del segnale ricevuto (che rappresenta un singolo impulso radar nel dominio temporale), viene generato un vettore di rumore complesso. Questo è ottenuto combinando due segnali: uno per la parte reale e uno per la parte immaginaria del rumore.

Successivamente, il rumore viene normalizzato affinché la sua potenza corrisponda al valore specificato in dB. Tale normalizzazione viene eseguita mediante la divisione per il valore RMS (*Root Mean Square*) del segnale di rumore e la moltiplicazione per il valore corrispondente in scala lineare, ottenuto da $10^{\mathrm{pow}/20}$. Inoltre, per evitare un'eccessiva amplificazione quando si somma il rumore su più righe, si effettua una divisione per il numero totale di righe del segnale.

Infine, il rumore normalizzato viene sommato alla riga corrente del segnale originale. Il risultato è un segnale rumoroso su cui sarà poi applicato il filtro adattato (*matched filter*) e le successive fasi della catena di elaborazione.

Questa procedura consente di testare l'efficacia dei metodi di rilevamento non solo in presenza di rumore bianco additivo gaussiano (AWGN), ma anche in condizioni più realistiche dove il rumore può avere una distribuzione spettrale non piatta.

Le curve ROC ottenute per diversi livelli di rumore rosa sono riportate in figura 4.8. Come previsto, le prestazioni di rilevamento peggiorano all'aumentare del rumore, ad eccezione del CA CFAR 1D, che mostra una notevole robustezza mantenendo pressoché costante la propria performance (la perdita massima è di circa 5% per $P_{FA} = 10^{-8}$). Questo comportamento è atteso, in quanto la soglia del metodo CFAR 1D è proporzionale alla stima della potenza media del rumore locale, calcolata indipendentemente per ciascuna frequenza Doppler. Da questo punto di vista, la *"non-bianchezza"* del rumore non comporta un aumento dei falsi allarmi.

Per quanto riguarda il CA CFAR 2D, la linea rossa tratteggiata evidenzia che la curva ROC è un effetto dell'interpolazione: in realtà, il CA CFAR 2D genera un FPR $\approx 10^{-3}$ indipendentemente dal valore di $P_{FA}$. Ciò è dovuto alla presenza di numerosi falsi allarmi a Doppler nullo (velocità radiale = 0 ), come illustrato in figura 4.9. Tali falsi positivi sono causati da una stima inaccurata della potenza media locale all'interno dell'insieme di celle disposte simmetricamente attorno alla CUT e le celle di guardia.

Anche i metodi YOLO mostrano un calo di performance al crescere del rumore, sebbene YOLOv5 si comporti meglio degli altri approcci quando $FPR < 10^{-5}$. Gli approcci CFAR garantiscono ancora un TPR superiore a 0.9, ma a scapito di un FPR elevato (maggiore di $10^{-4}$). Confrontando i modelli YOLO, la versione 5 risulta più robusta rispetto alla versione 8.

Nella figura 4.7 abbiamo le curve Roc anche in presenza di rumore rosa.

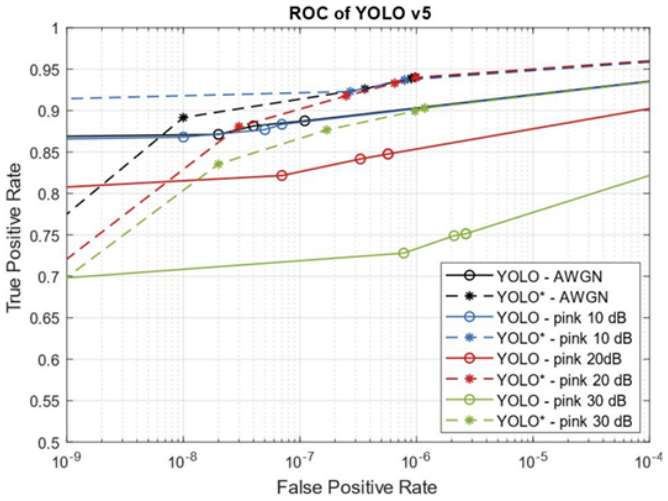



**Figure 4.7** Curve ROC su rumore gaussiano bianco additivo e rumore rosa prima e dopo l'addestramento sulle mappe con rumore

Il vantaggio del deep learning è proprio la capacità di apprendere dai dati. Nello specifico, i rivelatori basati su YOLO possiedono, una migliore capacità di separare il segnale di sfondo dall'oggetto di interesse. Questa caratteristica deriva dalla modellazione morfologica spaziale multi-scala (tramite filtri convoluzionali) e dall'uso di una funzione di perdita legata al punteggio di "objectness".

In quest'ottica, è stato condotto un esperimento finale ri-addestrando YOLOv5 con mappe affette da rumore rosa. In modo analogo a quanto fatto per il rumore bianco, al dataset di addestramento è stato aggiunto un 10% di mappe Range-Doppler contenenti solo rumore, senza target, per permettere alla rete di apprendere le caratteristiche spaziali del disturbo di fondo.

Come mostrato in figura 4.8, il ri-addestramento ha migliorato le prestazioni di rilevamento. Il miglioramento maggiore si è osservato nel caso peggiore, ovvero con rumore rosa a 30 dB (linee verdi nella figura). Tuttavia, si osserva ancora una perdita di prestazioni proporzionale all'intensità del rumore colorato. Questo comportamento è dovuto al fatto che il rumore colorato sovrasta il target, in particolare quando quest'ultimo si trova in prossimità di Doppler nullo.

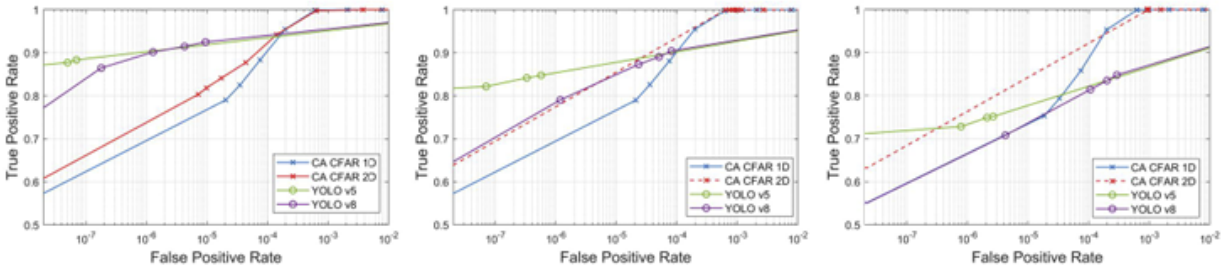



**Figure 4.8** Le curve ROC in presenza di rumore rosa a 10 dB (a sinistra), 20 dB (centrale) e 30 dB (destra) mostrano che, nei grafici (centrale) e (destra), la linea rossa tratteggiata evidenzia come il metodo CA CFAR 2D non sia in grado di mantenere un tasso di falsi positivi inferiore a $10^{-3}$, nonostante la probabilità teorica di falso allarme sia impostata a meno di $10^{-8}$.

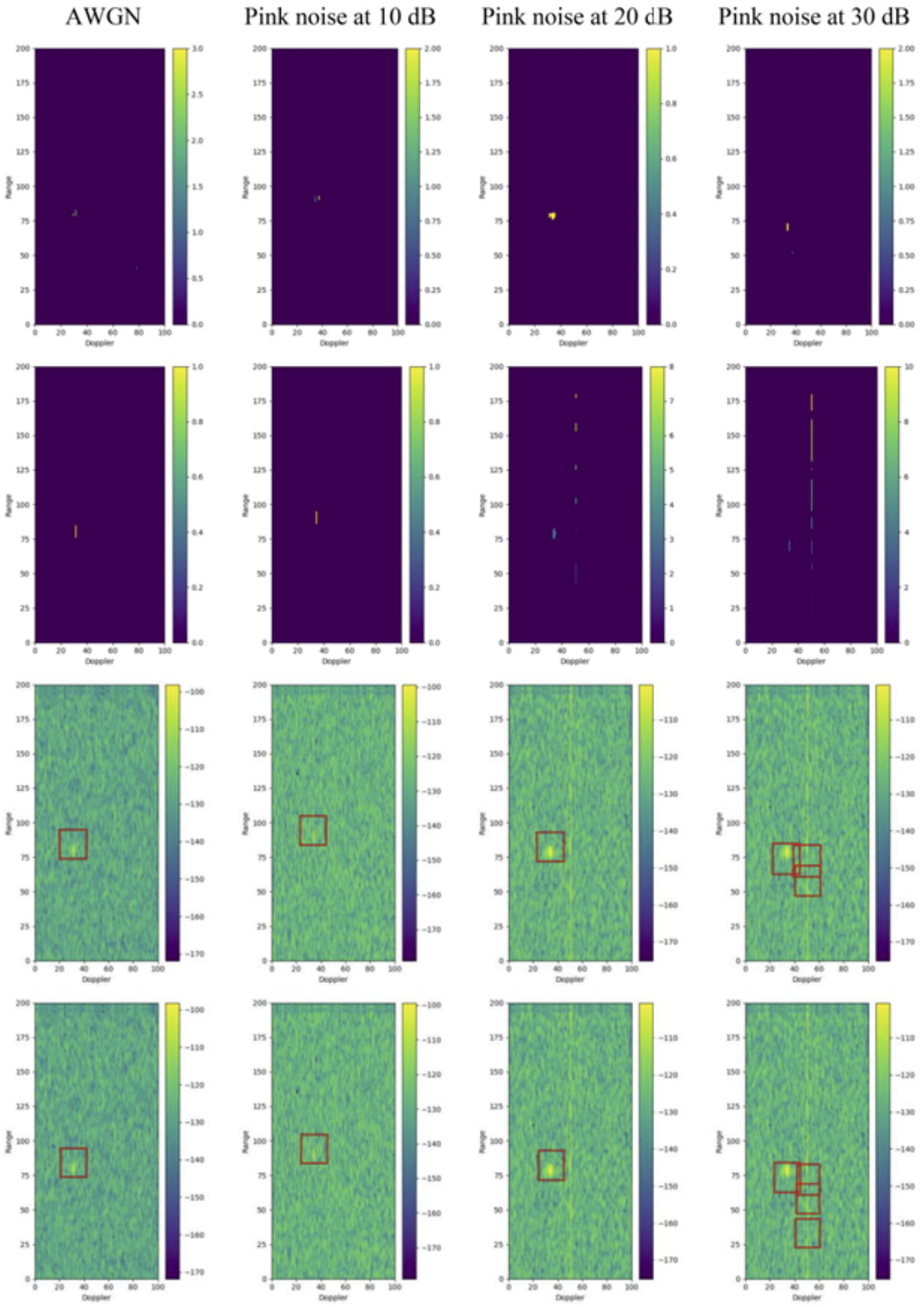



**Figure 4.9** Esempi di rilevamento dei target per CA CFAR 1D (prima riga), CA CFAR 2D (seconda riga), YOLOv5 (terza riga) e YOLOv8 (quarta riga). La barra dei colori in (prima riga) e (seconda riga) rappresenta il numero di target rilevati nella mappa Range-Doppler, mentre in (terza riga) e (quarta riga) è mostrata la mappa Range-Doppler originale (la stessa per ciascuna colonna), con sovrapposte le predizioni fornite da YOLO. Nei casi (prima riga) e (seconda riga), la probabilità di falso allarme è PFA $= 10^{-6}$, mentre per YOLO il livello di confidenza è pari a 0,5.

Le prestazioni ottenute indicano che le reti YOLO possono sostituire CFAR nei radar ground-based per la SSA, con maggiore robustezza in condizioni di rumore non stazionario. Nella tabella 4.3 vengono riassunti i risultati in termini di precision, recall e AUC.

| Metodo | Precision | Recall | AUC | Robustezza al rumore |
|---|---|---|---|---|
| CA CFAR 1D | 0.76 | 0.65 | 0.82 | Bassa |
| YOLOv5 | 0.88 | 0.84 | 0.94 | Alta |
| YOLOv8 | 0.91 | 0.89 | 0.96 | Molto alta |

**Table 4.3** Confronto delle prestazioni tra metodi di rilevamento

### 4.5.3 Considerazioni Finali

Abbiamo modellato il radar europeo TIRA in modalità di tracciamento al fine di generare dati sintetici per l'addestramento e il test dei modelli. Successivamente, abbiamo confrontato i sistemi di rilevamento radar classici con un rivelatore basato su YOLO. La valutazione in un ambiente simulato ha evidenziato che il rilevamento tramite YOLO supera l'approccio classico, garantendo un elevato tasso di rilevamento e mantenendo al contempo bassi i tassi di falsi allarmi.

Le future attività di ricerca si concentreranno sull'estensione dei framework di deep learning ad altri tipi di radar e radiotelescopi, nonché sull'integrazione di nuove metodologie orientate al miglioramento della sicurezza e della consapevolezza situazionale nello spazio (Space Situational Awareness, SSA).

Nel capitolo successivo affrontiamo il problema dell'ambiguità nella stima della distanza (range ambiguity), proponendo un approccio basato sul rilevamento multi-PRF (Pulse Repetition Frequency) per risolvere tale criticità.

# 5 Deep learning per la rilevazione di oggetti spaziali in radar con PRF multiple

Uno dei principali problemi nell'elaborazione dei segnali radar è rappresentato dall'ambiguità nelle misure di distanza (range) e velocità radiale (legata alla frequenza Doppler). In particolare, l'intervallo tra impulsi consecutivi, noto come Pulse Repetition Interval (PRI), il cui reciproco è la Pulse Repetition Frequency (PRF), definisce un limite massimo (in inglese, maximum non-ambiguous range) oltre il quale il sistema non è più in grado di determinare in modo univoco la distanza del bersaglio all'interno del Coherent Pulse Interval (CPI), a causa del fenomeno di aliasing. Allo stesso modo, la combinazione tra PRF e frequenza di portante introduce una soglia massima per la velocità osservabile (in inglese, maximum non-ambiguous Doppler frequency), generando ambiguità anche nella stima della frequenza Doppler. Di conseguenza, un bersaglio può essere erroneamente localizzato a una distanza o con una velocità diversa da quella reale.

La Fig 5.1 proveniente dal lavoro [125] evidenzia questo aspetto cruciale della radaristica, in particolare dell'ambiguità di range. In questa rappresentazione, il fenomeno viene illustrato mostrando come un target, che in realtà si trova a una certa distanza, possa essere erroneamente percepito a una distanza inferiore a causa del ritardo del segnale di ritorno. Questo ritardo si verifica quando l'eco del target arriva nel Coherent Processing Interval (CPI) successivo, un intervallo di tempo in cui il radar elabora i segnali ricevuti.

Nel contesto del radar a impulsi, l'ambiguità di range si verifica quando la frequenza di ripetizione degli impulsi (PRF, Pulse Repetition Frequency) è tale da non permettere una corretta distinzione tra le distanze di target che si trovano a più di un certo intervallo (legato alla velocità di propagazione del segnale e alla frequenza di campionamento). Quando un impulso viene trasmesso e il segnale di ritorno non arriva in tempo prima che il radar trasmetta il successivo, il target potrebbe essere erroneamente localizzato a una distanza più vicina di quanto non lo sia realmente, creando così un aliasing di range.

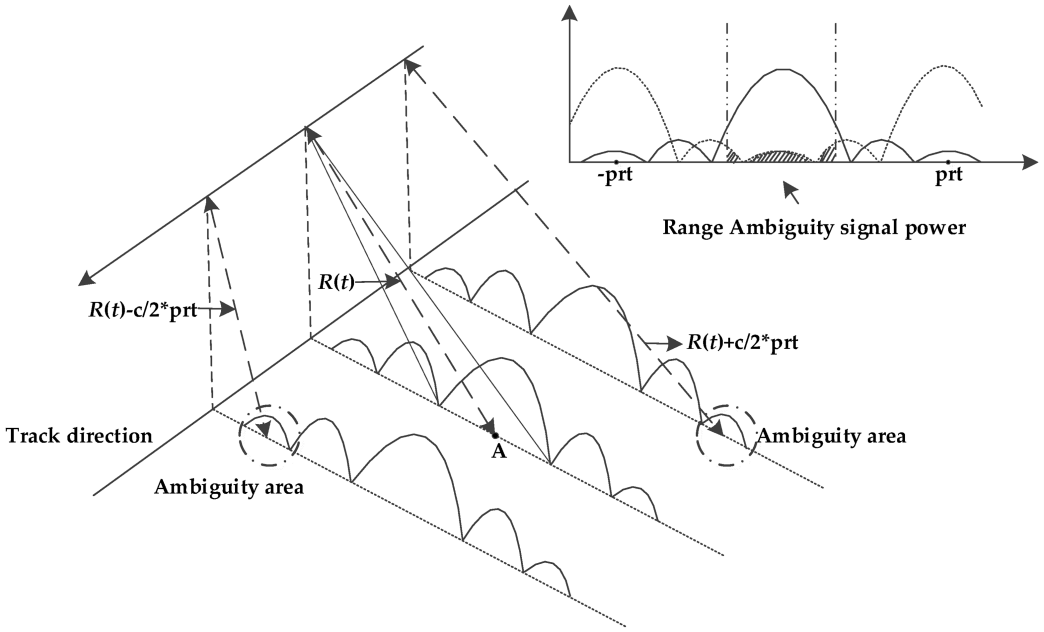



**Figure 5.1** Illustrazione dell'ambiguità del Range

L'obiettivo di questo lavoro è affrontare tali ambiguità utilizzando mappe radar acquisite con PRF differenti, e impiegare sia detector tradizionali, quali CFAR, sia innovativi modelli di apprendimento automatico per identificare correttamente i bersagli reali. Inoltre, il sistema mira a stimare con precisione i parametri fisici di interesse quali distanza, velocità e Radar Cross Section (RCS), superando i limiti delle tecniche tradizionali e migliorando l'affidabilità del rilevamento in scenari complessi.

La figura 5.2 illustra il processo di acquisizione radar suddiviso in più puntamenti, ciascuno dei quali corrisponde a un'osservazione dell'ambiente in una specifica direzione (identificato dagli angoli in elevazione e azimutali) ad un dato istante temporale. Per ogni puntamento, vengono inviati più gruppi (o burst) di impulsi a PRF differenti (PRF1 e PRF2). In ricezione vengono quindi processati in modo tale da ottenere 2 mappe range-Doppler con l'obiettivo di affrontare le ambiguità su range e Doppler: infatti ciascuna PRF genera aliasing differenti. Nella figura, le mappe sono rappresentate come finestre oblique contenenti puntini blu, che indicano la presenza di un bersaglio (un debris spaziale, nel nostro caso). Non tutti i puntamenti mostrano rilevamenti in entrambe le PRF: il target può risultare visibile in una sola mappa o in nessuna.

La pipeline di elaborazione mostrata in figura 5.3 si compone di più fasi, ciascuna delle quali contribuisce alla costruzione di un dataset strutturato per l'addestramento del modello neurale. Per ridurre la complessità del problema, si considera un solo puntamento, e dunque una sola coppia di mappe range-Doppler acquisite a PRF differenti.


**Figure 5.2** Puntamenti del Radar

- Vengono generate tramite simulazione due mappe: una con $PRF_1$ e una con $PRF_2$. Queste mappe contengono informazioni differenti sullo stesso puntamento, ciascuna caratterizzata da propri range e Doppler non ambigui.
- Su ciascuna mappa viene applicato un algoritmo di sogliaggio CFAR (Constant False Alarm Rate), che consente di identificare i possibili target, separando i segnali significativi dal rumore di fondo. Questo processo restituisce una serie di rilevamenti potenzialmente associabili a target reali.

  Per ogni rilevazione individuata (sia essa vera o dovuta al rumore), vengono estratti tre parametri fondamentali:

  - **Range**: la distanza apparente dal radar;
  - **Doppler**: che riflette la velocità radiale relativa;
  - **SNR** (Signal-to-Noise Ratio): indicatore della qualità del segnale rilevato.
- Le detection provenienti da $PRF_1$ e $PRF_2$ vengono combinate in tutte le possibili coppie. Questo passaggio consente di generare anche accoppiamenti ambigui o non corretti, utili per addestrare il modello a distinguerli. Nel caso in cui una delle due PRF non contenga alcuna detection, si procede con la replica dei dati disponibili per preservare la struttura del dataset.
- Le coppie di detection così ottenute vengono fornite in ingresso a modelli di apprendimento automatico strutturati come una rete neurale *multi-head*. I modelli eseguono due compiti principali:

  - **Regressione**, per stimare i valori corretti di range e SNR associati al target reale;
  - **Classificazione**, per determinare se la coppia in esame rappresenta un'associazione valida oppure no.

Il risultato finale è un vettore della forma:

$$[\text{range}, \text{snr}, \text{valid}]$$

dove `valid` è un flag binario che indica se la combinazione tra le due detection è plausibile e rappresenta un target reale, contribuendo così sia alla risoluzione delle ambiguità di misura che alla reiezione di false rilevazioni dovute al rumore.

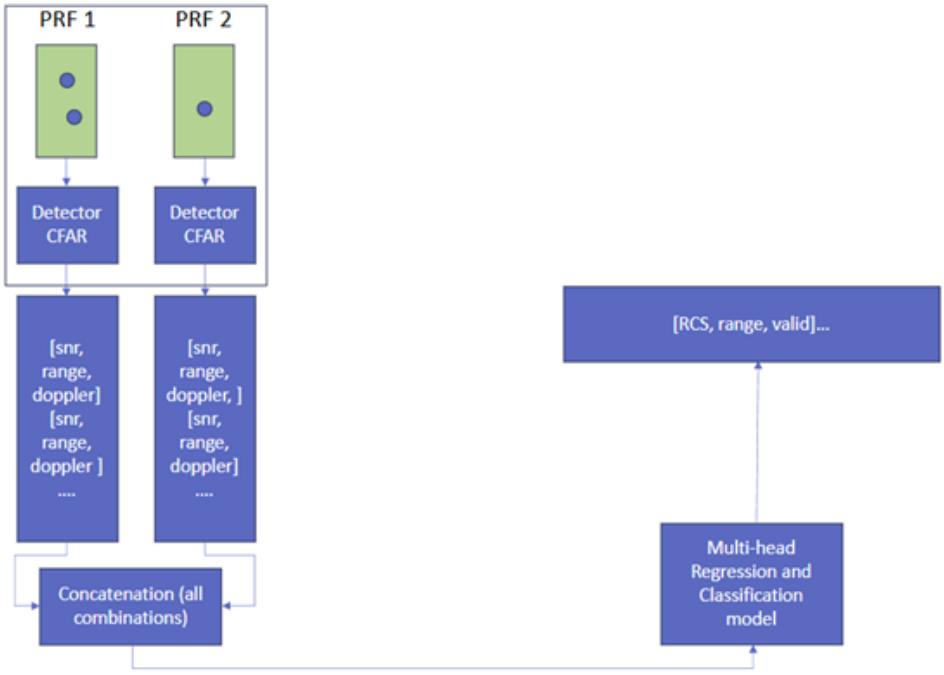



**Figure 5.3** Pipeline di elaborazione per la risoluzione del problema di ambiguità

## 5.1 Descrizione del radar e generazione delle mappe Range-Doppler

Come primo passo sono stati definiti i parametri principali che governano il comportamento del radar. Nella Tab 5.1 sono riportati i seguenti parametri fisici.

In questo progetto abbiamo voluto sperimentare l'impiego di due PRF differenti. Il radar preso come riferimento è reale e opera con due PRF distinte proprio per migliorare la risoluzione congiunta su distanza e velocità. Combinando le informazioni ottenute, è possibile disambiguare le misure e stimare in modo più accurato la posizione e la velocità dei target.

| **Parametro** | **Valore** |
|---|---|
| Velocità della luce | 299.792.458 m/s |
| Numero impulsi | 20 |
| Frequenza di campionamento | $3.698520 \times 10^6$ Hz |
| Distanza massima | 500 km |
| Frequenza operativa | $1.50 \times 10^9$ Hz |
| Potenza di picco | 1000 (lineare) |
| Guadagno dell'antenna | 57.78 dB |
| Durata dell'impulso | $1.00 \times 10^{-5}$ s |
| $PRF_1$ | 555 Hz |
| $PRF_2$ | 476 Hz |
| Temperatura del sistema | 300 K |
| Range di frequenza | 1–2 GHz |
| Range massimo non ambiguo $PRF_1$ | 268.584 km |
| Range massimo non ambiguo $PRF_2$ | 313.409 km |
| Velocità massima non ambigua $PRF_1$ | 27.63 m/s |
| Velocità massima non ambigua $PRF_2$ | 23.78 m/s |

**Table 5.1** Parametri radar e di misurazione utilizzati nella simulazione

Lo scenario radar è stato costruito simulando bersagli posizionati in tre distinti intervalli di distanza:

- 80–150 km: nessuna ambiguità per entrambe le PRF;
- 268–313 km: ambiguità per una sola delle due PRF;
- 320–450 km: ambiguità per entrambe le PRF.

Gli scaglioni sono stati scelti opportunamente per generare situazioni di ambiguità selettiva: in ciascun intervallo, i bersagli ricadono in una condizione di ambiguità rispetto a una delle due PRF. Questo approccio consente di studiare con maggiore precisione il comportamento del sistema radar e di testare la robustezza degli algoritmi di disambiguazione.

Per ciascuno scaglione di distanza, si eseguono le seguenti operazioni:

1. Assegnazione casuale della distanza e della Radar Cross Section (RCS) entro il range specificato;
2. Calcolo della posizione $(x, y, z)$ coerente con la distanza assegnata;
3. Generazione casuale della velocità del target;
4. Calcolo del segno della proiezione della velocità radiale, compresa tra $-40$ e $+40$ m/s, per determinare se il bersaglio si sta avvicinando o allontanando;

5. Calcolo della velocità totale relativa.

Ogni configurazione viene poi salvata in un file CSV con due righe: una per $PRF_1$ e una per $PRF_2$.

Il numero di bersagli da generare è definito dal parametro `N_targets`, impostato a 1. Anche la sezione radar (RCS) di ciascun bersaglio viene calcolata, poiché determina la "visibilità" del bersaglio da parte del radar. In questo progetto, la RCS è stata fatta variare per simulare oggetti con dimensioni comprese tra 2 e 12 cm.

Il radar monostatico invia impulsi verso i bersagli e raccoglie i segnali di ritorno. Il segnale riflesso dipende dalla RCS del bersaglio. Il canale di propagazione utilizzato simula la propagazione in spazio libero, modellando il comportamento del segnale in assenza di ostacoli o attenuazioni atmosferiche.

L'antenna del radar è direzionale e caratterizzata da un guadagno molto elevato nella direzione di puntamento. La potenza ricevuta dal radar viene calcolata utilizzando l'equazione radar.

$$P_r = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 R^4} \tag{5.1}$$

Questa equazione evidenzia come la potenza ricevuta diminuisca con il quarto della distanza dal bersaglio:

$$R^4 \tag{5.2}$$

implicando che il segnale ricevuto dal radar si indebolisce con l'aumentare della distanza. Inoltre, la potenza è direttamente proporzionale al guadagno dell'antenna (G), alla radar cross-section del bersaglio

$$\sigma \tag{5.3}$$

e al quadrato della lunghezza d'onda:

$$\lambda^2 \tag{5.4}$$

Una volta che i segnali vengono ricevuti dal radar, il codice applica un filtro correlato per migliorare la qualità del segnale ricevuto e ridurre il rumore. Questo passo è fondamentale per ottimizzare la rilevazione dei bersagli, in quanto i segnali radar sono frequentemente disturbati da rumore di fondo. Successivamente, viene eseguita l'analisi Doppler per determinare la velocità dei bersagli. Questa analisi sfrutta il cambiamento di frequenza dei segnali di ritorno per calcolare la velocità relativa tra il radar e il bersaglio, un passaggio cruciale per una corretta identificazione del movimento del bersaglio. Successivamente, vengono generate le mappe radar dove ogni mappa rappresenta un'istantanea della situazione in una determinata

finestra di distanza e velocità. Le mappe vengono anche convertite in formato immagine e TXT permettendo una visualizzazione chiara dei bersagli nel dominio spaziale e Doppler.

Ogni mappa radar è accompagnata da un'etichetta che descrive le caratteristiche del target, come RCS, Speed e Range. Questi risultati vengono salvati nell'apposito file CSV citato sopra.

## 5.2 CFAR Detector

Il secondo passo nella catena di elaborazione consiste nella segmentazione della mappa radar, al fine di individuare potenziali target. A questo scopo è stato implementato un algoritmo CFAR (Constant False Alarm Rate) nella sua variante monodimensionale (1D), applicato per ogni cella in range, fissata la frequenza Doppler. L'operazione viene ripetuta per tutte le frequenze Doppler disponibili.

Il CFAR è una tecnica largamente utilizzata in radaristica per il rilevamento di segnali in presenza di rumore e clutter, che consente di mantenere costante la probabilità di falso allarme anche in condizioni ambientali variabili. In particolare, il CFAR 1D opera confrontando ogni cella di test con una soglia calcolata dinamicamente a partire dal contenuto energetico delle celle circostanti, escludendo una finestra di guardia per evitare contaminazioni dovute al segnale del target.

L'utilizzo del CFAR monodimensionale, rispetto alla sua versione bidimensionale, è motivato da considerazioni computazionali e dalla struttura delle mappe Range-Doppler adottate. Nonostante la perdita di informazione spaziale completa, il CFAR 1D si è dimostrato efficace per il rilevamento preliminare dei target, offrendo un buon compromesso tra accuratezza e tempo di elaborazione, soprattutto in presenza di mappe di grandi dimensioni.

Una volta individuate le celle che superano la soglia imposta dal CFAR, è necessario procedere con l'aggregazione delle rilevazioni contigue che, verosimilmente, appartengono a uno stesso bersaglio. A tal fine, viene eseguita un'operazione di *clustering spaziale*, che consente di raggruppare celle adiacenti in oggetti coerenti.

Il clustering ha un duplice obiettivo:

- Ridurre la dimensionalità del problema, evitando di trattare ogni cella come un'entità autonoma;
- Migliorare l'accuratezza nella stima dei parametri del target, operando su insiemi più robusti di dati.

Per ciascun cluster identificato viene selezionato il *centroide*, ovvero il punto di massimo in termini di ampiezza del segnale. Questo consente di stimare, con buona affidabilità il range e la velocità radiale (Doppler) associati al bersaglio.

Viene anche calcolato il rapporto segnale-rumore (SNR), che rappresenta un indicatore fondamentale per valutare la qualità del rilevamento. Durante l'elaborazione, per ciascuna mappa viene selezionato il target rilevato più vicino alla posizione attesa, entro una soglia prestabilita minore di 5 km. Se tale corrispondenza è presente, si assume che il sistema abbia rilevato correttamente il target reale.

Questa fase è essenziale per la successiva classificazione dei dati: permette infatti di etichettare positivamente i rilevamenti corretti e di selezionare, i falsi positivi (è stato scelto un massimo di 10) che verranno utilizzati per addestrare modelli in grado di distinguere target reali da rumore.

Durante questa fase vengono salvati i parametri di ogni target individuato, insieme all'etichetta binaria che indica la correttezza del rilevamento. L'output della procedura consiste in due file `.csv`, uno per ciascuna PRF ($PRF_1$ e $PRF_2$), contenenti le seguenti colonne:

- `filename`$_{(1\text{ o }2)}$;
- `range`$_{(1\text{ o }2)}$;
- `doppler`$_{(1\text{ o }2)}$;
- `snr_db`$_{(1\text{ o }2)}$;
- `target_id` (0 se falso positivo, 1 altrimenti).

## 5.3 Preparazione del dataset per gli algoritmi di Deep-Learning

Una volta ottenuti i file `.csv` contenenti i dati acquisiti con $PRF_1$ e $PRF_2$, è stato creato un dataset unificato, utilizzabile per l'addestramento di una rete neurale. In questo processo, i dati relativi a *range*, *Doppler* e *SNR* vengono concatenati tra le due mappe radar, garantendo che ogni rilevazione disponga delle informazioni provenienti da entrambe le fonti.

Nel caso in cui una delle due mappe non presenti una rilevazione corrispondente, i valori mancanti vengono replicati nel dataset per mantenere la coerenza strutturale tra $PRF_1$ e $PRF_2$. Inoltre, per ciascuna rilevazione, viene calcolata una colonna denominata `valid`, che indica se i dati sono validi o meno:

- `valid` = 1 se esiste una corrispondenza tra $PRF_1$ e $PRF_2$;

- `valid` = 0 altrimenti.

Oltre a queste informazioni, vengono aggiunte colonne aggiuntive fondamentali per l'analisi e la supervisione dei dati,come RCS (Radar Cross Section), Range (distanza dal radar) e Speed (velocità del target).

Questi valori sono calcolati dalla generazione delle mappe range-Doppler (salvati su un CSV) e sono cruciali per l'addestramento delle reti neurali, poiché rappresentano la ground truth.

Il risultato finale è un dataset completo, che contiene tutte le informazioni necessarie per allenare l'algoritmo di intelligenza artificiale. I dati concatenati avranno le seguenti colonne:

- `filename1`
- `range1`
- `doppler1`
- `snr_db1`
- `filename2`
- `range2`
- `doppler2`
- `snr_db2`
- `valid`
- `RCS`
- `SNR`
- `Speed`
- `Range`

Questo dataset in formato .csv, una volta preparato, è pronto per essere utilizzato nel processo di training delle reti neurali, finalizzato all'analisi e alla previsione dei target radar.

Tutte le possibili casistiche sono riassunte nella Tabella 5.2.

## 5.4 Approccio basato su Deep Learning

Nel presente lavoro sono state sviluppate due reti neurali separate: una per la classificazione e una per la regressione. Entrambe le reti condividono un'idea architetturale comune basata su

| Casistica | $PRF_1$ | $PRF_2$ | Valid | Note |
|---|---|---|---|---|
| Presente $PRF_1$ e corrispettivo $PRF_2$ con `target_id = 1` | [snr1_1, range1_1, doppler1_1, tempo1_1] | [snr2_1, range2_1, doppler2_1, tempo2_1] | 1 | Merge riuscito: entrambi i PRF rilevano lo stesso target |
| Presente solo in $PRF_1$ | [snr1_1, range1_1, doppler1_1, tempo1_1] | Copia da $PRF_1$ | 1 | Valori della PRF mancante riempiti con quella esistente |
| Presente solo in $PRF_2$ | Copia da $PRF_2$ | [snr2_1, range2_1, doppler2_1, tempo2_1] | 1 | Valori della PRF mancante riempiti con quella esistente |
| Entrambi con `target_id = 0` | [snr1_1, range1_1, doppler1_1, tempo1_1] | [snr2_1, range2_1, doppler2_1, tempo2_1] | 0 | Target non valido |
| Solo uno dei due con `target_id = 1` | [snr1_1, range1_1, doppler1_1, tempo1_1] | [snr2_1, range2_1, doppler2_1, tempo2_1] | 0 | Target non valido |

**Table 5.2** Casistiche possibili nel processo di fusione dei dati tra $PRF_1$ e $PRF_2$. Le righe verdi indicano rilevamenti validi, quelle rosse rilevamenti non validi.

un ramo centrale che estrae rappresentazioni astratte dai dati radar in ingresso, ma differiscono nel numero e nella funzione delle teste.

Un elemento chiave introdotto in entrambe le architetture è il meccanismo di self-attention, applicato sul ramo condiviso (e anche sulle due teste di regressione). Questo componente consente al modello di pesare dinamicamente le informazioni in ingresso, adattando l'importanza di ciascuna feature in funzione del contesto globale. La self-attention è stata inizialmente proposta nel lavoro [36] dove ha rivoluzionato il campo del natural language processing. Successivamente, è stata adattata con successo alla visione artificiale tramite architetture come i Vision Transformer (ViT)[126] dimostrandosi efficace anche su dati strutturati spazialmente. Studi recenti [127] hanno evidenziato come possa catturare relazioni complesse anche in scenari ad alta dimensionalità e ambiguità, come quelli radaristici.

L'utilizzo congiunto della multi-head architecture e della self-attention apporta numerosi vantaggi. La struttura multi-head consente al modello di affrontare più task in modo sinergico, favorendo una generalizzazione migliore grazie alla condivisione di conoscenza nel ramo centrale. Questo approccio si è rivelato particolarmente utile in ambiti complessi, dove i compiti di classificazione e regressione condividono parte delle informazioni di input ma richiedono specializzazioni distinte nelle fasi finali [128].

### 5.4.1 Regressione

La prima rete è quella di regressione, con il compito di prevedere l'SNR e il Range, nella quale sono presenti i seguenti componenti funzionali:

- Il ramo condiviso elabora l'input attraverso più strati *fully connected*, combinati con tecniche di *Batch Normalization*, funzioni di attivazione non lineari e *dropout* per ridurre il rischio di overfitting. A questo livello, il modello estrae le caratteristiche generali, che vengono poi utilizzate nelle fasi successive per la previsione dei parametri.
- Self-Attention sul ramo condiviso: Su questo ramo viene applicato un meccanismo di *Self-Attention*, che consente al modello di focalizzarsi su informazioni rilevanti anche a lungo termine. Questo meccanismo migliora la capacità di rappresentare sequenze di dati, permettendo di modellare le dipendenze tra le variabili in ingresso. In tal modo, si migliora la qualità delle previsioni sia per il compito di regressione che per quello di classificazione.
- La rete presenta due teste distinte:
  - la prima dedicata alla previsione dello **SNR**;
  - la seconda dedicata alla previsione del **Range**.

  Ogni testa è composta da uno strato *fully connected*, seguito da *Batch Normalization*, attivazione non lineare e *dropout*. Le uscite di ciascuna testa forniscono valori continui, corrispondenti alle stime dei parametri fisici.

  Anche all'interno delle teste di regressione è stato implementato un meccanismo di *Self-Attention*, per migliorare ulteriormente la capacità del modello di catturare dipendenze rilevanti e affinare la precisione della previsione.

L'architettura complessiva della rete di regressione è illustrata in figura 5.4.

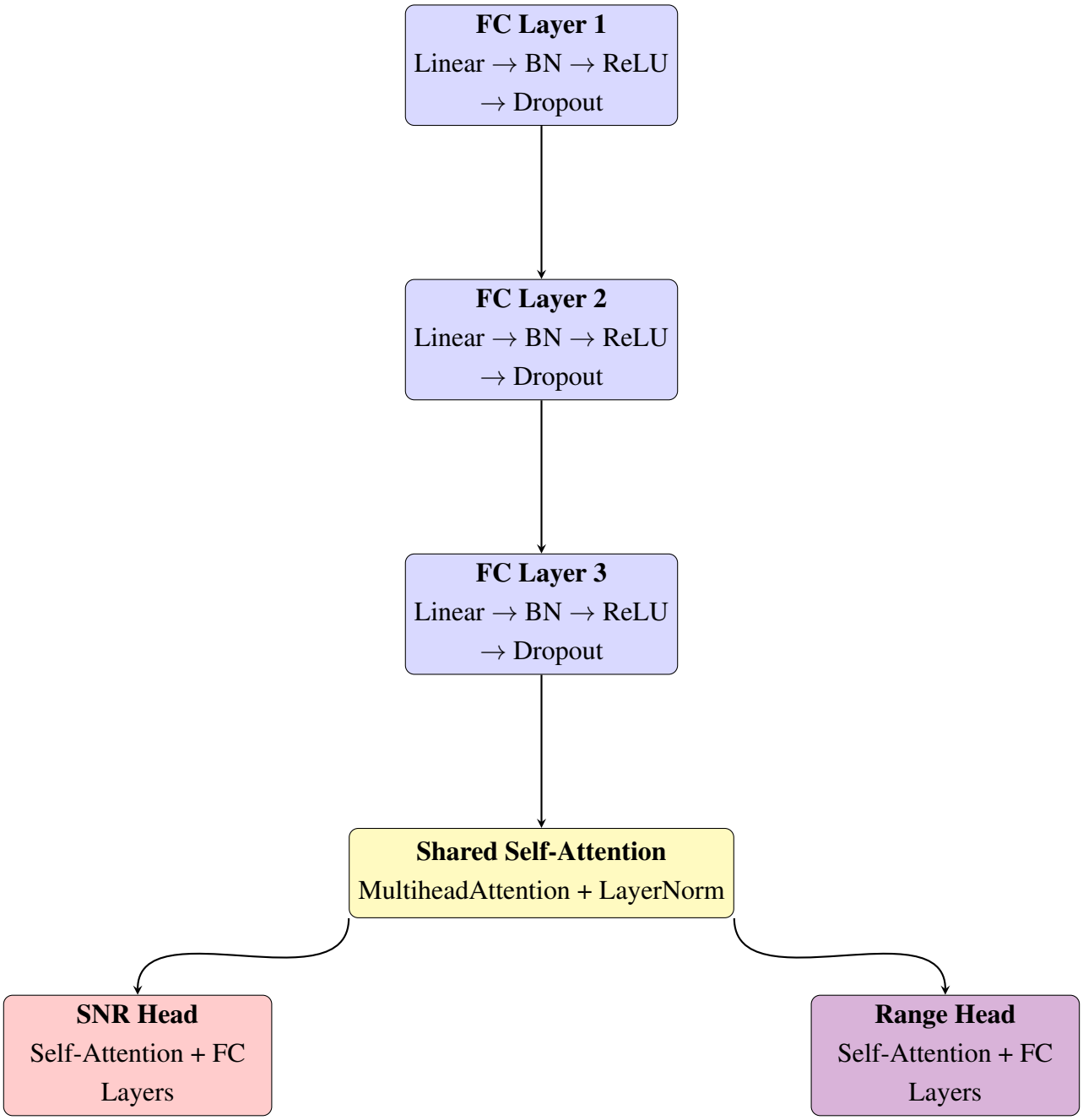



**Figure 5.4** Schema a blocchi della rete di regressione con Self-Attention condivisa e teste separate per SNR e Range.

### Training

Un aspetto importante dell'addestramento per la regressione è stata la scelta degli iper-parametri che possono fare una grande differenza nelle performance del modello.

In questo caso, è stato utilizzato Optuna, una libreria python che esegue una ricerca automatica degli iper-parametri, utilizzando tecniche avanzate di ottimizzazione. Optuna esplora diverse combinazioni di iper-parametri e trova quella che porta ai migliori risultati. In tab 5.3 abbiamo un elenco e intervalli degli iper-parametri presi in analisi per il modello.

| **Descrizione** | **Intervallo o Scelta** |
|---|---|
| Tasso di apprendimento (learning rate) | [1e-5, 1e-1] |
| Decadimento del peso (weight decay) | [1e-7, 1e-2] |
| Dimensione del batch | [32, 128] (step=32) |
| Numero di epoche | [20, 100] |
| Tasso di dropout | [0.1, 0.7] |
| Funzione di attivazione | `['ReLU', 'LeakyReLU']` |
| Momento per l'ottimizzatore SGD | [0.5, 0.9] |
| Scelta dell'ottimizzatore | `['Adam', 'AdamW', 'SGD']` |
| Numero di epoche di pazienza per gli scheduler | [5, 20] |
| Numero di layer nel modello | [2, 5] |
| Numero di neuroni per layer | [64, 1024] |
| Funzione di perdita | `['MSE', 'SmoothL1', 'L1']` |
| Tipo di scheduler per il tasso di apprendimento | `['ReduceLROnPlateau', 'CosineAnnealingLR', 'OneCycleLR']` |

**Table 5.3** Iperparametri esaminati con Optuna

### Test

Per trovare i parametri che garantissero performance ottimali sono state utilizzate 200 trial. In tab 5.4 vengono riportati i parametri che sono risultati ottimali per il problema.

Sono riportati i seguenti risultati nelle figure 5.5 e 5.6:

- **Training Loss:** 0.0138
  Questo valore indica quanto bene il modello si sta adattando ai dati di addestramento. Il valore finale è relativamente basso ed è un buon segno, poiché suggerisce che il modello sta imparando correttamente a partire dai dati forniti in fase di training.

| Descrizione | Parametri Ottimi |
|---|---|
| Tasso di apprendimento (learning rate) | 0.0022 |
| Decadimento del peso (weight decay) | 8.036e-06 |
| Dimensione del batch | 96 |
| Numero di epoche | 68 |
| Tasso di dropout | 0.11 |
| Funzione di attivazione | LeakyReLU |
| Scelta dell'ottimizzatore | Adam |
| Numero di epoche di pazienza per gli scheduler | 17 |
| Numero di layer nel modello | 3 |
| Numero di neuroni per layer | 905 |
| Funzione di perdita | SmoothL1 |
| Tipo di scheduler per il tasso di apprendimento | ReduceLROnPlateau |

**Table 5.4** Valori ottimali degli iperparametri ottenuti tramite procedura di tuning.

- **Validation Loss:** 0.0132
  Essendo molto simile alla Training Loss, questo valore suggerisce che il modello non solo si adatta bene ai dati di addestramento, ma riesce anche a generalizzare efficacemente su dati non visti. Inoltre, non si osservano segnali evidenti di overfitting.


**Figure 5.5** Scatter plot della rete di regressione

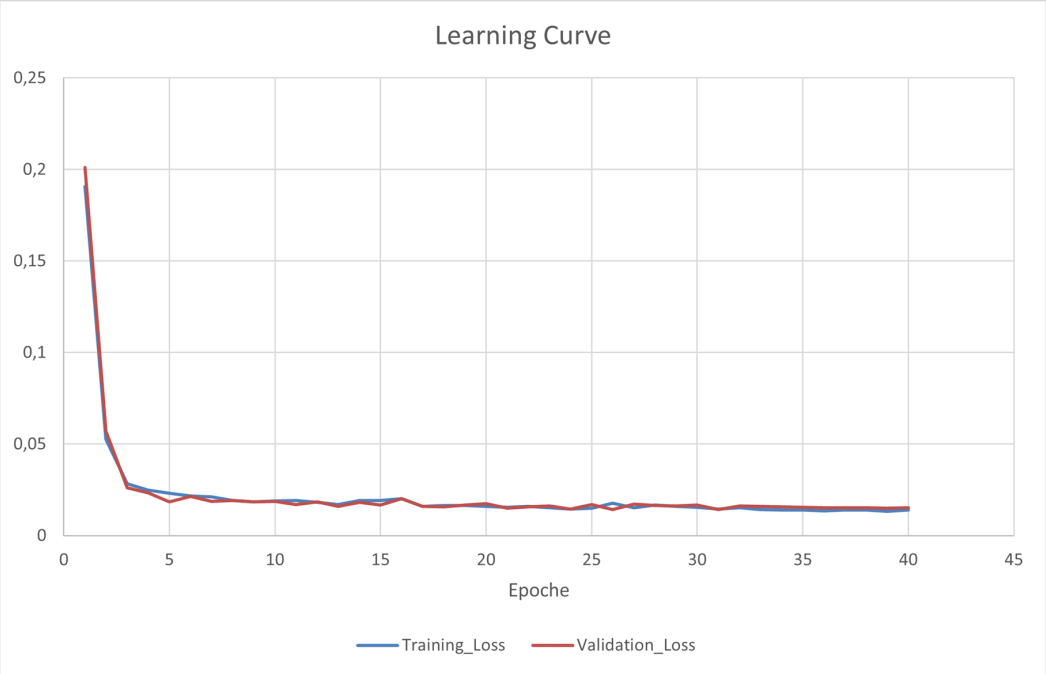



**Figure 5.6** Learning curve della rete di regressione

- **MAE SNR:** 0.0413
  Il modello è in grado di prevedere con buona precisione il valore dell'SNR, suggerendo un buon adattamento ai dati.
- **MAE Range:** 0.0759
  Questo valore indica una leggera imprecisione nella stima della distanza del bersaglio. Sebbene il valore sia comunque accettabile, mostra che il modello fatica un po' di più nel predire correttamente il range rispetto all'SNR.
- **Test Loss:** 0.0105
  Il modello dimostra una buona capacità di generalizzazione anche su dati completamente nuovi e non visti durante le fasi di addestramento e validazione.

| Distanza (Km) | MAE Range (Km) | MAE SNR (dB) |
|---|---|---|
| 100 | 10.972 | 3.299960 |
| 300 | 11.037 | 3.105901 |
| 400 | 41.389 | 4.666975 |

**Table 5.5** Valori di errore medio assoluto (MAE) per Range e SNR a diverse distanze.

Dai risultati in tab 5.5 si possono notare alcune tendenze significative legate alla distanza. In particolare, l'errore medio assoluto (MAE) sulla stima del range (distanza) aumenta

considerevolmente con l'aumento della distanza dal target. Per esempio, a una distanza di 100 km, l'errore medio è di circa 10.97 km, mentre a 400 km cresce drasticamente fino a 41.39 km. Questo suggerisce che il modello ha molta più difficoltà a stimare accuratamente la distanza quando il target è più lontano nonostante mantenga buoni risultati per le distanze 100-300 km.

Per quanto riguarda l'errore medio assoluto (MAE) sul SNR (rapporto segnale-rumore), possiamo osservare una tendenza simile, ma meno pronunciata. A 100 km, il MAE sul SNR è di 3.30 dB, mentre a 400 km cresce a 4.67 dB. Sebbene anche l'errore sul SNR aumenti con la distanza, lo fa in modo più contenuto rispetto al range, il che indica che il modello riesce a gestire meglio la previsione del SNR rispetto alla distanza quando il target è più lontano.

### 5.4.2 Classificazione

La seconda rete neurale è progettata per la classificazione, con l'obiettivo di prevedere la variabile valid. La variabile valid è di tipo binario e indica se il bersaglio è vero positivo o falso positivo. Il ramo condiviso è come quello della regressione. Successivamente, viene applicata una testa di classificazione dedicata, composta da strati fully connected,funzioni di attivazione, Dropout per regolarizzazione. Infine, un layer finale e attivazione sigmoid per output binario. Viene riportato il seguente diagramma a blocchi 5.7.

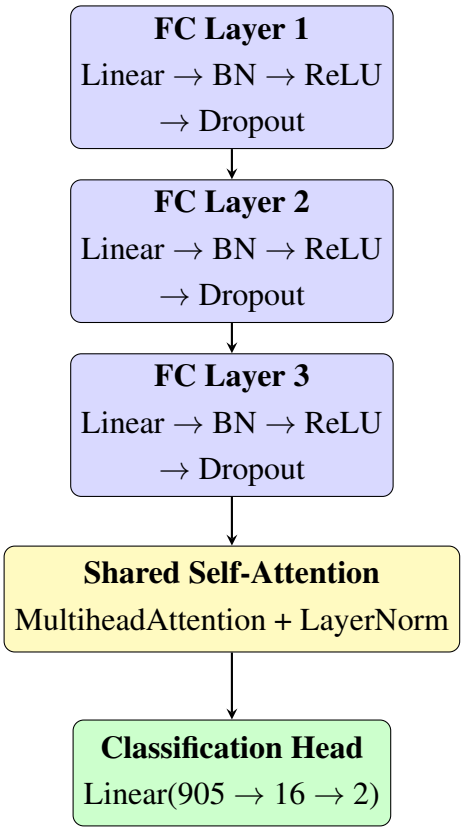



**Figure 5.7** Schema a blocchi della rete neurale con strato di self-attention condiviso.

### Training

Per la rete di classificazione è stato utilizzato un dataset contenente anche i falsi positivi. Le informazioni in input derivano da quattro caratteristiche: due valori di range e due di SNR. L'obiettivo del modello è distinguere tra target validi e non validi, restituendo in uscita un valore binario associato alla colonna valid. Per l'addestramento è stato adottato un approccio di transfer learning: la parte condivisa della rete (backbone) non è stata aggiornata poichè i pesi ottimi derivano dall'addestramento della rete di regressione, mentre la testa di classificazione è stata addestrata da zero. é stato fondamentale normalizzare tutti i valori secondo uno scaler specifico, nell'intervallo da 0 a 1.

La suddivisione dei dati in train, validation e test è stata rispettivamente per entrambe le reti per train, validation e test rispettivamente 60%,20% e 20%.

Nella rete di classificazione è stata utilizzata la focal loss per gestire lo sbilanciamento tra le classi, attribuendo maggiore peso ai veri positivi. Questo ha permesso al modello di concentrarsi maggiormente sugli esempi rilevanti, penalizzando i casi in cui un oggetto realmente presente non viene rilevato.

La curva in figura 5.8 mostra una diminuzione della training loss piuttosto contenuta nelle prime epoche, seguita da una fase di stabilizzazione, mentre la validation loss oscilla intorno a valori simili. La discesa della loss risulta quindi lenta e modesta, ma non si osservano segni di overfitting. Questo comportamento è coerente con il fatto che il modello rappresenta una rete di classificazione su cui è stato applicato il transfer learning: i pesi della rete sono stati inizializzati a partire dal modello addestrato, già ottimizzato sul task di regressione. Di conseguenza, la rete parte da una situazione già favorevole e il miglioramento nel corso dell'addestramento è solo marginale.

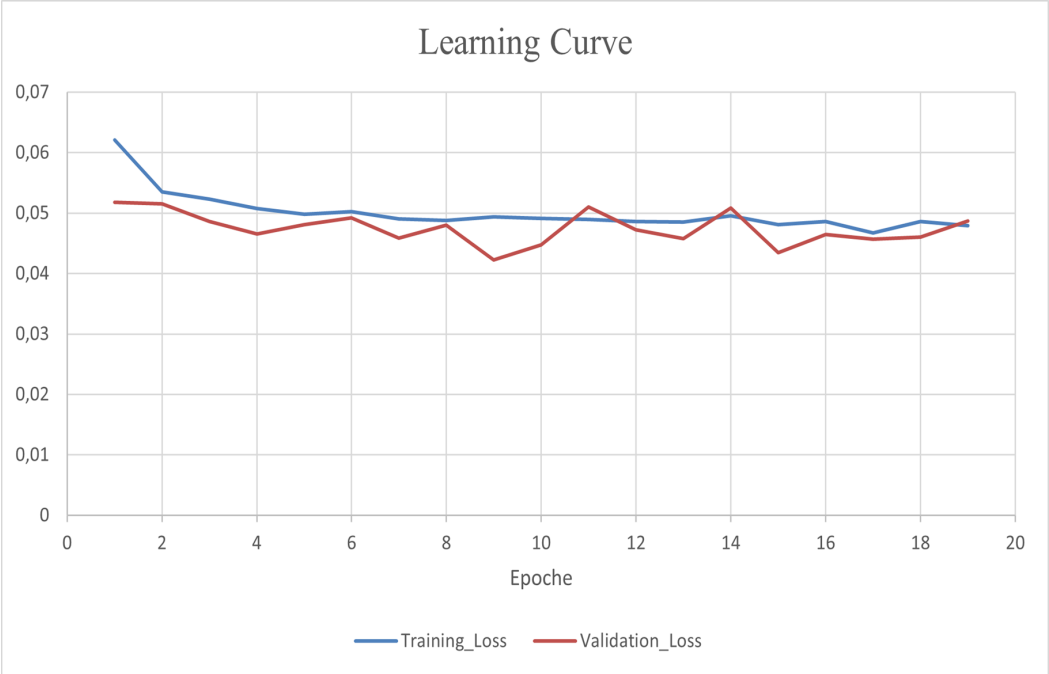



**Figure 5.8** Learning curve del modello di classificazione

### Test

In questa sezione siamo andati ad analizzare i risultati della classificazione in termini di confusion matrix 5.6 e curva ROC 5.9

| | **Predicted 0** | **Predicted 1** |
|---|---|---|
| **Class 0** | 0 | 0.26 |
| **Class 1** | 0.74 | 1 |

**Table 5.6** Confusion matrix normalizzata del classificatore. Class 1 rappresenta i target validi, Class 0 i falsi.

Un elemento importante da considerare nell'analisi della *confusion matrix* è lo sbilanciamento delle classi: nel dataset utilizzato per l'addestramento, i falsi positivi risultano essere numericamente molto più presenti rispetto ai veri positivi.

Nonostante questo squilibrio, la rete è in grado di classificare correttamente il 99% dei target validi, evidenziando un'elevata capacità di riconoscere i veri positivi, espressa da un *recall* molto alto. In un contesto radaristico, dove la rilevazione affidabile di un target reale è fondamentale, questa caratteristica rappresenta un punto di forza significativo.

Il modello riesce anche a identificare correttamente circa il 74% dei target non validi, mostrando una buona capacità di discriminazione anche nei confronti dei falsi. Tuttavia, a causa dello sbilanciamento e della forte presenza di falsi positivi nel dataset, la rete mostra una tendenza a sovrastimare la validità dei bersagli, portando a una misclassificazione del 26% dei falsi positivi come veri.

Il valore dell'AUC (*Area Under the Curve*) pari a 0,85 indica una buona capacità discriminativa del classificatore. Il modello è infatti in grado di distinguere in modo efficace tra classi positive (`valid = 1`) e negative (`valid = 0`). Sebbene non rappresenti un comportamento perfetto, questo valore suggerisce un buon compromesso tra sensibilità e specificità. Inoltre, l'introduzione della Self-Attention ha incrementato la robustezza del classificatore, migliorando il recall sui target veri in presenza di ambiguità.

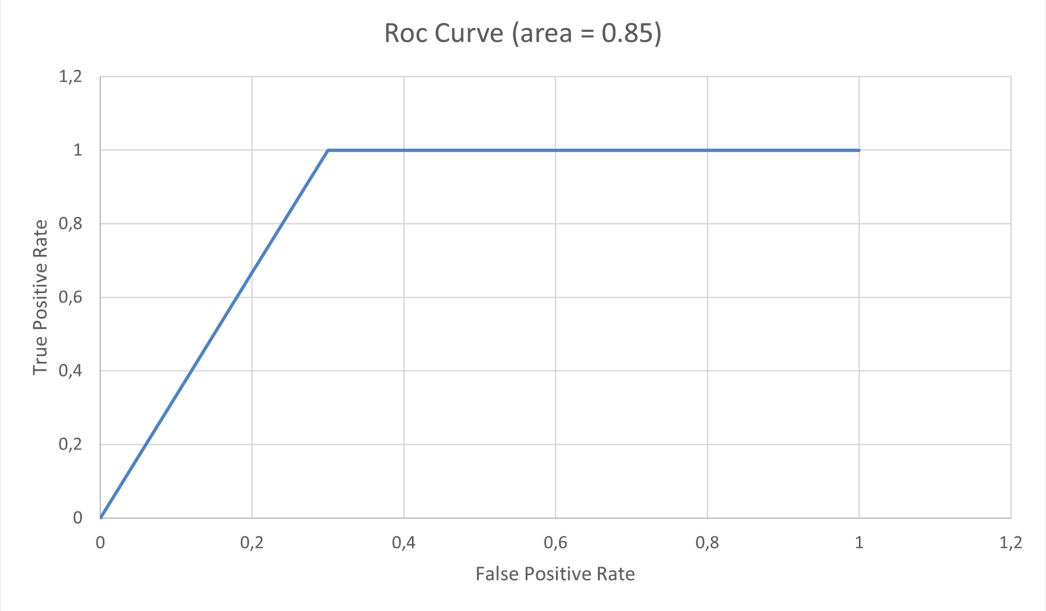



**Figure 5.9** Curva ROC del classificatore

## 5.5 Considerazioni Finali

I risultati ottenuti confermano l'efficacia dell'approccio proposto nell'affrontare le ambiguità di range utilizzando due PRF. La rete neurale ha mostrato una buona capacità di identificare correttamente i target reali, in particolare nelle distanze comprese tra 100 e 300 km, e una discreta precisione nella stima dei parametri fisici di interesse, soprattutto per quanto riguarda l' SNR. Tuttavia, sono emerse alcune criticità nella stima della distanza per target molto lontani, dove l'errore medio aumenta sensibilmente, indicando che il modello fatica a mantenere accuratezza in condizioni di segnale più debole e maggiore aliasing. Inoltre, lo sbilanciamento tra classi ha influito negativamente sulla capacità di riconoscere correttamente tutti i falsi positivi, portando a un certo numero di errori di classificazione.

Questo capitolo ha mostrato come architetture con self-attention condivisa possano gestire efficacemente dati radar da PRF multiple, combinando classificazione e regressione per un riconoscimento completo e robusto degli oggetti orbitali.

Sebbene i risultati siano incoraggianti, il sistema può essere ulteriormente migliorato. Una direzione promettente è l'estensione a più puntamenti radar, che permetterebbe di osservare i bersagli da angolazioni diverse e ridurre le ambiguità grazie alla ridondanza informativa, inoltre anche l'impiego di architetture neurali più sofisticate, capaci di catturare dipendenze spaziali e temporali più complesse, rappresenta un ulteriore passo verso l'applicazione operativa di questo approccio in contesti radar reali. Tra gli sviluppi futuri, si ipotizza l'estensione del framework a scenari con dati acquisiti da radar reali o tramite set sperimentali su campo, nonché l'integrazione con sistemi onboard come satelliti o droni. Inoltre, la struttura generale proposta potrebbe trovare applicazione anche in ambiti diversi, quali la cybersicurezza o l' identificazione di anomalie in reti neurali.

# Conclusioni

Il lavoro di ricerca presentato in questa tesi ha avuto come obiettivo principale affrontare le problematiche legate alla Space Situational Awareness. A partire dall'analisi delle esigenze operative dei sistemi radar, il progetto ha progressivamente integrato approcci tradizionali, come il rilevamento CFAR, confrontandoli con modelli neurali più evoluti e flessibili, capaci di apprendere strutture complesse e di adattarsi a contesti dinamici e rumorosi.

I risultati ottenuti dimostrano come l'impiego di reti come le YOLO possano migliorare significativamente le prestazioni dei sistemi radar, non solo in termini di accuratezza nella classificazione, ma anche nella capacità di rilevamento dei detriti e di generalizzazione a scenari non visti, riducendo il tasso di falsi allarmi. Le architetture con self-attention condivisa e tecniche multi-head, d'altra parte, hanno evidenziato il potenziale di queste soluzioni per affrontare problemi critici come l'ambiguità di range e la stima simultanea di parametri fisici, quali la distanza e il rapporto segnale-rumore.

Parallelamente, le esperienze condotte nei settori della sicurezza informatica e dell'elaborazione georadar hanno contribuito ad arricchire il lavoro, mostrando la versatilità dei modelli sviluppati e la possibilità di adattarli a domini molto diversi tra loro.

Dal punto di vista personale, questo percorso di dottorato ha rappresentato una tappa fondamentale nella mia formazione scientifica, permettendomi di approfondire strumenti teorici e pratici, di confrontarmi con solo con la comunità di ricerca ma anche di maturare una visione critica sul ruolo dell'IA nella società contemporanea. Il lavoro svolto ha confermato quanto possa essere importante la centralità dell'IA nei sistemi di monitoraggio avanzati, nonostante tuttavia sia ancora necessario continuare a interrogarsi sui limiti, l'affidabilità e l'etica di questi strumenti, soprattutto in ambiti sensibili come la sicurezza, l'ambiente e lo spazio.

Guardando al futuro, le tecniche proposte potranno essere estese, migliorate e validate su scenari operativi più complessi. La speranza è che questo lavoro possa offrire un contributo utile e concreto a queste sfide, aprendo nuove strade alla ricerca nel campo dell'IA applicata allo Spazio.

# References


[1] Precedence Research. Artificial intelligence market size, share, trends, growth 2024 to 2034. https://www.precedenceresearch.com/artificial-intelligence-market, 2025. Accessed: 2025-06-16.

[2] Khyati Hooda and Akash Mansukhani. 69 new ai market size stats to know for 2025. https://keywordseverywhere.com/blog/ai-market-size-stats/, December 2024. Accessed: 2025-06-16.

[3] Peter Dayan and L.F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press, 2001.

[4] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[5] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[6] Andrea Minini. Rete neurale semplice. https://www.andreaminini.com. Accessed: 2025-06-16.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[8] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[9] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2009.

[10] Yann LeCun et al. Self-supervised learning: The dark matter of intelligence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, 2019.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[14] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[15] David Silver, Aja Huang, Chris J Maddison, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[16] Frank Rosenblatt. *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*, volume 65. Psychological Review, 1958.

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[22] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[23] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021.

[24] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[25] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[26] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. In *Advances in Neural Information Processing Systems*, volume 34, pages 3965–3977, 2021.

[27] Kiruthika Sengodan, Balaji Raman, and Eswaran Duraisamy. Cbam-efficientnetv2: A deep learning framework for histopathological image classification. *Computerized Medical Imaging and Graphics*, 109:102276, 2024.

[28] Hani Alratrout, Turki Althobaiti, Sufyan Mahfooz, et al. Efficientnetv2-regnet fusion model for anomaly detection in iot networks. *Computers, Materials & Continua*, 78(1):1–17, 2024.

[29] Aldo Balderas, Héctor García, Genaro Zavala, et al. Ocnna: Optimal cnn architecture through pruning and knowledge distillation for edge ai. *IEEE Access*, 12:31712–31725, 2024.

[30] Google AI. Mobilenetv4. https://github.com/google-research/mobilenetv4, 2024. Accessed: 2025-04-19.

[31] Joseph Redmon, Santosh Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[32] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. 2018.

[33] Xiaoyang Wang, Yifan Zhang, and Chao Zhang. Yolov9: Improving object detection with attention mechanisms. *Journal of Machine Learning Research*, 24(45):1234–1247, 2023.

[34] Yun Liu, Wei Zhang, and Zhiqiang Wei. Yolov5: Enhancing real-time object detection performance with efficient backbone and anchor optimization. *IEEE Transactions on Image Processing*, 32(5):1557–1569, 2023.

[35] Zhen Chen, Xun Xu, and Hui Zhang. Yolov12: Transformer-based object detection with direct bounding box selection. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):987–999, 2024.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems (NeurIPS)*, 2017.

[37] Alexey Dosovitskiy, Thomas Schultz, Michael Tschannen, and Luc Van Gool. Transformers in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(6):1847–1867, 2021.

[38] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, 2020.

[39] Xizhou Zhu, Yuting Xie, Junyu Dong, Wei Zhan, Zhiqiang Wei, Xiangyu Zhang, and Jian Sun. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[40] Ze Liu, Zhiwei Xie, Pingping Zhang, Yichang Li, Xiaojuan Qi, Qi Tian, and Lewei Lu. Swin transformer: Hierarchical vision transformer using shifted windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1724–1735, 2023.

[41] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Shakir Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[42] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Machine Learning (ICML)*, 2015.

[43] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. In *International Conference on Machine Learning (ICML)*, 2017.

[44] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[45] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[46] Christian Ledig, Lucas Theis, Ferenc Huszar, William T. Freeman, Chao Dong, Wenzhe Shi, and Michael Bechthold. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[47] Amr R. Mohamed, Hongsong Yao, Lin Zhang, Zhiqiang Zeng, Bohan Shiu, Bing Li, and Yu Wu. Video generation from text. *arXiv*, 2022.

[48] Yang Miao, Lin Jin, Hao Xu, Tian Li, and Wei Xu. Synthetic data generation using generative adversarial networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(7):3067–3079, 2021.

[49] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[50] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 1137–1143, 1995.

[51] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[53] A. Author. Wavelet analysis: A powerful tool for time-frequency signal analysis. *Journal of Signal Processing*, 34(2):55–71, 2020.

[54] B. Author. Spectrogram representation in signal processing: Applications and techniques. *International Journal of Signal Processing*, 29(4):180–195, 2021.

[55] C. Author. Temporal patterns in signal processing: A review of time-frequency representations. *Signal Processing Review*, 22(3):122–136, 2019.

[56] D. Author. Visual interpretation of physiological signals: From raw data to meaningful representations. *Biomedical Signal Processing*, 17(1):45–59, 2022.

[57] E. Author. Applications of convolutional neural networks in signal processing. *Journal of Machine Learning Applications*, 43(4):234–249, 2020.

[58] F. Author. Convolutional neural networks: Architecture, applications, and future directions. *International Journal of Artificial Intelligence*, 28(2):65–81, 2019.

[59] G. Author. Pattern recognition in time-frequency representations: Deep learning approaches. *Journal of Pattern Recognition*, 39(5):105–120, 2022.

[60] H. Author. Hierarchical feature extraction with cnns for time-frequency analysis. *Neurocomputing*, 155:263–277, 2019.

[61] I. Author. Speech recognition using spectrograms: A convolutional neural network approach. *IEEE Transactions on Audio, Speech, and Language Processing*, 28(7):1213–1227, 2020.

[62] J. Author. Eeg signal analysis for medical diagnostics using deep learning. *Journal of Neuroengineering*, 14(3):123–138, 2022.

[63] K. Author. Machinery fault detection using spectrograms: A deep learning approach. *Mechanical Engineering Journal*, 45(8):800–815, 2021.

[64] Donald J. Kessler and Burton G. Cour-Palais. Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research*, 83:2637–2646, 1978.

[65] European Space Agency. Space debris by the numbers 2024, 2024. Accessed: 2024-04-27.

[66] James R. Wertz, David F. Everett, and Jeffery J. Puschell. *Space Mission Engineering: The New SMAD*. Microcosm Press, 2011.

[67] Jamie McClelland et al. Artificial intelligence for space applications: A review. *Acta Astronautica*, 192:38–51, 2022.

[68] Alessandro Rossi and Giovanni B. Valsecchi. Artificial intelligence and space debris mitigation. *Advances in Space Research*, 66(1):153–164, 2020.

[69] Jer-Chyi Liou. An active debris removal parametric study for leo environment remediation. *Advances in Space Research*, 47(11):1865–1876, 2008.

[70] Heiner Klinkrad. *Space Debris: Models and Risk Analysis*. Springer, 2006.

[71] European Space Agency. Evolution of space objects in all orbits. https://www.esa.int/Safety_Security/Space_Debris/Space_debris_by_the_numbers, 2023. Accessed: 2025-06-16.

[72] Jason C. Crusan et al. Nasa's lunar exploration program overview. In *Proceedings of the IEEE Aerospace Conference*, pages 1–9, 2018.

[73] Aimee van Wynsberghe et al. Ethics of artificial intelligence in space missions. *Nature Astronomy*, 7:403–410, 2023. Opportunities and risks of AI in space exploration.

[74] Mark Williamson. Space governance: Addressing the challenges of space debris. *Space Policy*, 51:101–104, 2020.

[75] Inioluwa Deborah Raji and Joy Buolamwini. Actionable auditing: Investigating the impact of publicly naming biased performance results of commercial ai products. *Proceedings of the 2020 AAAI/ACM Conference on AI, Ethics, and Society*, page 429–435, 2020.

[76] Market.us. Ai in space exploration market share by application, 2023. https://market.us/report/ai-in-space-exploration-market/, 2023. Accessed: 2025-06-16.

[77] The Business Research Company. Ai in space exploration global market report 2025. https://www.thebusinessresearchcompany.com/report/ai-in-space-exploration-global-market-report, 2025. Accessed: 2025-06-16.

[78] G. Muntoni et al. Crowded space: a review on radar measurements for space debris monitoring and tracking. *Applied Sciences*, 11(4):1364, 2021.

[79] J. Ender et al. Radar techniques for space situational awareness. In *Proc. 12th International Radar Symposium (IRS)*, pages 21–26, Leipzig, Germany, 2011.

[80] H. Wilden et al. Gestra — a phased-array based surveillance and tracking radar for space situational awareness. In *Proc. IEEE International Symposium on Phased Array Systems and Technology*, pages 1–5, Waltham, USA, 2016.

[81] A. Podda et al. Exploitation of bi-static radar architectures for leo space debris surveying and tracking: the birales/biralet project. In *Proc. IEEE Radar Conference*, pages 1–6, Florence, Italy, 2020.

[82] J. Vierinen et al. Use of eiscat 3d for observations of space debris. In *Proc. ESA Space Debris Office*, volume 7, 2017.

[83] Cosmo-skymed. https://earth.esa.int/eogateway/missions/cosmoskymed, accessed 8 May 2023.

[84] E-sar–the airborne sar system of dlr. https://www.dlr.de/hr/en/desktopdefault.aspx/tabid-2326/3776_read-5679/, accessed 8 May 2023.

[85] M. Maffei et al. An ontology for spaceborne radar debris detection and tracking: channel-target phenomenology and motion models. *IEEE Aerospace and Electronic Systems Magazine*, 36(6):18–42, 2021.

[86] M. Maffei et al. Spaceborne radar sensor architecture for debris detection and tracking. *IEEE Transactions on Geoscience and Remote Sensing*, 59(8):6621–6636, 2021.

[87] M. Maffei et al. Mimo sbr via code division multiplexing for track while simultaneous search. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–14, 2022.

[88] M. Maffei et al. Effects of plasma media with weak scintillation on the detection performance of spaceborne radars. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–13, 2022.

[89] V. Kothari, E. Liberis, and N.D. Lane. The final frontier: Deep learning in space. *arXiv*, arXiv:2001.10362, 2020.

[90] R.K. Mishra, G.Y.S. Reddy, and H. Pathak. The understanding of deep learning: A comprehensive review. *Mathematical Problems in Engineering*, 2021:5548884, 2021.

[91] W. Li, K. Wang, and L. You. A deep convolutional network for multitype signal detection and classification in spectrogram. *Mathematical Problems in Engineering*, 2021:9797302, 2021.

[92] T.J. Saleem and M.A. Chishti. Deep learning for the internet of things: Potential benefits and use-cases. *Digital Communications and Networks*, 7:526–542, 2021.

[93] J. Tao, Y. Cao, L. Zhuang, Z. Zhang, and M. Ding. Deep convolutional neural network based small space debris saliency detection. In *Proceedings of the 2019 25th International Conference on Automation and Computing (ICAC)*, pages 1–6, Lancaster, UK, September 2019.

[94] A. DeVittori, R. Cipollone, P. Di Lizia, and M. Massari. Real-time space object tracklet extraction from telescope survey images with machine learning. *Astrodyn*, 6:205–218, 2022.

[95] C. Pooja and K. Jaisharma. Novel framework for the improvement of object detection accuracy of smart surveillance camera visuals using modified convolutional neural network technique compared with support vector machine. In *Proceedings of the 2022 International Conference on Business Analytics for Technology and Security (ICBATS)*, pages 1–4, 2022.

[96] X. Zhang, Y. Liu, C. Huo, N. Xu, L. Wang, and C. Pan. Psnet: Perspective-sensitive convolutional network for object detection. *Neurocomputing*, 468:384–395, 2022.

[97] Y. Liu, M. Zhu, J. Wang, X. Guo, Y. Yang, and J. Wang. Multi-scale deep neural network based on dilated convolution for spacecraft image segmentation. *Sensors*, 22(4222), 2022.

[98] S. Long, H. Liu, X. Zhou, and H. Zhang. Long short-term memory (lstm) and its applications. Available online: https://arxiv.org/pdf/1909.09586.pdf. Accessed on 26 November 2022.

[99] C. Kim, J. Lee, T. Han, and Y.-M. Kim. A hybrid framework combining background subtraction and deep neural networks for rapid person detection. *J. Big Data*, 5:22, 2018.

[100] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. Available online: https://arxiv.org/pdf/1612.08242.pdf. Accessed on 26 November 2022.

[101] G. Liu, Y. Tan, L. Chen, W. Kuang, B. Li, F. Duan, and C. Zhu. The development of a uav target tracking system based on yolov3-tiny object detection algorithm. In *Proceedings of the 2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1636–1641, Sanya, China, December 2021.

[102] Q. Qi, H. Wang, T. Su, and X. Liu. Learning temporal information and object relation for zero-shot action recognition. *Displays*, 73:102177, 2022.

[103] J. Qin, H. Jiang, N. Lu, L. Yao, and C. Zhou. Enhancing solar pv output forecast by integrating ground and satellite observations with deep learning. *Renewable and Sustainable Energy Reviews*, 167:112680, 2022.

[104] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction, 2018. Available online: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf (accessed on 26 November 2022).

[105] B. Oakes, D. Richards, J. Barr, and J. Ralph. Double deep q networks for sensor management in space situational awareness. In *Proceedings of the 2022 25th International Conference on Information Fusion (FUSION)*, pages 1–6, Linköping, Sweden, 2022.

[106] W. Lei, H. Fu, and G. Sun. Active object tracking of free-floating space manipulators based on deep reinforcement learning. *Advances in Space Research*, 70(12):3506–3519, 2022.

[107] R. Linares and R. Furfaro. An autonomous sensor tasking approach for large scale space object cataloging. In *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, pages 19–22 September, Maui, HI, USA, 2017.

[108] Y. Xiang, J. Xi, M. Cong, Y. Yang, C. Ren, and L. Han. Space debris detection with fast grid-based learning. In *Proceedings of the 2020 IEEE 3rd International Conference of Safe Production and Informatization (IICSPI)*, pages 205–209, Chongqing, China, 2020.

[109] H. C. van de Hulst. *Light scattering by small particles*. Dover Publications, New York, 1981.

[110] Craig F. Bohren and Donald R. Huffman. *Absorption and Scattering of Light by Small Particles*. Wiley-VCH, Weinheim, 1998.

[111] G. Ruíz et al. Autonomous tracking of space objects with the fgan tracking and imaging radar. In *GI Jahrestagung*, 2006.

[112] J. Pandeirada et al. Development of the first portuguese radar tracking sensor for space debris. *Signals*, 2(1):122–137, 2021.

[113] J. Neyman and K. Pearson. On the problem of the most efficient test of statistical hypotheses. *Philosophical Transactions of the Royal Mathematical or Physical Character*, 231:289–337, 1933.

[114] Mark A. Richards. *Fundamentals of Radar Signal Processing*. McGraw-Hill Education, New York, 2nd edition, 2014.

[115] M.E. Smith and P.K. Varshney. Intelligent cfar processor based on data variability. *IEEE Transactions on Aerospace and Electronic Systems*, 36(3):837–847, 2000.

[116] P.P. Gandhi and S.A. Kassam. Analysis of cfar processors in nonhomogeneous background. *IEEE Transactions on Aerospace and Electronic Systems*, 24(4):427–445, 1988.

[117] T.-T.V. Cao. Constant false-alarm rate algorithm based on test cell information. *IET Radar, Sonar & Navigation*, 2(3):200–213, 2008.

[118] J.A. Ritcey and J.L. Hines. Performance of max family of order-statistic cfar detectors. *IEEE Transactions on Aerospace and Electronic Systems*, 27(1):48–57, 1991.

[119] T.-T.V. Cao. Design of low-loss cfar detectors. In *Proc. International Conference on Radar*, pages 712–717, Adelaide, SA, Australia, 2008.

[120] H.M. Finn. A cfar design for a window spanning two clutter fields. *IEEE Transactions on Aerospace and Electronic Systems*, AES-22(2):155–169, 1986.

[121] Robert Tarjan. Depth-first search and linear graph algorithms. In *Proc. 12th Annual Symposium on Switching and Automata Theory*, pages 114–121. IEEE, 1971.

[122] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, Las Vegas, NV, USA, 2016.

[123] Labellerr Team. Evolution of yolo object detection model from v5 to v8. https://www.labellerr.com, 2023. https://www.labellerr.com/blog/evolution-of-yolo-object-detection-model-from-v5-to-v8/.

[124] Peng Liu, Jianhua Li, and Yong Zhao. A lightweight object detection algorithm for remote sensing images based on attention mechanism and yolov5s. *Remote Sensing*, 15(9):2429, 2023.

[125] X. Wen, X. Qiu, B. Han, C. Ding, B. Lei, and Q. Chen. A range ambiguity suppression processing method for spaceborne sar with up and down chirp modulation. *Sensors*, 18(5):1454, 2018.

[126] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16×16 words: Transformers for image recognition at scale. *arXiv preprint*, arXiv:2010.11929, 2021.

[127] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint*, arXiv:2103.14030, 2021.

[128] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint*, arXiv:1706.05098, 2017.