**UNIVERSITÀ POLITECNICA DELLE MARCHE**
**FACOLTÀ DI INGEGNERIA**
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione


**TESI DI LAUREA**

# Progettazione e implementazione di approcci di Intelligenza Artificiale per il rilevamento di possesso di armi, di incidenti e di atti di violenza

# Design and implementation of Artificial Intelligence approaches for detecting weapon possession, accidents, and acts of violence

Relatore

Prof. Domenico Ursino

Candidato

Manuel Marrone

**ANNO ACCADEMICO 2023-2024**

*Success is not final, failure is not fatal: it is the courage to continue that counts.*

Winston Churchill

**Sommario**

La Computer Vision rappresenta una delle tecnologie emergenti più promettenti per migliorare la sicurezza e l'efficienza in vari contesti di sorveglianza. Questa tesi esplora l'intera filiera di produzione di modelli di Intelligenza Artificiale, concentrandosi sull'utilizzo della Computer Vision per il monitoraggio dei pericoli per l'essere umano. Attraverso una serie di esperimenti, si analizza l'efficacia di diverse tecniche di Computer Vision per automatizzare il rilevamento e il monitoraggio dei pericoli, come incidenti stradali, possesso illecito di armi e atti violenti. La tesi fornisce una panoramica completa dei processi implementati e dei risultati ottenuti, identificando i punti di forza ed i punti di debolezza delle tecniche analizzate; infine, essa conclude con suggerimenti per possibili miglioramenti e future applicazioni delle idee proposte.

**Keyword**: Deep Learning, Computer Vision, Instance Segmentation, Object Detection, Pose Estimation, YOLO, CNN

# Indice

# Elenco delle figure

# Introduzione

L'innovazione dell'Intelligenza Artificiale (IA) ha conosciuto un'espansione senza precedenti, segnando un boom di popolarità che ha coinvolto sia la comunità scientifica che il grande pubblico. L'IA, che una volta sembrava una tecnologia riservata a pochi esperti, è ora una presenza pervasiva in vari settori della nostra vita quotidiana. Dall'assistenza sanitaria alla finanza, dall'industria automobilistica all'intrattenimento, le applicazioni dell'IA sono diventate sempre più diffuse e sofisticate, portando benefici significativi in termini di efficienza, precisione e personalizzazione dei servizi.

Il dibattito attorno all'IA è altrettanto vivace quanto il suo sviluppo. Da un lato, ci sono entusiasmi per le sue potenzialità di trasformare radicalmente i processi lavorativi e le esperienze quotidiane; dall'altro, emergono preoccupazioni etiche e sociali riguardanti la privacy, la sicurezza e l'impatto sul mercato del lavoro. La sfida è trovare un equilibrio tra l'innovazione tecnologica e l'adozione responsabile di queste tecnologie.

Il lavoro svolto si concentra sull'analisi delle possibili applicazioni dell'Intelligenza Artificiale nel campo della Computer Vision, sui suoi punti di forza e sulle sue debolezze. Si affronta questo tema per fornire una panoramica delle varie tecniche attualmente esistenti.

La Computer Vision studia come i computer possano acquisire, elaborare e interpretare informazioni visive dal mondo circostante, imitando la capacità umana di vedere e comprendere le immagini. Attraverso un'analisi approfondita delle tecnologie e delle applicazioni della Computer Vision, dimostriamo come l'utilizzo dell'IA sia divenuto accessibile anche a chi non è esperto nel settore. Esaminiamo strumenti e piattaforme che permettono di implementare soluzioni avanzate di Computer Vision in modo intuitivo e user-friendly, aprendo la strada a nuove possibilità per innovatori, imprenditori e ricercatori di ogni ambito.

Durante il corso della nostra ricerca, studiamo tre macro-argomenti significativi che illustrano l'ampiezza e la complessità di questo campo in continua evoluzione. Il primo macro-argomento si concentra sull'object detection, una tecnologia fondamentale che consente ai sistemi di identificare e localizzare oggetti specifici all'interno di immagini o video. In particolare, vengono esplorati come modelli avanzati di object detection, come YOLO (You Only Look Once), possano essere adattati per il riconoscimento di incidenti stradali. Ciò non solo migliora la sicurezza stradale, ma dimostra anche come la Computer Vision possa contribuire significativamente alla gestione dei rischi e alla prevenzione degli incidenti.

Il secondo macro-argomento affronta l'instance segmentation, una tecnica più avanzata che non solo consente di riconoscere gli oggetti nelle immagini ma ne segmenta accuratamente i contorni pixel per pixel. L'obiettivo è quello di sviluppare e ottimizzare un modello per il rilevamento e la segmentazione di armi in diversi contesti. Questa tecnica è cruciale

per applicazioni in ambito di sicurezza e controllo, dove la precisione e l'attendibilità dei rilevamenti sono essenziali per decisioni rapide e informate.

Il terzo e ultimo macro-argomento si concentra sulla pose estimation per il rilevamento di atti di violenza, un'applicazione emergente della Computer Vision che si concentra sull'analisi delle pose umane per identificare comportamenti potenzialmente pericolosi. Tale approccio non solo richiede una comprensione profonda della biomeccanica umana, ma anche l'implementazione di algoritmi sofisticati capaci di rilevare e interpretare segnali sottili nei movimenti umani. Si esplorano le varie metodologie esistenti e le loro applicazioni pratiche, discutendo delle sfide tecniche e delle potenziali soluzioni per migliorare l'efficacia di tali sistemi.

Ogni macro-argomento viene approfondito attraverso l'analisi dei risultati, confrontando le prestazioni di diversi approcci e valutando le implicazioni pratiche delle tecnologie esaminate. Grazie a questa ricerca, cerchiamo di illuminare non solo le potenzialità della Computer Vision, ma anche le sfide e le opportunità che essa presenta per l'innovazione e il progresso in una varietà di settori applicativi.

Questa tesi è strutturata come di seguito specificato:

- Nel Capitolo 1 si fornisce un'introduzione generale sull'Intelligenza Artificiale, esaminando la sua evoluzione, dalla nascita della disciplina fino al contesto attuale. Viene dato particolare rilievo alla Computer Vision, seguita da una discussione sull'etica correlata all'IA. Infine, si esplorano brevemente gli strumenti impiegati per sviluppare i modelli.

- Nel Capitolo 2 si approfondisce il tema del Deep Learning e delle Convolutional Neural Network, esaminando le strutture e i meccanismi fondamentali di tali tecnologie. Vengono analizzate, anche, le diverse tecniche di Computer Vision per una comprensione approfondita dei loro funzionamenti e dei relativi campi applicativi.

- Nel Capitolo 3 ci si focalizza sugli strumenti utilizzati per la valutazione di un modello di Intelligenza Artificiale, in particolare sulle metriche utilizzate e sulle formule per ottenerle.

- Nel Capitolo 4 viene affrontato il primo esperimento, incentrato sul rilevamento degli incidenti stradali. Si propone di esaminare i dataset utilizzati e l'addestramento dei modelli, per poi valutare i risultati ottenuti.

- Nel Capitolo 5 si analizza il lavoro svolto per l'addestramento di un modello su diversi dataset per la segmentazione di immagini di armi. Si confrontano due metodologie di addestramento e si esplora lo stato dell'arte per questo specifico compito.

- Nel Capitolo 6 si discute l'ultimo esperimento sul rilevamento di atti di violenza. All'interno di questo capitolo viene presentata tutta la filiera di produzione per arrivare al modello finale. Si fornisce, anche, un'analisi dei lavori svolti in altre ricerche, che implementano il task con altre metodologie.

- Nel Capitolo 7 viene presentato un resoconto sulle esperienze affrontate, sulle lezioni apprese e sulle conoscenze acquisite; infine, si analizzano i punti di forza e i punti di debolezza di ogni tecnica implementata.

CAPITOLO 1

# Introduzione all'Intelligenza Artificiale

*In questo capitolo introduttivo si esplorerà che cos'è l'Intelligenza Artificiale, come è nata, come si è svillupata e come si sta sviluppando nel mondo di oggi. Si affronteranno inoltre i temi riguardanti l'etica e l'impatto sociale messi in discussione con l'avvento e la crescita della popolarità di queste macchine intelligenti.*

## 1.1 Panoramica dell'IA e la sua evoluzione

Per affrontare al meglio questo tema sarà necessario mettere in chiaro cosa intendiamo per Intelligenza Artificiale, nota anche con l'acronimo IA, e comprendere come siamo arrivati a possedere gli strumenti che utilizziamo oggi ormai divenuti di uso comune.

### 1.1.1 Definizione di Intelligenza Artificiale

L'Intelligenza Artificiale non può essere descritta da una semplice definizione, è un concetto che riguarda più discipline. Per dare un significato a quella che chiamiamo "Intelligenza Artificiale" possiamo partire dall'analizzare le parole che la compongono.

La parola "Intelligenza" deriva dal verbo latino "intelligere" ossia "capire"; quindi possiamo definire l'intelligenza come la capacità di comprendere fatti e concetti, i quali vengono immagazzinati come conoscenza da applicare a comportamenti in vari contesti. Da questa semplice definizione derivano tutte le capacità che ne conseguono di astrazione, pensiero critico e problem solving.

Il termine "Artificiale" dal latino "artificiālis", che, a sua volta, deriva da "artifex", composto da "ars" (arte) e "facere" (fare); si riferisce a qualcosa che è creato o fatto con l'arte, ossia a qualcosa che è prodotto dall'intervento dell'uomo.

Quindi stiamo unendo l'intelligenza, che solitamente viene attribuita a persone ed animali, con un termine che indica l'esatto opposto di un qualcosa definito come naturale. Si conclude, quindi, che l'IA non è altro che un sistema creato dall'uomo in grado di simulare l'intelligenza umana tramite l'apprendimento automatico e l'adattamento.

### 1.1.2 Definizione di rete neurale

La rete neurale è l'elemento centrale del Deep Learning, un sottoinsieme del Machine Learning che, a sua volta, è una branca dell'IA; essa utilizza nodi interconnessi in una struttura stratificata che simula il cervello umano. Le informazioni entrano come input nella

rete neurale, i nodi elaborano i dati, li analizzano o li categorizzano e producono un output da trasferire ad altri nodi.

Esistono diverse tipologie di reti neurali artificiali categorizzate in base a come i dati fluiscono dal nodo di input al nodo di output. Alcune tipologie sono: Feed Forward Neural Network (FNN), Recurrent Neural Network (RNN) e Convolutional Neural Network (CNN).

Comprendere cos'è una rete neurale è importante perchè questa tipologia di rete rappresenta uno dei pilastri dell'Intelligenza Artificiale moderna, visto che le reti neurali artificaili si sono dimostrate estremamente potenti in tanti e vari campi applicativi, dall'elaborazione del linguaggio naturale alla guida autonoma, dalla raccomandazione di contenuti online alla diagnosi medica.

### 1.1.3 I precursori dell'IA

Prima della nascita vera e propria dell'Intelligenza Artificiale ci sono stati diversi studiosi che hanno dato il loro contributo costruendo macchine e ragionando come nessuno aveva fatto prima di loro.

Si iniziò a parlare di IA nel secondo dopoguerra, grazie all'avvento dei calcolatori, macchine in grado di eseguire calcoli matematici. I primi a fare ricerca in questo ambito furono Warren McCulloch e Walter Pitts nel 1943, che proposero un modello di rete neurale mirato a simulare il funzionamento di un cervello umano. Ogni neurone poteva rappresentare uno stato binario i cui possibili valori "acceso" o "spento". I due ricercatori dimostrarono come unendo questi neuroni gli uni agli altri, fosse possibile implementare gli operatori fondamentali della logica booleana. Inoltre ipotizzarono che tali reti fossero in grado di apprendere, ipotesi che si concretizzò con Donald Hebb nel 1949, il quale ideò una regola di modifica dei pesi delle sinapsi che collegano i neuroni. L'insieme di queste scoperte porto a creare il primo computer a rete neurale, sviluppato da Marvin Minsky nel 1950, detto SNARC.

### 1.1.4 Il test di Turing

Nel frattempo uno dei padri dell'informatica, ovvero Alan Turing, aveva posto le fondamenta per tutti i futuri studi che ne sarebbero seguiti. Nel 1950 scrisse l'articolo "Computing machinery and intelligence" nel quale proponeva quello che, da quel momento in poi, sarebbe divenuto noto come il test di Turing. Secondo tale test una macchina poteva essere considerata intelligente se, in risposta a degli stimoli il comportamento sarebbe risultato identico a quello di un essere umano.

Grazie al lavoro di Turing, l'Intelligenza Artificiale acquisì una forte attenzione da parte della comunità scientifica e si avviarono i primi studi approfonditi.

### 1.1.5 La nascita

La nascita dell'IA come disciplina è attribuibile all'anno 1956. In quell'anno si tenne, infatti la conferenza del Dartmouth College nel New Hampshire, organizzata dal ricercatore John McCarthy, alla quale parteciparono i maggiori esponenti dell'informatica, ritratti nella Figura1.1.

Da quel convegno nacque un team di dieci persone che avrebbe dovuto creare in due mesi una macchina in grado di simulare l'intelligenza umana. Tra i ricercatori che parteciparono alla conferenza ci furono, anche, Trenchard More di Princeton, Arthur Samuel di IBM, e Ray Solomonoff e Oliver Selfridge del MIT.

Nello stesso convegno, ci si focalizzò anche su un'altra iniziativa, il programma di Allen Newell e Herbert Simon. Questi due ricercatori avevano già un programma capace di qualche forma di ragionamento, conosciuto con il nome di Logic Theorist (LP), in grado di dimostrare


**Figura 1.1:** I partecipanti alla Conferenza di Darmouth

teoremi partendo dai principi della matematica. Questi due ricercatori nel 1957 svilupparono il General Problem Solver (GPS), il quale venne creato principalmente per risolvere problemi teorici, geometrici, e anche per giocare a scacchi.

In quell'incontro organizzato da John McCarthy si raccolsero i principali contributi sul tema e si posero i quesiti sui futuri sviluppi. McCarthy introdusse l'espressione "Intelligenza Artificiale", che segnò, in maniera indelebile, la nascita effettiva della disciplina.

### 1.1.6 Le difficoltà

Tra il 1966 ed il 1973 si riscontrarono, però, i primi problemi che portarono ad un arresto parziale della ricerca in quell'ambito. Una delle applicazioni sulla quale i ricercatori puntarono fu la comprensione del linguaggio; tuttavia nella traduzione automatica l'IA portò ai primi fallimenti inattesi. Al contrario di quanto si credeva, non era possibile realizzare traduzioni automatizzate tramite una semplice manipolazione sintattica. Il fallimento di questi tentativi iniziali ebbe come conseguenza il taglio dei fondi a molti progetti di ricerca.

Un altro problema si riscontrò nel 1973 in Inghilterra, dove, con il rapporto Lighthill, l'IA venne definita incapace di risolvere molti dei problemi che si proponeva. Infatti, i primi ricercatori non tenevano conto della crescita esponenziale del tempo di calcolo al crescere dei dati.

Una terza difficoltà furono le limitazioni alla base del ragionamento. Nel libro di Marvin Minsky e Seymour Papert intitolato "Perceptrons: an introduction to Computational Geometry" nel 1969, si mostrò che i percettroni, una semplice forma di rete neurale, hanno delle limitazioni nel risolvere problemi non linearmente separabili; essi sono, quindi, limitati nella loro capacità di apprendere e generalizzare da dati complessi.

Inoltre, in questi anni, si sollevarono alcune critiche nei confronti dell'IA. Hubert L. Dreyfus, nel suo articolo "Alchemy and AI" del 1965, e poi nel libro "What computers can't do" del 1972, attaccò i ricercatori affermando che l'IA non era realizzabile dal punto di vista filosofico, sottolineando l'incapacità di realizzare i tanti successi promessi.

### 1.1.7 Il boom dell'IA

Queste difficoltà fornirono una base dalla quale ripartire definendo gli approcci adottati dalle macchine come approcci deboli, che necessitavano, quindi, di una conoscenza maggiore inerente al campo in cui venivano applicati.

Si teorizzarono, pertanto, i primi sistemi conosciuti come sistemi esperti, ovvero in grado di possedere una conoscenza specializzata in un determinato scenario applicativo. Si trattava di sistemi in cui l'uomo trasferiva direttamente la propria conoscenza alla macchina, stabilendo, mediante regole logiche, quali fossero le scelte da prendere in determinati contesti.

Il primo esempio fu DENDRAL, sviluppato da Edward Feigenbaum a Stanford a partire dal 1965, il cui compito era quello di mappare la struttura delle molecole.

Successivamente Feigenbaum e Buchanan implementarono il sistema esperto MYCIN, considerato un punto di svolta per la sua agilità e prestazione. MYCIN era pensato come uno strumento di aiuto per i medici nella diagnosi di malattie infettive del sangue.

La quantità di sistemi esperti prodotti dai ricercatori continuò a crescere attirando l'interesse delle aziende. Il primo sistema di Intelligenza Artificiale utilizzato in ambito commerciale fu R1, utilizzato dalla Digital Equipment Corporation nel 1982; lo scopo del programma era quello di aiutare a configurare gli ordini per nuovi computer. La scelta di introdurre l'IA nella propria industria si rivelò una scelta vincente, facendo risparmiare milioni di dollari all'azienda.

Questo contribuì alla comparsa dell'Intelligenza Artificiale nel campo industriale, con l'aumento dei fondi per la ricerca, portando ad una crescita repentina delle conoscenze nell'ambito.

L'industria dell'Intelligenza Artificiale raggiunse nel 1988 una cifra dell'ordine di miliardi di dollari, includendo centinaia di aziende che stavano creando sistemi esperti, robot e software e hardware specializzati in questi settori.

### 1.1.8 L'ultima frontiera: i Transformer

La disciplina dell'IA ha fatto passi enormi in relativamente pochi anni, un trend che potrebbe proseguire o rallentare nei prossimi anni.

Ad oggi l'ultima tecnologia riguardante il Deep Learning sono i Transformer. Un Transformer è un'architettura di Deep Learning sviluppata da Google e basata sul meccanismo di auto attenzione. Architettura mostrata nella Figura 1.2.

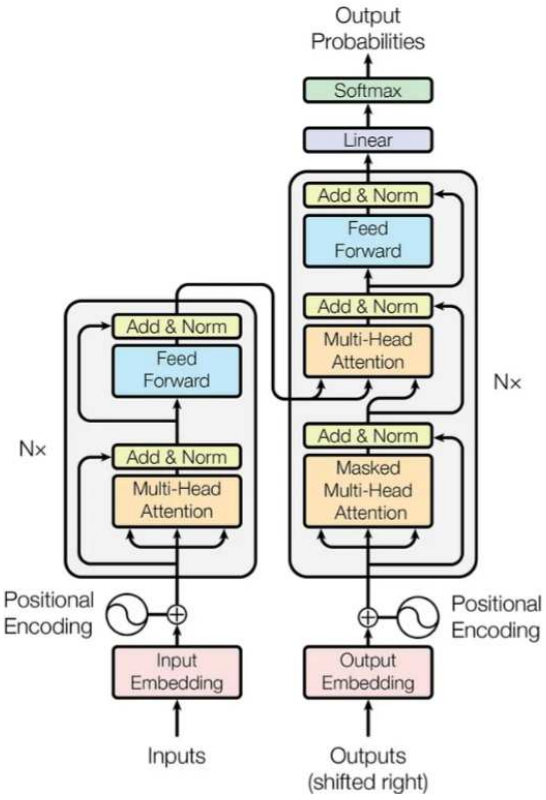


**Figura 1.2:** Architettura del modello Transformer

I primi modelli di Deep Learning che si concentravano nel Natural Language Learning (NLP) miravano a far comprendere alla macchina il linguaggio umano e predire la parola successiva basandosi sulla sequenza di parole precedenti.

I primi modelli di Machine Learning (ML) applicavano tecnologie simili su una scala più ampia, mappando la frequenza delle relazioni tra diverse coppie di parole, o gruppi di parole, nel loro training set e provando a indovinare la parola successiva. Tuttavia, superata una certa lunghezza di input, tali modelli non riuscivano a mantenere il contesto coerente; ad esempio, non erano in grado di generare un paragrafo significativo perché non riuscivano a mantenere il contesto tra la prima e l'ultima frase di un paragrafo.

I Transformer hanno abbattuto questa barriera consentendo la gestione di dipendenze tra le parole a lungo raggio. Grazie al meccanismo dell'attenzione, essi riescono a concentrarsi su diverse parti della sequenza di parole per stabilire, poi, quale parte risulta più significativa. Tale meccanismo consente, quindi, di dare più attenzione alle parti rilevanti del testo e di combinarle per effettuare delle predizioni sulle prossime parole.

## 1.2 Computer Vision

Dopo aver discusso in modo generico sull'Intelligenza Artificiale, approfondiamo ora un campo specifico di questa disciplina, ovvero la Computer Vision o Visione Artificiale, tema principale sul quale ci si è focalizzati per la conduzione degli esperimenti che verranno esposti più dettagliatamente nei prossimi capitoli.

### 1.2.1 Cos'è la Computer Vision

La Computer Vision è un campo che studia algoritmi e architetture con l'obiettivo di riprodurre tutte le funzionalità dell'apparato visivo umano. In particolare, si vuole permettere alle macchine di vedere e comprendere da immagini e video, con lo scopo di estrarre e immagazzinare le informazioni utili all'elaborazione e comprensione dei dati.

Per noi esseri umani risulta un compito particolarmente semplice ed intuitivo grazie al vantaggio di disporre di tantissimi anni di esperienza nel riconoscimento degli oggetti con la vista; non si può dire però lo stesso per le macchine.

### 1.2.2 Come funziona

Per poter dare significato alle immagini e ai video, la macchina necessita di una grande mole di dati. Il processo durante il quale vengono forniti al computer i dati opportunamente etichettati fino a quando non riesce ad estrarre le informazioni corrette prende il nome di addestramento.

Nel dettaglio, il funzionamento si basa su una serie di processi di seguito elencati:

- *acquisizione* dell'immagine da una sorgente;
- *pre-elaborazione dell'immagine* per migliorarne la qualità; vi è inoltre la possibilità di utilizzare tecniche come il filtraggio, la trasformazione, la segmentazione e la normalizzazione dell'immagine;
- *estrazione delle caratteristiche dall'immagine*, come i bordi, i contorni, i colori, le texture, i punti chiave e le regioni d'interesse;
- *riconoscimento delle caratteristiche* e classificazione dell'immagine tramite l'ausilio di tecniche di Machine Learning e Deep Learning;

- *utilizzo di tecniche di ragionamento*, d'inferenza e di decisione, per generare risposte in base alla classificazione effettuata.

L'insieme di questi passaggi porta, di conseguenza, al riconoscimento di oggetti, animali, persone e a molti altri possibili task. Tutto questo senza non poche problematiche, legate alle qualità delle immagini fornite nel dataset e alla limitata capacità della rete neurale di generalizzare ed adattarsi ai diversi contesti.

### 1.2.3 Computer Vision e Deep learning

Analizzando il funzionamento della Computer Vision non si può non notare lo stretto legame che ha con il Deep Learning.

Esistono diverse modalità con le quali un'architettura di Visione Artificiale può estrarre informazioni dalle immagini avvalendosi del Deep Learning.

Queste sono:

- *L'Hand Crafted Features*; si basa sul concetto che gli algoritmi possano estrarre e definire ciò che è più rilevante nell'immagine (ad esempio, uno specifico colore/forma, area, grandezza).
- *La Computer Vision Features*; si concentra sulla suddivisione dell'immagine in piccole regioni per permettere un'analisi più approfondita dell'immagine.
- *La Data Driven Features*; è la più evoluta e permette il riconoscimento e la classificazione delle immagini (anche naturali) senza dover progettare la fase di estrazione delle features, che viene svolta dalle reti neurali convoluzionali.

### 1.2.4 Computer Vision e Generative AI

L'Intelligenza Artificiale Generativa nel campo della Computer Vision ha aperto nuove possibilità per la creazione e l'analisi di contenuti visivi.

Tramite questa nuova tecnologia è possibile ampliare la gamma di dati a disposizione per l'addestramento, generando, così, immagini sintetiche utili per amplicare il dataset. Un esempio pratico è la creazione di immagini sintetiche simulando varie condizioni di luminosità, occlusioni o prospettive

L'Intelligenza Artificiale Generativa apre sviluppi anche in altri campi; basti pensare al recente traguardo raggiunto da OpenAI con il rilascio del suo modello Sora, un modello Text-to-video, basato su complesse reti neurali, in grado di realizzare video realistici e dettagliati partendo da un input testuale.

## 1.3 Etica e filosofia

Nel parlare di Intelligenza Artificiale e di quello che ne consegue, bisogna menzionare, oltre che il lato tecnico, anche il lato umanistico. Difatti la crescente popolarità di questa disciplina porta con sè un importante impatto nella società.

### 1.3.1 Le leggi che ne stabiliscono l'utilizzo

Come tutte le discipline, l'IA necessita di regolamentazioni che ne limitino l'utilizzo e i rischi. Tutto il mondo si sta movimentando per produrre le nuove leggi ed essendo una materia nata da poco le criticità per regolamentarla non sono poche.

L'Europa si sta muovendo nella regolamentazione con l'approvazione dell'AI Act da parte degli Stati membri nel dicembre 2023.

L'obiettivo che si pone è quello di proteggere i diritti fondamentali, la democrazia, lo Stato di diritto e la sostenibilità ambientale dai sistemi di IA ad alto rischio, utilizzando una visione "human centered", nella quale si dà priorità alla salvaguardia dei diritti fondamentali dell'uomo.

Il fulcro è il cosiddetto "risk-based approach" che consente di classificare i sistemi di IA proprio sulla base dei rischi che potrebbero derivare ai diritti fondamentali dei singoli, nonché ai valori dell'Unione, la cui violazione attiverebbe un regime sanzionatorio pecuniario ad hoc.

Le nuove norme mettono fuori legge alcune applicazioni di IA che minacciano i diritti dei cittadini, come i sistemi di categorizzazione biometrica per creare banche dati di riconoscimento facciale; saranno vietati, anche, i sistemi di riconoscimento delle emozioni sul luogo di lavoro e nelle scuole.

Per i sistemi ad alto rischio si prevedono obblighi più stringenti, ovvero di valutare e ridurre i rischi, mantenere registri d'uso, essere trasparenti e accurati e garantire la sorveglianza umana. I cittadini avranno diritto a presentare reclami sui sistemi di IA e a ricevere spiegazioni sulle decisioni che incidono sui loro diritti.

Si impongono, inoltre, obblighi di trasparenza e di rispetto delle norme UE sul diritto d'autore durante le fasi di addestramento dei vari modelli. Inoltre, le immagini e i contenuti audio o video artificiali o manipolati, conosciuti come deepfake, dovranno essere chiaramente etichettati come tali.

Nel frattempo, negli Stati Uniti, il 30 ottobre 2023 viene emanato l'Executive Order on Safe, Secure, and Trustworthy Artificial Intelligence. A differenza della visione human-centric europea, qui l'approccio è orientato a favorire le imprese e lo sviluppo dei sistemi di IA, poiché solo così può essere mantenuta la leadership americana. Gli obiettivi imposti sono lo sviluppo sicuro, protetto e affidabile dell'IA, la salvaguardia della privacy degli americani, l'uguaglianza, la protezione dei diritti civili, la tutela dei consumatori e dei lavoratori, nonché la promozione dell'innovazione e della competizione. Il problema del rischio viene affrontato, ad eccezione dell'aspetto sanzionatorio.

Si riconosce la necessità di standard per un'Intelligenza Artificiale "safe and secure" che si traduce in obblighi di trasparenza in capo agli sviluppatori dei sistemi di IA, potenzialmente più rischiosi per la sicurezza nazionale o per la salute degli individui.

Lo sviluppo di questi standard generali di sicurezza, nonché di linee guida e di criteri di previsione dei potenziali danni, viene delegata ai dipartimenti e alle agenzie preposte, quali il National Institute of Standards and Technology.

In conclusione, da un lato c'è l'Europa, con una visione human-centric fondata sulla classificazione dei sistemi di IA in base al rischio e sulla predisposizione di una serie di obblighi orizzontali e relative previsioni sanzionatorie. Dall'altro lato, gli Stati Uniti d'America, con un approccio maggiormente business friendly che prediligono una normativa più frammentata e di più ampio respiro, delegano alle singole agenzie e dipartimenti la definizione di standard generali, al fine di incentivare lo sviluppo e il mantenimento del primato mondiale in tema di IA, senza ricorrere a rigide regolamentazioni, che potrebbero rendere eccessivamente gravoso il progresso.

L'emazione di leggi e linee guida sono il primo passo per la regolamentazione dell'IA; ora inizia la fase in cui queste leggi dovranno essere rispettate con controlli rigidi e costanti, consapevoli del fatto che non sarà semplice trovare un equilibrio tra le parti coinvolte.

### 1.3.2 Fattori negativi e principi morali

Tutte queste regolamentazioni sono fondamentali al fine di limitare e controllare tutti i rischi e i fattori negativi che l'Intelligenza Artificiale porta con sè.

I rischi ai quali si può incorrere sono:

- *La violazione del copyright*, ad esempio tramite l'IA generativa, in grado di produrre contenuti apparentemente originali ma che, in realtà, risultano imitazioni del lavoro di altre persone.
- *La sostituzione di persone con il deep fake* che, partendo da immagini, video e audio reali, riescono a modificare o ricreare, in modo estremamente realistico, la persona simulata, donando ad esempio la possibilità ai malintenzionati di manipolare immagini di bambini per la realizzazione di contenuti pedopornografici. Tale violazione è punita in Italia con la reclusione da sei a dodici anni e con la multa da 24.000 a 240.000 euro.
- *Propaganda e disinformazione* dovute all'aumento della facilità di creazione e diffusione di informazioni false e dalla convinzione che gli strumenti di IA, come, ad esempio, ChatGPT, diano sempre risposte vere e verificate.
- *Bias e discriminazione*; la presenza di pregiudizi contenuti, volontariamente o meno, nei dati che addestrano gli algoritmi potrebbe generare un risultato immorale o discriminatorio.
- *Responsabilità*; in caso di danni non è semplice individuare il soggetto che ne risponderebbe. Ad esempio, se un'auto a guida autonoma causasse un incidente stradale, gli attori potenzialmente colpevoli potrebbero essere la casa produttrice, chi ha creato il software, chi ha scritto l'algoritmo.

Essendo i confini tra macchine ed umani sempre più sottili entra in gioco la filosofia con l'etica morale, la quale fornisce una serie di norme comportamentali, insieme a criteri che permettono agli esseri umani di giudicare i comportamenti degli altri rispetto ai concetti di bene e male.

Negli anni sono stati prodotti molti documenti per definire i principi per l'etica dell'IA. Luciano Floridi, filosofo e professore di filosofia ed etica dell'informazione all'Università di Oxford e all'Università di Bologna, ha svolto un'analisi comparativa di diversi trattati e ha evidenziato come tra i molti principi etici individuati si possa definire una convergenza sui quattro principi fondamentali. A questi quattro principi, Floridi ha poi aggiunto un nuovo principio. Di seguito i principi secondo Floridi:

- *beneficienza*: lo sviluppo dell'Intelligenza Artificiale dovrebbe promuovere il benessere di tutti gli esseri viventi;
- *non maleficenza*: è necessario prevenire le violazioni della privacy personale, evitare usi impropri e poter individuare delle responsabilità;
- *autonomia*: lo sviluppo dell'IA dovrebbe promuovere l'autonomia di tutti gli esseri umani; i sistemi autonomi non devono compromettere la libertà degli esseri umani e quest'ultima deve essere limitata e sempre reversibile;
- *giustizia*: l'IA dovrebbe promuovere la giustizia e cercare di eliminare tutti i tipi di discriminazione;

- *esplicabilità*: include sia il senso di intelligibilità (come risposta alla domanda “Come funziona?”), sia il senso etico di responsabilità (come risposta alla domanda “Chi è responsabile del modo in cui funziona?).

Altrettanta attenzione viene posta per un’altra preoccupazione oltre l’etica, ossia il timore della perdita di posti di lavoro. Come già visto in passato, ogni nuova tecnologia comporta una variazione della domanda di mercato dei profili occupazionali. Sicuramente molti dei lavori che conosciamo oggi svaniranno nel tempo ma saranno presumibilmente sostituiti da nuovi sbocchi occupazionali volti alla gestione di questi sistemi.

### 1.3.3 Benefici e impatto sulla società

Ovviamente l’Intelligenza Artificiale non porta solo rischi e pericoli per gli esseri umani, ma offre diversi benefici. Questi vantaggi consentiranno di migliorare lo stile di vita del singolo individuo.

Di seguito un elenco dei potenziali vantaggi che si riscontreranno da oggi ai prossimi anni:

- *assistenza sanitaria migliore*, grazie a diagnosi più accurate e alla prevenzione di sempre più malattie;
- *maggiore sicurezza*; le forze dell’ordine potranno fare affidamento a sistemi di identificazione dei reati e di criminali in tempo reale, facilitando, inoltre, la ricerca di una persona scomparsa o la prevenzione di un attacco terroristico;
- *trasporti più sicuri* e servizi pubblici migliori, attraverso l’automatizzazione dei mezzi di trasporto;
- *cybersecurity avanzata*; l’IA potrebbe essere usata per la difesa e le strategie di attacco in caso di crimini informatici;
- *riduzione dell’errore umano* tramite l’automatizzazione di operazioni ad alto rischio.

Questo elenco di benefici però non ferma le preoccupazioni della maggior parte della popolazione ancora scettica su questa nuova tecnologia, ponendosi domande come "l’Intelligenza Artificiale prenderà il controllo sugli umani?".

In generale, si registra una mancanza di conoscenza nella popolazione che porta comprensibilmente ad uno scetticismo e alla paura dell’ignoto. Infatti, la visione delle opportunità legate all’Intelligenza Artificiale è ancora confusa; la credenza comune è quella che l’IA sia capace di replicare completamente la mente umana, mentre sono in pochi ad aver compreso che, in realtà, mira a replicare specifiche capacità tipiche dell’essere umano: l’interazione con l’ambiente, l’apprendimento e adattamento, il ragionamento e la pianificazione.

L’unico modo per contrastare questi timori è tramite campagne di informazione per distribuire la conoscenza anche a chi è meno coinvolto in prima persona dall’argomento.

## 1.4 Obiettivi degli esperimenti e strumenti utilizzati

Dopo aver introdotto il tema fondante della tesi, si presentano gli esperimenti condotti e gli strumenti coinvolti nella loro realizzazione. Si motivano, inoltre, le scelte effettuate e le aspettative a riguardo.

L’obiettivo posto è quello di acquisire esperienze nel campo dell’Intelligenza Artificiale e inoltrarsi in campi non ancora pienamente sviluppati.

### 1.4.1 Motivazione della scelta dei task

Prima di effettuare la scelta delle applicazioni da realizzare, si è presa familiarità con i vari strumenti utilizzando AWS Rekognition per la classificazione di oggetti nelle immagini, per poi passare all'algoritmo YOLOv5 per il rilevamento di oggetti in video.

Tramite questi test si sono analizzate alcune criticità, ovvero:

- Difficoltà nel trovare risorse complete sulle quali costruire il dataset; un dataset fatto male porta ad una detection ambigua e non accurata invalidando il modello addestrato, che genererà un numero eccessivo di false detection.
- Nella detection di persone si riscontrano problemi in caso di zone affollate, video con bassa risoluzione e, in generale, in presenza di trasformazioni, come condizioni di luminosità non ottimale, deformazione o copertura parziale del soggetto, variazioni di scala.
- Abbassando la frequenza di frame a 15 fps durante la rilevazione delle macchine su un'autostrada si osserva un aumento della confidence dei risultati. Tuttavia, aumentando la frequenza dei frame, si manifestano false detection e incertezza delle rilevazioni;
- aumento del carico computazionale man mano che si usano modelli più complessi (ad esempio, al crescere del numero di parametri utilizzati).

A seguito dei vari test si scelgono le seguenti idee applicative:

- rilevamento di comportamenti aggressivi tramite telecamere;
- riconoscimento del possesso di armi tramite telecamere;
- riconoscimento di incidenti stradali tramite telecamere di sorveglianza.

Si sono scelte queste applicazioni nello specifico perchè non risultano essere campi ancora sufficientemente esplorati; i campi applicativi sono abbastanza eterogenei per acquisire un'esperienza quanto più varia nella realizzazione degli esperimenti ed, inoltre, ciascuno di essi possiede un elemento in comune, ossia la protezione e la salvaguardia dell'essere umano.

### 1.4.2 Cosa ci si aspetta dagli esperimenti

Attraverso gli esperimenti ci si aspetta di riuscire ad addestrare dei modelli e metterli a confronto; non si pone la priorità nell'ottenere la massima accuratezza dei modelli ma semplicemente che riescano a svolgere il loro compito con sufficienza.

Le aspettative maggiori si rivolgono alla comprensione del materiale che verrà realizzato e all'acquisizione delle competenze da poter applicare in altri contesti.

### 1.4.3 YOLO

Uno degli strumenti utilizzati nel progetto è YOLO, acronimo di "You Only Look Once"; esso è un framework di Deep Learning utilizzato per l'object detection in tempo reale.

La caratteristica principale di YOLO è la sua capacità di rilevare oggetti in un'immagine o in un video in una singola Convolutional Neural Network effettuando contestualmente la predizione dei bounding box e la predizione della probabilità di appartenenza alle classi per ogni bounding box, rendendo tale task estremamente veloce rispetto ad altri approcci.

YOLO realizza la detection dividendo concettualmente l'immagine in una griglia; ogni cella della griglia è responsabile della predizione di un singolo oggetto il cui centro ricade all'interno della stessa. Per ogni cella, quindi, esso predice delle bounding box, un punteggio di confidenza e la probabilità di appartenenza alle classi.

L'architettura è basata su GoogLeNet, viene adattata per l'object detection e comprende 24 strati convoluzionali più 2 strati completamente connessi, struttura rappresentata nella Figura 1.3.

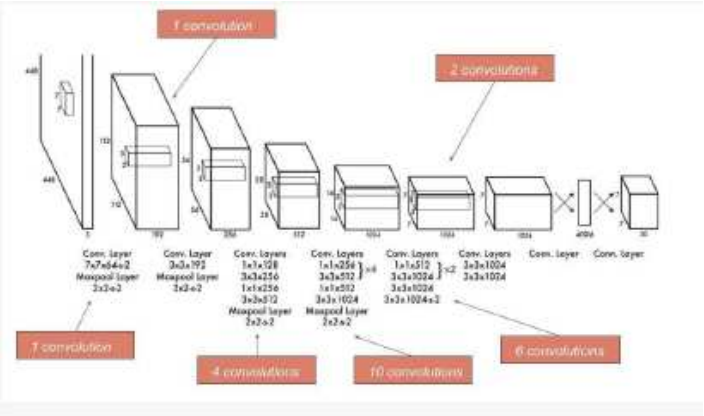


**Figura 1.3:** Architettura di YOLO

### 1.4.4 Roboflow

Una piattaforma che verrà spesso mezionata sarà Roboflow; si tratta di una piattaforma che consente di trasformare in pochi minuti le immagini grezze in un modello di computer vision addestrato.

Il suo intento è quello di offrire agli sviluppatori tutto ciò di cui hanno bisogno per iniziare a incorporare la tecnologia di Computer Vision nelle proprie applicazioni, anche senza essere esperti nel Machine Learning.

Verrà utilizzato all'interno dei task principalmente per attingere a dataset per l'addestramento dei modelli, essendo la creazione di un dataset un processo oneroso che richiede molto tempo.

### 1.4.5 Keras e Tensorflow

TensorFlow è una libreria open source per l'apprendimento automatico, che fornisce moduli sperimentati e ottimizzati, utili nella realizzazione di algoritmi per diversi tipi di compiti.

Questa libreria offre un'ampia gamma di strumenti e risorse per lo sviluppo di modelli di Machine Learning, inclusi strumenti per la costruzione di reti neurali complesse, per la gestione dei dati, l'ottimizzazione dei modelli e il deployment su diverse piattaforme, tra cui dispositivi mobili e cloud.

Keras è una libreria di Deep Learning open-source che fornisce un'interfaccia user-friendly per la creazione e l'addestramento di reti neurali artificiali; essa è stata integrata in TensorFlow come parte della libreria TensorFlow 2.0. Essa semplifica notevolmente il processo di costruzione dei modelli di Deep Learning e consente agli sviluppatori di concentrarsi sulla progettazione del modello piuttosto che sulla complessità dell'implementazione.

Con l'integrazione di Keras in TensorFlow, è possibile sfruttare la semplicità di Keras insieme alla potenza e alla flessibilità di TensorFlow.

### 1.4.6 OpenPose

OpenPose è una libreria open-source che usa Deep Neural Network per stimare le pose del corpo umano in tempo reale. Permette l'identificazione delle giunture chiave del corpo, come spalle, ginocchia e mani, nonché la loro connessione da un input video.

OpenPose ha diverse applicazioni pratiche, che vanno dalla sorveglianza video alla realtà aumentata, dall'analisi del movimento umano in ambito sportivo e di fitness alla creazione di giochi e applicazioni interattive.

CAPITOLO 2

# Il Deep Learning e le Convolutional Neural Network

*In questa sezione si approfondirà il tema del Deep Learning e delle Convolutional Neural Network, citati nel capitolo introduttivo. Si volgerà lo sguardo al funzionamento interno, ai casi d'uso di applicazione e si studieranno, nel dettaglio, le tecnologie utilizzate per l'esecuzione degli esperimenti.*

## 2.1 Deep Learning

Il Deep Learning è la branca più avanzata del Machine Learning; è una tecnologia alla base di molti prodotti e servizi d'uso quotidiano. Esso sfrutta tecniche di apprendimento nelle quali si espongono reti neurali ad una grande quantità di dati, in modo tale da insegnare loro a svolgere determinati task.

### 2.1.1 Come funziona

Alla base del Deep Learning vi sono le reti neurali artificiali organizzate per livelli. L'argomento delle reti neurali è già stato approfondito nel capitolo precedente; si ricorda che esse hanno lo scopo di imitare il cervello umano tramite un insieme di input, dati e pesi.

Gli algoritmi di Deep Learning sono molto complessi e ne esistono di diverse tipologie, ciascuno specializzato in un particolare compito. Ad esempio:

- la *Convolutional Neural Network* o *CNN* viene utilizzata per applicazioni di Computer Vision e classificazione di immagini, tale tipologia di rete verrà approfondita più avanti;
- la *Reccurrent Neural Network* o *RNN* è specializzata in dati sequenziali e serie temporali; viene utilizzata in applicazioni di riconoscimento del linguaggio naturale e vocale.

Ciasuna di queste tipologie ha in comune il funzionamento di base, ossia l'estrapolazione delle informazioni dai dati, i quali vengono, poi, analizzati e classificati, arrivando a produrre modelli (pattern) tra i dati stessi. Tutte queste operazioni sono realizzabili grazie alla struttura portante, che permette queste particolari elaborazioni di dati.

### 2.1.2 La struttura

Le Deep Neural Network sono composte da molti livelli di nodi interconessi, che insieme realizzano la rete neurale; ogni livello incrementa i risultati ricevuti dal precedente e calcola i

valori da inviare al livello successivo. Questo tipo di trasferimento dell'informazione è noto come propagazione in avanti.

Oltre alla propagazione in avanti esiste anche la retropropagazione, la quale utilizza algoritmi come la discesa del gradiente, che agisce sulla regolazione dei pesi e delle distorsioni; solitamente questa propagazione viene utilizzata per correggere errori e migliorare la precisione.

Una Deep Neural Network è suddivisa in livelli di input, livelli di output e livelli nascosti, come mostrato in Figura 2.1.

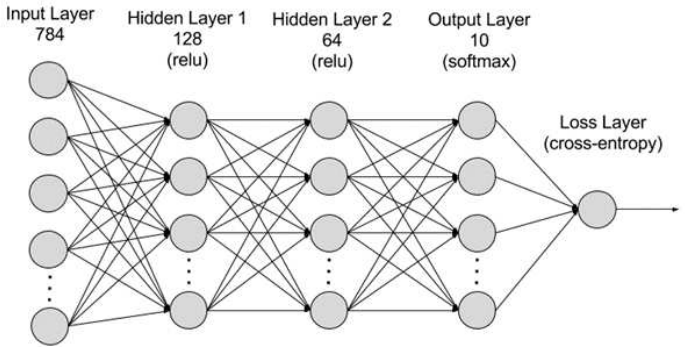


**Figura 2.1:** Stuttura di una Deep Neural Network

Si analizzano, ora, le tipologie di livelli per comprendere qual è il loro compito all'interno della rete:

- il livello di input ha il compito di inserire i dati attraverso i nodi; esso elabora e trasmette queste informazioni ai livelli successivi della rete neurale;
- i livelli nascosti elaborano le informazioni ricevute, adattando il loro comportamento sulla base delle nuove informazioni; ogni livello nascosto si occupa di elaborare una caratteristica diversa dell'entità da analizzare, ad esempio di una immagine;
- il livello di output possiede molti meno nodi rispetto ai livelli iniziali; la quantità di nodi varia in base alla gamma di risposte che viene richiesta alla rete neurale.

### 2.1.3 Deep Learning e Machine Learning

Il Deep Learning differisce dal Machine Learning per le tecniche di apprendimento che utilizza.

Gli algoritmi di Machine Learning fanno uso di dati strutturati ed etichettati per fare previsioni; pertanto, effettuano una pre-elaborazione per organizzare i dati in un formato strutturato.

Invece, nel Deep Learning si elimina una parte della pre-elaborazione non necessaria, consentendo l'acquisizione e l'elaborazione di dati non strutturati, come immagini e testi. Il Deep Learning, avvalendosi dei processi di discesa del gradiente e retropropagazione, regola e adatta la precisione effettuando previsioni sempre più accurate.

Si conclude, quindi, che il Deep Learning si distingue dal Machine Learning poichè nel primo si salta la parte manuale di estrazione delle feature del dato da classificare. Nella Figura 2.2 viene mostrata questa differenza con uno schema.

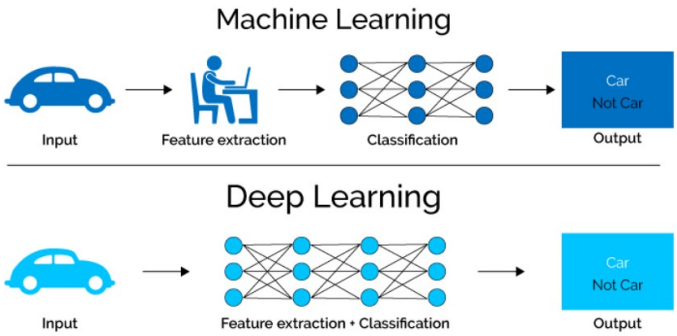


**Figura 2.2:** Confronto tra Machine Learning e Deep Learning

### 2.1.4 Pro e contro

Tuttavia, come ogni tecnologia, anche il Deep Learning presenta dei pro e dei contro. I vantaggi dell'utilizzo del Deep Learning sono i seguenti:

- *assenza di supervisione*; i modelli apprendono e migliorano nel tempo in base ai dati in input, senza necessità di un controllo costante o di annotazioni manuali;
- *grande scalabilità*; più è grande il dataset e più i risultati prodotti sono accurati; ciò rende il Deep Learning adatto a gestire grandi quantità di dati;
- *non è necessario etichettare manualmente il dataset*, un'operazione che risulta dispediosa a livello di tempo ed energie.

Mentre tra gli svantaggi troviamo che:

- *l'addestramento richiede tante risorse computazionali* a causa della complessità della rete, comportando costi elevati in termini di hardware;
- *i tempi di addestramento sono molto estesi* a causa della grande mole di dati richiesti.

Analizzando i pro e i contro è possibile, quindi, dedurre se il Deep Learning è adatto ad un contesto per una determinata applicazione o se non rispetta le caratteristiche desiderate.

### 2.1.5 Applicazioni pratiche

Il Deep Learning ha tantissime possibili applicazioni e si adatta anche ai contesti più variegati.

Viene spesso utilizzato nel settore della salute, ad esempio nel riconoscimento di malattie e tumori da immagini, per affiancare il medico nell'analisi e nella valutazione in tempi più rapidi.

Un altro ambito di applicazione è nel settore automobilistico, tramite la guida autonoma che fa uso di modelli di Deep Learning per rilevare segnali stradali e pedoni.

I modelli vengono anche utilizzati nel settore finanziario, in particolare nell'analitica predittiva, per guidare il trading dei titoli, valutare i rischi per la banca nelle approvazioni dei prestiti, rilevare le frodi e contribuire a gestire finanziamenti e depositi titoli dei clienti.

Il Deep Learning viene anche ampiamente utilizzato nel servizio clienti, nel riconoscimento vocale e nella raccomandazione di prodotti e servizi.

Tutte queste sono applicazioni usufruite dall'utenza con frequenza giornaliera, ma sono in pochi a rendersi conto della complessa elaborazione dei dati che avviene a loro insaputa.

## 2.2 CNN

La Convolutional Neural Network è un'architettura di Deep Learning che impara direttamente dalle immagini; essa è composta da diversi livelli che processano e trasformano un input per produrre un output. Viene utilizzata per analizzare, segmentare immagini e rilevare oggetti.

### 2.2.1 Definizione e funzionamento

In una Convolutional Neural Network, solo una piccola regione di nodi appartenenti al livello di input è connessa a nodi del livello nascosto; queste regioni sono note come campi ricettivi locali. Il passaggio dal livello di input al livello di output produce una mappa delle feature dell'immagine.

Mentre una classica rete neurale ha nodi con pesi e bias, che si aggiornano continuamente durante il processo di allenamento, in una CNN i valori dei pesi e dei bias è lo stesso per tutti i nodi del livello nascosto; tutti i nodi rilevano la stessa caratteristica ma in diverse regioni dell'immagine; ciò rende la rete tollerante alla traslazione degli oggetti in una immagine.

Esistono 3 modi per allenare una CNN, ovvero:

- *Allenamento da zero*, nel quale si inseriscono in input immagini etichettate; è un allenamento che richiede significative risorse computazionali.
- *Trasferimento dell'apprendimento*; esso è basato sull'idea per la quale è possibile utilizzare la conoscenza di un determinato problema per risolvere un problema simile; esso richiede meno dati e risorse computazionali della prima tipologia di allenamento.
- *Estrazione delle feature*; esso sfrutta una CNN pre-addestrata per svolgere un altro compito per estrarre delle feature utili; ad esempio, è possibile utilizzare uno strato nascosto che ha imparato a rilevare i bordi di una immagine per far svolgere ad esso altri task; ciò richiede risorse computazionali ridotte.

### 2.2.2 Livello convoluzionale, livello di pooling e livello completamente connesso

Per svolgere le funzionalità sopra descritte, una CNN deve possedere una struttura specifica. In particolare fa affidamento su due funzioni:

- *Attivazione*; con questa operazione si applica una trasformazione agli output di ogni nodo, utilizzando delle funzioni di attivazione. Un esempio di funzione comunemente utilizzata è la Rectified Linear Unit, o ReLU, la quale prende l'output dei nodi e li "mappa" al valore più alto; se l'output è negativo la funzione lo mappa a zero.
- *Pooling*; esso riceve un'immagine che rappresenta una grande mole di dati e cerca di diminuire lo spazio occupato in memoria conservando, però, le caratteristiche.

Una CNN è composta da 3 tipologie di livelli; ovvero il *livello convoluzionale*, il *livello di pooling* e il *livello completamente connesso*. Ad ogni strato la CNN aumenta la sua complessità e identifica porzioni maggiori dell'immagine.

Il *livello convoluzionale* è l'elemento fondamentale di una CNN; in esso avvengono la maggior parte dei calcoli. Esso richiede alcuni componenti, ovvero dati di input, un filtro e una mappa delle feature. Con la convoluzione si filtra l'immagine tramite filtri, detti kernel, che riescono a individuare diverse proprietà in base a come sono costituiti. Il rilevatore di feature che svolge la convoluzione è una matrice di pesi bidimensionale, la quale rappresenta parte dell'immagine. La dimensione del filtro è, in genere, una matrice 3x3 sebbene la dimensione

dei 9 elementi possa variare; ciò determina anche la dimensione del campo recettivo. Il filtro viene quindi applicato a un'area dell'immagine e viene calcolato un prodotto scalare tra i pixel di input ed esso. Il risultato del prodotto scalare viene, quindi, inserito in un array di output. Successivamente, il filtro si sposta gradualmente ripetendo il processo finché il kernel non ha attraversato l'intera immagine. L'output finale della serie di prodotti scalari tra input e filtro è noto come mappa di funzionalità, mappa di attivazione o funzionalità convoluta.

Esistono tre iperparametri che influenzano la dimensione del volume dell'output e che devono essere impostati prima che si inizi l'addestramento della rete neurale. Questi includono:

- *il numero di filtri*, che influisce sulla profondità dell'output; ad esempio, tre filtri distinti produrrebbero tre diverse mappe di feature, creando profondità di tre;
- *il passo*, che rappresenta la distanza, o numero di pixel, che il kernel sposta sulla matrice di inpu;, un passo maggiore produce un risultato inferiore;
- *lo zero-padding*, che viene solitamente utilizzato quando i filtri non si adattano all'immagine di input; esso imposta a zero tutti gli elementi che ricadono al di fuori della matrice di input, producendo un output maggiore o di dimensioni uguali.

Esistono tre tipi di padding, ovvero:

- *valid padding*; è noto anche come assenza di padding; in questo caso, l'ultima convoluzione viene eliminata se le dimensioni non si allineano;
- *same padding*; garantisce che il livello di output abbia la stessa dimensione del livello di input;
- *full padding*; aumenta la dimensione dell'output aggiungendo zeri al bordo dell'input.

Dopo ogni operazione di convoluzione, la CNN applica una trasformazione Rectified Linear Unit (ReLU) alla mappa delle feature, introducendo la non linearità nel modello. Un esempio della convoluzione viene illustrato nella Figura 2.3.

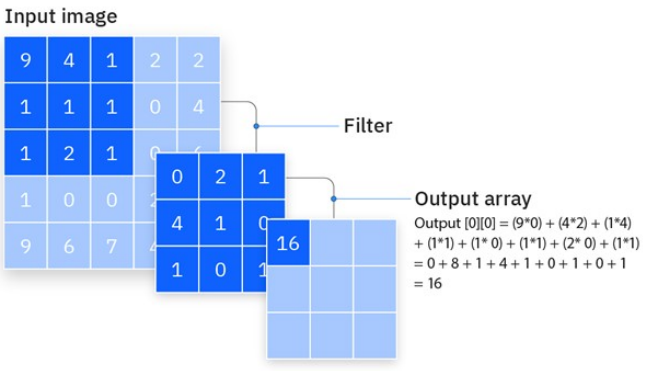


**Figura 2.3:** Un esempio di convoluzione

Un altro livello di convoluzione può seguire il livello di convoluzione iniziale. Quando ciò accade, la struttura della CNN può diventare gerarchica, poiché i livelli successivi possono vedere i pixel all'interno dei campi ricettivi dei livelli precedenti. In definitiva, il livello convoluzionale converte l'immagine in valori numerici, consentendo alla rete neurale di interpretare ed estrarre modelli rilevanti.

Il *livello di pooling*, noto anche come sottocampionamento, effettua la riduzione della dimensionalità, riducendo il numero di parametri nell'input. Similmente al livello convoluzionale, l'operazione di pooling applica un filtro sull'intero input, con la differenza che tale filtro non ha alcun peso. Il kernel applica una funzione di aggregazione ai valori all'interno del campo ricettivo, popolando l'array di output. Esistono due tipi principali di pooling, ovvero:

- *pooling massimo*; quando il filtro si sposta sull'input, seleziona il pixel con il valore massimo da inviare all'array di output;
- *pooling medio*; quando il filtro si sposta sull'input, calcola il valore medio all'interno del campo ricettivo da inviare all'array di output.

Sebbene molte informazioni vadano perse nel livello di pooling, ciò presenta anche numerosi vantaggi per la CNN; infatti, ciò aiuta a ridurre la complessità, migliorare l'efficienza e limitare il rischio di overfitting.

Nel *livello completamente connesso* ogni nodo nel livello di output si collega direttamente a un nodo nel livello precedente. Tale livello svolge il compito di classificazione in base alle feature estratte tramite i livelli precedenti e i loro diversi filtri. Mentre i livelli convoluzionali e di pooling tendono a utilizzare le funzioni ReLu, i livelli completamente connessi solitamente sfruttano una funzione di attivazione softmax per classificare gli input in modo appropriato, producendo una probabilità da 0 a 1.

### 2.2.3 Algoritmo Sliding Window

Per poter svolgere uno dei suoi task più comuni, ossia quello dell'object detection, la CNN fa affidamento su un algoritmo noto come algoritmo *Sliding Window*, in italiano *Finestra Scorrevole*.

Per predire la presenza di un oggetto, l'algoritmo fa scorrere una "finestra" sull'immagine per effettuare una predizione per ogni porzione. Esistono diversi approcci per implementare l'algoritmo, il migliore è l'approccio convoluzionale, poichè le operazioni di convoluzione condividono le elaborazioni comuni alle porzioni di immagine che si sovrappongono.

I passi da seguire per realizzare l'algoritmo sono i seguenti:

- viene scelta una cella della griglia di una dimensione specifica;
- si passa la cella attraverso l'immagine e si effettua la convoluzione sulla parte dell'immagine nella cella, per prevedere l'output;
- successivamente si fa scorrere di nuovo la cella e si ripete il secondo passo fino a coprire l'intera immagine.

Ovviamente verranno utilizzate celle della griglia di diverse dimensioni per poter rilevare oggetti di tutte le possibili dimensioni.

Un esempio dell'algortimo viene mostrato nelle Figure 2.4 e 2.5.

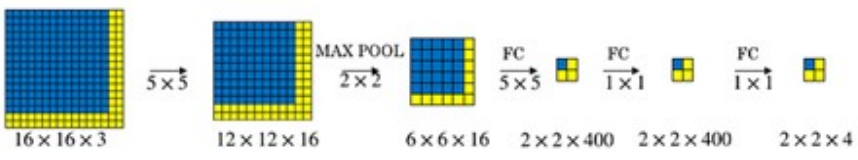


**Figura 2.4:** Esempio di un'applicazione dell'algoritmo Sliding Window

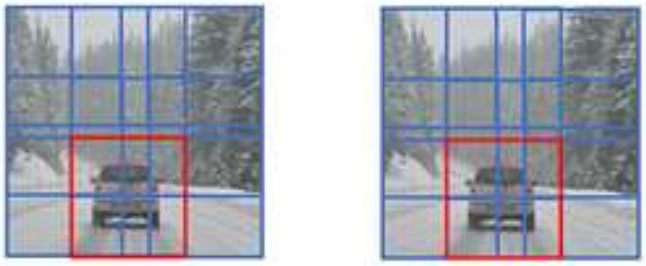

**Figura 2.5:** Visualizzazione della finestra nell'algortimo Sliding Window

Nell'esempio si applicano tecniche convoluzionali all'intera immagine per produrre un'immagine 2x2x4; ogni 1x1x4 dell'output corrisponde ad una delle finestre scorrevoli.

Tale algoritmo, però, soffre alcuni svantaggi; infatti, possono essere trovati solo oggetti con stessa dimensione della finestra e muovere la finestra sull'immagine richiede lunghi intervalli temporali.

### 2.2.4 Differenza tra classificazione binaria e classificazione multiclasse

Così come il termine CNN, anche i termini "classificazione binaria" e "classificazione multiclasse" verranno spesso utlizzati all'interno degli esperimenti condotti. Quindi è opportuno spiegare cosa si intende con queste due tipologie di classificazione.

La classificazione è un problema tipico dell'apprendimento automatico; essa consiste nell'assegnare un dato a una categoria, sulla base di un modello di classificazione appreso dalla macchina tramite l'Intelligenza Artificiale. Dunque si parla di classificazione quando la variabile target, ovvero quella da predire, è di tipo categorico.

Nella classificazione binaria i possibili valori che la variabile può assumere sono solo due (positivo o negativo). L'output di molti algoritmi di classificazione binaria è un punteggio di previsione. Quest'ultimo indica la certezza del sistema che una data osservazione appartenga alla classe positiva; per decidere se l'osservazione debba essere classificata come positiva o negativa si confronterà il punteggio con una soglia di classificazione. Qualsiasi osservazione con punteggi superiori alla soglia viene prevista come classe positiva, mentre i punteggi inferiori alla soglia vengono previsti come classe negativa

Nella classificazione multiclasse, invece, le variabili target possono assumere più di due valori. A differenza dei problemi di classificazione binaria, non è necessario scegliere una soglia per effettuare previsioni. La risposta prevista è la classe, ad esempio un'etichetta, con il miglior punteggio previsto. In alcuni casi, può essere opportuno mostrare la classe in output solo se il suo punteggio è elevato. In questo caso, è possibile scegliere una soglia per i punteggi previsti in base alla quale si accetta o meno la previsione della classe.

## 2.3 Region-based Convolutional Neural Network

Le Region-based Convolutional Neural Network, o R-CNN, rappresentano un'evoluzione delle tradizionali CNN. Esse comportano un'evoluzione nel task di rilevamento di oggetti nelle immagini, combinando la potenza delle reti convoluzionali con tecniche di selezione regionale e permettendo una localizzazione precisa degli oggetti.

### 2.3.1 Come funziona una R-CNN

I principali passi per poter realizzare il funzionamento di una R-CNN sono tre, ovvero:

- trovare regioni nell'immagine che potrebbero contenere un oggetto; tali regioni sono chiamate "region proposal";
- estrarre le feature della CNN dalle region proposal;
- classificare gli oggetti in base alle feature estratte.

Analizziamo, ora, i passi nel dettaglio.

La R-CNN inizia dividendo l'immagine in input in più regioni, che potrebbero contenere oggetti. La R-CNN non genera queste proposte da sola, ma si affida a metodi esterni come la Selective Search o gli EdgeBox.

Una volta generate le region proposal, queste ultime vengono estratte e deformate anisotropicamente, fino a raggiungere una dimensione di input coerente prevista dalla CNN (ad esempio, 224x224 pixel); a questo punto ogni regione viene elaborata attraverso la CNN per estrarre le feature. Prima della deformazione, la dimensione della regione viene espansa a una nuova dimensione, che risulterà in 16 pixel di contesto nel fotogramma deformato. Una CNN utilizzata è AlexNet ed è, in genere, ottimizzata su un dataset di grandi dimensioni, come ImageNet, per la rappresentazione di feature generiche. L'output della CNN è un vettore di feature di grandi dimensioni, che rappresenta il contenuto della region proposal.

Il vettore delle feature estratte viene inserito in un classificatore di apprendimento automatico, separato per ciascuna classe di oggetti di interesse. La R-CNN utilizza tipicamente Support Vector Machines (SVM) per la classificazione; per ciascuna classe viene addestrato un SVM univoco per determinare se la region proposal contiene, o meno, un'istanza di quella classe. Durante l'addestramento, i campioni positivi sono regioni che contengono un'istanza della classe.

La R-CNN esegue, poi, la regressione del riquadro di delimitazione. Per ciascuna classe viene addestrato un modello di regressione separato per perfezionare la posizione e la dimensione del riquadro di delimitazione attorno all'oggetto rilevato. La regressione del riquadro di delimitazione aiuta a migliorare la precisione della localizzazione dell'oggetto, regolando il riquadro di delimitazione inizialmente proposto per adattarlo meglio ai confini effettivi dell'oggetto.

Dopo aver classificato e regredito i riquadri di delimitazione per ciascuna regione, la R-CNN applica la Non-Maximun Suppression (NMS) per eliminare i riquadri di delimitazione duplicati o altamente sovrapposti. La NMS garantisce che solo i riquadri di delimitazione più sicuri e non sovrapposti vengano conservati come rilevamenti finali degli oggetti.

Si conclude, così, la realizzazione di una R-CNN.

### 2.3.2 Selective Search

Per poter trovare le region proposal, la R-CNN fa affidamento sulla Selective Search.

La Selective Search funziona unendo o dividendo segmenti dell'immagine, in base a varie caratteristiche, come colore, trama e forma, per creare un insieme diversificato di region proposal. Si utilizza una metodologia bottom-up per combinare le aree segmentate più piccole, creando, così, regioni di dimensioni maggiori. Nella Figura 2.6, si mostra come funziona la Selective Search.

### 2.3.3 Struttura

Per poter realizzare i task sopra mezionati, la R-CNN utilizza una specifica struttura, illustrata nella Figura 2.7.

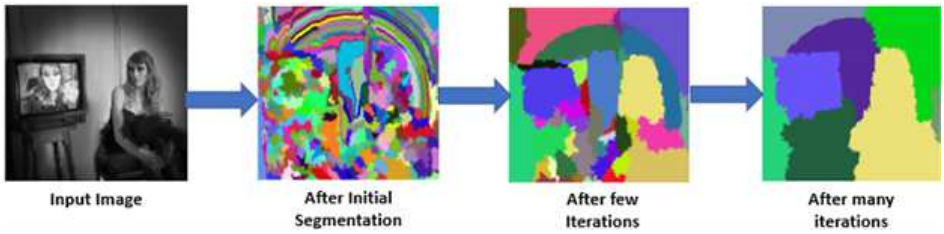


**Figura 2.6:** Esempio di un algoritmo di Selective Search

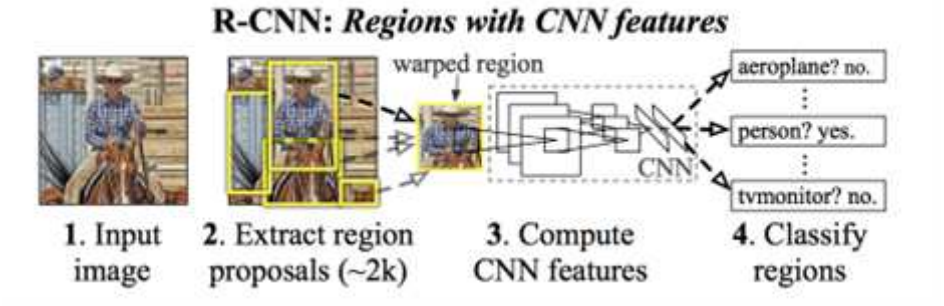


**Figura 2.7:** L'architettura di una R-CNN

Nell'immagine si possono distinguere le fasi di acquisizione di un'immagine in input, estrazione delle region proposal, calcolo delle feature tramite CNN e classificazione delle regioni.

### 2.3.4 Fast R-CNN

La Fast R-CNN rappresenta un'evoluzione significativa rispetto alla R-CNN, affrontando alcune delle sue principali limitazioni in termini di efficienza e velocità. Mentre la R-CNN richiede un lungo processo di estrazione e classificazione delle region proposal, la Fast R-CNN introduce un metodo più efficiente per l'elaborazione delle immagini.In particolare, al posto di convertire ciascuna delle regioni nelle corrispondenti feature map, essa converte l'intera immagine una sola volta.

Invece di fornire le region proposal direttamente alla CNN, l'immagine di input viene utilizzata per generare una feature map, attraverso la parte convoluzionale della rete. Le region proposal, o Region of Interest (RoI), sono identificate a partire dall'immagine di input e mappate sulla feature map tramite un'operazione di RoI projection. Le region proposal sulla feature map vengono, poi, trasformate in quadrati e scalate a una dimensione fissa, per poter essere fornite in input ai livelli completamente connessi.

Questa trasformazione delle regioni estratte dalla feature map è realizzata attraverso un layer, denominato RoI pooling layer. L'output del RoI pooling layer genera un vettore di feature che viene elaborato da una classification head, per classificare l'eventuale oggetto presente, e da una regression head, per predirre i valori di offset, necessari al raffinamento del bounding box candidato, ossia la region proposal o region of interest.

La struttura della Fast R-CNN viene mostrata nella Figura 2.8.

## 2.4 Object detection

L'object detection è uno dei temi che verranno affrontati negli esperimenti; in questa sezione verrà analizzato cosa si intende per object detection e come essa viene utilizzata.

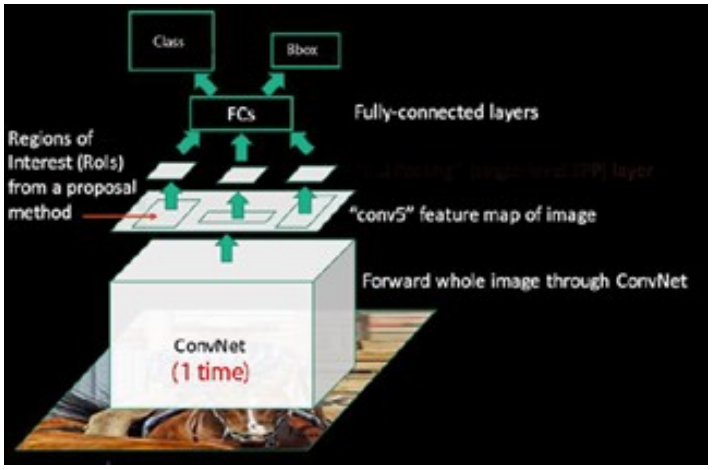


**Figura 2.8:** Struttura della Fast R-CNN

L'object detection è una tecnica utilizzata dalla Computer Vision per individuare oggetti in immagini o video. Questi tipi di algoritmi sfruttano tipicamente l'apprendimento automatico, o il Deep Learning, con l'obiettivo di replicare il riconoscimento degli oggetti, così come farebbe un umano con la vista.

### 2.4.1 Cos'è il riconoscimento delle immagini

Per riconoscimento delle immagini nel contesto dell'object detection si intende il processo attraverso il quale un sistema rileva la presenza di oggetti in un'immagine e identifica la tipologia di oggetto. Per realizzare ciò, è necessario combinare la localizzazione precisa degli oggetti con la loro classificazione il più accurata possibile.

Localizzare un oggetto implica determinare la posizione esatta all'interno dell'immagine; la localizzazione viene realizzata attraverso la generazione di bounding box, ossia rettangoli che delimitano l'area dell'immagine occupata dall'oggetto.

Successivamente alla localizzazione si procede con la classificazione, che comporta l'assegnazione di etichette riferite a delle classi per ogni tipologia di oggetto rilevato.

### 2.4.2 Come vengono identificati gli oggetti

Nel dettaglio, per identificare gli oggetti sono necessari i seguenti passi fondamentali:

- *pre-elaborazione dell'immagine*; la pre-elaborazione può includere operazioni come ridimensionamento dell'immagine, normalizzazione dei colori e data augmentation;
- *estrazione delle feature*; l'immagine pre-elaborata viene passata attraverso una rete neurale che ne estrae le feature;
- *classificazione e regressione delle bounding box*; tramite le feature si identificano le classi degli oggetti presenti e si analizzano le celle della griglia per affinare le coordinate delle bounding box.

### 2.4.3 Approcci non neurali

Analizziamo, di seguito, gli approcci non neurali per l'object detection, ampiamente utilizzati prima dell'avvento delle Deep Neural Network.

- *L'algortimo Haar Cascade* utilizza una serie di filtri, i quali analizzano variazioni di intensità dei pixel. Viene implementato come un cascade classifier, in cui i vari stadi di filtraggio vengono applicati in sequenza; ciascuno stadio è progettato per ridurre il numero di falsi positivi. L'utilizzo di tale tipologia di algortimi è diffuso nel rilevamento di volti.
- *I modelli di Mixture di Gaussiane* sono una tecnica statistica utilizzata per rappresentare la distribuzione di dati, in termini di una combinazione di diverse distribuzioni gaussiane. Ogni distribuzione gaussiana nella mixture rappresenta un componente del modello e viene caratterizzata da una media e una varianza.
- *La tecnica Hough Transform* viene utilizzata per l'elaborazione delle immagini, principalmente per il rilevamento di forme geometriche, come linee, cerchi e altre forme specifiche in un'immagine. Tale metodologia si basa sulla trasformazione matematica, che converte un'immagine da coordinate cartesiane a coordinate polari.
- *Il metodo basato sul Template Matching* consiste nel far scorrere un template su tutta l'immagine target e nel misurare la somiglianza in ogni posizione; esso si basa, pertanto, su tecniche di correlazione e confronto diretto dei pixel.

Questi approcci non neurali sono stati fondamentali nello sviluppo iniziale dell'object detection, anche se le tecniche basate sulle Deep Neural Network, come le CNN (Convolutional Neural Network), sono ora prevalenti.

### 2.4.4 Approcci basati sulle reti neurali

Gli approcci per l'object detection basati sulle reti neurali hanno rivoluzionato il campo della Computer Vision negli ultimi anni, grazie alla loro capacità di apprendere caratteristiche complesse direttamente dai dati. Questi metodi utilizzano principalmente Convolutional Neural Network per l'estrazione delle feature e la classificazione.

È possibile utilizzare diverse tecniche per eseguire il rilevamento degli oggetti. Gli approcci più diffusi basati sul Deep Learning che utilizzano CNN, come R-CNN e YOLO, imparano automaticamente a rilevare oggetti all'interno delle immagini.

Esistono due tipologie di reti utilizzabili:

- Le *Single-Stage Network*: sono reti nelle quali la CNN produce previsioni per le regioni dell'intera immagine, utilizzando anchor box; le previsioni vengono decodificate per generare i riquadri di delimitazione finali per gli oggetti. Tali reti possono essere molto più veloci delle Two-Stage Network, ma potrebbero non raggiungere lo stesso livello di precisione, soprattutto per scene contenenti piccoli oggetti.
- *Two-Stage Network*; la fase iniziale di tale rete identifica le region proposal o i sottoinsiemi dell'immagine che potrebbero contenere un oggetto. La seconda fase classifica gli oggetti all'interno delle region proposal. Le Two-Stage Network possono ottenere risultati di rilevamento degli oggetti molto accurati; tuttavia, sono più lente delle Single-Stage Network.

Oltre alle Deep Neural Network, anche le tecniche di Machine Learning sono comunemente utilizzate per il rilevamento di oggetti e offrono approcci diversi rispetto al Deep Learning. Le tecniche comuni di Machine Learning includono:

- *Aggregate Channel Features (ACF)*;
- *Classificazione SVM utilizzando Histogram of Oriented Gradients (HOG)*;

### 2.4.5 Utilizzi

Il riconoscimento delle immagini nell'object detection ha numerose applicazioni pratiche; in particolare

- nei veicoli autonomi consentono di rilevare e classificare pedoni, segnali stradali, veicoli e ostacoli;
- nella sorveglianza permettono di identificare persone e armi;
- nella medicina consentono identificare fratture e tumori.

Ciasuno di questi possibili utilizzi offre supporto all'attività umana.

## 2.5 Instance segmentation

L'instance segmentation, la quale è un sottoinsieme del campo della segmentazione delle immagini, fornisce risultati più dettagliati e sofisticati rispetto agli algoritmi convenzionali di object detection. Essa è un'applicazione della Computer Vision, che prevede gli esatti confini in termini di pixel di ogni singola istanza di oggetto in un'immagine.

### 2.5.1 La segmentazione

Instance segmentation e object detection differiscono poichè la prima prevede i confini a livello di pixel di ciascun oggetto, mentre la seconda si limita solo a prevedere la posizione approssimativa di un oggetto. Gli algoritmi di instance segmentation generano una "segmentation mask" pixel per pixel della forma e dell'area precise per ogni istanza. Molte delle principali architetture di modelli di instance segmentation, come Mask R-CNN, eseguono l'object detection come passaggio preliminare nel processo di generazione delle segmentation mask. Tali modelli "a due stadi" forniscono un'elevata precisione, anche se con un compromesso in termini di velocità di elaborazione.

### 2.5.2 Instance Segmentation, semantic segmentation e panoptic segmentation

Esistono diverse metodologie per la segmentazione delle immagini, ognuna con scopi e applicazioni specifiche. Le principali tecniche di segmentazione includono la *semantic segmentation*, l'*instance segmentation* e la *panoptic segmentation*.

La *semantic segmentation* è una tecnica meno complessa della instance segmentation. A differenza di quest'ultima, la semantic segmentation non si occupa del conteggio o della distinzione tra diverse istanze; l'unico scopo della semantic segmentation è annotare ogni pixel in un'immagine con l'etichetta di una classe. Se più istanze di oggetti della stessa classe sono strettamente adiacenti o si sovrappongono l'una all'altra, il modello le raggrupperà insieme, all'interno di un singolo segmento.

La *panoptic segmentation*, invece, esegue sia la classificazione semantica di ogni pixel in un'immagine, sia la delineazione di ogni diversa istanza di oggetto. I tentativi iniziali per realizzare questo tipo di segmentazione, eseguivano sia la segmentazione delle istanze, che quella semantica separatamente, quindi, combinavano i loro risultati in una fase di post-elaborazione. Tale metodo è computazionalmente inefficiente e incontra difficoltà nell'unire semanticamente gli output delle due segmentazioni. Approcci più recenti collegano una "head" di semantic segmentation e una "head" di instance segmentation, responsabili di classificare ogni pixel dell'immagine in una delle categorie semantiche predefinite e di individuare e separare le diverse istanze di oggetti presenti nell'immagine.

Si riporta in Figura 2.9 un'immagine di esempio, per confrontare le varie metodologie.

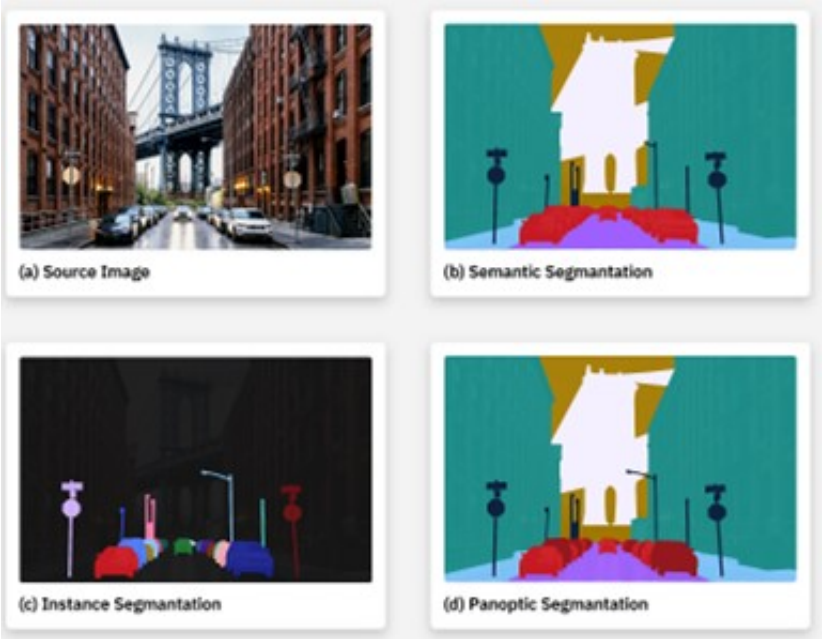


**Figura 2.9:** Confronto tra instance segmentation, semantic segmentation e panoptic segmentation

### 2.5.3 Funzionamento pratico

Il Deep Learning è diventato fondamentale per l'instance segmentation; quasi tutti i moderni metodi di segmentazione delle immagini utilizzano reti neurali. Sebbene negli ultimi anni i modelli Transformer siano emersi come una valida alternativa, la maggior parte dei metodi di instance segmentation sfruttano alcune forme di Convolutional Neural Network (CNN). Tali modelli utilizzano una struttura codificatore-decodificatore, in cui una rete codificatore viene utilizzata per estrarre i dati rilevanti dall'immagine di input e una rete decodificatore utilizza i dati delle feature estratte per ricostruire l'immagine con una mappa di segmentazione.

Un esempio di architettura spesso utilizzata è l'architettura Mask R-CNN, la quale ha abbinato l'object detection di una Faster R-CNN con le capacità di segmentazione di una Fully Convolutional Network (FCN). Dopo che la Region Proposal Network (RPN) ha generato riquadri di delimitazione per gli oggetti proposti, il resto della rete Faster R-CNN conferma quali region proposal contengono effettivamente degli oggetti; la FCN crea, poi, una segmentation mask degli oggetti contenuti all'interno di ogni riquadro di delimitazione. Tale processo è efficace anche quando gli oggetti sono occlusi poiché la Faster R-CNN riesce a distinguere ciascuna istanza di oggetto per garantire che le istanze siano segmentate individualmente.

Gli algoritmi di Deep Learning utilizzati per l'instance segmentation devono essere addestrati. L'addestramento avviene con la backpropagation, tramite la quale i modelli effettuano il reverse engineering su immagini di addestramento annotate, per apprendere i pesi e i bias appropriati per il task da svolgere. L'annotazione dei training set deve essere molto accurata per massimizzare il corretto apprendimento automatico e fungere, inoltre, da punto di riferimento "di base" rispetto al quale i modelli addestrati possano essere valutati e ottimizzati. Poiché le capacità umane superano di gran lunga anche i modelli di Computer Vision più accurati, questa annotazione viene eseguita manualmente; ciò la rende un processo costoso e ad alta intensità di manodopera. Per evitare i tempi e i costi di dataset personalizzati, la maggior parte dei modelli utilizza training set open source di grandi dimensioni.

### 2.5.4 Possibili utilizzi

L'instance segmentation è essenziale per una serie di attività di Computer Vision, ovvero:

- *diagnostica per immagini*: la segmentazione viene utilizzata per rilevare i confini specifici di tessuti e patologie, come i tumori;
- *guida autonoma*: l'instance segmentation consente alle auto a guida autonoma di rilevare e classificare con precisione auto, oggetti, persone ed elementi stradali;
- *immagini satellitari*: la segmentazione può aiutare a identificare e isolare oggetti di interesse, ad esempio distinguendo tra più edifici lungo una strada per offrire un servizio GPS;
- *robotica*: l'instance segmentation consente lo smistamento degli articoli, il rilevamento dei difetti e, analogamente alle auto a guida autonoma, consente ai robot di discernere e spostarsi attorno agli oggetti nel loro ambiente.

## 2.6 Pose estimation

La stima della posa 3D nasce dalla Computer Vision; è un processo di previsione della trasformazione di un oggetto da una posa di riferimento definita dall'utente, data un'immagine o una scansione 3D. Gli oggetti presi in considerazione possono essere piuttosto generali, potendo comprendere persone, animali o parti del corpo. I metodi utilizzati per determinare la posa di un oggetto, tuttavia, sono solitamente specifici per una classe di oggetti e generalmente non ci si può aspettare che lo stesso modello funzioni bene per altri tipi di oggetti.

### 2.6.1 Cosa vuol dire stimare una posa

La stima della posa umana mira a prevedere le pose delle parti e delle articolazioni del corpo umano in immagini o video; tale stima può essere effettuata da dati in 2D oppure in 3D.

La stima della posa umana 2D viene utilizzata per stimare la posizione spaziale dei punti chiave del corpo umano da elementi visivi come immagini e video. I tradizionali metodi di stima della posa umana 2D utilizzano diverse tecniche di estrazione delle feature. I primi lavori di Computer Vision descrivevano il corpo umano come una figura stilizzata per ottenere strutture di posa globali. Tuttavia, i moderni approcci basati sul Deep Learning hanno raggiunto importanti progressi migliorando significativamente le prestazioni per la stima delle pose sia di una sola persona che di più persone. Alcuni metodi popolari di stima della posa umana 2D includono OpenPose, CPN, AlphaPose e HRNet.

La stima della posa umana 3D, invece, viene utilizzata per prevedere la posizione delle articolazioni del corpo nello spazio 3D; oltre alla posa, alcuni metodi recuperano anche mesh umane 3D da immagini o video. L'analisi della posa umana 3D può essere eseguita su immagini o video monoculari, oppure utilizzando più punti di vista o sensori aggiuntivi. La raccolta di annotazioni accurate delle immagini di posa 3D richiede molto tempo e l'etichettatura manuale non è pratica ed è costosa. Pertanto, ci sono ancora diverse sfide da superare; tra cui la generalizzazione del modello, la robustezza all'occlusione e l'efficienza di calcolo.

### 2.6.2 Funzionamento

La stima della posa utilizza la posa e l'orientamento per prevedere e tracciare la posizione di una persona o di un oggetto. In generale, la maggior parte degli stimatori di posa sono strutture a 2 passaggi, che rilevano i riquadri di delimitazione umani e, quindi, stimano la posa all'interno di ciascun riquadro. La stima della posa funziona trovando i punti chiave di una persona o di un oggetto. Prendendo, ad esempio una persona, i punti chiave sarebbero le articolazioni, come il gomito, le ginocchia, i polsi, le caviglie, e così via. Esistono due tipi di stima: multipose e single pose. La stima single pose viene utilizzata per stimare le pose di un singolo oggetto in una determinata scena, mentre la stima multipose viene utilizzata quando si rilevano pose per più soggetti. Ogni punto chiave è annotato con tre variabili $(x, y, v)$, dove $x$ e $y$ indicano le coordinate e $v$ indica se il punto chiave è visibile.

OpenPose è uno degli approcci bottom-up più popolari per la stima di posa in tempo reale e per più persone. Esso è un framework open source adatto per ottenere un'elevata precisione nel rilevamento dei punti chiave del corpo, del piede, della mano e del viso. Un vantaggio di OpenPose è che si tratta di un'API che offre agli utenti la flessibilità di selezionare immagini sorgente da campi di telecamere, webcam, etc., soprattutto per applicazioni di sistemi embedded. OpenPose supporta diverse architetture hardware, come GPU CUDA, GPU OpenCL o dispositivi solo CPU. La versione leggera è sufficientemente efficiente per le applicazioni di inferenza Edge con elaborazione sul dispositivo in tempo reale con dispositivi Edge. In Figura 2.10 si presenta un esempio di Pose Estimation.

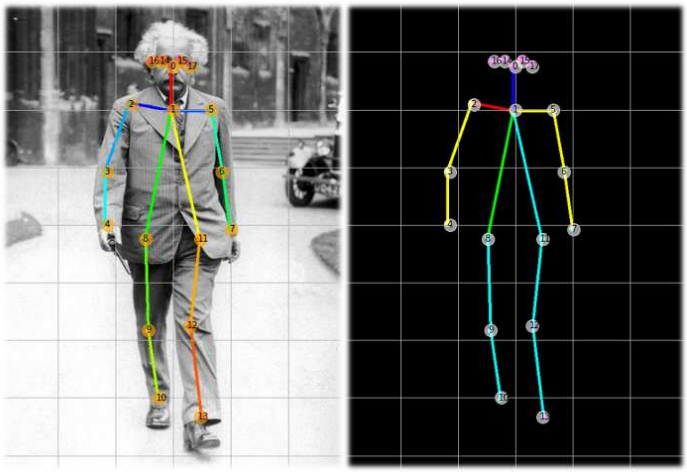

**Figura 2.10:** Un esempio di pose estimation

La stima AlphaPose è, invece, un popolare metodo top-down per la stima della posa. È utile per rilevare le pose in presenza di riquadri di delimitazione umani imprecisi. In altre parole, essa è un'architettura per stimare le pose umane tramite riquadri di delimitazione rilevati in modo ottimale. L'architettura AlphaPose è applicabile per rilevare pose singole e multiple in immagini o campi video.

La differenza tra i due approcci nominati risiede nel modo in cui affrontano il problema della stima della posa. Nell'approccio bottom-up, il processo inizia con il rilevamento delle parti del corpo in un'immagine, indipendentemente dagli individui presenti. Successivamente, queste parti rilevate vengono raggruppate per formare pose individuali per ciascuna persona. Nell'approccio top-down, invece, il processo inizia con il rilevamento delle persone nell'immagine. Una volta che una persona è stata rilevata, viene applicato un modello di stima della posa su ciascuna regione rilevata per determinare le articolazioni del corpo.

### 2.6.3 Utilizzi comuni

La human pose estimation viene utilizzata in un'ampia gamma di applicazioni, tra cui il riconoscimento delle azioni, la cattura del movimento, l'analisi del movimento, la realtà aumentata, lo sport, il fitness e la robotica.

Architetture come DensePose, PoseNet o OpenPose vengono spesso utilizzate per il riconoscimento di attività, gesti o andatura. Esempi di monitoraggio dell'attività umana tramite l'uso della stima della posa includono:

- applicazioni per riconoscere i gesti delle mani, per analizzare le espressioni facciali oppure per analizzare il linguaggio del corpo/dei segni;
- applicazioni per analizzare tecniche di danza;
- applicazioni intelligenti per rilevare la caduta di persone o il processo di determinate malattie come il morbo di Alzheimer;
- applicazioni fitness per rilevare la forma di esecuzione degli esercizi e per contare le ripetizioni.

# CAPITOLO 3

## Metriche utilizzate per la valutazione del modello

*In questo capitolo si analizzeranno gli strumenti utilizzati per la valutazione di un modello di Intelligenza Artificiale. Le metriche sono una parte fondamentale per un modello; esse consentono di confrontare diversi modelli tra loro e di identificare quale funziona meglio per uno scopo specifico. In secondo luogo, aiutano a comprendere le aree in cui il nostro modello può migliorare, consentendo di iterare e ottimizzare il processo di addestramento.*

## 3.1 Principali parametri

Di seguito sono riportate alcune delle principali metriche di valutazione:

- La *Precision* è la frazione delle predizioni corrette positive rispetto al totale delle predizioni positive. Tale metrica è indicatrice di quanto sia affidabile il modello quando predice la classe positiva. La Precision è descritta dalla seguente formula:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- La *MeanAveragePrecision* è definita come la media delle Average Precision (AP) per ogni classe. L'AP misura la precisione media del modello su un insieme di classi o categorie. Dal momento che la MeanAveragePrecision rappresenta la media delle AP, essa tiene conto delle prestazioni del modello su tutte le categorie di oggetti da rilevare o classificare.

- Il *Recall* è il rapporto tra le previsioni corrette per una classe e totale dei casi in cui si verifica effettivamente.

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

- L'*Accuracy* è indicatrice di quante volte il modello ha correttamente classificato un'istanza nel dataset rispetto al totale. Infatti, la formula per l'Accuracy consiste nel rapporto tra il numero di predizioni corrette e il totale delle predizioni:

$$\text{Accuracy} = \frac{\text{Numero di predizioni corrette}}{\text{Totale delle predizioni}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

  L'Accuracy tiene conto delle performance del modello in senso stretto. In altre parole, essa non permette di comprendere il contesto nel quale si sta operando. Ad esempio,

è sconsigliato usare l'Accuracy come metrica di valutazione quando si opera con un dataset sbilanciato, dove le classi sono distribuite in maniera impari, poichè se una classe rappresenta il 90% del dataset e il modello classifica erroneamente ogni campione come appartenente a quella classe, l'Accuracy risulterà del 90%. Nonostante si abbia un'alta Accuracy, il modello non sarà realmente performante, portando a una falsa interpretazione delle prestazioni.

- L'*F1 Score* è una metrica che combina la Precision e il Recall per restituire un unico valore. Essa è utile quando è necessario bilanciare queste due metriche, cioè, quando entrambe sono considerate importanti al fine di valutare il modello. L'F1 è la media armonica di Precision e Recall:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- L'*IoU*, o *Intersection over Union*, misura l'Accuracy della localizzazione di un object detector in uno specifico dataset. Essa misura la sovrapposizione tra il box di ground truth con il box predetto.

$$\text{IoU} = \frac{\text{Area di sovrapposizione}}{\text{Area di unione}}$$

- La *loss* si divide in *loss_bbox*, che misura quanto sono vicine le box predette rispetto all'oggetto di ground truth, in *loss_cls*, che misura la correttezza della classificazione di ogni box predetta, e in *dfl_loss*, che considera il problema dello squilibrio di classe, ossia la presenza di uno scenario in cui vi è una classe che si verifica troppo frequentemente e un'altra che si verifica più raramente.

Tali metriche verranno utilizzate all'interno degli esperimenti per quantificare la qualità del lavoro svolto.

## 3.2 Confusion matrix

Una *confusion matrix*, in italiano *matrice di confusione*, è una matrice che riassume le prestazioni di un modello di Machine Learning su un insieme di dati di test. Essa è un mezzo per visualizzare il numero di istanze corrette e di istanze incorrette predette dal modello. La confusion matrix viene spesso utilizzata per misurare le prestazioni dei modelli di classificazione, che mirano a prevedere un'etichetta di classificazione per ciascuna istanza.

Tale matrice offre un'analisi approfondita delle previsioni veri positivi, veri negativi, falsi positivi e falsi negativi, consentendo il calcolo delle metriche citate in precedenza; si presenta come una tabella con 4 diverse combinazioni di valori previsti e reali, come in Figura 3.1.

### 3.2.1 TP, TN, FP e FN

La confusion matrix (Figura3.1) si divide in 4 regioni, ovvero:

- *true positive* (TP) o *veri positivi*: si verificano quando il modello prevede un valore positivo e il valore è effettivamente positivo;
- *true negative* (TN) o *veri negativi*: si verificano quando il modello prevede un valore negativo e il valore è effettivamente negativo;

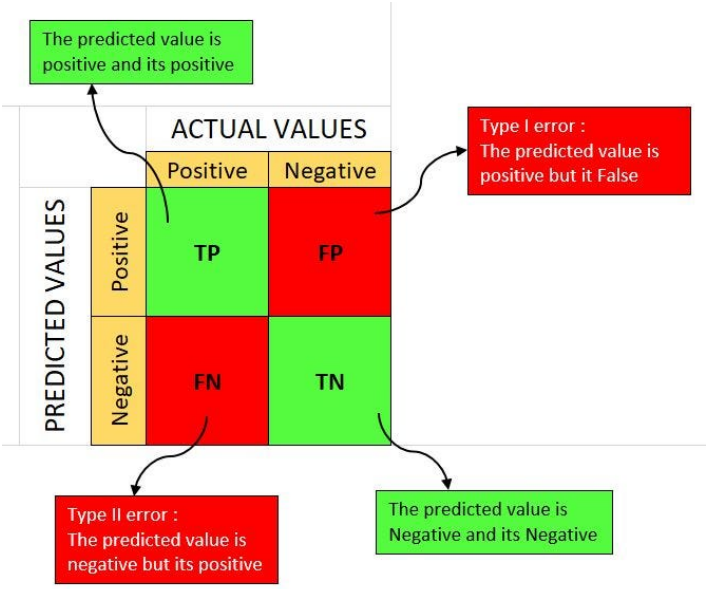


**Figura 3.1:** Confusion matrix

- *false positive* (FP) o *falsi positivi*: si verificano quando il modello prevede un valore positivo ma il valore reale è negativo;
- *false negative* (FN) o *falsi negativi*: si verificano quando il modello prevede un valore negativo ma il valore è positivo.

Tali valori sono gli stessi che vengono utilizzati per calcolare le metriche fondamentali sopra citate.

## 3.3 Overfitting

Quando vengono costruiti gli algoritmi di Machine Learning, questi utilizzano un training set. Tuttavia, quando il modello si addestra troppo a lungo sui dati o quando esso è troppo complesso, può iniziare ad apprendere il "rumore", o informazioni irrilevanti, all'interno del set di dati. Se il modello memorizza il rumore e si adatta troppo strettamente al training set, esso diventa soggetto a "overfitting" e non è in grado di effettuare previsioni o conclusioni accurate da dati diversi da quelli di addestramento. Un modello che non è in grado di generalizzare correttamente sui nuovi dati, non sarà in grado di eseguire le attività di classificazione o stima per le quali è stato progettato.

Ad esempio, considerando un caso d'uso in cui un modello di Machine Learning deve analizzare delle foto e identificare quelle che contengono cani, se il modello è stato addestrato su un training set che contiene per la maggior parte foto di cani all'aperto nei parchi, potrebbe imparare a usare l'erba come caratteristica per la classificazione e non riconoscere un cane all'interno di una stanza.

### 3.3.1 Come rilevarlo

Il metodo migliore per rilevare l'overfitting nei modelli consiste nel testarli su più dati con una rappresentazione completa dei possibili valori e tipi di dati di input. In genere, una parte del training set viene utilizzata come test set per verificare l'overfitting. Se i dati di

addestramento hanno un tasso di errore basso e i dati di test hanno un tasso di errore elevato, allora siamo in presenza di un chiaro segnale di overfitting.

La convalida incrociata K-fold, è uno dei metodi di test utilizzati nella pratica. Con tale metodologia il training set viene diviso in K sottoinsiemi di dimensioni uguali chiamati fold. Uno dei k-fold fungerà da set di test, noto anche come set di holdout o set di convalida, e i restanti fold addestreranno il modello. Dopo ogni valutazione viene mantenuto un punteggio e, al termine di tutte le iterazioni, viene calcolata la media dei punteggi per valutare le prestazioni del modello complessivo. Le iterazioni si ripetono finché il modello non viene testato su ogni set di campioni. È, quindi, possibile calcolare la media dei punteggi in tutte le iterazioni per ottenere la valutazione finale del modello predittivo.

### 3.3.2 Soluzione

È possibile prevenire l'overfitting diversificando e dimensionando il set di dati di addestramento o utilizzando altre strategie, come:

- *Arresto anticipato*: consiste nell'interrompere la fase di addestramento prima che il modello riconosca il rumore nei dati; tale approccio rischia di arrestare troppo presto il processo di addestramento, portando al problema opposto dell'underfitting.
- *Eliminazione*: identifica le feature più importanti all'interno del training set ed elimina le feature irrilevanti che potrebbero generare rumore.
- *Regolarizzazione*: applica una "penalità" ai parametri di input con i coefficienti maggiori, il che limita la quantità di varianza nel modello e cerca di identificare e ridurre il rumore all'interno dei dati.
- *Ensembling*: combina le previsioni di diversi algoritmi di Machine Learning separati. I due principali metodi di ensemble sono il bagging e il boosting. Il boosting addestra diversi modelli uno dopo l'altro per ottenere il risultato finale, mentre il bagging addestra i modelli in parallelo.
- *Aumento del training set*: l'ampliamento del training set con un maggior numero di dati può aumentare l'Accuracy del modello, offrendo maggiori opportunità di individuare la relazione dominante tra le variabili di input e di output. Tale metodo risulta efficace quando nel modello vengono inseriti dati pertinenti e privi di rumore; altrimenti, si aggiungerebbe ulteriore complessità al modello, causandone l'overfitting.

## 3.4 Come si valuta la bontà di un modello

Tutte le metriche sopra citate sono misure quantitative della performance di un modello. Tuttavia, esistono vari aspetti qualitativi che sono cruciali per valutare la "bontà" di un modello di IA, come:

- *generalizzazione*: capacità di mantenere alte prestazioni quando viene applicato a nuovi dati;
- *scalabilità*: capacità di gestire dataset crescenti e problemi sempre più complessi;
- *affidabilità*: robustezza di fronte a dati perturbati, rumorosi o mancanti;
- *usabilità*: facilità con cui il modello può essere utilizzato e interpretato anche da persone senza competenze tecniche avanzate;

- *efficienza*: uso ottimale delle risorse, inclusi tempi di addestramento e requisiti hardware.

È bene specificare che non esiste un modello che massimizzi contemporaneamente tutti questi parametri; migliorare certi aspetti può spesso deteriorarne altri, per cui, in genere, è necessario trovare un trade-off. Tale aspetto è centrale nella progettazione e nell'ottimizzazione dei modelli di Machine Learning.

Ad esempio, se il modello cerca di aumentare il Recall, tenterà di ridurre il numero di FN abbassando la soglia per poter classificare il maggior numero di istanze positive, aumentando così i TP e i FP. Il Recall aumenta perché il modello identifica più TP, ma la Precision diminuisce perché l'aumento di FP incrementa il denominatore (TP + FP) nella corrispettiva formula.

# CAPITOLO 4

# Rilevamento degli incidenti

*All'interno di questo capitolo si discuterà del primo dei tre esperimenti condotti. Si darà una breve descrizione e si mostreranno i passi fondamentali percorsi per l'allenamento dei modelli; si confronteranno, inoltre, i modelli ottenuti.*

## 4.1 Descrizione

La selezione del task per questo esperimento è stata motivata da una questione di vitale importanza, con conseguenze che talvolta coinvolgono vite umane. Il rilevamento degli incidenti stradali è essenziale per massimizzare le possibilità di salvataggio, poiché la prontezza nell'individuare un incidente può drasticamente ridurre i tempi di intervento dei soccorsi e facilitare la gestione del traffico, soprattutto in contesti come le autostrade. In questo campo l'impiego di modelli di Intelligenza Artificiale può migliorare l'efficienza nei tempi di rilevamento, dato che un controllo simultaneo di migliaia di telecamere risulterebbe oneroso per un operatore umano.

L'obiettivo di questo esperimento è quello di ottenere un modello di Intelligenza Artificiale in grado effettuare una object detection degli incidenti stradali, con la funzionalità aggiuntiva di poter riconoscere il tratto di carreggiata nella quale è avvenuto l'incidente.

Il processo per arrivare ad ottenere un modello di IA è iniziato dall'acquisizione dei dataset, che hanno permesso l'allenamento di modelli YOLOv8 e YOLOv9, che verranno confrontati in seguito.

## 4.2 Dataset

Nel processo di sviluppo del modello per il rilevamento degli incidenti stradali, si è optato per l'utilizzo di dataset già etichettati, poiché la creazione di un dataset da zero avrebbe richiesto un considerevole impegno temporale che non era al momento prioritario.

Per garantire un addestramento efficace del modello, è essenziale disporre di un dataset ampio e variegato che copra una vasta gamma di scenari di incidenti. A tal fine, si è scelto di utilizzare i dataset forniti da Roboflow, data la loro ampia disponibilità e diversità.

In particolare si sono scelti i dataset considerati più idonei tra tutte le possibili scelte, ovvero:

- *Primo dataset*: consiste in 2632 immagini, suddivise nel 94% come training set, nel 4% come validation set e nel restante 2% come test set. Le immagini sono state pre-elaborate con un ridimensionamento a 640x640 e sono soggette a un processo di augmentation che include la conversione in scala di grigi per il 5% di esse, variazioni di esposizione tra il -23% e il +23%, e inversione delle bounding box orizzontalmente. Le immagini sono classificate come "accident", "motorcycle", "car" oppure "null".
- *Secondo dataset*: comprende 3092 immagini, suddivise nel 78%, 11% e 11% rispettivamente come training set, validation set e test set. Anche in questo caso, le immagini sono state pre-elaborate con un ridimensionamento a 640x640 e sono soggette a un processo di augmentation che include la conversione in scala di grigi per il 5% di esse e variazioni di luminosità tra il -20% e il +20%. Le immagini sono classificate come "accident", "non-accident" oppure "null".
- *Terzo dataset*: utilizza 7869 immagini divise in 84% di training set, 10% di validation set, 5% di test set. Le immagini vengono pre-processate con un ridimensionamento a 640x640. Si effettua, anche, un'augmentation con un capovolgimento delle immagini in orizzontale e verticale. Le immagini vengono classificate come "mild", "moderate", "severe" oppure "null".

## 4.3 Training

Per l'addestramento dei modelli si è fatto affidamento sui notebook messi a disposizione da Roboflow progettati per l'utilizzo tramite Google Colab.

Durante il processo di addestramento e validazione, si sono testati sia il modello YOLOv8 che il modello YOLOv9 su tutti e tre i dataset citati. Tuttavia, i risultati mostrati riflettono le prestazioni migliori ottenute utilizzando il terzo dataset.

Gli step per il training nel notebook si dividono nelle seguenti sezioni:

- Clonazione del Github "yolov9" di SkalskiP contenente l'implementazione del paper "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information." Il repository include il codice per l'addestramento e la valutazione dei modelli YOLOv9.

  Per YOLOv8 si è utilizzato, invece, il GitHub "Jocher_Ultralytics_YOLO_2023".
- Installazione del package di RoboFlow e dei requisiti del repository GitHub.
- Installazione dei pesi.
- Caricamento del dataset.
- Allenamento tramite le istruzioni indicate nelle Figure 4.1 e 4.2.

```
!yolo task=detect mode=train model=yolov8m.pt data={dataset.location}/data.yaml epochs=20 imgsz=800 plots=True
```

**Figura 4.1:** Istruzioni per l'allenamento tramite YOLOv8

- Validazione tramite le istruzioni indicate nelle Figure 4.3 e 4.4.
- Inferenza tramite le istruzioni indicate nelle Figure 4.5 e 4.6.

```
!python train.py \
--batch 8 --epochs 20 --img 640 --device 0 --min-items 0 --close-mosaic 15 \
--data {dataset.location}/data.yaml \
--weights {HOME}/weights/gelan-c.pt \
--cfg models/detect/gelan-c.yaml \
--hyp hyp.scratch-high.yaml
```

**Figura 4.2:** Istruzioni per l'allenamento tramite YOLOv9

```
!yolo task=detect mode=val model={HOME}/runs/detect/train/weights/best.pt data={dataset.location}/data.yaml
```

**Figura 4.3:** Istruzioni per la validazione di YOLOv8

```
!python val.py \
--img 640 --batch 8 --conf 0.001 --iou 0.7 --device 0 \
--data {dataset.location}/data.yaml \
--weights {HOME}/yolov9/runs/train/exp/weights/best.pt
```

**Figura 4.4:** Istruzioni per la validazione di YOLOv9

```
!yolo task=detect mode=predict model={HOME}/runs/detect/train/weights/best.pt conf=0.25 source={dataset.location}/test/images save=True
```

**Figura 4.5:** Istruzioni per l'inferenza del modello in YOLOv8

```
!python detect.py \
--img 640 --conf 0.1 --device 0 \
--weights {HOME}/yolov9/runs/train/exp/weights/best.pt \
--source {dataset.location}/valid/images
```

**Figura 4.6:** Istruzioni per l'inferenza del modello in YOLOv9

Per YOLOv9 è stata utilizzata la versione più leggera, ovvero gelan-c. Esso è ottimizzato per risorse computazionali limitate, mantenendo comunque buone prestazioni di rilevamento. GELAN sta per Guided Efficient Light-weight Attention Network; è un meccanismo di attenzione leggero ed efficiente che aiuta a catturare relazioni spaziali tra le feature map.

Inoltre, per YOLOv8 si è utilizzato yolov8m, la versione di YOLOv8 di dimensioni medie, addestrata per il rilevamento degli oggetti su COCO a una risoluzione di 640x640.

### 4.3.1 Confronto tra YOLOv8 e YOLOV9

Una parte cruciale dell'implementazione del modello di rilevamento degli incidenti è la scelta della sua architettura. Si confrontano di seguito due versioni popolari del modello YOLO, ovvero YOLOv8 e YOLOv9, mettendo in evidenza le differenze prestazionali, la complessità computazionale e i risultati ottenuti.

Nelle Figure 4.7 e 4.8 si mostrano i risultati dell'addestramento con YOLOV9; si può notare che l'addestramento ha impiegato 2 ore. I risultati indicano che il modello mostra una buona Precision (P) pari a 0.811 e Recall (R) di 0.824. Le classi "moderate" e "severe" hanno entrambe Precision e Recall alti, indicando che il modello riesce a distinguere efficacemente queste categorie. La classe "mild" ha una bassa Precision di 0.605 e un Recall pari a 0.619, indicando che il modello può non riconoscere alcuni casi "mild". Il mAP50-95 per la classe "mild" è 0.443, relativamente basso rispetto alle altre categorie, suggerendo che ci sono opportunità di miglioramento per questa classe.

In Figura 4.9 si mostrano i risultati dell'inferenza del modello YOLOv9 effettuato su immagini di test.

I risultati della confusion matrix in Figura 4.8 hanno permesso di calcolare le seguenti metriche:

```
20 epochs completed in 1.998 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, saved as runs/train/exp/weights/last_striped.pt, 51.5MB
Optimizer stripped from runs/train/exp/weights/best.pt, saved as runs/train/exp/weights/best_striped.pt, 51.5MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
gelan-c summary: 467 layers, 25413273 parameters, 0 gradients, 102.5 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 52/52 [00:21<00:00,  2.40it/s]
                   all        823        856      0.811      0.824      0.857      0.724
                  mild        823         21      0.605      0.619      0.646      0.443
              moderate        823        216      0.869      0.894      0.939      0.838
                severe        823        619       0.96      0.959      0.987      0.892
Results saved to runs/train/exp
```

**Figura 4.7:** Risultato dell'addestramento in YOLOv9

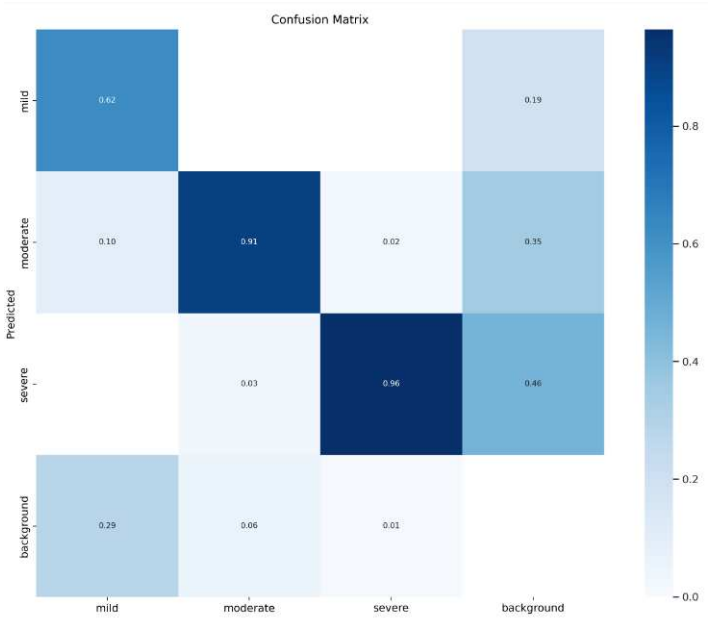


**Figura 4.8:** Confusion matrix con YOLOv9


**Figura 4.9:** Risultati dell'inferenza con YOLOV9

$TPMild = 0.62 \times 21 = 13$
$TPModerate = 0.91 \times 216 = 197$
$TPSevere = 0.96 \times 619 = 594$
$TP = TPMild + TPModerate + TPSevere = 804$
L'Accuracy è calcolata come il rapporto tra il totale dei True Positive (TP) e il numero totale di istanze:

$Accuracy = TP/TotalInstances = 804/856 = 0.939 = 93.9\%$

Questo valore di Accuracy indica un risultato soddisfacente del modello nell'identificare correttamente le diverse classi di incidenti stradali.

Nelle Figure 4.10 e 4.11 si mostrano i risultati dell'addestramento con YOLOV8; si può notare che l'addestramento ha impiegato poco meno di 2 ore; i risultati indicano che il modello si comporta bene nelle categorie "moderate" e "severe" con Precision e Recall elevati, mentre le prestazioni per la categoria "mild" sono più basse.

```
20 epochs completed in 1.971 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 52.1MB
Optimizer stripped from runs/detect/train/weights/best.pt, 52.1MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 218 layers, 25841497 parameters, 0 gradients, 78.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 26/26 [00:20<00:00,
                   all        823        856      0.939      0.786      0.857      0.753
                  mild        823         21      0.916      0.524      0.634      0.492
              moderate        823        216      0.918      0.894      0.947      0.858
                severe        823        619      0.983      0.942      0.989       0.91
Speed: 0.4ms preprocess, 15.1ms inference, 0.0ms loss, 2.2ms postprocess per image
Results saved to runs/detect/train
💡 Learn more at https://docs.ultralytics.com/modes/train
```

**Figura 4.10:** Risultato dell'allenamento in YOLOv8

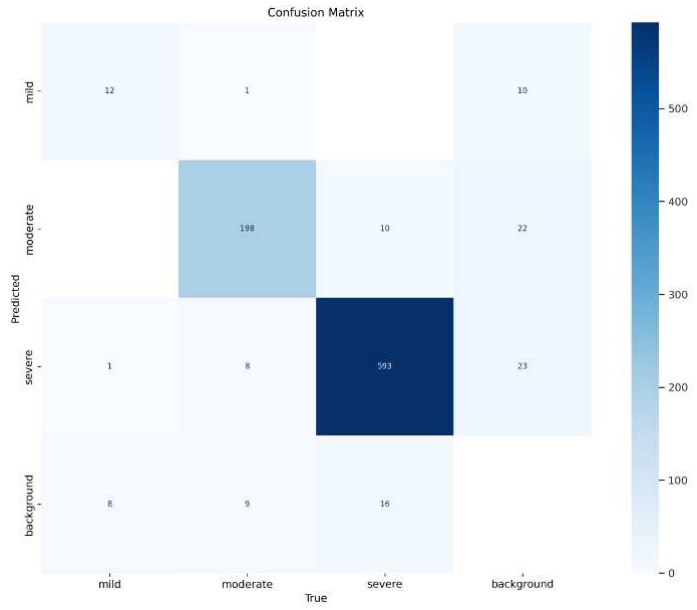


**Figura 4.11:** Confusion matrix con YOLOv8

In Figura 4.12 si mostrano i risultati dell'inferenza del modello effettuata su immagini di test.

I risultati della confusion matrix presentati nella Figura 4.11 hanno permesso di calcolare le seguenti metriche:

$TP = TP_{Mild} + TP_{Moderate} + TP_{Severe} = 12 + 198 + 593 = 803$

L'Accuracy è calcolata come il rapporto tra il totale dei True Positive (TP) e il numero totale di istanze:

$Accuracy = TP/TotalInstances = 803/856 = 0.938 = 93.8\%$

Questo valore di Accuracy indica un risultato molto simile a quello ottenuto precedentemente, confermando l'efficacia del modello nel rilevare correttamente gli incidenti stradali.

A seguito del calcolo di tutte le metriche visualizziamo, nelle Figure 4.13 e 4.14, un grafico di confronto tra le metriche calcolate per YOLOv8 e per YOLOv9.

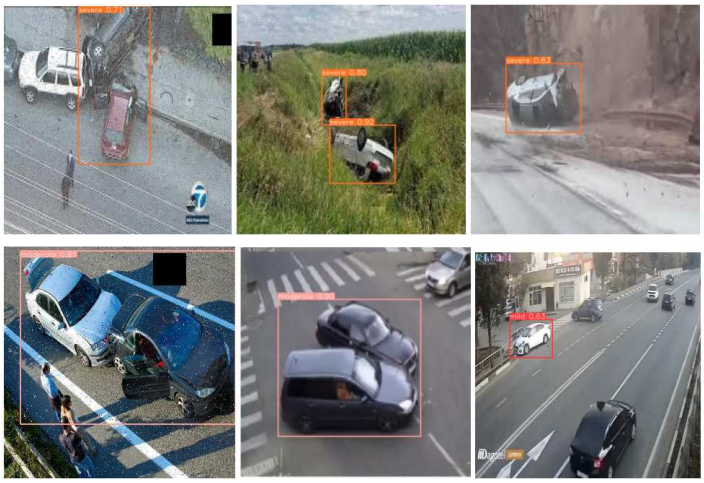

**Figura 4.12:** Risultati dell'inferenza con YOLOv8

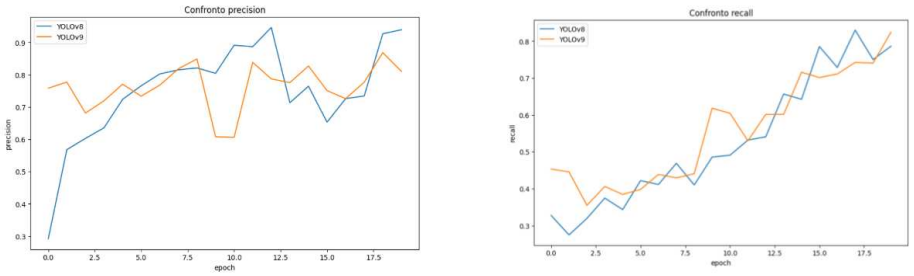


**Figura 4.13:** Confronto tra Precision e Recall per YOLOv8 e YOLOv9

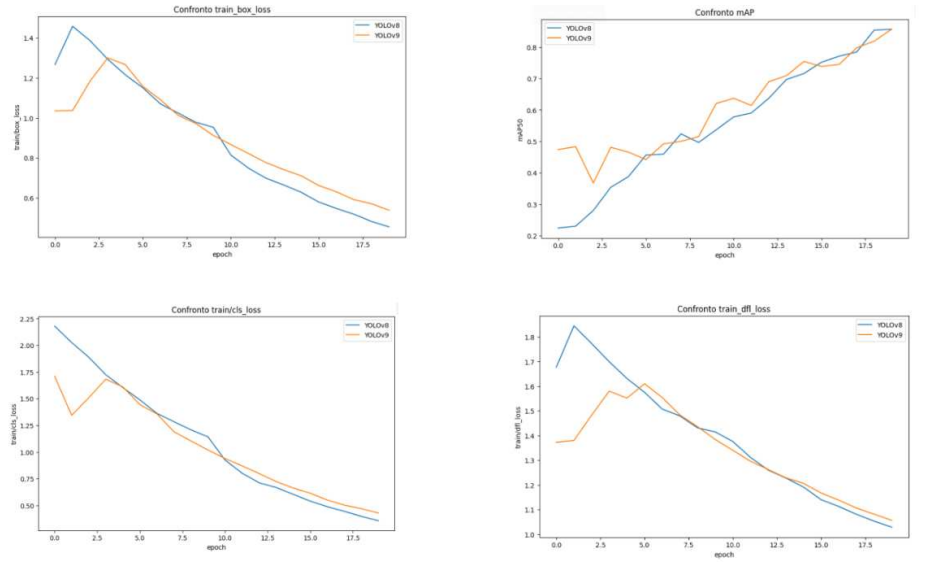


**Figura 4.14:** Confronto tra ulteriori metriche per YOLOv8 e YOLOv9

In Figura 4.15 viene illustrato un confronto diretto tra le due inferenze; si può notare che il punteggio di confidence è leggermente maggiore in YOLOv9.


**Figura 4.15:** Confronto diretto tra le inferenze di YOLOv8 e YOLOv9

## 4.4 Stato dell'arte

È stata condotta un'analisi approfondita su vari studi al fine di esaminare lo stato attuale della ricerca nel campo del rilevamento degli incidenti attraverso l'impiego di modelli di Intelligenza Artificiale.

### 4.4.1 Prima Fonte: Accident-Detection-System

Uno dei primi studi considerati presenta un programma composto da quattro sezioni:

- *data*: utilizza un dataset di Kaggle sul rilevamento degli incidenti dalle riprese delle telecamere di sorveglianza;
- *accident-classification.ipynb*: si tratta di un notebook Jupyter che genera un modello per classificare i dati sopra menzionati; questo file produce due file importanti, ovvero `model.json` e `model_weights.h5`;
- *detection.py*: carica il sistema di rilevamento degli incidenti utilizzando i file `model.json` e `model_weights.h5`;
- *camera.py*: gestisce la telecamera ed esegue il file `detection.py` sul video, suddividendolo in frame e mostrando la percentuale di previsione di incidenti (se presenti) in ciascun frame.

Il modello in Figura 4.16 è stato addestrato con 20 epoche. Nella Figura 4.17 si mostrano i grafici correlati ai risultati ottenuti.

```
model = tf.keras.models.Sequential([
  layers.BatchNormalization(),
  layers.Conv2D(32, 3, activation='relu'), # Conv2D(f_size, filter_size, activation)
  layers.MaxPooling2D(), # Max Pooling
  layers.Conv2D(64, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(128, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(256, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(512, activation='relu'),
  layers.Dense(len(class_names), activation= 'softmax')
])
```

**Figura 4.16:** Modello utilizzato

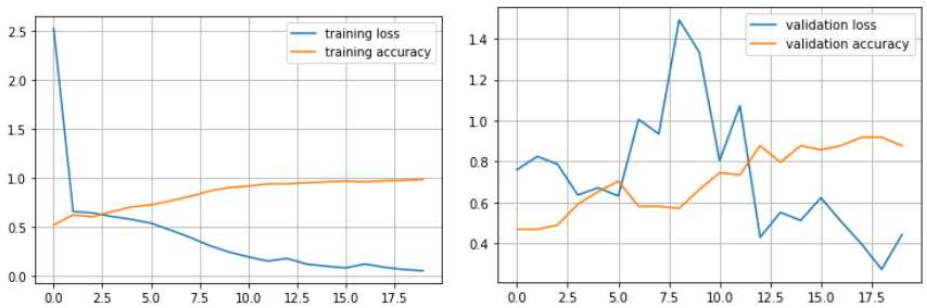


**Figura 4.17:** Grafici dei risultati

Dalle figure si nota che il grafico della perdita di training mostra una diminuzione costante, mentre la perdita di validation è instabile e non segue un trend chiaro. Questo è un segno di possibile overfitting, dove il modello si adatta troppo bene ai dati di training ma non generalizza bene sui dati di validation. Il modello, quindi, mostra un buon apprendimento sui dati di training; la sua performance sui dati di validation indica che c'è spazio per migliorare la sua capacità di generalizzazione.

### 4.4.2 Seconda Fonte: Vehicle Collision Detection based on Synthetic Data using Deep Learning

Nel secondo studio il dataset viene creato da immagini prese da un videogioco, al posto di utilizzare immagini reali.

La prima fase si occupa del rilevamento degli oggetti: rileva i veicoli da video frame e associa ad essi delle bounding box.

La seconda fase si occupa di tracciare gli oggetti: segue il movimento dei veicoli da un frame al successivo del video.

Le informazioni ottenute da queste fasi vengono, poi, utilizzate per la terza fase, dove un rilevatore di collisione deduce se il veicolo si muove normalmente o sta partecipando ad una collisione.

I modelli proposti sono i seguenti:

- *CM:5-1*: questo modello utilizza un singolo branch per elaborare una sequenza di cinque immagini consecutive, di cui l'ultima è la più recente e proviene dal frame

appena visto dal tracker. Oltre a questa immagine di track, vengono utilizzate quattro immagini precedenti per la rilevazione.

- *CM:5-2*: questo modello ha due branch differenti: uno per elaborare cinque immagini consecutive e uno con un comportamento di "skipping" delle immagini. Il branch di "skipping" ignora tutte le rilevazioni e le immagini tra la prima e l'ultima immagine, al fine di catturare i cambiamenti drastici tra l'inizio e la fine della sequenza.
- *CM:10-1*: questo modello è molto simile al CM:5-1, ma utilizza sequenze di dieci immagini consecutive per la rilevazione. Esso utilizza solo un singolo branch durante l'elaborazione delle immagini.
- *CM:10-2*: questo modello è molto simile al CM:5-2. Il primo branch elabora una sequenza di dieci immagini consecutive, mentre il secondo branch è di "skipping" delle immagini e considera solo la prima e l'ultima immagine nella sequenza di dieci immagini.
- Il modello di base utilizzato per valutare le prestazioni dei modelli di collisione è una rete VGG16 tarata finemente. La rilevazione del tipo di sequenza (normale o collisione) si basa solo su una singola immagine, mentre le altre immagini nella sequenza vengono ignorate. Il modello di base utilizza la quarta immagine nella sequenza, poiché, di solito, essa rappresenta meglio una collisione ed è osservata sia dai modelli di sequenza di 5 che di 10, garantendo uniformità tra i modelli. La struttura di VGG16 viene modificata rimuovendo i livelli fully connected e impostando le ultime convoluzioni come addestrabili.

Per il dataset inizialmente sono stati utilizzati dati sintetici per addestrare i modelli di rilevamento delle collisioni proposti (modelli CM e il modello di base). Successivamente, i modelli sono stati valutati su un set di dati reali. Il training set è stato diviso utilizzando il 70% dei dati per l'addestramento e il 30% per la validazione. Questo set di dati comprendeva una collezione di 300 video di collisioni, ognuno con circa 150 frame.

La dimensione del training set è stata aumentata con l'augmentation tramite flipping (capovolgimento dell'immagine). Nessuna augmentation è stata eseguita sul validation set.

I risultati del training e della validation vengono riportati in Figura 4.18.

I risultati iniziali indicano una connessione promettente, sebbene limitata, tra i dati sintetici e quelli reali; il modello proposto riesce a superare leggermente le prestazioni di un banale modello di base. Tuttavia, i dati di addestramento sintetici generati non sono completamente rappresentativi del loro corrispettivo del mondo reale, il che rende alcune delle collisioni molto difficili da rilevare correttamente.

### 4.4.3 Terza Fonte: Efficient Vehicle Accident Detection System using Tensorflow and Transfer Learning

La CNN richiede molto tempo, dati e potenza di calcolo per essere addestrata. Per mitigare questi problemi, è stata incorporata la tecnica di transfer learning, che comporta il retraining della rete già addestrata.

Inception-v3 è un classificatore di immagini sviluppato da Google, che è stato incorporato a questo scopo. In questo lavoro, il sistema di rilevamento degli incidenti è stato progettato utilizzando un algoritmo di transfer learning avanzato ed efficiente, che fornisce un'accuratezza del 84,5%. Il classificatore di immagini Inception-v3 di Google è addestrato sul dataset ImageNet di 100.000 immagini per classificare tra 1000 classi. Ha 22 layer nascosti e ci sono volute settimane per addestrarlo. Si effettua un retraining solo sull'ultimo layer per classificare le immagini tra "incidente" e "non incidente".

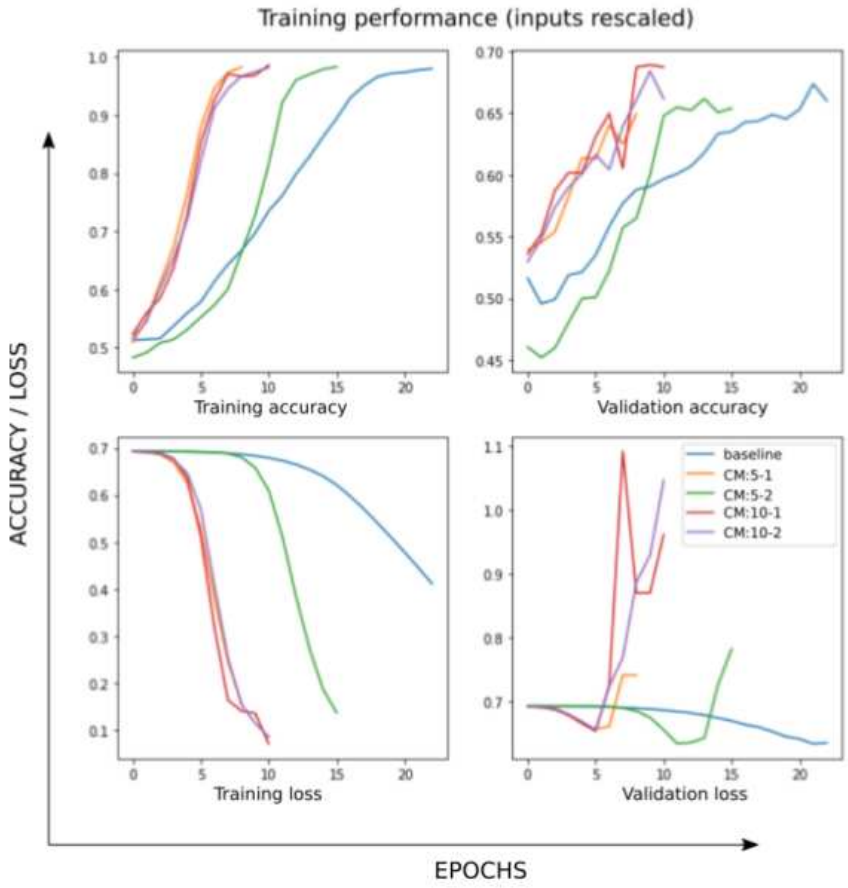


**Figura 4.18:** Risultati della seconda fonte

L'accuratezza finale del metodo di transfer learning per rilevare gli incidenti sembra soddisfacente. Tuttavia, l'affidabilità del sistema dipende dalla capacità di classificare correttamente le immagini in tempo reale durante il suo impiego su un veicolo. Questo metodo funziona con meno dati e tempo, ma solleva alcune domande, come: Quante immagini minime devono essere prese per riaddestrare il modello? Possiamo affidarci a questo metodo per usarlo nella vita reale dove possono apparire immagini di tutti i tipi con illuminazione diversa? Un'accuratezza dell'84,5% è accettabile in questo tipo di problema critico? Più immagini di addestramento sono disponibili, migliore è l'accuratezza finale. Tuttavia, anche avere solo 3 immagini dà un'accuratezza del 40%. Questo perché la rete è in grado di estrarre caratteristiche visive generali dalle immagini. La tecnica di transfer learning fornisce un'accuratezza dell'84,5%, che in questo tipo di situazione critica potrebbe non essere accettabile. Quindi, al momento attuale, la tecnica di transfer learning ha bisogno di miglioramenti prima di essere implementata praticamente.

### 4.4.4 Conclusione stato dell'arte

Si nota che i progetti analizzati nello stato dell'arte non effettuano una detection vera e propria, ossia non rintracciano in quale porzione dell'immagine si trova l'incidente, ma si limitano a controllare se è presente o meno un incidente nell'immagine. Invece, il modello sviluppato in questo progetto effettua una detection evidenziando dove si verifica l'incidente all'interno dell'immagine, questa funzione aggiuntiva può risultare utile per applicazioni

nelle quali si desidera capire quali macchine sono coinvolte nell'incidente e, di conseguenza, in quale porzione della strada avviene l'evento.

## 4.5 Conclusioni

In conclusione, i risultati ottenuti dall'esperimento possono ritenersi soddisfacenti. Il confronto tra YOLOv8 e YOLOv9 non ha portato alla luce grandi differenze in termini di prestazioni complessive, ma è stato, comunque, utile esplorare le versioni di YOLO per una comprensione più approfondita delle loro capacità e limitazioni. Entrambe le versioni hanno mostrato un comportamento simile in termini di accuratezza e perdita, evidenziando una buona capacità di rilevamento e classificazione degli oggetti.

Tuttavia, l'area di miglioramento identificata per la classe "mild" suggerisce che ulteriori azioni possano essere intraprese per ottimizzare le prestazioni del modello. La combinazione di tecniche di bilanciamento dei dati, ottimizzazione del modello e analisi degli errori può portare a un miglioramento significativo delle prestazioni e garantire una maggiore robustezza e affidabilità nelle applicazioni reali. Continuare a esplorare e confrontare diverse versioni e approcci rimane una strategia cruciale per mantenere e migliorare l'efficacia dei sistemi di rilevamento basati su YOLO.

# CAPITOLO 5

# Rilevamento del possesso di armi

*All'interno di questo capitolo si discuterà del secondo esperimento condotto. Si mostreranno i risultati ottenuti e si farà un confronto tra due approcci di addestramento.*

## 5.1 Descrizione

La scelta del compito per questa ricerca è stata guidata da una preoccupazione sociale, con implicazioni che spesso riguardano la sicurezza pubblica. Il rilevamento del possesso di armi tramite videocamere di sorveglianza è fondamentale per prevenire tragedie e garantire la protezione delle persone. Essere in grado di identificare in tempo reale la presenza di armi può consentire alle autorità di intervenire prontamente e impedire potenziali situazioni pericolose, sia in ambienti pubblici che privati. L'obiettivo principale di questa ricerca è sviluppare un modello di Intelligenza Artificiale in grado di rilevare la presenza di armi tramite segmentazione.

## 5.2 Dataset

Si sono scelti 3 dataset, in particolare:

- *Primo dataset*: dataset per realizzare una classificazione binaria.

  Esso è composto da 713 immagini divise in 88% come training set, 8% come validation set e 4% come test set. Le immagini sono state pre-elaborate con un ridimensionamento a 640x640 e sono soggette a un processo di augmentation con variazione della tonalità tra -25° e +25°, di saturazione tra -25% e +25%, di luminosità tra -25% e +25%, di esposizione tra -25% e +25%, di sfocamento fino a 2.5px, di rumore fino al 2% dei pixel. Essendo una classificazione binaria le immagini possono essere classificate in "gun" oppure "null".

- *Secondo dataset*: per realizzare una classificazione multiclasse.

  Tale dataset è formato da 991 immagini di cui l'84% compone il training set, il 10% il validation set e il 6% il test set. Le immagini sono state pre-elaborate con un ridimensionamento a 240x240 ed è stata effettuata un'augmentation tramite rotazione di esse tra -15° e +15°, scala di grigi applicata al 25% delle immagini, esposizione tra il -25% e il

+25% e rumore non superiore al 5% dei pixel. Le immagini possono essere classificate in “3D Printed Firearm”, “Danger”, “Gun”, “Null” o “Traditional Firearm”.

- *Terzo dataset*: possiede 1470 immagini ritagliate per rappresentare solo immagini di armi, il dataset viene suddiviso in 85% di training set, 9% di validation set e 6% di test set. Esse sono state pre-elaborate con un ridimensionamento a 240x240 ed è stata applicata un’augmentation tramite una rotazione di esse tra -15° e +15°, scala di grigi applicata al 25% delle immagini, esposizione tra il -25% e il +25% e rumore non superiore al 5% dei pixel. Le classi utilizzate sono le stesse del secondo dataset essendo la quinta versione dello stesso dataset.

Per l’addestramento dei modelli si è fatto sempre affidamento sui notebook messi a disposizione da Roboflow progettati per l’utilizzo tramite Google Colab.

Il modello utilizzato per l’addestramento è yolov8x-seg; esso include funzionalità avanzate di segmentazione basandosi sull’architettura di YOLOv8; ciò permette al modello non solo di rilevare dove si trovano gli oggetti, ma anche di delineare i contorni precisi di ogni oggetto all’interno dell’immagine. Il suffisso "x" indica che si tratta di una versione "extra-large" del modello, che offre una maggiore capacità di apprendimento grazie a un numero superiore di parametri e una rete neurale più profonda (71.8 milioni di parametri).

## 5.3 Classificazione binaria

Inizialmente si è addestrato il modello per effettuare una classificazione binaria tramite l’utilizzo del primo dataset. I risultati ottenuti vengono mostrati nelle Figure 5.1 e 5.2.

```
100 epochs completed in 2.341 hours.
Optimizer stripped from runs/segment/train/weights/last.pt, 144.0MB
Optimizer stripped from runs/segment/train/weights/best.pt, 144.0MB

Validating runs/segment/train/weights/best.pt...
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8x-seg summary (fused): 295 layers, 71721619 parameters, 0 gradients, 343.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 2/2 [00:02<00:00,
                   all         59         83      0.964      0.867      0.951      0.802      0.964      0.867      0.942      0.697
Speed: 0.3ms preprocess, 34.0ms inference, 0.0ms loss, 1.5ms postprocess per image
Results saved to runs/segment/train
💡 Learn more at https://docs.ultralytics.com/modes/train
```

**Figura 5.1:** Addestramento del modello per la classificazione binaria

Tramite la confusion matrix in Figura 5.2 è stato possibile calcolare l’accuracy.
$TP = 76, TN = 0, FN = 7, FP = 10$
$Accuracy = TP + TN/TotalInstances = 76/93 = 0.817 = 81,7\%$
Alcuni risultati dell’inferenza sul test set vengono mostrati in Figura 5.3.

## 5.4 Classificazione multiclasse

Si è proseguito in seguito con l’addestramento di un modello per la classificazione multiclasse, sempre tramite instance segmentation, utilizzando il secondo dataset.

I risultati sono osservabili nelle Figure 5.4 e 5.5.

Tramite la confusion matrix in Figura 5.5 è stato possibile calcolare l’accuracy.
$TP3DPrinted = 20, TPFireArm = 41, TPDanger = 29, TPGun = 20$
$Accuracy = TP/TotalInstances = 110/136 = 0.808 = 80,8\%$

## 5.5 Progressive fine-tuning

Oltre ai "classici" approcci sopra descritti, si è provato ad addestrare il modello tramite una variante del progressive fine-tuning.

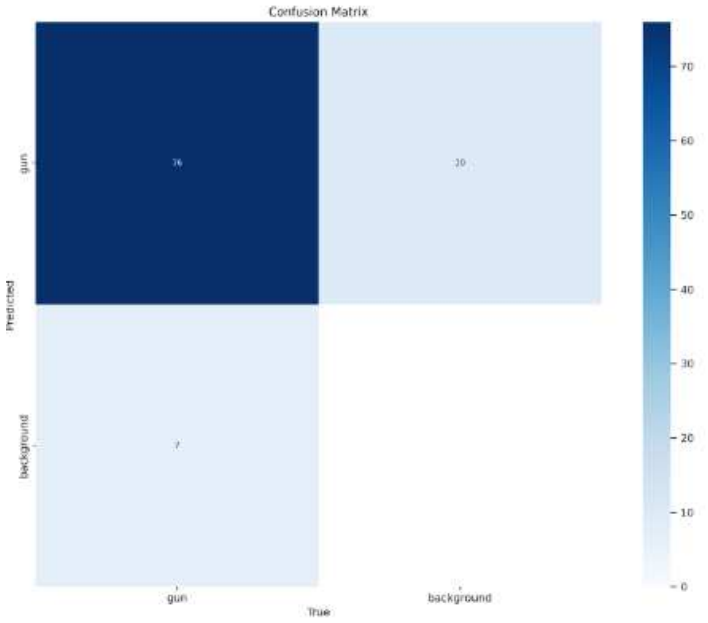


**Figura 5.2:** Confusion matrix della classificazione binaria


**Figura 5.3:** Risultato dell'inferenza del primo dataset

In primo luogo, si è condotto un addestramento su un dataset specifico contentente solo immagini di armi (terzo dataset), permettendo, quindi, di focalizzarsi esclusivamente sulle caratteristiche distintive delle armi. Questo approccio ha consentito al modello di immergersi nei dettagli più sottili, affinando la sua capacità di riconoscere le armi con precisione. Grazie a questo primo passaggio, il modello ha acquisito una comprensione approfondita delle caratteristiche visive che definiscono un'arma, senza essere distratto da altre variabili presenti nei contesti più ampi. Il risultato di tale addestramento viene visualizzato in Figura 5.6.

```
100 epochs completed in 2.724 hours.
Optimizer stripped from runs/segment/train/weights/last.pt, 144.0MB
Optimizer stripped from runs/segment/train/weights/best.pt, 144.0MB

Validating runs/segment/train/weights/best.pt...
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8x-seg summary (fused): 295 layers, 71724508 parameters, 0 gradients, 343.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 3/3 [00:05<00:00,
                   all         96        136      0.825      0.832      0.841      0.706      0.824      0.834      0.848      0.638
    3D Printed Firearm         96         25      0.726        0.8      0.719      0.655      0.762       0.84      0.765      0.615
   Traditional Firearm         96         59      0.703      0.644      0.714      0.514      0.666       0.61      0.676      0.461
                danger         96         31      0.967      0.932      0.989      0.877      0.967      0.932      0.985      0.786
                   gun         96         21      0.903      0.952      0.943      0.779      0.903      0.952      0.964      0.689
Speed: 0.4ms preprocess, 32.0ms inference, 0.0ms loss, 2.8ms postprocess per image
Results saved to runs/segment/train
💡 Learn more at https://docs.ultralytics.com/modes/train
```

**Figura 5.4:** Addestramento del modello per la classificazione multiclasse

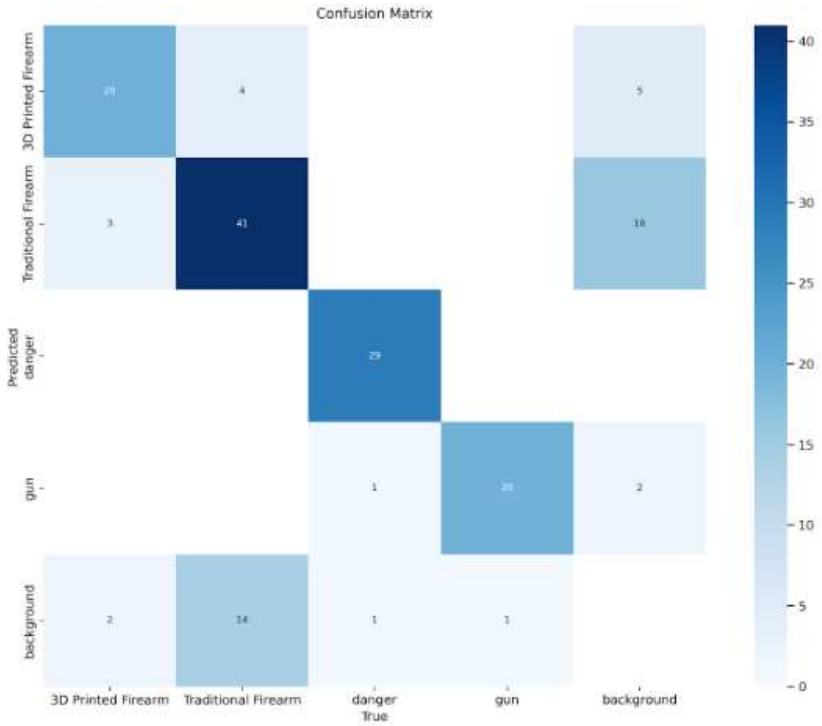


**Figura 5.5:** Confusion matrix della classificazione multiclasse

```
20 epochs completed in 0.793 hours.
Optimizer stripped from runs/segment/train/weights/last.pt, 143.9MB
Optimizer stripped from runs/segment/train/weights/best.pt, 143.9MB

Validating runs/segment/train/weights/best.pt...
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8x-seg summary (fused): 295 layers, 71722582 parameters, 0 gradients, 343.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95):
                   all        136        136      0.994       0.99      0.995      0.993      0.994       0.99      0.995      0.975
                danger        136         31          1       0.98      0.995      0.995          1       0.98      0.995      0.989
                   gun        136        105      0.988          1      0.995       0.99      0.988          1      0.995      0.962
Speed: 0.5ms preprocess, 33.8ms inference, 0.0ms loss, 5.0ms postprocess per image
Results saved to runs/segment/train
💡 Learn more at https://docs.ultralytics.com/modes/train
```

**Figura 5.6:** Addestramento con dataset contenente solo armi

Successivamente, si è realizzato un fine-tuning sul modello prodotto dalla prima fase, questa volta utilizzando un dataset più ampio e variegato, ossia il secondo dataset, in cui le immagini includevano non solo le armi, ma anche persone che le tenevano in mano. Questo passaggio ha dato la possibilità al modello di generalizzare le sue conoscenze acquisite, affrontando contesti reali in cui le armi possono essere presenti insieme ad altre variabili, come persone, posture e sfondi complessi. È possibile osservare il risultato di questo fine-tuning in Figura 5.7, mentre in Figura 5.8 si mostra la confusion matrix ottenuta.

Tramite la confusion matrix in Figura 5.8 è stato possibile calcolare l'accuracy.

```
80 epochs completed in 2.161 hours.
Optimizer stripped from runs/segment/train/weights/last.pt, 144.0MB
Optimizer stripped from runs/segment/train/weights/best.pt, 144.0MB

Validating runs/segment/train/weights/best.pt...
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8x-seg summary (fused): 295 layers, 71724508 parameters, 0 gradients, 343.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100%
                   all         96        136      0.852      0.788      0.852      0.678      0.856      0.771      0.838      0.563
     3D Printed Firearm         96         25      0.737       0.76      0.767      0.673      0.757      0.749      0.767      0.563
   Traditional Firearm         96         59      0.758       0.61      0.689      0.504      0.751      0.559      0.651      0.364
                danger         96         31          1      0.878      0.982      0.821      0.997      0.871      0.963      0.702
                   gun         96         21      0.911      0.905       0.97      0.712       0.92      0.905       0.97      0.624
Speed: 0.4ms preprocess, 31.2ms inference, 0.0ms loss, 5.8ms postprocess per image
Results saved to runs/segment/train
💡 Learn more at https://docs.ultralytics.com/modes/train
```

**Figura 5.7:** Fine-tuning sul modello

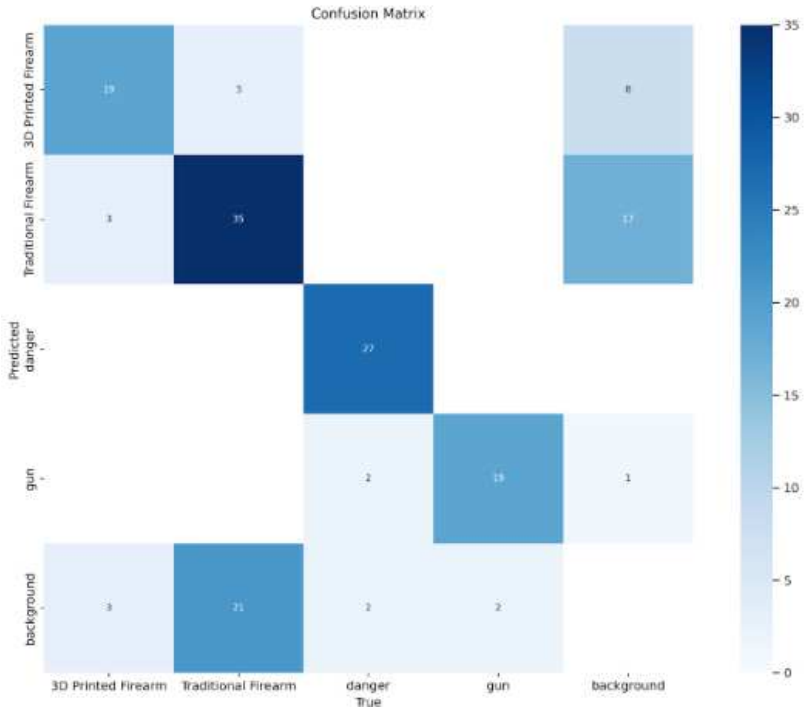


**Figura 5.8:** Confusion matrix del modello finale

$TP3DPrinted = 19, TPFireArm = 35, TPDanger = 27, TPGun = 19$
$Accuracy = TP/TotalInstances = 100/136 = 0.735 = 73,5\%$

## 5.6 Confronto tra i due approcci

Per il primo approccio "classico" di classificazione multiclasse, il modello è stato addestrato per 100 epoche. Per l'approccio basato sul fine-tuning, invece, nella prima fase sono state utilizzate 20 epoche, seguite da 80 epoche per il fine-tuning con il secondo dataset.

I risultati mostrano che il primo approccio offre una precisione complessiva migliore. In particolare, le metriche, come la MeanAveragePrecision delle bounding box (mAP50-95), sono risultate più elevate; questo trend si è ripetuto anche nelle singole classi.

Inaspettatamente il secondo approccio si è dimostrato meno performante; possono esserci vari motivi per i quali si sono riscontrati tali risultati; ad esempio, il modello originale su cui è stato eseguito il fine-tuning potrebbe non essere stato addestrato abbastanza a lungo o potrebbe non possedere una qualità sufficiente per fungere da buona base per ulteriori addestramenti.

In Figura 5.9 è possibile vedere un confronto tra due approcci: il primo viene indicato come "yolov8x-seg", e il secondo, basato sul fine-tuning, viene indicato come "pre-trained". Nella prima colonna sono riportate le metriche riguardanti le bounding box, mentre nella seconda colonna sono riportate le metriche riguardanti la maschera.

Allo stesso modo, in Figura 5.10, viene effettuato un confronto tra training e validation.

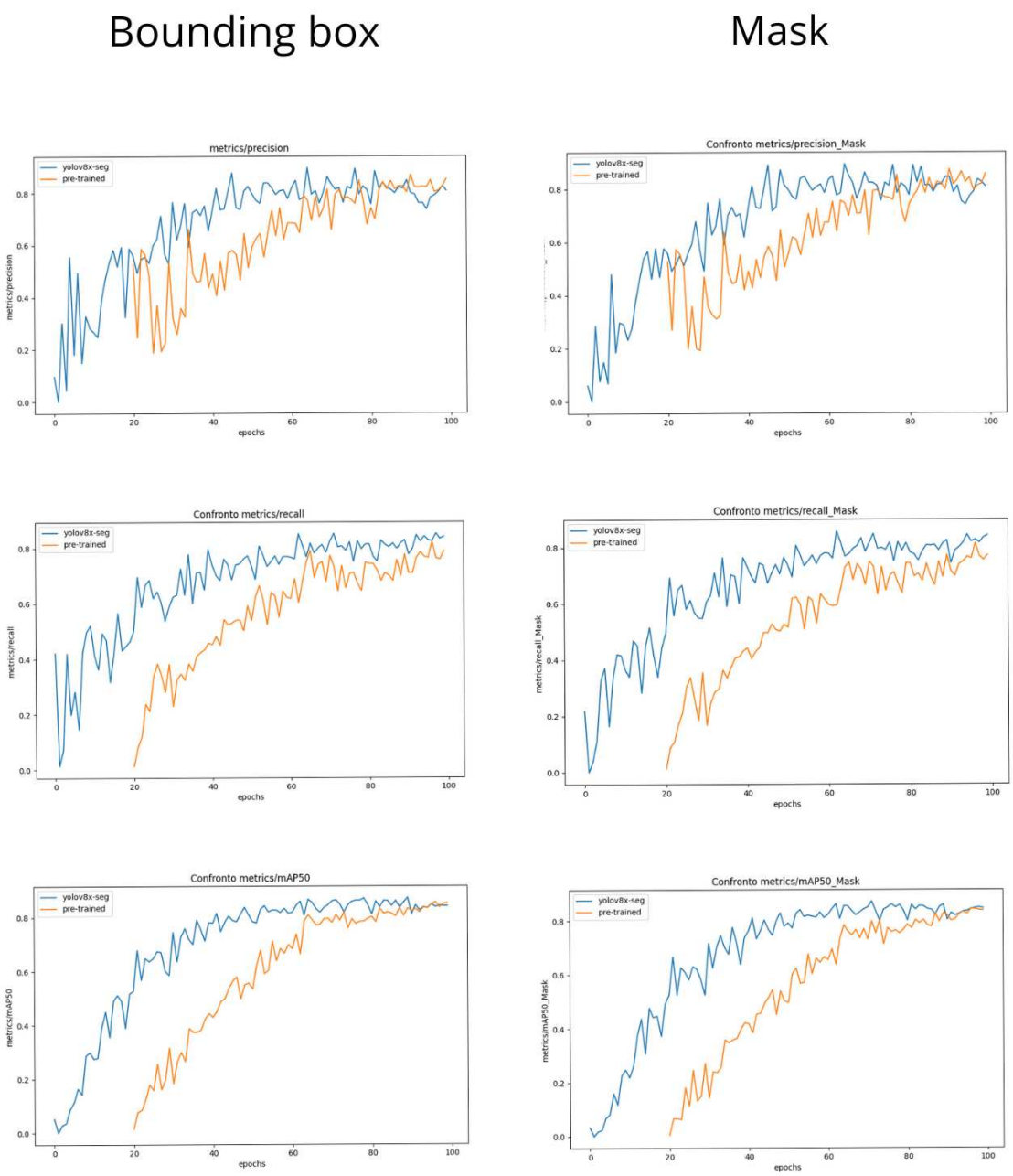


**Figura 5.9:** Confronto tra bounding box e mask

In Figura 5.11 viene mostrato un confronto tra le inferenze sul test set.

## 5.7 Stato dell'arte

È stata condotta un'analisi approfondita su vari studi per esaminare lo stato attuale della ricerca nel campo del rilevamento del possesso di armi mediante l'uso di modelli di

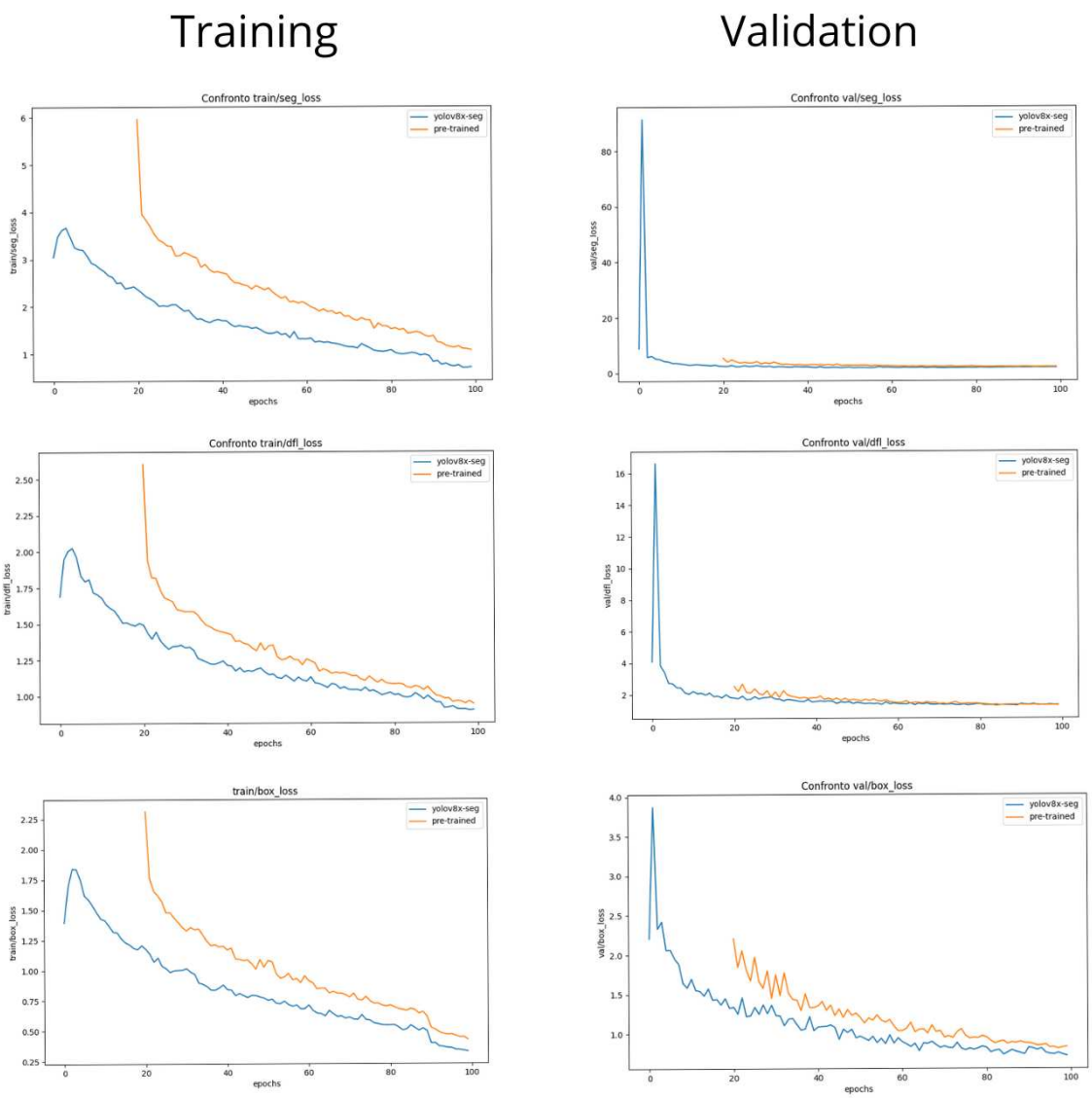


**Figura 5.10:** Confronto tra training e validation

Intelligenza Artificiale.

### 5.7.1 Prima fonte Recognizing Firearms from Images and Videos in Real-Time with Deep Learning and Computer Vision

La documentazione consultata fa parte di alcune sezioni di un report tecnico scritto nel Dicembre 2018, all'interno della quale viene mostrato come sono stati addestrati i modelli di Machine Learning che riconoscono determinate armi in immagini e video, in alcuni casi in real-time.

Si è data la massima priorità all'accuratezza, con il vincolo che le predizioni false positive devono essere inferiori al 10% e i falsi negativi non devono superare il 3%. Verranno implementati sia Darknet YOLO che Mask R-CNN per effettuare un'analisi comparativa.

L'addestramento del modello DarkNet YOLO è stato avviato utilizzando i pesi preaddestrati del dataset COCO. Nella Figura 5.12 si illustra l'output del training effettuato su un dataset di 50 immagini di mitragliatrici, per le quali le bounding box sono state create manualmente. Inoltre, nella medesima figura, viene mostrato il risultato dell'addestramento


**Figura 5.11:** Confronto tra le inferenze dei due approcci

di un modello per il riconoscimento delle pistole, effettuato utilizzando un dataset composto da 100 immagini.

Nella prima immagine della Figura 5.12, risultata dall'addestramento tramite mitragliatrici, sono state riconosciute correttamente 2 mitragliatrici, non la terza a causa delle caratteristiche dello scanning singolo di YOLO, che porta a una velocità maggiore ma ad un'accuratezza minore.

Grazie al grande volume del dataset, il modello addestrato con le pistole risulta più accurato.

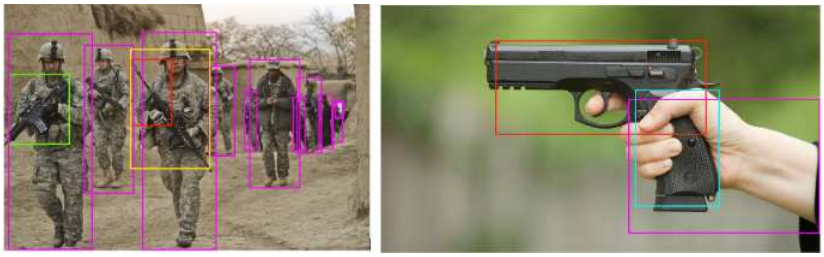

**Figura 5.12:** Inferenza con Darknet YOLO

Successivamente si è addestrato il modello Mask R-CNN utilizzando i pesi del pre-trained COCO. La Figura 5.13 mostra l'output ottenuto dall'addestramento con un dataset di 50 mitragliatrici e quello ottenuto dall'addestramento con un dataset di 100 immagini di pistole. Nella prima immagine della Figura 5.13 i contorni non coprono l'intero spazio delle due mitragliatrici a causa della complessità data dalle immagini. Se si aggiungessero più immagini al dataset i risultati sarebbero più accurati.

**Figura 5.13:** Inferenza con R-CNN

L'elaborazione delle immagini è più lenta, circa 1.5 secondi per immagine.

In conclusione, si ritiene che dovrebbe essere utilizzata una combinazione tra YOLO e Mask R-CNN. YOLO può essere utilizzato per uno scan preliminare del video mentre Mask R-CNN per la soluzione finale.

### 5.7.2 Seconda fonte: Object detection using mask-RCNN on custom Dataset

In questo progetto si utilizza Mask-RCNN. L'obiettivo è quello di addestrare il modello su un dataset composto da pistole a spade.

Il dataset è formato da 40 immagini di pistole e 40 di spade; le immagini sono state procurate da open image dataset e annotate utilizzando l'image annotator VIA. Sono state utilizzate due classi, ovvero 'gun' e 'sword'.

Il codice utilizzato proviene da `matterport/Mask_RCNN` GitHub repository.

Il risultato ottenuto a seguito del training è osservabile nella Figure 5.14 e 5.15.

The command with `weights=last` will resume training from the last epoch. The weights are going to be saved in the `logs` directory in the `Mask_RCNN` folder.

This is how the loss looks after our final epoch.

```
rcnn_class_loss: 0.0170 - mrcnn_bbox_loss: 0.0399 - mrcnn_mask_loss: 0.1560 - val_loss: 2.5081 - val_rpn_class_loss: 0.0766
 - val_rpn_bbox_loss: 1.7466 - val_mrcnn_class_loss: 0.0645 - val_mrcnn_bbox_loss: 0.3578 - val_mrcnn_mask_loss: 0.2626
```

**Figura 5.14:** Risultato dell'addestramento

Si nota che la validation loss si muove in maniera brusca. Questo era un risultato prevedibile visto che si sono utilizzate solo 20 immagini nel validation set. In Figura 5.16 viene mostrato il risultato dell'inferenza.

Il risultato non si dimostra molto promettente, ma ciò era prevedibile visto che si sono utilizzate solo 60 immagini per il training set. Per migliorare il risultato si dovrebbero utilizzare più immagini, si dovrebbe addestrare il modello per più tempo e si dovrebbero regolare gli iperparametri nel file `mrcnn/config` della cartella `Mask_RCNN`.


**Figura 5.15:** Grafici del risultato dell'addestramento

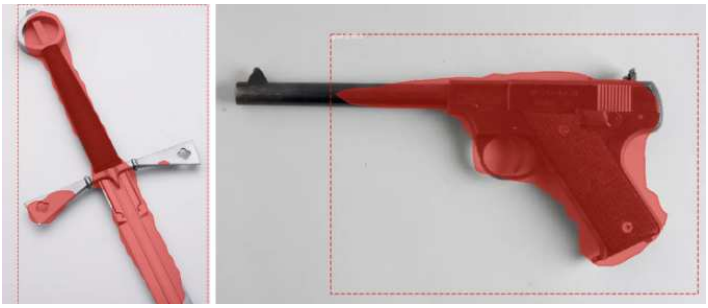

**Figura 5.16:** Inferenza relativa alla seconda fonte

### 5.7.3 Terza fonte: Scalable 3D Semantic Segmentation for Gun Detection in CT Scans

Una società di sicurezza privata ha creato un dataset con scansioni TC (tomografia computerizzata) in 3D di bagagli a mano contenenti armi da fuoco. Di seguito, viene proposto un nuovo metodo di semantic segmentation in 3D tramite HiLo-Network. L'obiettivo principale è ridurre il consumo di memoria pur mantenendo un processo di addestramento veloce.

Il dataset comprende 2925 scansioni TC etichettate di bagagli contenenti armi da fuoco. Le scansioni TC sono rappresentate come volumi di voxel, dove ogni voxel contiene informazioni sul materiale. Ogni voxel indica se è parte o meno della sagoma dell'arma; pertanto, l'attività di semantic segmentation è binaria. La rappresentazione tramite voxel può essere descritta come un volume cubico diviso in una griglia regolare. I voxel sono densi, cioè lo sfondo fa parte della griglia. Un vantaggio di questo tipo di rappresentazione è che possono essere applicate CNN con convoluzioni 3D, sostituendo l'estrazione manuale delle caratteristiche con una automatica.

Le scansioni TC hanno una risoluzione di $n \times 416 \times 616$, dove $n$ varia tra le scansioni. Per uniformare le dimensioni, $n$ è limitato o riempito fino a un valore di 640, con dimensioni finali di $640 \times 416 \times 616$, che si traducono in un consumo di memoria di circa 650 Megabyte per scansione. Dopo la pre-elaborazione, il dataset è diviso in un training set di 2600 campioni, un validation set di 128 campioni e un test set di 196 campioni.

Nella ricerca, vengono valutate e confrontate le prestazioni delle architetture proposte delle Super-resolution Occupancy Network per la semantic segmentation. Viene testato un encoder CNN ampio e uno superficiale, con quello ampio che utilizza il doppio dei filtri rispetto a quello superficiale per ogni strato. Per confrontare i risultati delle diverse architetture, viene calcolata l'Intersection over Union (IoU) sul test set. Dai punteggi bassi di IoU, mostrati nella Figura 5.17, è possibile concludere che nessuna delle architetture O-Net è in grado di eseguire contemporaneamente la super risoluzione e la segmentazione semantica in 3D.

| Cat | CBN | Wide | Parameters | IoU |
|---|---|---|---|---|
| ✓ | ✓ | ✓ | 3.4M | 0.1744 |
| ✓ | ✗ | ✓ | 2.2M | 0.1591 |
| ✗ | ✓ | ✓ | 3.3M | 0.1689 |
| ✓ | ✓ | ✗ | 2.0M | **0.1943** |
| ✓ | ✗ | ✗ | 0.7M | 0.1612 |
| ✗ | ✓ | ✗ | 1.9M | 0.1128 |

**Figura 5.17:** Caratteristiche della terza fonte

Dai risultati della segmentazione analizzati visivamente emerge un'incapacità delle O-Net nel prevedere forme dettagliate, sebbene siano in grado di individuare con precisione la posizione approssimativa di un oggetto. Questo suggerisce l'opportunità di utilizzare le O-Net per prevedere una segmentazione grossolana, piuttosto che una segmentazione dettagliata. Inoltre, durante il processo di addestramento si osserva una certa instabilità, evidenziata dalla fluttuazione dei metriche di validazione osservabili in Figura 5.18, attribuibile probabilmente alla natura casuale delle O-Net. I risultati dell'inferenza vengono mostrati in Figura 5.19.

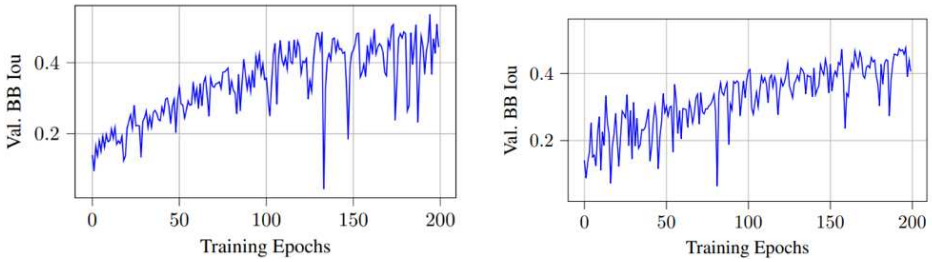


**Figura 5.18:** Grafico della terza fonte

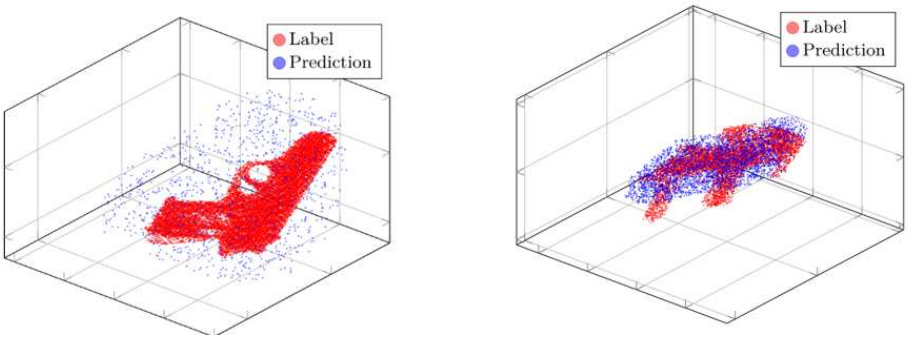


**Figura 5.19:** Inferenza della terza fonte

Successivamente viene confrontato l'approccio CNN con l'approccio O-Net. Ogni rete viene addestrata per 20 epoche. In media, l'addestramento di una HiLo-Network richiede 3 giorni su una singola GPU. Inoltre, sono state addestrate molte altre reti durante lo sviluppo delle HiLo-Network. Purtroppo, non solo il consumo di memoria dei volumi, ma anche il tempo di esecuzione dell'inferenza delle HiLo-Network crescono in modo non lineare. La visualizzazione dei risultati dell'approccio CNN, è riportata in Figura 5.20.

Complessivamente, gli IoU di test delle O-Net sono significativamente peggiori rispetto a quelli dell'approccio CNN e altamente instabili. Ciò indica un problema generale delle O-Net per la segmentazione semantica in 3D.

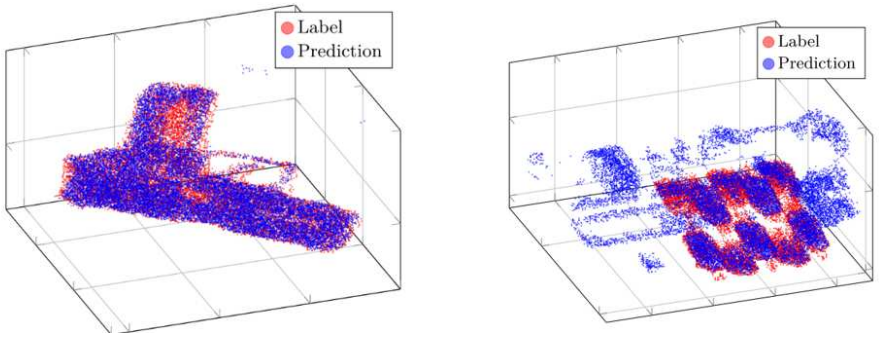


**Figura 5.20:** Inferenza della terza fonte tramite CNN

In conclusione, in questo capitolo vengono esaminate due architetture innovative per la segmentazione semantica 3D di volumi voxel. In primo luogo, si introduce una Super-resolution Occupancy Network. Purtroppo, le architetture di Occupancy Network testate non sono in grado di ottenere i risultati desiderati. In secondo luogo, viene proposta la HiLo-Network, una nuova architettura di rete neurale scalabile per eseguire la segmentazione semantica 3D. Gli esperimenti mostrano che le HiLo-Network possono segmentare in modo affidabile le armi all'interno del bagaglio. In questo senso, sono efficienti in termini di memoria e scalabili in termini di risoluzione di input.

## 5.8 Conclusioni

In conclusione, l'esperimento condotto sulla segmentazione non ha portato a risultati pienamente soddisfacenti; il fine-tuning non ha prodotto gli esiti sperati. Tuttavia, questo processo ha evidenziato significativi margini di miglioramento, aprendo la strada a ulteriori ottimizzazioni future.

Questo esperimento è stato cruciale per comprendere meglio la distinzione tra l'object detection, affrontata nel primo esperimento, e l'instance segmentation. Mentre l'object detection ha ottenuto risultati più immediati, l'instance segmentation si è rivelata una sfida più complessa, soprattutto per quanto riguarda l'ottenimento di buone metriche e la disponibilità di dataset di alta qualità.

Nonostante le difficoltà riscontrate, queste esperienze offrono un'opportunità preziosa per affinare le tecniche e migliorare i modelli. La consapevolezza delle problematiche e delle aree di miglioramento fornisce una solida base per sviluppare soluzioni più efficaci in futuro. Con ulteriori ricerche e un approccio mirato all'ottimizzazione, si possono raggiungere risultati significativamente migliori.

# CAPITOLO 6

## Rilevamento di atti di violenza

*All'interno di questo capitolo si discuterà del terzo ed ultimo esperimento condotto. In particolare, verrà esplorato l'addestramento di un modello per il rilevamento di atti di violenza.*

## 6.1 Descrizione

Questa ricerca nasce dalla necessità di affrontare una questione sociale critica, ossia la prevenzione della violenza. Rilevare rapidamente comportamenti violenti è essenziale per evitare incidenti gravi e garantire la sicurezza delle persone. La possibilità di identificare atti violenti in tempo reale permette alle autorità di intervenire prontamente, riducendo il rischio di escalation in situazioni pericolose, sia in contesti pubblici che privati. Il focus di questa ricerca è lo sviluppo di un modello di Intelligenza Artificiale, progettato per individuare episodi di violenza attraverso la tecnica di pose estimation. Per svolgere l'esperimento si è fatto riferimento al repository GitHub "Human-Pose-Estimation-Benchmarking-and-Action-Recognition" di ChengeYang.

## 6.2 Modello

Per affrontare il problema della prevenzione della violenza tramite il riconoscimento in tempo reale, ci si è ispirati a un approccio consolidato nel repository GitHub "Online-Realtime-Action-Recognition-based-on-OpenPose". Tale metodo utilizza OpenPose, una delle tecnologie più avanzate per il rilevamento delle pose umane, per monitorare e analizzare i movimenti in tempo reale. Inzialmente si è implementato il modello utilizzando Keras e TensorFlow.

Il modello è composto da tre strati nascosti, ciascuno progettato per catturare e interpretare diverse caratteristiche dei dati di input. Essi lavorano insieme per estrarre rappresentazioni significative dei movimenti umani, fondamentali per distinguere la differenza tra comportamenti normali e atti di violenza. L'ultimo strato del modello è uno strato di output Softmax, che esegue la classificazione finale in cinque classi distinte. Questa scelta consente di categorizzare accuratamente i comportamenti osservati, assegnandoli a una delle seguenti categorie: "squat", "stand", "punch", "kick" e "wave". L'architettura del modello è stata progettata con attenzione per bilanciare complessità e prestazioni, garantendo al contempo una risposta rapida e accurata.

## 6.3 Dataset

Il dataset utilizzato per questa ricerca comprende un totale di 3916 immagini estratte da video registrati a una frequenza di 10 fotogrammi al secondo (fps), ciascuna con una risoluzione di 640 x 480 pixel. In Figura 6.1 viene mostrato il numero di immagini appartenenti ad ogni classe.

| Class label | squat | stand | punch | kick | wave | Total |
|---|---|---|---|---|---|---|
| Number of training images | 711 | 907 | 583 | 784 | 931 | 3916 |

**Figura 6.1:** Divisione delle classi nel dataset

## 6.4 Training

Per ottenere lo scheletro della pose estimation si è utilizzato "tf-pose-estimation". Questo strumento utilizza reti neurali convoluzionali per rilevare e stimare le posizioni dei giunti umani, fornendo una rappresentazione precisa delle posture nelle immagini.

Successivamente, i dati estratti relativi allo scheletro sono stati preparati per l'input nella rete neurale mediante l'estrazione di tre caratteristiche principali:

- *Riferimento testa*: tutte le posizioni dei giunti vengono convertite nelle coordinate $xy$; le coordinate $(0, 0)$ fanno riferimento al giunto della testa, in modo tale da avere un metodo standardizzato di confronto tra le pose.
- *Posizioni articolari*: le 18 posizioni articolari identificate sono state tradotte in 8 angoli articolari significativi. Questi angoli includono spalla sinistra/destra, gomito sinistro/destro, anca sinistra/destra, ginocchio sinistro/destro.
- *Normalizzazione*: tutte le posizioni dei giunti sono state normalizzate rispetto alle coordinate $xy$ del riquadro di delimitazione dello scheletro. Questo processo standardizza le posizioni dei giunti rispetto alle dimensioni e alla posizione nello spazio dell'immagine, migliorando la robustezza del modello nei confronti di variazioni di scala e posizione delle persone nei video.

Questo approccio dettagliato nell'estrazione e nella preparazione dei dati consente alla rete neurale di apprendere in modo efficace.

A seguito dell'addestramento si ottengono i risultati mostrati nelle Figure 6.2 e 6.3.

Dall'analisi dei risultati dell'addestramento del modello per il riconoscimento di atti violenti tramite pose estimation si evidenziamo vari aspetti positivi. Si è raggiunta un'accuracy nel training del 99.72%, con una loss che è scesa costantemente fino a 0.0276 all'epoca 50/50, indicando una buona capacità di apprendimento e di minimizzazione dell'errore. Analogamente, l'accuracy della validation si è stabilizzata al 99.74%, con una loss ridotta a 0.0110, segnalando una solida capacità di generalizzazione. Le curve di accuracy e loss mostrano una buona convergenza, suggerendo che il modello ha raggiunto un punto di saturazione; la leggera differenza tra loss di training e di validation indica una buona generalizzazione, dimostrando l'assenza di overfitting.

In conclusione, il modello ha dimostrato un'elevata accuracy e una buona capacità di generalizzazione, rendendolo promettente per il riconoscimento di atti violenti tramite pose estimation.

```
Epoch 40/50
111/111 - 0s - loss: 0.0386 - accuracy: 0.9952 - val_loss: 0.0144 - val_accuracy: 0.9974 - 276ms/epoch - 2ms/step
Epoch 41/50
111/111 - 0s - loss: 0.0352 - accuracy: 0.9957 - val_loss: 0.0155 - val_accuracy: 0.9974 - 266ms/epoch - 2ms/step
Epoch 42/50
111/111 - 0s - loss: 0.0377 - accuracy: 0.9946 - val_loss: 0.0178 - val_accuracy: 0.9974 - 305ms/epoch - 3ms/step
Epoch 43/50
111/111 - 0s - loss: 0.0328 - accuracy: 0.9952 - val_loss: 0.0166 - val_accuracy: 0.9974 - 268ms/epoch - 2ms/step
Epoch 44/50
111/111 - 0s - loss: 0.0381 - accuracy: 0.9926 - val_loss: 0.0146 - val_accuracy: 0.9974 - 268ms/epoch - 2ms/step
Epoch 45/50
111/111 - 0s - loss: 0.0354 - accuracy: 0.9952 - val_loss: 0.0137 - val_accuracy: 0.9974 - 262ms/epoch - 2ms/step
Epoch 46/50
111/111 - 0s - loss: 0.0358 - accuracy: 0.9946 - val_loss: 0.0132 - val_accuracy: 0.9974 - 283ms/epoch - 3ms/step
Epoch 47/50
111/111 - 0s - loss: 0.0336 - accuracy: 0.9949 - val_loss: 0.0129 - val_accuracy: 0.9974 - 455ms/epoch - 4ms/step
Epoch 48/50
111/111 - 1s - loss: 0.0308 - accuracy: 0.9966 - val_loss: 0.0126 - val_accuracy: 0.9974 - 526ms/epoch - 5ms/step
Epoch 49/50
111/111 - 1s - loss: 0.0306 - accuracy: 0.9963 - val_loss: 0.0117 - val_accuracy: 0.9974 - 562ms/epoch - 5ms/step
Epoch 50/50
111/111 - 1s - loss: 0.0276 - accuracy: 0.9972 - val_loss: 0.0110 - val_accuracy: 0.9974 - 552ms/epoch - 5ms/step
```

**Figura 6.2:** Risultato dell'addestramento

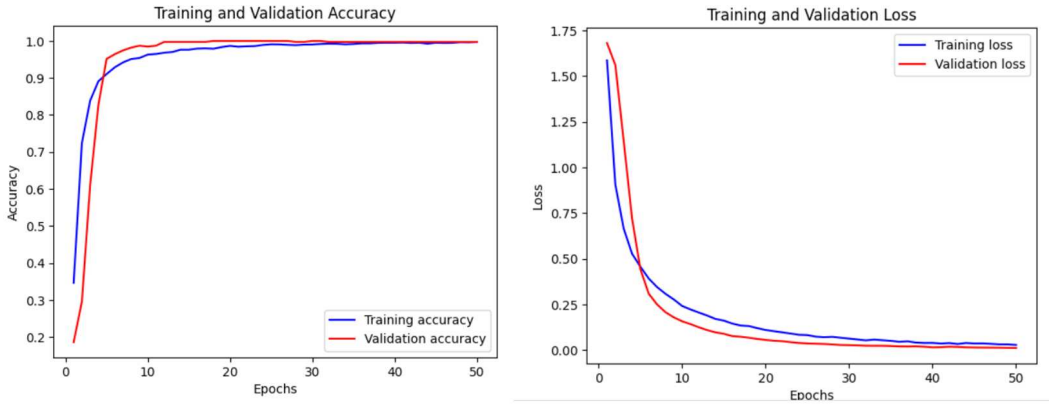


**Figura 6.3:** Grafici di training e validation

## 6.5 Stato dell'arte

Si sono analizzati vari studi sul tema per comprendere le tecnologie che vengono applicate ad oggi e le problematiche che si riscontrano.

### 6.5.1 Prima fonte: Human Action Recognition using Detectron2 and LSTM

All'interno di questo articolo viene analizzata la realizzazione di un'applicazione per il riconoscimento delle azioni umane utilizzando pose estimation e LSTM (Long Short-Term Memory). Viene utilizzato PyTorch lightning per il training e la validation del modello.

Si adopera il modello pre-trained 'R50-FPN', procurato dal model zoo per la pose estimation di Detectron2. Tale modello è già stato allenato sul COCO dataset contenente più di 200.000 immagini e 250.000 istanze di persone, etichettate con i keypoint. L'output del modello è di 17 keypoint per ogni persona, come mostrato in Figura 6.4.

Si addestra il modello sul dataset fornito nel repository GitHub "RNN-for-Human-Activity-Recognition-using-2D-Pose-Input".

Per eseguire la classificazione delle azioni su una sequenza di keypoint da un video, viene utilizzata una LSTM. I dati di input dell'addestramento contengono una sequenza di keypoint e le etichette delle azioni associate. Per identificare una particolare azione viene utilizzata una sequenza continua di 32 fotogrammi. A differenza dei 18 punti chiave di un corpo umano rilevati dal modello OpenPose nel dataset originale, l'applicazione realizzata in

| Index | Key point |
|---|---|
| 0 | Nose |
| 1 | Left-eye |
| 2 | Right-eye |
| 3 | Left-ear |
| 4 | Right-ear |
| 5 | Left-shoulder |
| 6 | Right-shoulder |
| 7 | Left-elbow |
| 8 | Right-elbow |
| 9 | Left-wrist |
| 10 | Right-wrist |
| 11 | Left-hip |
| 12 | Right-hip |
| 13 | Left-knee |
| 14 | Right-knee |
| 15 | Left-ankle |
| 16 | Right-ankle |

**Figura 6.4:** Keypoint applicati su una persona

questa ricerca ha solo 17 keypoint rilevati da Detectron2; quindi il dataset viene convertito in un formato a 17 keypoint prima di essere utilizzato per l'addestramento.

Si allena il modello per 400 epoche ottenendo un'accuracy nella validation di 0.913 e curve di loss in decrescita costante, come mostrato in Figura 6.5.

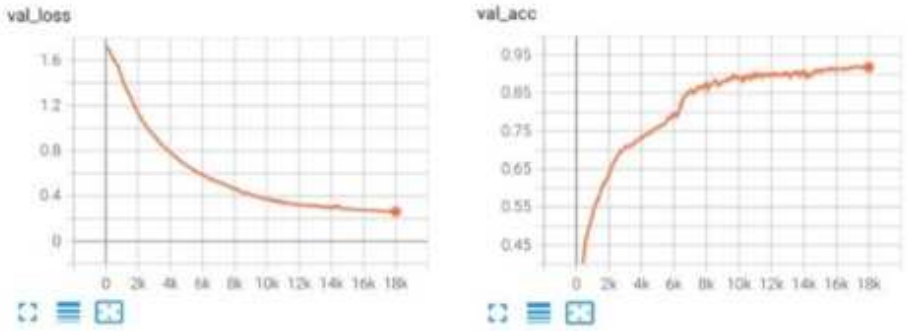


**Figura 6.5:** Grafici di loss e Accuracy

### 6.5.2 Seconda fonte: Violence Detection From Videos Captured By CCTV

Il dataset utilizzato in questo articolo è composto da 2000 video, 1000 per la violenza e 1000 per la non violenza. Esso viene diviso in 1600 video per il training e 400 video per la validation.

Per l'architettura del modello si fa riferimento al paper "Channel-wise Attention in 3D Convolutional Networks for Violence Detection". Dopo ogni blocco 3DConvNet, viene aggiunto un SELayer. Quest'ultimo è una parte interessante di questa architettura e viene utilizzato per fornire pesi a ciascun fotogramma del video. Alla fine, si aggiunge un livello FC, ovvero un livello completamente connesso. L'ultimo FC contiene una singola unità e l'attivazione sigmoidea per la classificazione binaria. L'architettura viene implementata tramite TensorFlow e Keras.

Vengono realizzati diversi modelli per poter testare vari approcci:

- *Modello 1*: vengono aggiunti 0.1 dropout dopo ogni livello di convoluzione e 0.4 dropout dopo gli strati completamente connessi. Si è mantenuto il learning rate basso a 0.0007. Si è aggiunto, inoltre, TensorBoard per ottenere i log, mostrati in Figura 6.6.

Il modello viene addestrato per 50 epoche, ottenendo un'accuracy nella validation del 76%. Nonostante ci sia una differenza nella loss tra training e validation, le accuracy del training e della validation sono simili, quindi non vi è alcun overfitting.

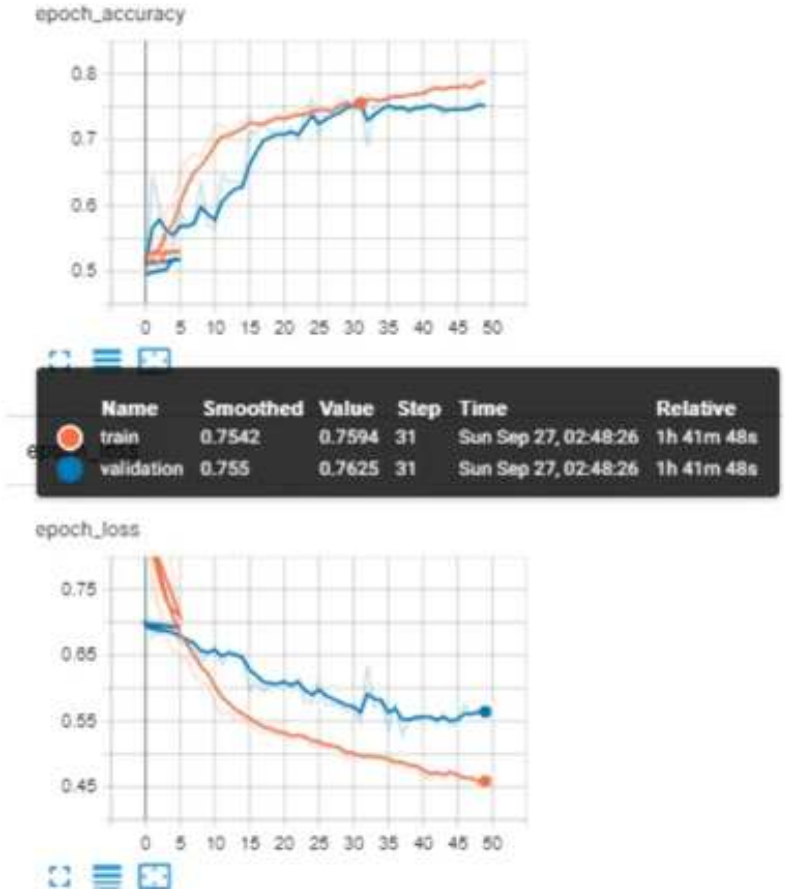


**Figura 6.6:** Grafici del primo modello

- *Modello 2*: dai log del primo modello è possibile osservare che, dopo un po' di tempo, l'accuracy del training e della validation non è aumentata in modo significativo. Ecco perché si deciso di addestrare un altro modello in cui viene rimosso il dropout dopo lo strato di convoluzione, mantenendolo, invece, dopo gli strati completamente connessi. Si è aumentato il learning rate a 0.001.

  I risultati vengono mostrati in Figura 6.7.

  Nel secondo modello l'accuracy della validation è del 75% nonostante si siano utilizzate 80 epoche. Dai log è possible notare un grande overfitting.

- *Modello 3*: nei primi due modelli si utilizzano 20 frame al secondo; con questo terzo modello si provano ad utilizzare 50 frame al secondo e un learning rate di 0.001.

  I risultati non sono cambiati rispetto agli altri modelli in quanto si verifica lo stesso overfitting.

  Per ridurre l'overfitting si agisce aumentando la dimensione del training set; l'augmentation viene realizzata dai seguenti metodi: cambio della luminosità, rotazione da sinistra a destra, rotazione dall'alto al basso, cambio del contrasto.

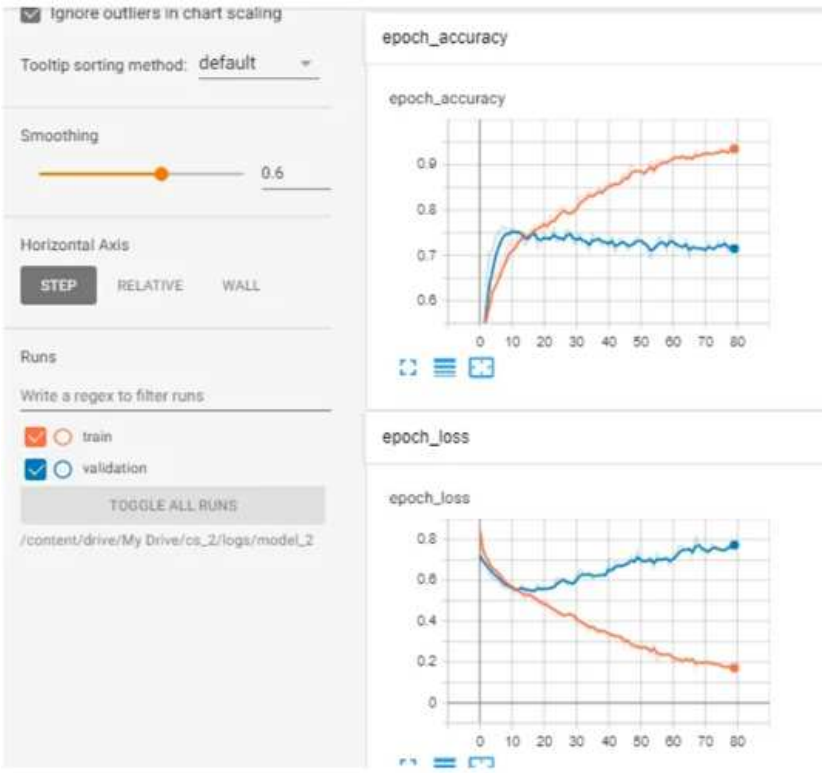


**Figura 6.7:** Grafici del secondo modello

- *Modello 4 e Modello 5*: per il modello 4 i dropout non sono stati aggiunti dopo lo strato di convoluzione, ma dopo lo strato completamente connesso e anche il learning rate viene tenuto alto, pari a 0.001. Questa configurazione porta ad un grande overfitting.

  Osservando le prestazioni del modello, si è compreso che bisogna mantenere i dropout dopo lo strato di convoluzione e mantenere basso il learning rate. Tali accorgimenti vengono applicati nel Modello 5. Finora il Modello 5 è quello con il miglior risultato; infatti, a seguito di un addestramento con 100 epoche si ottiene un'accuracy di validation pari al 79%.

- *Modello 6*: finora, con l'aggiunta di 800 video aumentati si è ottenuta la massima accuracy. C'è la possibilità di ottenere un valore più alto di tale metrica aggiungendo 1600 video aumentati, questa configurazione viene applicata nel Modello 6. I dropout e il learning rate sono gli stessi di quelli utilizzati nel Modello 5. In Figura 6.8 si può osservare il risultato.

  Come ci si aspettava, l'accuracy della validation aumenta del 2%, raggiungendo l'81%.

- *Modello 7*: in questo metodo si aggiungono alcune informazioni in più sul video, ossia informazioni sul flusso ottico.

  Le informazioni sul flusso ottico forniscono dati sul movimento dei pixel nelle direzioni orizzontale e verticale. Con l'aggiunta di due parametri del flusso ottico, cioè il movimento verticale e orizzontale, si ottiene una profondità pari a 5. Ora la forma di ciascun video è $20 \times 224 \times 224 \times 5$.

  Anche aggiungendo ulteriori informazioni sui dati non si riscontra alcun miglioramento. L'accuracy di validation con questo metodo è del 79%, inferiore a quella ottenuta con il Modello 6.

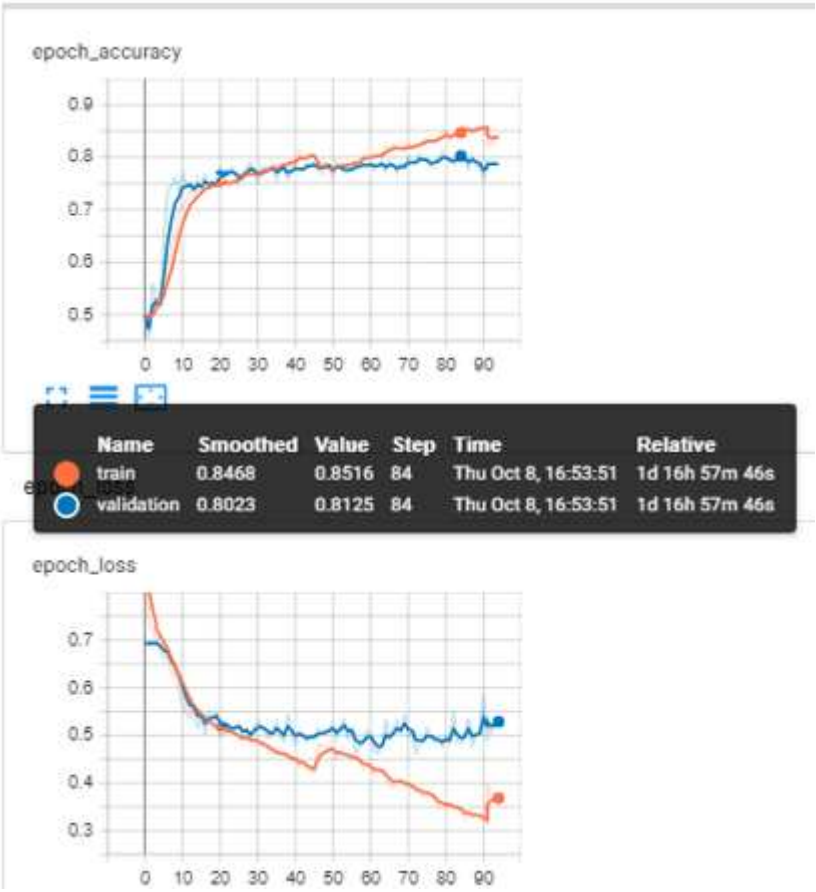


**Figura 6.8:** Grafici sesto modello

In conclusione, addestrando sette modelli, si è scoperto che il Modello 6 offre le prestazioni migliori tra tutti. In generale guardando i test non si sono ottenuti dei risultati pienamente soddisfacenti.

### 6.5.3 Terza fonte: Motion Direction Inconsistency-Based Fight Detection for Multiview Surveillance Videos

In questa ricerca, al posto di riconoscere la struttura corporea e il movimento degli arti, si utilizzano le informazioni del flusso ottico per generare dei descrittori e l'algoritmo Random Forest per identificare atti di violenza. Il rilevamento delle risse utilizza il flusso ottico per distinguere i comportamenti violenti da quelli non violenti. Le risse sono caratterizzate da movimenti improvvisi, violenti e disordinati, a differenza di azioni come parlare o camminare. La sfida principale è distinguere le risse da altri comportamenti ad alta velocità, come correre.

Il metodo proposto analizza la coerenza e l'intensità dei movimenti in video di sorveglianza da diverse angolazioni. Nei video di rissa, i corner point mostrano cambiamenti di direzione caotici e velocità irregolari, mentre nei video non violenti i movimenti sono più stabili.

Si utilizza il modello di Deep Learning Yolo-V3, mostrato in Figura 6.9; per identificare le aree di movimento esso calcola i vettori del flusso ottico ed estrae una serie di descrittori. Le caratteristiche statistiche di questi descrittori vengono concatenate per creare un vettore finale di caratteristiche. Infine, l'algoritmo Random Forest viene utilizzato per classificare i fotogrammi.

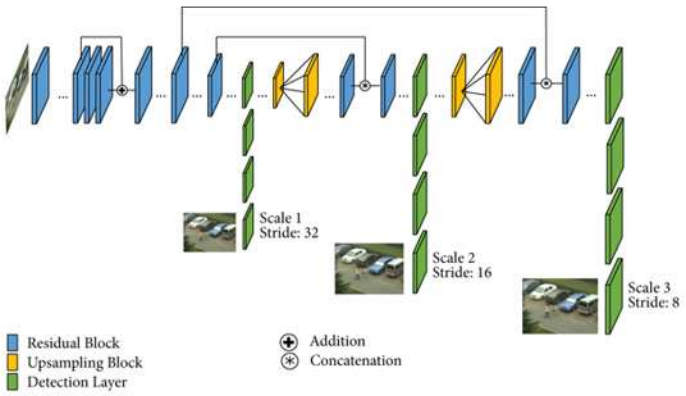


**Figura 6.9:** Architettura utilizzata nella terza fonte

Il classificatore è addestrato sul dataset delle azioni CASIA. Tale dataset contiene 432 video, inclusi 15 tipi di comportamento, catturati da 3 fotocamere fisse in vista angolare, orizzontale e dall'alto verso il basso.

Al fine di valutare l'efficacia dei descrittori esaminati, per ciascun descrittore vengono illustrate, nella Figura 6.10, le curve ROC (Receiver Operating Characteristic).

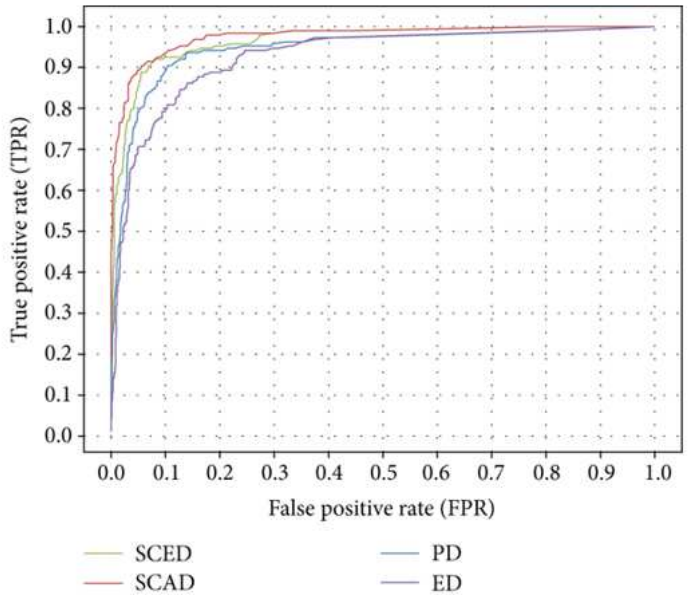


**Figura 6.10:** Grafico delle curve ROC

In particolare, si considerano le curve ROC dei seguenti indicatori:

- SCED: Statistical Characteristics of Existing Descriptors;
- SCAD: Statistical Characteristics of All Descriptors;
- PD: Proposed Descriptors;
- ED: Existing Descriptors.

Si valutano le prestazioni del classificatore Random Forest (RF) messo a confronto con altri algoritmi, in termini di: accuracy, mancate allerte (MA), falsi allarmi (FA) ed F1-score. I

parametri di valutazione sono calcolati utilizzando quattro misure: True Positive (TP), False Positive (FP), True Negative (TN) e False Negative (FN). I risultati degli esperimenti vengono mostrati in Figura 6.11.

| Classifier | Accuracy | F1-score | MA | FA |
|---|---|---|---|---|
| SVM | 92.00% | 92.36% | 11.58% | 3.67% |
| AdaBoost | 97.00% | 96.95% | 4.67% | 1.33% |
| Bagging | 92.67% | 92.14% | 1.40% | 0.60% |
| RF | 97.66% | 97.66% | 2.67% | 2.00% |

**Figura 6.11:** Risultato dell'addestramento

In sintesi, i risultati sperimentali mostrano che il metodo proposto è efficace nel riconoscere comportamenti di rissa in video ripresi da diverse angolazioni. Questo successo è dovuto principalmente a tre fattori principali. Innanzitutto, i vettori delle caratteristiche proposte catturano efficacemente le informazioni rilevanti indipendentemente dall'orientamento o dalla variazione dell'angolazione della telecamera. Secondo, il processo di smussamento dei descrittori riduce il rumore e aumenta la robustezza del metodo. Terzo, il Random Forest si dimostra un classificatore efficiente per questo tipo di riconoscimento.

Nella ricerca, si è sviluppato un metodo efficace per il rilevamento di comportamenti di rissa. Tale metodo riduce gli errori nella valutazione di comportamenti non rilevanti e aumenta la capacità di rilevamento in video girati da diverse prospettive. Invece di utilizzare i descrittori direttamente, sono state selezionate cinque caratteristiche statistiche per generare un vettore finale.

I risultati sperimentali confermano l'efficacia del metodo proposto nel riconoscere comportamenti di rissa nei video con diverse angolazioni di ripresa, ottenendo la massima precisione nel dataset CASIA.

## 6.6 Conclusioni

In conclusione, quest'ultimo esperimento si è rivelato particolarmente impegnativo, ponendo una serie di sfide significative che hanno influenzato il processo di addestramento del modello. Tra le difficoltà più rilevanti è emerso il complesso recupero delle informazioni necessarie per un corretto allenamento del modello. Questo problema ha evidenziato la necessità di applicare strategie avanzate di raccolta e di gestione dei dati, al fine di garantire la qualità e la completezza delle informazioni utilizzate nel processo di apprendimento.

Analizzando lo stato dell'arte nel campo del riconoscimento di azioni umane, tale campo si è rilevato un panorama ricco di possibilità e scelte. Le diverse tecniche e diversi approcci offrono molteplici strade per affrontare le sfide del riconoscimento di azioni umane, ognuna con caratteristiche uniche che ne determinano l'applicabilità e l'efficacia nei diversi contesti.

Nonostante le sfide incontrate, questo esperimento ha contribuito a migliorare la comprensione delle complessità coinvolte nel riconoscimento di azioni umane e nell'addestramento di modelli di Intelligenza Artificiale.

# CAPITOLO 7

# Discussione in merito al lavoro svolto

*In questo capitolo si discuteranno gli esperimenti condotti e si trarranno le conclusioni dalle esperienze effettuate, analizzando i punti di forza, di debolezza e le lezioni apprese da ciascuna di esse.*

## 7.1 Discussione sull'esperienza nel rilevamento degli incidenti stradali

Grazie a questa esperienza si è avuta l'opportunità di esplorare per la prima volta l'universo dei modelli YOLO, riuscendo così a comprendere il funzionamento di questa tipologia di modelli, i loro punti di forza e di debolezza.

Tra i vantaggi dei modelli YOLO nel campo dell'object detection si individuano:

- *Velocità elevata*: uno dei vantaggi più significativi dei modelli YOLO è la loro velocità. Essi sono in grado di processare immagini in tempo reale, il che è cruciale per applicazioni real-time, come la guida autonoma e i sistemi di sorveglianza.
- *Architettura unificata*: YOLO utilizza una singola Convolutional Neural Network (CNN) per effettuare la rilevazione degli oggetti. Questo approccio unificato permette di semplificare il processo di addestramento e inferenza.
- *Capacità di generalizzazione*: YOLO tende a generalizzare bene su nuove immagini, il che significa che può rilevare oggetti in contesti diversi da quelli visti durante l'addestramento.

Tra gli svantaggi rilevati durante l'addestramento dei modelli, si osservano:

- *Poca precisione in situazioni complesse*: sebbene YOLO sia accurato, può avere difficoltà con immagini particolarmente complesse o affollate, dove molti oggetti sono vicini tra loro o parzialmente sovrapposti.
- *Dimensioni degli oggetti*: YOLO può avere problemi nel rilevare oggetti molto piccoli o molto grandi rispetto alla dimensione dell'immagine. Questo perché il modello divide l'immagine in una griglia e ogni cella della griglia può rilevare solo un numero limitato di oggetti.

- *Compromesso velocità-accuratezza*: anche se YOLO è veloce, questo spesso implica un compromesso in termini di accuratezza rispetto a modelli più lenti e complessi. In alcune applicazioni, la massima accuratezza è preferita alla velocità.

Grazie a tale esperienza, è stato, inoltre, possibile acquisire le conoscenze per valutare con spirito critico i migliori dataset per l'addestramento di un modello, in base allo specifico compito da perseguire.

## 7.2 Analisi dell'esperienza nel rilevamento del possesso di armi

A seguito della seconda esperienza focalizzata sul compito di rilevamento del possesso di armi, si è esplorata un'altra tecnica di Computer Vision, ossia l'instance segmentation. Essa consente di superare la limitazione principale dell'object detection, cioè l'impossibilità di rilevare i dettagli riguardo alla struttura interna degli oggetti rilevati. L'instance segmentation offre una soluzione avanzata, consentendo non solo di rilevare gli oggetti ma anche di delineare i loro contorni esatti, migliorando significativamente la comprensione del contesto visivo. Questo approccio, tuttavia, richiede risorse computazionali più elevate e un processo di implementazione più complesso.

Le conoscenze acquisite finora sono state ampliate grazie allo studio del comportamento dei modelli YOLO durante le attività di segmentazione; in particolare, è stato studiato in dettaglio il modello YOLOV8-seg. Tale versione mantiene i punti di forza dei modelli YOLO precedenti, come la velocità e l'efficienza, aggiungendo funzionalità avanzate per la segmentazione a livello di pixel. YOLOv8-Seg, rispetto a YOLOv8 per l'object detection, fornisce, oltre alle bounding box e alle etichette di classe, delle maschere di segmentazione a livello di pixel per ogni oggetto. Ciò permette una comprensione molto più dettagliata della scena, utile in applicazioni dove è necessario conoscere i contorni esatti degli oggetti. Mentre entrambe le versioni di YOLOv8 condividono la base dell'architettura CNN, YOLOv8-Seg include strati aggiuntivi per la predizione delle maschere di segmentazione. Questi strati sono progettati per elaborare e affinare le caratteristiche necessarie per delineare con precisione i bordi degli oggetti.

Oltre all'utilizzo di YOLO per la segmentazione si è esplorato un tipo di approccio sull'addestramento conosciuto come progressive fine-tuning. Tale procedura viene realizzata tramite un addestramento su un dataset specifico, permettendo, quindi, di focalizzarsi esclusivamente sulle caratteristiche distintive dell'oggetto da identificare; successivamente si addestra il modello in output su un dataset più generico per adattarlo al suo vero task.

La lezione imparata da questa esperienza porta alla conclusione che bisognerebbe dapprima addestrare il modello in modo generale per poter costruire i livelli di base, per poi specializzarlo con un addestramento sul dataset specifico.

## 7.3 Rilevamento di atti di violenza: analisi e riflessioni sul lavoro eseguito

L'elemento chiave dell'ultimo task è la pose estimation, la quale ha permesso di ampliare il nostro bagaglio di conoscenze e di analizzare e determinare con precisione la posa e l'orientamento delle parti del corpo umano.

Una delle principali difficoltà rilevate è stata quella di ottenere un dataset di qualità sufficiente per addestrare il modello in modo efficace. Inoltre, la scelta dell'architettura del modello e degli algoritmi appropriati è cruciale per ottenere buone prestazioni.

Una seconda difficoltà riscontrata è stata l'incapacità di riuscire a riconoscere, ad esempio, una corsa o un abbraccio da un atto violento. Per superare questi ostacoli è necessario

mettere in campo più tecnologie insieme, integrando, ad esempio, la pose estimation con un rilevamento del flusso ottico.

Per quanto riguarda l'esperimento condotto, invece di adottare YOLO, è stata seguita un'altra metodologia che ha utilizzato del codice in Python per implementare un modello supportato da OpenPose. Quest'ultimo è noto per la sua capacità avanzata nel rilevare con alta precisione le pose umane attraverso un complesso sistema di reti neurali convoluzionali (CNN). Inoltre, esso è in grado di identificare le pose di più individui in una singola immagine, il che lo rende particolarmente idoneo per applicazioni in contesti affollati.

Nonostante le difficoltà iniziali nell'installazione e nella configurazione, l'integrazione di OpenPose in un progetto ha condotto a risultati notevoli, grazie alla sua robustezza e flessibilità. Con il continuo miglioramento delle tecnologie di Deep Learning e l'accesso a dataset sempre più vasti, le potenzialità di strumenti come OpenPose sono destinate a crescere ulteriormente.

Questo esperimento si è dimostrato il più complesso tra tutti quelli effettuati, poiché non sono state sufficienti le conoscenze di base, essendo un tema che va molto più a fondo nell'ampia materia dell'Intelligenza Artificiale. Pertanto, è stato necessario ingegnarsi per trovare un approccio innovativo ed individuare una soluzione valida ed efficace.

# Conclusioni

Nel corso della presente tesi, abbiamo esplorato approfonditamente le tecnologie e le metodologie nel vasto campo della Computer Vision. Il focus principale è stato posto sulla progettazione e sull'implementazione di modelli al fine di esplorare le limitazioni del Deep Learning e affrontare tematiche della Computer Vision ancora poco note.

Siamo partiti fornendo un'introduzione sull'Intelligenza Artificiale odierna, delineando le definizioni fondamentali e sollevando diverse discussioni critiche che evidenziano le questioni e i problemi ancora irrisolti nel campo. Successivamente, ci siamo immersi nei dettagli analizzando la materia in maniera approfondita, concentrandoci, in particolare, sul Deep Learning e sulle Convolutional Neural Network. Di quest'ultime, sono stati studiati le strutture, il funzionamento, i vantaggi e gli svantaggi.

In seguito, abbiamo descritto dettagliatamente il processo di sviluppo per ciascuno dei tre esperimenti condotti. Abbiamo analizzato i risultati ottenuti traendo le relative conclusioni e valutato le conoscenze attuali della ricerca sui temi trattati.

In definitiva, riteniamo di aver ottenuto buoni risultati e di aver acquisito le conoscenze di base necessarie per avviare un approfondimento del tema e immergerci nei dettagli dell'Intelligenza Artificiale.

Questo approccio ci ha permesso non solo di esplorare le potenzialità delle tecnologie attuali, ma anche di contribuire a colmare alcune lacune nel campo della Computer Vision, aprendo nuove prospettive per la ricerca futura.

Guardando al futuro, uno dei progetti più ambiziosi potrebbe essere l'implementazione di un sistema di Intelligenza Artificiale per riconoscere eventi critici in tempo reale e segnalarli immediatamente alle autorità competenti, come polizia e soccorso. Utilizzando modelli avanzati di object detection, instance segmentation e pose estimation, l'obiettivo potrebbe essere quello di identificare incidenti, emergenze mediche o attività sospette e fornire segnalazioni automatiche precise. Si immagini un sistema che monitora costantemente strade, edifici e altri ambienti critici; quando rileva un incidente o un'emergenza notifica le autorità con la localizzazione precisa e la descrizione dell'evento.

Attraverso queste esperienze, non sono solo state affinate competenze tecniche, ma è stata anche sviluppata una profonda consapevolezza delle sfide e delle opportunità nel campo della visione artificiale. Questa ricerca non rappresenta solo un percorso accademico, ma una via per comprendere come tali tecnologie possano trasformare settori cruciali, quali la sanità, la sicurezza e l'automazione industriale.

# Bibliografia


ACCIDENT AND NONACCIDENT (2023), «Accident and Non-accident label Image Dataset», Roboflow Universe, open Source Dataset.

AGARWAL, R. (2019), «Creating a Weapon Detector in 5 Simple Steps: Object Detection using Mask-RCNN on Custom Dataset», Blog post.

ANN (2024), «Accident Dataset», Roboflow Universe, open Source Dataset.

BORISAGAR, P., AGRAWAL, Y. e PAREKH, R. (2018), «Efficient Vehicle Accident Detection System using Tensorflow and Transfer Learning», in «2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)», p. 1–6.

CHANG, H.-S., WANG, C.-Y., WANG, R. R., CHOU, G. e LIAO, H.-Y. M. (2023), «YOLOR-Based Multi-Task Learning», *arXiv preprint arXiv:2309.16921*.

CREVIER, D. (1993), *Ai: The Tumultuous History of the Search for Artificial Intelligence*, BasicBooks.

DABLE, R. (2020), «Violence Detection From Videos Captured By CCTV», Medium.

DEPARTMENT OF COMPUTING, TIMO JOKELA (2021), «Vehicle Collision Detection based on Synthetic Data using Deep Learning», *University of Turku*.

EIFFERT, S. (2021), «RNN for Human Activity Recognition using 2D Pose Input», GitHub repository.

FIRSTTEST (2024), «Accident Detection Model Dataset», Roboflow Universe.

HE, K., GKIOXARI, G., DOLLAR, P. e GIRSHICK, R. (2018), «Mask R-CNN», URL `https://arxiv.org/abs/1703.06870`.

JIANG, B., XU, F., TU, W. e YANG, C. (2019), «Channel-wise Attention in 3D Convolutional Networks for Violence Detection», in «2019 International Conference on Intelligent Computing and its Emerging Applications (ICEA)», p. 59–64.

JOCHER, G., CHAURASIA, A. e QIU, J. (2023), «Ultralytics YOLOv8», URL `https://github.com/ultralytics/ultralytics`.

JOHN, M. (1955), *A proposal for the Dartmouth summer research project on artificial intelligence*.

KRISH RUSTAGI (2023), «Accident-Detection-System», GitHub repository.

LZQTHEPLANE (2019), «Online-Realtime-Action-Recognition-based-on-OpenPose», GitHub repository.

MCCORDUCK, P. (1979), *Machines Who Think*, Routledge.

MEMMEL, M., REICH, C., WAGNER, N. e SAEEDAN, F. (2021), «Scalable 3D Semantic Segmentation for Gun Detection in CT Scans», *arXiv preprint arXiv:2112.03917*.

PROAÑO, M. A. P. (2022), «Guns Segmentation Dataset», Roboflow Universe.

PYTHON STUFF (2023), «g Dataset», Roboflow Universe.

RUSSELL STUART, N. P. (2010), *Artificial Intelligence: Pearson New International Edition:A Modern Approach*, Pearson Education.

SEBASTIAN, B. (2021), «Human Action Recognition using Detectron2 and LSTM», LearnOpenCV.

TURING, A. M. (1950), *Computing machinery and intelligence*, Alphascript Publishing.

WANG, C.-Y. e LIAO, H.-Y. M. (2024), «YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information», *arXiv preprint arXiv:2402.13616*.

WANG, T. (2019), «Recognizing Firearms from Images and Videos in Real-Time with Deep Learning and Computer Vision», Medium.

YANG, C., YU, Z. e CHEN, F. (2019), «Human Pose Estimation Benchmarking and Action Recognition», Deep Learning Project, Winter 2019, Northwestern University.

YAO, C., SU, X., WANG, X., KANG, X., ZHANG, J. e REN, J. (2021), «Motion Direction Inconsistency-Based Fight Detection for Multiview Surveillance Videos», URL `https://doi.org/10.1155/2021/9965781`.

# Sitografia


- Agenzia ANSA – www.ansa.it
- Amazon Web Service – www.aws.amazon.com
- Blog degli Osservatori Digital Innovation – blog.osservatori.net
- CreditNews – www.creditnews.it
- IBM – www.ibm.com
- Linkedin – it.linkedin.com
- Mathworks – www.mathworks.com
- Medium – www.medium.com
- PFM Research – www.pmf-research.eu
- Parlamento Europeo – www.europarl.europa.eu
- Pulp Learning – www.pulplearning.altervista.org
- Skilla – www.skilla.com
- TD SYNNEX – blog.tdsynnex.it
- Treccani – www.treccani.it
- UAltalex - Quotidiano di informazione giuridica – www.altalex.com
- Viso.ai – www.viso.ai
- Wikipedia – www.wikipedia.org
- 01net – www.01net.it

# Ringraziamenti

Il primo ringraziamento va a mia madre Romina e mio padre Tiberio, che con i loro sacrifici mi hanno dato l'opportunità di poter intraprendere questo percorso; sono sempre rimasti al mio fianco in ogni situazione, mi hanno sempre dato il massimo supporto e ci sono sempre stati per darmi conforto e festeggiare ogni piccolo successo; vi voglio bene.

Ci tengo a ringraziare il Professor Domenico Ursino per la sua professionalità e disponibilità dimostrate durante tutto il processo di stesura della tesi. Ringrazio, inoltre, il Dott. Davide Traini, il quale mi ha fornito un supporto fondamentale trasferendomi le sue conoscenze per la realizzazione della parte tecnica della tesi. Grazie per il vostro prezioso contributo.

Vorrei ringraziare anche tutte le zie, gli zii, i cugini e le cugine per tutta la vicinanza e l'affetto incondizionato che mi hanno dimostrato e per avermi sempre spronato ad andare avanti. La vostra presenza nella mia vita è stata una fonte inestimabile di forza e ispirazione. Grazie di cuore a tutti voi. Dedico un pensiero in particolare a nonno Gabriele, nonna Lucia, nonno Graziano, nonna Carmela e lo zio compare Carlo, che, nonostante non siano più tra noi, porto con me nel cuore e spero stiano festeggiando con me da lassù.

Come potrei poi non ringraziare i miei amici in trasferta Gabriele, Daniele, Matteo Cillo, Filippo, Matteo Giuliani e Davide, senza i quali, vi assicuro, che starei ancora provando a passare gli esami di Fisica e Analisi; siete stati una parte fondamentale sia per lo studio ma anche per lo svago; avete portato un pò di Abruzzo facendomi sentire a casa nonostante i chilometri di distanza.

Un ringraziamento particolare va al mio amico Samuele, con il quale ho condiviso gli ultimi passi di questo percorso; il tuo aiuto è stato fondamentale per raggiungere questo traguardo.

Ringrazio Alessia per tutto l'aiuto che mi ha fornito nella correzione della tesi e per avermi supportato in questi mesi; ringrazio anche i miei coinquilini Alessandro, Marco e Mauro e tutti gli amici del palazzo (vorrei ringraziarvi uno a uno ma siete tantissimi!), i quali mi hanno permesso di vivere serenamente questi anni di permanenza ad Ancona con tutte le grigliate, i momenti di festa, ma anche di studio, che porterò in ricordo per sempre.

Ci tengo a ringraziare i miei amici di San Pellegrino: Danilo, Lorenzo e Pierluca, e tutti gli amici di Penne e dintorni. Siamo un gruppo meraviglioso, siete come una seconda famiglia per me. Grazie per il vostro affetto e il vostro sostegno.

Per ultimi, ma non per importanza, ringrazio i miei amici Marco, Davide, Vincenzo e Icaro, che ogni volta che tornavo da Ancona mi accoglievano con entusiasmo, pronti a farmi compagnia e a divertirci insieme. La vostra presenza e il vostro affetto hanno reso ogni ritorno speciale e indimenticabile

Ringrazio tutte le persone che non ho avuto modo di nominare e chiunque mi abbia dedicato anche un solo pensiero o mi abbia fornito una parola di sostegno. Il vostro supporto è stato prezioso e significativo per me. Grazie di cuore a tutti.

Infine, ringrazio me stesso per non aver mai mollato e per esser stato forte in ogni situazione che ho affrontato a testa alta.