ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN
INGEGNERIA INFORMATICA

RILEVAMENTO DI INTRUSIONI IN TEMPO REALE MEDIANTE VIDEO DI SORVEGLIANZA A BASSA RISOLUZIONE E RICONOSCIMENTO FACCIALE

Tesi di laurea magistrale in

Computer Vision and Image Processing

Relatore

Prof. Luigi Di Stefano

Presentata da

Rabin Sikder

Sessione Marzo 2024

Anno Accademico 2022/2023

Ringraziamenti

Grazie a te mamma, che da quando ne ho memoria mi hai sempre indicato la via più luminosa.
Grazie a te Luuuuuuu, che in questo percorso sei stata la mia fonte di luce. Sei e sarai sempre nel mio cuore.
Grazie a voi, per essere gli amici del passato, del presente e spero del futuro. Vi devo tanto.
Grazie a voi dell'UniOne, che mi avete permesso di restare sano di mente anche nei momenti di studio più intensi.

Contents

Introduzione

L'arte di insegnare alle macchine a vedere e comprendere il nostro mondo è stata una delle sfide più affascinanti e complesse del ventunesimo secolo. Questa tesi esplora il core di questa sfida: l'intersezione tra il rilevamento di oggetti, il riconoscimento facciale e l'elaborazione di video di sorveglianza a bassa risoluzione per il rilevamento automatico di intrusioni.

Attraverso l'analisi di vasti insiemi di dati e il ricorso a metodologie avanzate di apprendimento automatico e Deep Learning, questa ricerca spalanca nuovi orizzonti nell'automazione della sicurezza. Il documento inizia presentando i principi fondamentali del Machine Learning e le architetture delle reti neurali, per poi procedere con un'analisi dettagliata degli algoritmi classici e all'avanguardia per la localizzazione e l'identificazione delle persone.

Mentre il riconoscimento facciale si configura come uno strumento di importanza crescente in una varietà di applicazioni di sicurezza, questo lavoro approfondisce gli algoritmi specifici utilizzati per l'analisi dei video di sorveglianza. Con un'attenzione particolare all'implementazione pratica, viene analizzata l'efficacia del sistema di rilevamento degli intrusi, integrando teoria e pratica per offrire una visione completa e approfondita dell'attuale stato dell'arte e delle sue potenziali evoluzioni.

I capitoli successivi delineano il percorso che va dalla teoria alla pratica, esaminando non solo le tecniche di rilevamento e riconoscimento ma anche l'implementazione del sistema di rilevamento di intrusioni proposto e le sue prestazione in termini di precisione e velocità.

Infine, il testo si conclude con una discussione approfondita sulla conformità alle leggi vigenti, in particolare all'AI Act, e sulle implicazioni per la privacy e la sicurezza nell'uso di tali tecnologie in ambienti pubblici.

Questa tesi è stata sviluppata presso l'azienda Data Reply S.r.l durante il mio periodo di tirocinio curriculare. Data Reply è la società del gruppo Reply che offre servizi di eccellenza per trarre valore dai dati attraverso soluzioni di Advanced Analytics e AI. Costruiscono piattaforme Big Data e implementano modelli di ML e AI, in modo sicuro, efficiente, replicabile e scalabile, attraverso le competenze in Big Data Engineering, Data Science e IPA.

Chapter 1

Machine Learning

Il machine learning (ML) è una branca dell'intelligenza artificiale (AI) che si occupa di sviluppare algoritmi in grado di apprendere dai dati senza che sia necessario fornire loro istruzioni dettagliate su come farlo. Nel mondo contemporaneo, il Machine Learning è onnipresente, spesso in modo invisibile:

- **Riconoscimento facciale**: per identificare persone in immagini o video.
- **Assistenza clienti**: per rispondere alle domande dei clienti in modo automatico.
- **Diagnostica medica**: per aiutare i medici a diagnosticare malattie.
- **Guida autonoma**: per guidare le automobili senza intervento umano.
- **Filtri di bellezza**: utilizzato nei social per modificare il proprio aspetto estetico.

Grazie all'avvento di hardware più potente, big data e algoritmi più efficienti, le applicazioni del Machine Learning sono sempre più utilizzate in ambito ingegneristico e scientifico.

1.1 Dati

Il ML dipende fortemente dalla qualità e dalla quantità dei dati a disposizione. I dati rappresentano il cuore pulsante di qualsiasi algoritmo di apprendimento automatico e ne influenzano direttamente le prestazioni. Il termine "DataSet" si riferisce all'insieme di dati utilizzati per addestrare, validare e testare un modello.

1.1.1 Training set e test set

Per valutare efficacemente le prestazioni di un modello, è usuale dividere il DataSet in due sottoinsiemi: l'insieme di addestramento e quello di test. Generalmente, l'insieme di addestramento rappresenta circa il 70% del DataSet completo, mentre il restante 30% costituisce l'insieme di test.

1.1.2 Data augmentation

La Data Augmentation, nel contesto dell'elaborazione delle immagini, è una tecnica di regolarizzazione che permette di aumentare artificialmente la dimensione del dataset di immagini applicando trasformazioni quali rotazioni, ritagli e variazioni cromatiche. Questo processo migliora

notevolmente la capacità del modello di Machine Learning di generalizzare su nuovi dati, aiutando a prevenire l'overfitting e migliorando la robustezza del modello.

1.1.3 Qualità dei dati

La qualità dei dati può influenzare l'efficacia degli algoritmi di apprendimento. Dati rumorosi, valori anomali (outliers), dati mancanti e dati duplicati possono alterare seriamente le prestazioni del modello. Per questo motivo il pre-trattamento dei dati è un passaggio cruciale nel flusso di lavoro del ML.

1.2 Apprendimento automatico

Il pilastro del Machine Learning è l'apprendimento automatico che può avvenire attraverso l'uso di dati di esempio, esperienze passate e algoritmi matematici che cercano di replicare il processo di apprendimento umano.

1.2.1 Tecniche di apprendimento

Abbiamo diverse tecniche che possono essere classificate grossolanamente in tre grandi classi:

- **Tecniche simboliche**: In queste tecniche, la conoscenza estratta dai dati è rappresentata in una forma simbolica, facilmente interpretabile da un esperto umano. L'obiettivo è non solo prevedere o classificare, ma anche comprendere i pattern e i fenomeni sottostanti ai dati. Algoritmi come gli alberi di decisione e le regole associative sono esempi comuni di tecniche simboliche.
- **Tecniche statistiche**: Questo approccio si avvale dei metodi statistici per analizzare e interpretare i dati da una mole di dati molto ampia. L'obiettivo è spesso estrarre misure numeriche che possano riassumere le caratteristiche salienti del dataset. La regressione lineare e le tecniche di inferenza statistica ne sono degli esempi.
- **Tecniche sub-simboliche**: Queste tecniche operano a un livello più basso di astrazione rispetto alle tecniche simboliche. Invece di cercare di formulare regole o simboli chiari, questi metodi lavorano scomponendo il problema in sotto-problemi più piccoli e risolvendoli in modo iterativo. Le reti neurali e gli algoritmi genetici sono esempi tipici di tecniche sub-simboliche. Questi metodi sono spesso utilizzati quando la complessità del problema rende difficile o impraticabile una soluzione simbolica o statistica.

1.2.2 Processo di addestramento

Il processo di apprendimento si basa sull'idea che ogni aspetto del mondo possa essere descritto da una funzione matematica. Inizialmente, questa funzione è ignota, ma viene progressivamente identificata e perfezionata attraverso l'analisi dei dati forniti. In sintesi, il ML si occupa della creazione di algoritmi che identificano e assimilano le relazioni presenti nei dati.
Ogni esempio analizzato rappresenta una singola istanza, caratterizzata da diverse proprietà o "**feature**". Da queste, l'algoritmo ricava informazioni essenziali per la sua funzione. Durante la fase di addestramento, l'obiettivo primario è stabilire una correlazione tra le feature e gli output, definendo una funzione che avvicini il più possibile la funzione desiderata, cioè quella che connette correttamente gli esempi ai rispettivi output.
La **Funzione di Costo** è essenziale per determinare l'efficacia dell'algoritmo nel replicare la

funzione target. Può essere interpretata come la discrepanza tra le predizioni dell'algoritmo e i risultati attesi. Così, il compito dell'addestramento diventa un'operazione di ottimizzazione, mirata a minimizzare questa Funzione di Costo, localizzando il suo punto di minimo assoluto.
Tra i metodi utilizzati per questa ottimizzazione, la **Discesa del Gradiente** è tra i più comuni. L'idea principale è quella di iniziare con una scelta casuale dei parametri del modello e successivamente apportare correzioni iterativamente in direzione opposta al gradiente della funzione di perdita. Il calcolo del gradiente può essere oneroso, specialmente con insiemi di dati ampi. Per superare questa sfida, sono state proposte tecniche come:

- **Discesa del Gradiente Stocastico (SGD)**: Questa tecnica implica l'aggiornamento dei parametri del modello utilizzando un solo esempio di training per volta. Ciò significa che il modello viene aggiornato per ogni esempio. Sebbene questo metodo possa presentare una maggiore varianza negli aggiornamenti, aiutando a evitare minimi locali (nella superficie di perdita), può essere più lento e richiedere molte più iterazioni per convergere.

- **Discesa del Gradiente su Mini-Batch**: Questo approccio rappresenta un compromesso tra la Discesa del Gradiente Batch completa e la Discesa del Gradiente Stocastico. Invece di utilizzare l'intero set di dati o un singolo esempio per ogni aggiornamento, un mini-batch di esempi viene utilizzato. Questa tecnica combina il meglio di entrambi i metodi: ha la velocità computazionale della SGD pur mantenendo la stabilità degli aggiornamenti. La dimensione del mini-batch (solitamente espressa come potenza di 2, ad esempio 16, 32, 64 ecc.) diventa un iperparametro importante nel processo di addestramento.

L'addestramento di un modello segue una sequenza strutturata di passi:

- Inizializzazione dei parametri.
- Shuffling dei dati e suddivisione in mini-batch.
- Per ogni mini-batch:
  - Propagazione in avanti per ottenere le previsioni.
  - Calcolo della funzione di perdita.
  - Propagazione all'indietro (back-propagation) per calcolare il gradiente.
  - Aggiornamento dei parametri.

Nel tentativo di migliorare la velocità e la stabilità della convergenza durante l'addestramento, sono stati proposti vari metodi di ottimizzazione. Due tra i più noti sono il momentum (cap. 2.8.2) e l'ottimizzatore ADAM (cap. 2.8.5).

1.3 Tipi di apprendimento

Nel campo dell'apprendimento automatico, gli algoritmi possono essere generalmente categorizzati in tre grandi famiglie:

1.3.1 Apprendimento supervisionato

Nell'apprendimento supervisionato, un algoritmo viene addestrato su un set di dati in cui l'output desiderato è già noto.

Classificazione

È una tecnica in cui i dati usati per l'apprendimento sono accompagnati da ulteriori informazioni che specificano quale sia la classificazione corretta. Queste informazioni aggiuntive vengono utilizzate per imparare il corretto schema di classificazione, schema che poi può essere usato per classificare gli individui sprovvisti di classe.
Il processo standard per implementare un modello di classificazione comprende i seguenti passi:

- **Apprendimento Induttivo**: Si addestra il modello utilizzando un training set di dati etichettati. Questi dati dovrebbero essere rappresentativi dell'intero dataset e selezionati casualmente, pur mantenendo le proporzioni relative delle diverse classi.
- **Valutazione del Modello**: Si valuta l'accuratezza del modello attraverso un test set, confrontando le classificazioni del modello con le etichette reali. Questa fase consente di calcolare una percentuale di successo, che indica quanto è probabile che il modello classifichi correttamente nuovi dati.
- **Processo deduttivo**: Una volta validato, il modello può essere utilizzato per classificare nuovi individui per cui non esiste una classificazione precedente.

Regressione

A differenza della classificazione, che assegna una etichetta categorica a un oggetto, la regressione punta a prevedere una quantità continua o numerica. Ad esempio nella classificazione, potresti voler prevedere se una squadra vincerà, pareggerà o perderà una partita futura. D'altra parte, utilizzando la regressione, potresti cercare di stimare il numero esatto di gol che la squadra segnerà in quella partita.
Il workflow per generare il modello:

- **Fase di Addestramento**: Addestramento del modello su un set di dati con variabili indipendenti e una variabile dipendente continua. L'obiettivo è di trovare la funzione che meglio approssima la relazione tra le variabili.
- **Valutazione del Modello**: Dopo l'addestramento, il modello viene valutato su un test set. Una delle metriche più comuni per la valutazione è l'errore quadratico medio (MSE).
- **Processo deduttivo**: Una volta che il modello ha mostrato un'adeguata accuratezza e performance, può essere utilizzato per fare previsioni su nuovi dati per cui la variabile di risposta è sconosciuta.

1.3.2 Apprendimento non supervisionato

A differenza dell'apprendimento supervisionato, l'apprendimento non supervisionato non utilizza dati etichettati. L'algoritmo impara autonomamente dai dati, identificando strutture o pattern all'interno di essi. Metodologie comuni includono:

Clustering

Il clustering è una delle tecniche più utilizzate nell'apprendimento non supervisionato. Il suo obiettivo è di dividere un insieme di dati in gruppi, o "cluster", in modo tale che gli elementi all'interno di ogni gruppo siano più simili tra loro rispetto a quelli negli altri gruppi. La "somiglianza" è spesso definita in termini di una certa metrica di distanza, come la distanza euclidea. Metodologie comuni:

- **K-means**: Questo algoritmo partiziona i dati in 'K' gruppi, minimizzando la somma delle distanze quadrate tra i punti e i centroidi dei cluster.

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**: Questo algoritmo raggruppa insieme punti che sono vicini nello spazio dei dati, separando punti che sono in aree di bassa densità.

- **Hierarchical Clustering**: Crea una gerarchia di cluster partendo da singoli punti e combinandoli progressivamente.

Workflow tipico per implementare un modello di clustering:

- **Selezione delle Features**

- **Pre-elaborazione dei Dati**

- **Scelta dell'Algoritmo e dei Parametri**

- **Addestramento del Modello**

- **Valutazione dei Cluster**: Utilizzare metriche come il Silhouette Score per valutare la qualità dei cluster.

- **Interpretazione e Applicazione**: Una volta identificati i cluster, si passa alla fase di interpretazione, che potrebbe includere l'etichettatura dei cluster in base alle caratteristiche comuni o l'utilizzo dei cluster per la segmentazione del mercato, la raccomandazione di prodotti, ecc.

Regole di associazione

L'obiettivo è scoprire relazioni interessanti o "regole" che mettano in correlazione la presenza di un insieme di variabili con la presenza di un altro insieme di variabili in grandi dataset. Vengono addestrate su un set di dati di transazioni, dove una transazione è una raccolta di oggetti o eventi acquistati o accaduti insieme. L'algoritmo di apprendimento automatico individua le regole con alta probabilità di co-presenza, espresse in forma di implicazioni. Esistono diversi tipi di regole d'associazione:

- **Regole di consistenza**: Indicano la frequente co-presenza di oggetti o eventi.

- **Regole di copertura**: Mostrano la comune inclusione di oggetti o eventi in transazioni.

- **Regole di lift**: Misurano la forza della relazione tra oggetti o eventi.

Tra gli algoritmi più usati ci sono:

- **Apriori**: Utilizza un approccio "bottom-up" per generare regole, iniziando da itemset di dimensione 1.

- **FP-Growth**: Più veloce di Apriori, codifica il database in una struttura ad albero compatta, l'FP-tree.

1.3.3 Apprendimento per rinforzo

L'apprendimento per rinforzo è un paradigma dove l'algoritmo, o "agente", apprende eseguendo azioni in un ambiente e ricevendo un feedback in termini di "ricompense" o "punizioni".
Non ci sono dati etichettati, ma l'algoritmo apprende comunque dalla sua esperienza, ottimizzando una funzione di ricompensa nel tempo. In ambienti come la guida autonoma, non è una soluzione ottimale principalmente perché i sistemi apprendono attraverso un processo di tentativo ed errore, il che potrebbe essere pericoloso senza un simulatore accurato e sicuro.

Chapter 2

Deep Learning

Il passaggio dal ragionamento simbolico al connessionismo rappresenta un momento fondamentale nello sviluppo dell'intelligenza artificiale (IA) e della scienza della computazione. Gli approcci simbolici, che si basano sull'utilizzo di simboli e regole per manipolarli, sono stati giudicati insufficienti per emulare l'intelligenza umana. I connessionisti propongono un modello alternativo ispirato dalla neurofisiologia, mirato a riprodurre la complessa elaborazione delle informazioni nel cervello umano.
La pratica ha mostrato i limiti degli approcci simbolici, soprattutto nelle attività umane come la percezione sensoriale e il riconoscimento di pattern, dove i computer tradizionali non riescono a eguagliare le capacità umane. Questo ha portato alla convinzione che le azioni umane complesse non si basino esclusivamente su regole fisse, ma richiedano l'apprendimento dall'esperienza.
In questo scenario, il machine learning, e in particolare le reti neurali e il deep learning, emergono come paradigmi molto più efficaci. Ispirandosi al cervello umano, queste tecnologie cercano di imitare la capacità dei neuroni di formare e rafforzare le connessioni sinaptiche attraverso l'apprendimento basato sull'esperienza. Il deep learning utilizza reti neurali profonde per apprendere gerarchie complesse di caratteristiche dai dati, grazie alla loro capacità di rappresentare funzioni non lineari complesse, offrendo un approccio più vicino alla simulazione dell'intelligenza umana rispetto ai metodi tradizionali. Le informazioni presenti nei capitoli 2.1 e 2.2 sono stati reperiti dal libro di S. Haykin "Neural Networks and Learning Machines" [1].

2.1 Cervello umano

Il neurone è l'unità fondamentale del sistema nervoso e funziona come un interruttore elettrico. Composto da dendriti, un soma e un assone, il neurone è specializzato nella ricezione, elaborazione e trasmissione di informazioni sotto forma di impulsi elettrici.

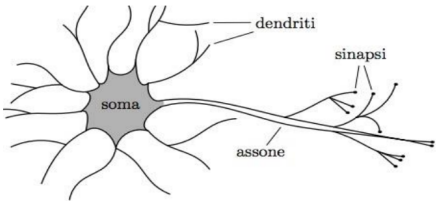

Figure 2.1: Cervello Umano

I dendriti sono estensioni ramificate del neurone che funzionano come antenna, raccogliendo segnali elettrici da altre cellule nervose.
Il corpo cellulare, o soma, è il "centro di elaborazione" del neurone. Riceve segnali dai dendriti e li elabora. Se il segnale è abbastanza forte da superare una certa soglia, viene generato un potenziale d'azione.
L'assone è la "linea di trasmissione", estendendosi dal soma per trasmettere il potenziale d'azione a altre cellule attraverso le sinapsi.
Il potenziale d'azione è un evento elettrico che rappresenta la trasmissione dell'informazione lungo l'assone. Una serie di eventi ionici e molecolari, facilitata da canali ionici e pompe ioniche, contribuisce al generarsi del potenziale d'azione, permettendo il flusso di ioni attraverso la membrana cellulare. Questo è un meccanismo "tutto o nulla": o il segnale raggiunge la soglia per generare un potenziale d'azione, o non succede nulla.
Il punto in cui un neurone comunica con un altro è chiamato sinapsi. L'assone del neurone "pre-sinaptico" rilascia neurotrasmettitori nella fessura sinaptica, una piccola lacuna tra i due neuroni. Questi neurotrasmettitori sono poi rilevati dai dendriti del neurone "post-sinaptico", generando un nuovo segnale elettrico se le condizioni lo permettono.
Come ha teorizzato Donald Hebb nel 1949, l'apprendimento e la memoria sono il risultato di cambiamenti nelle forze delle connessioni sinaptiche. Se un neurone A è abbastanza vicino a un neurone B e contribuisce regolarmente all'attivazione di B, la sinapsi tra A e B si rafforza. Questa è la base per l'apprendimento e la formazione della memoria nel sistema nervoso.

2.2 Neuroni artificiali

Il neurone artificiale è una componente fondamentale delle reti neurali artificiali, che sono algoritmi progettati per emulare l'approccio del sistema nervoso umano alla risoluzione di problemi specifici. Simile ai neuroni biologici, un neurone artificiale è una unità di elaborazione semplice che acquisisce informazioni (input) da altre unità tramite connessioni chiamate "sinapsi". La forza di queste connessioni è quantificata da "pesi sinaptici", che cambiano durante il processo di apprendimento per adattarsi meglio alla risoluzione del problema specifico.

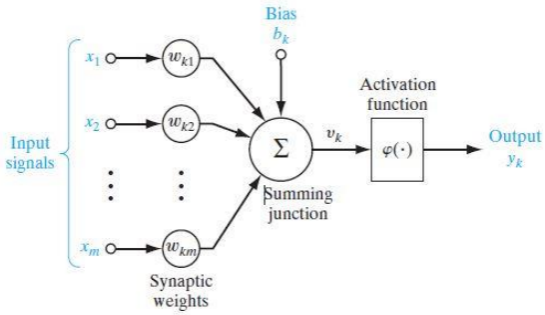

Figure 2.2: Neurone artificiale

Un modello computazionale classico di un neurone artificiale è il Percettrone [2], introdotto da F. Rosenblatt. In questa architettura, un vettore di input viene associato a un vettore di pesi sinaptici. La somma ponderata di questi input e pesi viene poi calcolata secondo la formula:

$$y = \theta \left(\sum_{i=1}^{n} w_i x_i - b \right) \tag{2.1}$$

dove w_i sono i pesi sinaptici e x_i sono i valori di input. Il termine b è noto come "bias", e serve per spostare la funzione di attivazione in modo che possa meglio adattarsi ai dati in input.
Una volta calcolata la somma ponderata, questa viene passata attraverso una funzione di attivazione θ, che modula l'output del neurone. La funzione di attivazione serve anche a limitare l'ampiezza del segnale di output, evitando che raggiunga valori eccessivamente alti o bassi che potrebbero essere problematici per l'apprendimento o la successiva elaborazione del segnale.
Le reti neurali artificiali imparano adattando i pesi sinaptici e il bias durante il processo di apprendimento, che è guidato da un insieme di esempi e dal metodo di apprendimento scelto.
La "memoria" di una rete neurale è intrinseca nei suoi pesi sinaptici, che vengono aggiornati per minimizzare una funzione di errore relativo al compito specifico che la rete è progettata per svolgere.

2.3 Funzioni di attivazione

Nelle reti neurali, l'adozione di funzioni di attivazione non lineari è essenziale per garantire la capacità del modello di rappresentare e apprendere relazioni complesse e non lineari tra gli input. Se si utilizzassero esclusivamente funzioni di attivazione lineari, anche impilando numerosi layer, la rete complessiva avrebbe ancora una funzione di trasformazione lineare. In pratica, la composizione di funzioni lineari resta lineare.
Tuttavia, introducendo non linearità attraverso funzioni di attivazione appropriate, ogni layer aggiuntivo può effettivamente aumentare la capacità espressiva della rete, permettendo di modellare relazioni più complesse tra i dati. Questo è uno dei principi fondamentali che sta alla base del deep learning. Di seguito vengono esaminate alcune delle funzioni di attivazione più comuni:

2.3.1 Funzione Sigmoid (o Logistica)

- **Formula**: $f(x) = \frac{1}{1+e^{-x}}$
- **Range**: (0, 1)
- **Caratteristiche**: Questa funzione mappa qualsiasi input in un valore compreso tra 0 e 1. Tuttavia, può soffrire di scomparsa del gradiente, un fenomeno in cui i pesi e i bias cambiano molto lentamente e impediscono alla rete di apprendere.

2.3.2 Funzione Tangente Iperbolica (tanh)

- **Formula**: $f(x) = \frac{2}{1+e^{-2x}} - 1$
- **Range**: (-1, 1)
- **Caratteristiche**: Simile alla funzione sigmoid, ma mappa l'input in un intervallo tra -1 e 1, rendendola centrata sullo zero. Anche questa può soffrire di scomparsa del gradiente.

2.3.3 Funzione ReLU (Rectified Linear Activation)

- **Formula**: $f(x) = \max(0, x)$
- **Range**: $[0, \infty)$

- **Caratteristiche**: È attualmente una delle funzioni di attivazione più utilizzate nelle reti neurali, soprattutto nelle CNN. È computazionalmente efficiente e aiuta a mitigare il problema della scomparsa del gradiente. Tuttavia, può soffrire del problema delle unità "morte", dove alcuni neuroni possono non attivarsi mai.

2.3.4 Leaky ReLU

- **Formula**: $f(x) = \begin{cases} x & \text{se } x > 0, \\ \alpha x & \text{altrimenti (dove } \alpha \text{ è una piccola costante)} \end{cases}$
- **Range**: $(-\infty, \infty)$
- **Caratteristiche**: Una variazione della ReLU progettata per affrontare il problema delle unità morte introducendo una pendenza leggera per valori negativi.

2.3.5 Softmax

- **Formula**: $f(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_i}}$ per $j = 1, \ldots, K$
- **Range**: (-1, 1)
- **Caratteristiche**: Utilizzata principalmente nell'ultimo strato di reti neurali per problemi di classificazione multiclasse. Converte l'input in una distribuzione di probabilità su diverse classi.

2.4 Back-propagation

La Back-propagation è un algoritmo chiave per l'addestramento di reti neurali artificiali. Questo algoritmo è fondamentalmente suddiviso in due fasi:

- **Forward Propagation**: In questa fase, l'input viene passato attraverso la rete, layer per layer, fino a produrre un output. Questo processo implica la propagazione dei dati attraverso la rete e la produzione di una predizione.
- **Backward Propagation**: Una volta ottenuta la predizione, si calcola l'errore (o la perdita) tra la predizione e il valore reale. L'obiettivo della backpropagation è di minimizzare questo errore aggiornando i pesi della rete. Durante questa fase, l'errore viene "propagato all'indietro" attraverso la rete. Questa propagazione retrograda consente di determinare quanto ogni neurone ha contribuito all'errore totale, e i pesi vengono quindi aggiornati di conseguenza.

L'equazione

$$W_i = W_{i-1} - \eta \frac{\partial \text{Loss}(\omega)}{\partial \omega} \tag{2.2}$$

fornisce una rappresentazione matematica dell'aggiornamento dei pesi durante la backpropagation. Qui, W_{i-1} rappresenta i pesi al passo precedente, $\frac{\partial}{\partial \omega}\text{Loss}(\omega)$ è il gradiente dell'errore rispetto ai pesi, e η è il learning rate.

2.5 Learning rate

Il Learning Rate (α) regola la velocità con cui i pesi della rete vengono aggiornati. Se è troppo alto, la rete potrebbe "saltare" oltre il minimo della funzione di perdita, un fenomeno noto come "overshooting". Ciò potrebbe portare la rete a non convergere o a convergere a una soluzione non ottimale. D'altra parte, se il learning rate è troppo basso, la rete potrebbe impiegare molto tempo per convergere e potrebbe facilmente rimanere intrappolata in minimi locali, piuttosto che trovare il minimo globale.

2.6 Reti neurali

Le reti neurali rappresentano uno degli strumenti più potenti per affrontare problemi di alta complessità, come il riconoscimento di immagini e la comprensione del linguaggio naturale. Esse sono composte da un insieme interconnesso di neuroni artificiali organizzati in strati o "layer". Per ogni coppia di livelli adiacenti i neuroni del livello precedente sono connessi a quelli del livello successivo attraverso collegamenti a cui sono associati dei pesi di valore numerico, che rappresentano la forza della connessione tra neuroni. Il livello di input estrae le feature dai singoli casi e li immette nella rete, mentre quello di output produce il risultato. Ogni neurone dei livelli nascosti (intermedi) riceve in input una serie di valori ponderati, quindi li somma tra loro e si serve di una funzione di attivazione per elaborare il risultato. In base alla loro architettura e al modo in cui i dati vengono elaborati, possiamo classificare le reti neurali in diverse categorie, tra le più importanti:

2.6.1 Feed Forward Neural Network

Queste reti hanno una struttura in cui l'informazione fluisce solo in una direzione, dall'input all'output. Non ci sono cicli, né connessioni all'indietro. Dopo l'input, l'informazione viene elaborata attraverso uno o più strati nascosti per poi raggiungere lo strato di output.

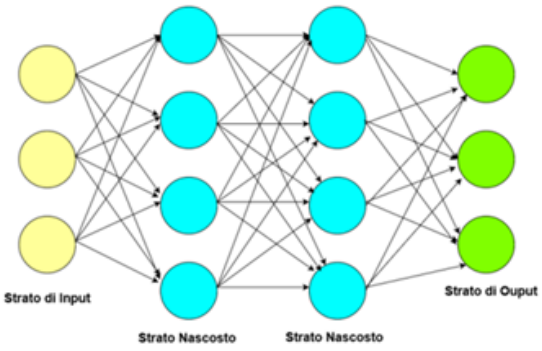

Figure 2.3: Feed Forward NN

2.6.2 Reti Neurali Ricorrenti (RNN)

A differenza delle reti feed-forward, le RNN hanno connessioni cicliche che permettono all'informazione di circolare all'interno della rete. Ciò le rende particolarmente adatte per gestire sequenze di dati,

come serie temporali o frasi, dato che possiedono una sorta di "memoria" delle informazioni passate.

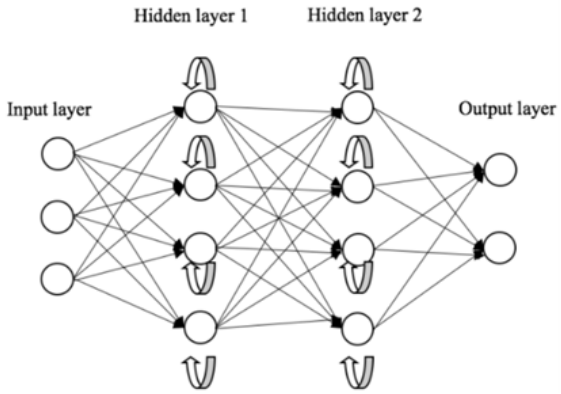

Figure 2.4: Reti Neurali Ricorrenti

2.6.3 Reti Neurali Convoluzionali (CNN)

Queste sono progettate per l'elaborazione di dati con una topologia a griglia, come le immagini. Utilizzano operazioni di convoluzione per scansionare porzioni localizzate dei dati, rendendole estremamente efficienti nel riconoscimento di pattern spaziali.
Le CNN sono di fondamentale importanza nell'ambito dell'elaborazione delle immagini, per questo motivo verranno trattate approfonditamente in un capitolo dedicato.

2.7 Tipi di layer

Come precedentemente accennato, una rete neurale è composta da unità di elaborazione chiamate neuroni, organizzate in strati o layer. Questi layer sono la colonna portante della rete, determinando come l'informazione viene processata, filtrata e infine prodotta come output.
A seconda della natura e della complessità del problema da risolvere, potrebbero essere necessarie diverse architetture di rete e, di conseguenza, diversi tipi di layer. La combinazione e la sequenza di questi layer influenzano la capacità della rete di apprendere dai dati e di generalizzare su nuovi input.
Ecco una panoramica di alcuni dei tipi di layer più comuni utilizzati nelle reti neurali:

2.7.1 Layer densi o totalmente connessi

Ogni neurone in questo layer è connesso a tutti i neuroni del layer precedente e a tutti quelli del layer successivo. Trovano applicazione spesso come layer finali nelle reti CNN e come layer principali nelle MLP (MultiLayer Perceptrons).

2.7.2 Layer convoluzionali

Questi layer sono caratterizzati dalla presenza di filtri (o kernel) che "scivolano" sull'input per produrre una mappa delle caratteristiche (feature map). Permettono di catturare le caratteris-

tiche locali e spaziali dell'input. Sono fondamentali nelle reti neurali convolutive (CNN) utilizzate prevalentemente nel riconoscimento delle immagini.

2.7.3 Layer di pooling

Riducono la dimensione spaziale (altezza e larghezza) dell'input mantenendo le caratteristiche più importanti. Esistono vari tipi di pooling come il max-pooling e il mean-pooling. Sono spesso intercalati tra layer convolutivi nelle CNN per ridurre la computazione e catturare le caratteristiche invarianti.

2.7.4 Layer ricorrenti

Hanno connessioni che agiscono da loop all'interno del layer, permettendo alla rete di avere una memoria del passato. Questo rende i layer ricorrenti adatti a dati sequenziali. Sono il cuore delle reti neurali ricorrenti (RNN) e sono usati in applicazioni come la modellazione di sequenze, traduzione automatica e riconoscimento vocale.

2.7.5 Layer di dropout

Durante l'addestramento, spegne (rende inattivi) randomicamente una frazione dei neuroni, aiutando a prevenire l'overfitting e migliorare la generalizzazione del modello.

2.7.6 Layer di embedding

Trasformano gli input categorici o interi in vettori densi e continui. Sono utilizzati per rappresentare parole o altri elementi discreti in uno spazio vettoriale continuo. Sono fondamentali nei modelli di elaborazione del linguaggio naturale NLP.

2.7.7 Layer di normalizzazione (Batch, Layer, Instance)

Normalizzano l'output dei neuroni per assicurare che abbiano una distribuzione coerente, facilitando e accelerando l'addestramento. Sono frequentemente utilizzati nelle CNN e nelle reti profonde per stabilizzare l'apprendimento.

2.8 Ottimizzatori

Gli ottimizzatori svolgono un ruolo fondamentale nell'allenamento delle reti neurali, consentendo di aggiornare i pesi delle reti in modo da minimizzare una certa funzione di perdita. Diversi tipi di ottimizzatori utilizzano varie strategie per raggiungere questo obiettivo. Ecco una panoramica dei più comuni e delle loro caratteristiche principali:

2.8.1 Discesa del gradiente stocastico (SGD)

La Discesa del Gradiente Stocastico (SGD) rappresenta una delle metodologie di ottimizzazione più note e utilizzate. Essa è una variante della Discesa del Gradiente, un algoritmo di ottimizzazione che mira a trovare i parametri (o pesi) di una funzione che minimizzino una determinata funzione obiettivo, di solito la Funzione di Costo o Funzione di perdita (Loss Function). Vediamo in dettaglio il funzionamento e le caratteristiche principali di SGD:

La Discesa del Gradiente classica aggiorna i pesi di una rete neurale o di un modello di apprendimento basandosi sull'intero set di dati di addestramento. Ciò significa che, prima di ogni aggiornamento, si deve calcolare il gradiente della funzione di costo rispetto a tutti gli esempi nel dataset. Questo processo può diventare molto oneroso e inefficiente specialmente con set di dati di grandi dimensioni. Al contrario, SGD effettua un aggiornamento dei pesi per ogni singolo esempio di addestramento.
Per ogni esempio, l'algoritmo calcola l'errore e aggiorna i pesi in modo proporzionale al gradiente negativo della funzione di costo, utilizzando una formula del tipo:

$$W := W - \eta \nabla J(W) \tag{2.3}$$

dove η è la velocità di apprendimento e $\nabla J(W)$ è il gradiente della funzione di costo rispetto ai pesi W.
Vantaggi: Poiché l'aggiornamento viene effettuato per ogni esempio, SGD tende ad essere molto più veloce rispetto alla Discesa del Gradiente tradizionale. Riesce anche a sfuggire ai minimi locali grazie alla sua natura stocastica anche se non può convergere al minimo glabale.
Svantaggi: Poiché ogni aggiornamento è basato su un singolo esempio, gli aggiornamenti dei pesi possono variare considerevolmente, portando a una convergenza instabile. La scelta di un'adeguata velocità di apprendimento è cruciale. Un valore troppo alto può far "rimbalzare" l'algoritmo intorno al minimo, mentre un valore troppo basso può rendere la convergenza estremamente lenta.

2.8.2 Momentum

Momentum è una tecnica che aiuta ad accelerare la SGD (Discesa del Gradiente Stocastico) nella direzione rilevante e ad attenuare le oscillazioni in direzioni irrilevanti. Esso può essere visto come una palla che rotola lungo una superficie scivolosa: la palla accumula velocità (momento) in direzioni con pendenza costante e rallenta rapidamente in zone pianeggianti. L'idea alla base del Momentum è quella di introdurre una variabile (il "momento") che tiene traccia della direzione degli aggiornamenti passati. Questa variabile viene combinata con il gradiente corrente per determinare l'aggiornamento dei pesi nel passo attuale. La formula per l'aggiornamento:

$$v = \beta v - \eta \nabla J(W) \tag{2.4}$$

$$W = W + v \tag{2.5}$$

dove, $\nabla J(W)$ rappresenta il gradiente della funzione di costo rispetto ai pesi correnti W, η è la velocità di apprendimento, β è il coefficiente di momentum (generalmente compreso tra 0 e 1), e v è il termine di momentum che accumula i gradienti passati.
Vantaggi: Se la direzione del gradiente cambia rapidamente, il momento contribuirà a smorzare gli aggiornamenti, evitando oscillazioni e instabilità. La componente di momentum può aiutare l'algoritmo a sfuggire da minimi locali o da punti di sella (zone dove il gradiente è zero ma che non sono né un minimo né un massimo), accelerando la convergenza.
Svantaggi: L'introduzione del coefficiente di momentum β aggiunge un altro parametro da sintonizzare. Anche se esistono valori consigliati (ad esempio, 0.9), in alcuni scenari potrebbe essere necessario regolare questo valore.

2.8.3 Adagrad

La caratteristica distintiva di Adagrad rispetto ad altri algoritmi di ottimizzazione è il modo in cui adatta dinamicamente il tasso di apprendimento di ciascun parametro in base alla storia

dei gradienti. L'idea principale è di adattare il tasso di apprendimento di ciascun parametro in modo inversamente proporzionale alla radice quadrata della somma dei quadrati dei gradienti passati per quel parametro. Questo significa che i parametri che hanno gradienti grandi avranno un tasso di apprendimento più lento, mentre quelli con gradienti piccoli avranno un tasso di apprendimento più rapido.

Formule di aggiornamento

Siano g_t il gradiente al tempo t, G_t una matrice diagonale dove ogni diagonale i contiene la somma dei quadrati dei gradienti per il parametro i fino al tempo t, e η il tasso di apprendimento iniziale. L'aggiornamento del parametro W in Adagrad può essere:

$$G_t = G_{t-1} + g_t g_t^T \tag{2.6}$$

$$W_{t+1} = W_t - \eta G_t^{-1/2} \otimes g_t \tag{2.7}$$

Qui, $G_t^{-1/2}$ denota la matrice diagonale con la radice quadrata inversa di G_t lungo la diagonale.
Vantaggi: Adagrad può essere particolarmente vantaggioso per problemi sparsi, dove alcuni parametri sono aggiornati più frequentemente di altri. L'adattamento del tasso di apprendimento per ciascun parametro permette un'ottimizzazione più mirata. Riduce anche la necessità di sintonizzazione manuale del tasso di apprendimento, dato che adatta dinamicamente il tasso per ciascun parametro.
Svantaggi: Poiché AdaGrad accumula il quadrato dei gradienti nel denominatore, il tasso di apprendimento effettivo per ogni parametro tende a decrescere molto rapidamente, il che può portare a una convergenza prematura o lenta nei problemi di deep learning. Questo è il motivo per cui, in contesti pratici, sono spesso preferiti altri ottimizzatori, come Adam (cap. 2.8.5), che combinano l'idea di AdaGrad con altre tecniche per mitigare questo problema.

2.8.4 RMSProp

L'algoritmo RMSProp cerca di risolvere alcuni dei problemi associati alla Discesa del Gradiente Stocastico (SGD) e all'Adagrad, in particolare la riduzione rapida e monotonica del tasso di apprendimento nel caso di Adagrad. Il concetto chiave di RMSProp è di utilizzare una media mobile esponenziale decrescente del quadrato dei gradienti passati per aggiornare i pesi, anziché accumulare semplicemente tutti i gradienti precedenti come fa Adagrad.

Formule di aggiornamento

Dato un gradiente g_t al tempo t, una costante di decadimento γ (tipicamente impostata a valori come 0.9), e un tasso di apprendimento η, l'aggiornamento del parametro W in RMSProp può essere formulato come:

$$v_t = \gamma v_{t-1} + (1-\gamma) g_t^2 \tag{2.8}$$

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{v_t + \epsilon}} g_t \tag{2.9}$$

Qui, v_t è una stima del quadrato medio decrescente del gradiente. ϵ è un piccolo valore di smorzamento per evitare divisioni per zero (tipicamente $\epsilon = 10^{-7}$).
Vantaggi: Analogamente ad Adagrad, RMSProp adatta il tasso di apprendimento per ogni

parametro individualmente, il che lo rende meno sensibile ai valori iniziali del tasso di apprendimento. A differenza di Adagrad, RMSProp non ha una riduzione monotonica del tasso di apprendimento grazie all'utilizzo della media mobile esponenziale decrescente.
Svantaggi: Nonostante RMSProp risolva alcune delle limitazioni di Adagrad, in alcuni contesti potrebbe non convergere altrettanto rapidamente come altre tecniche di ottimizzazione avanzate, come Adam, che combina concetti sia da RMSProp che da Momentum.

2.8.5 Adam (Adaptive Moment Estimation)

Adam [3], che sta per "Adaptive Moment Estimation", è un metodo di ottimizzazione che combina le idee di due altre tecniche di ottimizzazione: il Momentum e l'AdaGrad. È uno degli algoritmi di ottimizzazione più popolari e viene spesso consigliato come l'ottimizzatore di default per molte applicazioni di deep learning, grazie alla sua capacità di adattarsi ai diversi problemi e al fatto che richiede meno sintonizzazione manuale degli iperparametri rispetto ad altri metodi.
Vediamo in dettaglio le sue caratteristiche:

Combinazione di Metodi

- **Momentum**: Adam mantiene una stima in movimento (media mobile esponenziale) dei gradienti passati, simile al metodo Momentum.
- **Adaptive Learning Rates** Adam mantiene anche una stima in movimento del quadrato dei gradienti, simile ad AdaGrad e RMSProp, permettendo così un adattamento del learning rate per ogni parametro.

Formule di aggiornamento

- $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Momento del primo ordine)
- $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Momento del secondo ordine)
- $m_t = \frac{m_t}{(1-\beta_1^t)}$ (Correzione del bias per il primo momento)
- $v_t = \frac{v_t}{(1-\beta_2^t)}$ (Correzione del bias per il secondo momento)
- $W = W - \eta \frac{\hat{m}_t}{(\sqrt{\hat{v}_t} + \epsilon)}$

Qui, g_t rappresenta il gradiente al tempo t, m_t e v_t sono stime dei momenti del primo e secondo ordine dei gradienti, rispettivamente, η è il learning rate, ϵ è un piccolo valore per prevenire la divisione per zero (solitamente intorno a 10^{-7}), β_1 e β_2 sono iperparametri che controllano il decadimento delle medie mobili; valori comuni sono 0.9 per β_1 e 0.999 per β_2.
Vantaggi: Adam è computazionalmente efficiente e richiede poca memoria. Adatta i learning rates per ciascun parametro basandosi sulla stima del primo e secondo momento dei gradienti. Questo rende Adam meno sensibile alla scelta iniziale del learning rate rispetto alla SGD standard. Spesso funziona bene con poche modifiche degli iperparametri.
Svantaggi: Anche se Adam è generalmente robusto e si comprota bene in molte situazioni, alcuni studi hanno mostrato che potrebbe non convergere all'ottimo in alcuni scenari, specialmente in task di training profondo e non regolarizzato.

2.9 Vanishing ed exploding Gradient

Il problema del vanishing gradient si verifica durante l'addestramento di reti neurali profonde, in particolare quando i gradienti tendono a diventare estremamente piccoli man mano che vengono retropropagati attraverso la rete. Di conseguenza, gli aggiornamenti ai pesi nelle prime fasi (o strati) della rete sono trascurabili, il che rende difficile o impossibile l'addestramento di questi strati.
Le cause possono essere le seguenti:

- **Funzioni di Attivazione**: Funzioni come la sigmoide e la tanh, quando saturate, hanno derivati vicini a zero. Questo può portare a gradienti molto piccoli durante la retropropagazione.
- **Inizializzazione dei Pesi**: L'inizializzazione con valori molto piccoli può far saturare le attivazioni nelle regioni dove le derivazioni sono piccole.
- **Profondità della Rete**: Nelle reti profonde, il problema può essere amplificato poiché il gradiente viene moltiplicato molte volte attraverso gli strati.

Questo porta a un addestramento lento e a una pessima convergenza e performance del modello. Opposto al problema del vanishing gradient, l'exploding gradient si verifica quando i gradienti diventano troppo grandi, causando aggiornamenti eccessivamente grandi ai pesi. Questo può portare a pesi infiniti e, di conseguenza, a un modello instabile e inutilizzabile.

Soluzioni ad entrambi problemi possono essere:

- **Funzioni di Attivazione Adeguata**: Usare funzioni come ReLU o le sue varianti può aiutare a contrastare entrambi i problemi.
- **Inizializzazione dei Pesi**: L'adozione di metodi di inizializzazione, come l'inizializzazione di He o Xavier, può mitigare questi problemi.
- **Batch Normalization**: Normalizzando gli output di ogni strato, si possono evitare saturazioni estreme.
- **Troncamento del Gradiente**: Per l'exploding gradient, si può troncare (limitare) il gradiente a un valore massimo durante la retropropagazione.
- **Architetture Residuali**: I "collegamenti residuali" nelle architetture come ResNet aiutano a preservare il gradiente attraverso la rete, contrastando il vanishing gradient.
- **Regolarizzazione**: L'aggiunta di regolarizzazione L1 o L2 ai pesi può aiutare a prevenire l'exploding gradient.

2.10 Transfer Learning e Fine tuning

Il **transfer learning** si riferisce all'approccio di utilizzare un modello pre-addestrato su un grande dataset (come ImageNet, che contiene milioni di immagini appartenenti a migliaia di classi) come punto di partenza per addestrare un modello su un compito diverso con un dataset di dimensioni più ridotte. La motivazione dietro il transfer learning è che le caratteristiche apprese dal modello sul dataset originale possono essere utili anche per risolvere compiti simili. Per esempio, i primi livelli di una rete neurale addestrata su immagini spesso imparano a riconoscere

bordi, texture e altre caratteristiche basilari, che sono utili per molte applicazioni di computer vision.
Le fasi del transfer learning sono i seguenti:

1. **Pre-addestramento**: Si addestra un modello su un grande dataset.

2. **Estrazione di Caratteristiche**: Si usa il modello pre-addestrato (senza gli ultimi livelli) per estrarre le caratteristiche dal nuovo dataset.

3. **Addestramento**: Si addestra un nuovo modello (spesso solo una parte del modello, come gli ultimi livelli) utilizzando le caratteristiche estratte.

Il **fine-tuning** è un'estensione del transfer learning. Invece di tenere fissi i pesi dei livelli precedenti del modello e addestrare solo gli ultimi livelli, nel fine-tuning si continua l'addestramento di (alcuni o tutti) i livelli della rete neurale pre-addestrata. Questo consente al modello di adattare anche le caratteristiche dei livelli più profondi al nuovo compito. Le fasi del fine-tuning sono simili a quelle del transfer learning, con la differenza che, nella fase di addestramento, si addestrano anche i livelli precedenti (con un tasso di apprendimento solitamente più basso per evitare di perdere le caratteristiche utili apprese durante il pre-addestramento).

2.11 Convolutional Neural Network

Le Convolutional Neural Networks (CNN), o reti neurali convolutive, sono una categoria di reti neurali profonde particolarmente adatte al riconoscimento di pattern in immagini. Rispetto alle tradizionali reti neurali completamente connesse, le CNN sono progettate per riconoscere automaticamente ed in modo gerarchico i pattern visivi dai dati delle immagini.
Quando si analizza un'immagine con i tradizionali Fully Connected (FC) layers, il computo risulterebbe molto oneroso, anche per la rilevazione di semplici caratteristiche come bordi e angoli. Questo perché ogni neurone in un FC layer è collegato a tutti i neuroni del layer precedente, rendendo la struttura molto pesante in termini computazionali. Inoltre, nel caso delle immagini, questo approccio non tiene conto della struttura spaziale delle immagini, ovvero della correlazione tra i pixel vicini.
Ecco perché, nel contesto del riconoscimento di immagini, si utilizzano le reti neurali convolutive. Le principali differenze tra i layers convolutivi e i FC layers sono:

1. **Mantenimento della struttura spaziale**: A differenza dei FC layers che richiedono un "flattening" (appiattimento) dell'immagine in un vettore unidimensionale, le CNN mantengono la struttura 2D dell'immagine.

2. **Campo recettivo locale**: Ogni unità di output in un layer convolutivo è collegata solo ad un piccolo insieme di unità di input vicine, noto come campo recettivo.

3. **Condivisione dei pesi**: Tutte le unità di output in un layer convolutivo condividono lo stesso set di pesi. Ciò significa che stanno cercando la stessa caratteristica in diverse parti dell'immagine.

4. **Bias induttivo**: Le CNN introducono un bias induttivo, supponendo che i pattern locali (come bordi, angoli) possano apparire in qualsiasi parte dell'immagine.

Quando parliamo di "convoluzione" in questo contesto, ci riferiamo in realtà alla correlazione.

2.11.1 Convoluzione

La **convoluzione** è una operazione matematica che descrive il modo in cui due funzioni si combinano per produrre una terza funzione. Si può pensare alla convoluzione come al processo di applicazione di un filtro (o kernel) a un segnale o a un'immagine. L'operazione di convoluzione può essere formalmente definita come:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2.10}$$

Nel contesto della computer vision, l'equazione sopra viene generalmente semplificata in una somma discreta per immagini 2D:

$$(I * K)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(i, j)K(x-i, y-j) \tag{2.11}$$

Dove I è l'immagine di input, K è il filtro o kernel e I*K è l'immagine di output.

2.11.2 Correlazione

La convoluzione è definita come:

$$(f * g)(t) = \int_{-\infty}^{\infty} f^*(\tau)g(t+\tau)d\tau \tag{2.12}$$

E per immagini 2D:

$$(I * K)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(i, j)K(x+i, y+j) \tag{2.13}$$

In termini generali, un layer convolutivo prende un'attivazione di input multi-canale e produce un'attivazione di output multi-canale applicando diversi filtri. Ogni filtro è progettato per rilevare una caratteristica specifica nell'immagine.
Un layer convolutivo da solo è una forma speciale di layer lineare e può essere espresso come moltiplicazione di matrici. Per introdurre non linearità nelle CNN, si utilizzano funzioni di attivazione, tipicamente la **ReLU (Rectified Linear Unit)**.
Per evitare che le attivazioni si riducano lungo la catena, è comune l'utilizzo del **padding**, che aggiunge zeri ai bordi dell'immagine prima della convoluzione.
Vengono utilizzate i **pooling layer** per prendere l'output da un'area dell'immagine e riassumerlo in un singolo valore. Il pooling più comune è il **max pooling**, che prende il valore massimo da un'area dell'immagine.
La struttura generale di una CNN include layers convolutivi seguiti da layers completamente connessi. Questa porzione della rete viene spesso chiamata "feature extractor" perché estrae automaticamente le caratteristiche dalle immagini. L'ultimo layer completamente connesso è spesso chiamato "classificatore" poiché prende le caratteristiche estratte e produce una previsione.
Le prime CNN come **LeNet-5** erano utilizzate per la classificazione di cifre scritte a mano. Con l'evoluzione della tecnologia e l'aumento della potenza di calcolo, sono state sviluppate reti più profonde e complesse come **AlexNet** e **VGG**. Un'altra innovazione significativa nel campo delle CNN è stata l'introduzione delle **Residual Networks (ResNet)**.

2.11.3 AlexNet

Una delle novità introdotte da AlexNet è l'uso della funzione di attivazione ReLU, che ha permesso un addestramento più veloce risolvendo problemi di vanishing gradient. L'architettura ha anche incorporato tecniche come il dropout per combattere l'overfitting e ha sfruttato l'uso di GPU per l'addestramento, facilitando l'addestramento di modelli più grandi in tempi più rapidi.

2.11.4 VGGNet

VGGNet ha introdotto l'idea di impilare molteplici layer convolutivi con piccoli kernel 3x3. Questo ha permesso alla rete di catturare relazioni spaziali complesse con meno parametri. Sebbene la rete sia profonda, con varianti come VGG16 e VGG19, la sua struttura è intuitiva e facile da seguire.

2.11.5 GoogLeNet

GoogLeNet, spesso chiamato Inception, ha portato una nuova prospettiva al design delle reti neurali. Ha introdotto l'architettura "Inception", che consente alla rete di apprendere caratteristiche su diverse scale simultaneamente. All'interno di ogni modulo Inception, convoluzioni di dimensioni diverse (1x1, 3x3 e 5x5) e operazioni di pooling sono eseguite in parallelo e i risultati sono concatenati. Questo ha permesso una maggiore flessibilità nell'apprendimento delle caratteristiche. Un altro concetto innovativo introdotto da GoogLeNet è il "network-in-network", utilizzando convoluzioni 1x1 per ridurre la dimensionalità.

2.11.6 ResNet

ResNet, o Reti Residue, ha rivoluzionato ulteriormente il modo in cui vediamo l'addestramento delle reti neurali profonde. La sua innovazione chiave è l'introduzione di connessioni residue che consentono al segnale di bypassare uno o più layer. Questo semplice, ma potente, meccanismo ha permesso l'addestramento di reti con centinaia di layer senza incorrere in problemi di vanishing gradient.

2.11.7 MobileNet

MobileNet è stato progettato per fornire una rete neurale convoluzionale efficace per dispositivi con risorse limitate come smartphone o dispositivi IoT. A differenza delle reti dense come VGGNet o ResNet, MobileNet sfrutta convoluzioni separabili in profondità, che scompongono la convoluzione in operazioni più leggere, riducendo notevolmente il numero di parametri e calcoli.

Chapter 3

Object Detection

La Computer Vision ha fatto passi da gigante negli ultimi anni, e una delle sue applicazioni principali è l'object detection. A differenza della classificazione di immagini, dove l'obiettivo è identificare l'oggetto principale in un'immagine, l'object detection si basa su due componenti principali: la localizzazione, che si occupa di identificare la posizione dell'oggetto, e la classificazione, che etichetta l'oggetto rilevato. Per la localizzazione, una bounding box viene spesso definita da quattro valori:

- Coordinate x, y del centro dell'oggetto.
- Altezza e larghezza della bounding box.

3.1 Tecniche tradizionali

Prima dell'avvento delle reti neurali profonde, l'object detection si basava su tecniche come i descrittori di caratteristiche, come SIFT e HOG, accoppiate con classificatori come le macchine a vettori di supporto (SVM).

3.1.1 SIFT (Scale Invariant Feature Transform)

Il descrittore SIFT (Scale-Invariant Feature Transform) è un algoritmo per rilevare e descrivere le caratteristiche locali nelle immagini. Fu introdotto da David Lowe nel 1999 e da allora ha avuto un impatto significativo sull'analisi e l'interpretazione delle immagini e dei video, diventando uno dei descrittori di caratteristiche più utilizzati e influenti. Prima di addentrarsi nei dettagli tecnici di SIFT, è utile comprendere alcuni concetti fondamentali:

1. **Caratteristiche Locali**: Sono punti distintivi in un'immagine, come angoli e/o bordi, che possono essere riconosciuti in modo affidabile tra diverse immagini dello stesso oggetto o scena.
2. **Invarianza di Scala**: La capacità di identificare le stesse caratteristiche a diverse scale (dimensione dell'oggetto nell'immagine).
3. **Invarianza alla Rotazione**: La capacità di riconoscere le stesse caratteristiche indipendentemente dall'orientamento dell'immagine.
4. **Robustezza alle variazioni di illuminazione**: La capacità di riconoscere le stesse caratteristiche sotto diversi livelli di illuminazione.

L'algoritmo SIFT può essere suddiviso in quattro fasi principali:

1. **Rilevamento dei keypoints**: La ricerca di caratteristiche stabili inizia con la costruzione dello spazio delle scale dell'immagine. Questo viene fatto attraverso la convoluzione dell'immagine originale con filtri Gaussiani di varie deviazioni standard. Si ottiene una serie di immagini "smussate" a scale crescenti. Per trovare caratteristiche che siano invarianti alla scala, l'algoritmo calcola la Differenza di Gaussiane (DoG) sottraendo immagini Gaussiane successive. I massimi e i minimi locali della DoG sono i candidati per le posizioni delle caratteristiche invarianti alla scala.

2. **Localizzazione dei keypoints**: Ogni punto estremo trovato nella fase 1 viene esaminato per determinare se è un punto di interesse valido. Ciò avviene tramite un processo di eliminazione basato sul contrasto e sul rapporto della curvatura principale. Questo passo riduce il numero di punti di interesse errati, come quelli causati dal rumore.

3. **Assegnazione dell'Orientamento**: Per ogni caratteristica, viene calcolato l'istogramma dei gradienti orientati nell'intorno della caratteristica. L'orientamento dominante dell'istogramma è assegnato alla caratteristica. Questo fornisce invarianza alla rotazione, consentendo alla caratteristica di essere riconosciuta indipendentemente dall'orientamento dell'immagine.

4. **Descrizione dei keypoints**: L'ultimo passo è creare un descrittore per ogni caratteristica. Intorno al punto di interesse, l'immagine viene divisa in sotto-regioni. Per ogni sotto-regione, viene calcolato l'istogramma dei gradienti orientati. Questi istogrammi sono poi concatenati per formare un vettore di 128 elementi che proviene da 16 istogrammi (4x4 sotto-regioni intorno al punto di interesse) con 8 bin ciascuno (per 8 direzioni principali del gradiente). Questo vettore è normalizzato per garantire la robustezza alle variazioni di illuminazione.

Nonostante la sua robustezza, SIFT ha delle limitazioni:

- **Elevato costo computazionale**: L'elaborazione di SIFT può essere computazionalmente onerosa, specialmente con immagini di grandi dimensioni o in applicazioni in tempo reale.

- **Performance in condizioni di visione variabile**: In presenza di occlusione importanti, cambiamenti di vista drastici o compressione dell'immagine, le prestazioni di SIFT possono degradare.

3.2 HOG (Histogram of Oriented Gradients)

L'Histogram of Oriented Gradients (HOG) è una tecnica per l'estrazione delle caratteristiche per il riconoscimento di oggetti. L'idea chiave è che l'aspetto locale e la forma di un oggetto all'interno di un'immagine possono essere descritti dalla distribuzione dell'intensità del gradiente o dalle direzioni del bordo. L'algoritmo SIFT può essere nelle seguenti principali fasi:

1. **Calcolo del gradiente**: La prima fase nel calcolo di HOG è il calcolo del gradiente dell'immagine. Per ogni pixel, il gradiente è un vettore bidimensionale che punta nella direzione del maggiore cambiamento di intensità e la cui lunghezza è proporzionale alla grandezza di quel cambiamento. I gradienti sono generalmente calcolati usando filtri di Sobel, che sono maschere di convoluzione applicate all'immagine per ottenere le derivate parziali in direzione orizzontale e verticale.

2. **Suddivisione in celle**: L'immagine viene suddivisa in piccole regioni spaziali chiamate celle, e per ciascuna cella viene calcolato un istogramma dei gradienti orientati. L'istogramma conta quanti pixel nella cella hanno il loro gradiente orientato in una particolare direzione. Le direzioni possono essere suddivise in un numero fisso di bin angolari. Per esempio, si potrebbe utilizzare un istogramma a 9 bin, coprendo 0-180 gradi.

3. **Normalizzazione del Blocco**: Le celle vengono poi raggruppate in blocchi più grandi, e gli istogrammi delle celle all'interno di ciascun blocco vengono normalizzati insieme. Questo passo è cruciale per introdurre qualche livello di invarianza alle variazioni di illuminazione e ombreggiatura.

4. **Calcolo del descrittore**: Il descrittore HOG per l'intera immagine è ottenuto concatenando i vettori normalizzati di tutti i blocchi dell'immagine. Questo descrittore può essere poi utilizzato come input per un classificatore basato su macchine a vettori di supporto (SVM) o altri algoritmi di machine learning per il riconoscimento di oggetti.

HOG ha le segenti limitazioni:

- **Sensibilità alla Posizione e alla Pose**: Non è invariante alla posizione e non gestisce bene le variazioni di pose.
- **Elaborazione Computazionalmente Intensiva**: L'estrazione di HOG può essere onerosa in termini di calcolo.
- **Performance in Condizioni di Illuminazione Estrema**: Sebbene la normalizzazione migliori la robustezza, variazioni estreme di illuminazione possono comunque degradare le prestazioni.

3.3 Classificazione tramite SVM

Le Support Vector Machines (SVM) sono uno degli algoritmi di apprendimento supervisionato più robusti e accurati utilizzati sia per la classificazione che per la regressione, ma prevalentemente nel primo caso. Le SVM sono basate sulla teoria del margine e sul concetto di iperpiano ottimale. In un contesto bidimensionale, un iperpiano è semplicemente una linea che divide uno spazio in due parti. Nel caso delle SVM, lo spazio è tipicamente multidimensionale e l'iperpiano è una struttura che divide lo spazio di caratteristiche in due parti, ciascuna delle quali è associata a una differente classe.

3.3.1 Massimizzazione del margine

L'idea fondamentale dietro le SVM è trovare l'iperpiano che ha il massimo margine, cioè la massima distanza tra l'iperpiano stesso e i punti di dati più vicini a questo, che sono noti come vettori di supporto. Un margine più ampio offre una maggior confidenza nella corretta classificazione dei dati e una migliore generalizzazione dal training set al test set.

3.3.2 Utilizzo delle SVM

Nella **classificazione binaria**, l'obiettivo delle SVM è di costruire un modello che assegni nuovi esempi a una delle due categorie. Ciò viene fatto costruendo un iperpiano che separa al meglio le due classi di punti.
Nella **regressione**, invece di trovare un iperpiano che separa due classi, l'obiettivo è trovare un

iperpiano che meglio si adatta ai dati continuando a considerare il margine.
Il **training di una SVM** si riduce a un problema di ottimizzazione quadratica, dove la funzione obiettivo è quella di massimizzare il margine tra le classi. Ciò si ottiene minimizzando una funzione che dipende dai pesi assegnati alle caratteristiche, con un vincolo che i dati di training siano classificati correttamente.
Le SVM tradizionali sono state progettate per la classificazione binaria. Tuttavia, esistono approcci come "one-vs-one" e "one-vs-all" per estenderle al problema multi classe. La **scelta dei parametri** del kernel e il valore di regolarizzazione C (che controlla il trade-off tra il margine e l'errore di classificazione) sono fondamentali per le prestazioni dell'SVM e vengono tipicamente selezionati tramite cross-validation.
Nella computer vision, le SVM sono spesso utilizzate come classificatori di immagini. Gli input per l'SVM sono i descrittori delle immagini, come HOG e SIFT che codificano informazioni importanti riguardo la forma e la texture degli oggetti nelle immagini. La capacità dell'SVM di operare in spazi ad alta dimensione e la sua efficacia nel gestire dati non linearmente separabili lo rendono particolarmente adatto per lavorare con le ricche rappresentazioni fornite da HOG e SIFT.

3.4 Localizzazione tramite corrispondenze

Come illustrato all'inizio di questo capitolo, l'Object Detection integra sia la classificazione, attraverso algoritmi come quelli basati sulle Support Vector Machines (SVM) descritti nel capitolo dedicato, sia la localizzazione degli oggetti. La localizzazione si avvale dell'uso di "bounding box", ovvero rettangoli che incorniciano l'oggetto da identificare all'interno di una scena, per indicarne la posizione precisa.
Per realizzare questo processo, si utilizza l'algoritmo Scale-Invariant Feature Transform (SIFT), che permette di rilevare i keypoints e calcolare i loro descrittori unici. Successivamente, si impiegano specifici algoritmi per stabilire le corrispondenze tra i keypoints dell'immagine dell'oggetto da riconoscere e quelli presenti nelle immagini di scena. Questo passaggio è cruciale per identificare e localizzare l'oggetto desiderato all'interno di nuovi contesti visivi.
Per l'immagine dell'oggetto da riconoscere viene quindi calcolata un'omografia. Questa trasformazione prospettica consente di mappare i punti dell'immagine dell'oggetto sui corrispondenti punti nell'immagine di scena. L'omografia è essenziale per determinare con precisione la posizione e l'orientamento dell'oggetto all'interno della scena, consentendo di definire i confini dell'oggetto attraverso una bounding box che avvolge i vertici trasformati.

3.5 Verso il Deep Learning

In un'epoca dominata dall'apprendimento profondo, le SVM rimangono rilevanti, specialmente in scenari in cui i set di dati sono più piccoli e le reti neurali profonde potrebbero essere soggette a sovradimensionamento. La capacità delle SVM di generalizzare bene con un limitato numero di esempi di training le rende uno strumento prezioso nel toolkit di machine learning e computer vision. Tuttavia, questi metodi non erano abbastanza robusti o accurati per applicazioni in tempo reale o per immagini complesse con molteplici oggetti. Con l'avvento delle reti neurali convoluzionali (CNN), l'object detection ha vissuto una rivoluzione. Le CNN sono particolarmente adatte a lavorare con immagini grazie alla loro capacità di estrarre gerarchicamente caratteristiche dalle immagini, dalle più semplici alle più complesse.

3.6 R-CNN (Region-based CNN)

Le R-CNN [4], o Region-based Convolutional Neural Networks, sono una famiglia di algoritmi di computer vision che combinano le reti neurali convoluzionali (CNN) con la ricerca selettiva di regioni per rilevare oggetti all'interno di immagini. Questa famiglia di metodi ha segnato un punto di svolta nel riconoscimento di oggetti, in quanto ha fornito un approccio che può riconoscere sia la posizione che la classe di oggetti in un'immagine con un'elevata precisione. Nel dettaglio, il processo R-CNN si articola nelle seguenti fasi:

3.6.1 Selezione delle regioni

La prima fase nel processo R-CNN è la selezione delle regioni che potrebbero contenere oggetti. Questo passaggio riduce lo spazio di ricerca da tutta l'immagine a regioni probabili che contengono oggetti di interesse.

1. **Segmentazione**: Inizia con la segmentazione dell'immagine in un insieme di piccole regioni candidate, comunemente dette superpixel. Questi superpixel sono raggruppati sulla base della similitudine del colore, della texture e/o della posizione, per formare regioni più grandi che hanno maggiori probabilità di corrispondere a interi oggetti.
2. **Gerarchia**: La ricerca selettiva applica una gerarchia di raggruppamenti che unisce questi superpixel in regioni sempre più grandi. Il processo è iterativo e ad ogni passo, i raggruppamenti più simili vengono fusi insieme.
3. **Strategie di Fusione**: Vengono impiegate diverse strategie per decidere quali regioni fondere, come la similitudine di colore, la texture, la dimensione, la forma e la "buona continuità" che preferisce unire i superpixel adiacenti che non introducono una grande variazione nella direzione del contorno.
4. **Diversità delle Proposte**: Questo processo produce circa 2000 regioni candidate (proposte di regioni) per immagine.

3.6.2 Estrazione delle Caratteristiche

Una volta identificate le regioni di interesse, ogni regione viene processata per estrarre un vettore di caratteristiche utile per la classificazione.

- **Vettore multi-dimensionale**: Un vettore di caratteristiche 4096-dimensionale viene estratto da ogni proposta di regione.
- **Ridimensionamento**: Ogni regione proposta viene ridimensionata a una dimensione fissa in modo che possa essere processata da una CNN. Questo è essenziale perché le CNN richiedono un input di dimensione uniforme.
- **Struttura della CNN**: Le regioni ridimensionate vengono passate attraverso una serie di strati convoluzionali, strati di pooling e, infine, strati completamente connessi. Ogni strato convoluzionale applica diversi filtri all'input e cattura varie caratteristiche visive.
- **Training della CNN**: La CNN può essere pre-addestrata su un grande set di dati (come ImageNet) per rilevare caratteristiche generiche e poi fine-tuned sul set di dati specifico per l'applicazione desiderata.
- **Output della CNN**: L'output della CNN è un vettore di caratteristiche per ogni regione, che cattura informazioni visive cruciali necessarie per la classificazione.

3.6.3 Classificazione

Dopo l'estrazione delle caratteristiche, ciascuna regione viene classificata in una delle categorie di oggetti predefinite.

- **Allenamento dell'SVM**: Un classificatore SVM lineare è addestrato per ogni categoria di oggetto utilizzando il vettore di caratteristiche come input. Gli SVM sono adatti per questo compito grazie alla loro efficacia nella gestione di dati ad alta dimensione e nella separazione di classi complesse.
- **Score di Classificazione**: Ogni regione riceve uno score di classificazione per ogni categoria di oggetto, e la categoria con lo score più alto (a patto che superi una soglia di confidenza) viene assegnata alla regione.

3.6.4 Regressione dei Bounding Box

L'ultima fase del processo R-CNN è la regressione dei bounding box per affinare la posizione e le dimensioni delle caselle di delimitazione delle regioni proposte.

- **Allenamento dei Regressori**: Per ogni categoria di oggetti, viene addestrato un regressore lineare sui vettori di caratteristiche per prevedere gli offset rispetto alle proposte di regioni originali. Questi offset sono utilizzati per correggere la posizione e le dimensioni del bounding box proposto.
- **Affinamento del Bounding Box**: Gli offset predetti vengono applicati alle coordinate del bounding box proposto per adattarlo meglio alla forma e alla posizione effettive dell'oggetto all'interno dell'immagine.

3.6.5 Conclusione

Le R-CNN ha mostrato come l'utilizzo di un modello pre-addestrato e successivamente fine-tuned su dati specifici aumenti la precisione del modello in presenza di dati limitati. Nello specifico per il dataset PASCAL VOC migliora le prestazioni di mAP dell'8% con una precisione mAP del 54% dopo il fine tuning. Tuttavia, a causa della loro natura frammentaria e della necessità di addestrare più modelli separati (SVM e regressori per ogni categoria di oggetto), le R-CNN erano piuttosto lente e computazionalmente dispendiose. Questo ha portato allo sviluppo di architetture successive come Fast R-CNN e Faster R-CNN, che hanno migliorato significativamente l'efficienza integrando e semplificando questi passaggi in un framework unificato end-to-end.

3.7 Fast R-CNN

Fast R-CNN [5] ha costruito sul successo dell'originale R-CNN e ha risolto alcune delle sue principali inefficienze, migliorando la velocità e la semplicità del training dei modelli per la rilevazione degli oggetti.

3.7.1 Architettura di Fast R-CNN

L'architettura di Fast R-CNN segue un approccio unificato per la classificazione e la localizzazione degli oggetti (al contrario di R-CNN), ed è composta dai seguenti elementi chiave:

1. **Input di Rete**: La rete riceve come input un'immagine completa e un insieme di proposte di oggetti.

2. **Strati di Convoluzione e Max Pooling**: L'immagine è prima elaborata attraverso strati convoluzionali e di max pooling per produrre una mappa delle caratteristiche convoluzionale.

3. **Strato di RoI Pooling**: Il RoI Pool è un componente fondamentale che consente al Fast R-CNN di elaborare input di dimensioni variabili attraverso una rete convoluzionale che richiede dimensioni fisse di output. Il RoI Pooling funziona seguendo i seguenti passaggi:

   - **Proiezione delle Proposte**: Le regioni di interesse (ROI), che sono i bounding box proposti dalla fase di selezione delle regioni, vengono proiettate sulle mappe di attivazione corrispondenti dell'immagine di input elaborata dalla CNN.
   - **Pooling**: Per ogni ROI, l'operazione di RoI Pool divide la regione proiettata in una griglia fissa (ad esempio, 7x7 celle) e applica un'operazione di max pooling in ciascuna cella per produrre un output di dimensione fissa, indipendentemente dalle dimensioni originali della ROI.

4. **Strati Completamente Connessi**: I vettori di caratteristiche estratti sono poi alimentati in una serie di strati completamente connessi.

5. **Strati di Output**: L'architettura si suddivide in due strati di output "fratelli":

   - Uno strato che produce stime di probabilità softmax per la classificazione di oggetti più una classe "background".
   - Un altro strato che emette quattro numeri reali per ciascuna delle K classi di oggetti, codificando le posizioni raffinate dei bounding box.

R-CNN utilizza reti pre-addestrate da ImageNet, con trasformazioni specifiche per adattarle a Fast R-CNN. Include la sostituzione dell'ultimo strato di max pooling con uno strato di RoI pooling e la modifica degli ultimi strati completamente connessi e softmax.
Si utilizzano minibatch composti da N immagini e R/N RoI per ogni immagine, condividendo la computazione e la memoria nei passaggi in avanti e all'indietro.
Si utilizza una funzione di perdita multi-task per l'addestramento. Questa loss function è una combinazione lineare di due funzioni di perdita separate: una per la classificazione (perdita logaritmica) e una per la regressione dei bounding box (perdita L1 o L2 sui quattro parametri spaziali del box). L'obiettivo dell'addestramento è minimizzare una combinazione ponderata di queste due perdite.

3.7.2 Training

1. **Pre-processing**: Il training inizia con il pre-processing delle immagini di input, durante il quale vengono generate proposte di regioni di interesse.

2. **Forward Pass**: Successivamente, si esegue un forward pass dell'intera immagine attraverso la rete neurale convoluzionale (CNN). In questa fase, l'immagine viene processata una sola volta per produrre mappe di attivazione, che rappresentano le caratteristiche salienti dell'immagine.

3. **RoIPooling**: Viene poi utilizzata l'operazione di RoIPooling per estrarre le caratteristiche da ciascuna delle proposte di regione. Questo passaggio trasforma le caratteristiche estratte in una dimensione fissa, necessaria per il successivo processo di classificazione e regressione.

4. **Classificazione e Regressione**: Le caratteristiche estratte vengono passate attraverso layer completamente connessi, che producono score di classificazione per l'identificazione degli oggetti e offset di bounding box per la loro localizzazione.
5. **Backward Pass**: Il backward pass implica l'aggiornamento dei pesi della rete basandosi sui gradienti derivati dalla funzione di perdita.

3.7.3 Inferenza

L'inferenza in una rete Fast R-CNN addestrata consiste principalmente nell'esecuzione di un passaggio in avanti, con l'assunzione che le proposte di oggetti siano già state pre-calcolate. Di seguito sono descritte le fasi principali di questo processo:

1. **Input della Rete**: La rete riceve come input un'immagine (o una piramide di immagini, che è una lista di immagini) e una lista di R proposte di oggetti. Nel contesto di test, R è generalmente circa 2000, sebbene possano essere considerati casi con un numero maggiore di proposte.
2. **Assegnazione della Scala alle RoI**: Nell'uso di una piramide di immagini, ogni Region of Interest (RoI) viene assegnata a una scala specifica. Questo è fatto in modo che l'area della RoI scalata sia il più vicino possibile a 224x224 pixel.
3. **Elaborazione delle RoI**: Per ogni RoI di test, il passaggio in avanti della rete produce una distribuzione di probabilità a posteriori per le varie classi e un insieme di offset predetti per le bounding box, con ogni classe che ottiene la propria bounding box raffinata.
4. **Assegnazione della Confidenza di Rilevamento**: Viene assegnata una confidenza di rilevamento a ogni RoI per ciascuna classe di oggetti. Questo viene fatto utilizzando la probabilità stimata $Pr(\text{classe} = k|\text{RoI})$.
5. **Soppressione dei Non Massimi**: Successivamente, viene eseguita la soppressione dei non massimi in modo indipendente per ciascuna classe utilizzando l'algoritmo e le impostazioni di R-CNN.
6. **Ottimizzazione con SVD Troncata**: A differenza della classificazione dell'intera immagine, nella rilevazione l'elaborazione di un gran numero di RoI rende gli strati completamente connessi particolarmente onerosi in termini di tempo. Qui interviene la SVD troncata (Decomposizione ai Valori Singolari).
   In questo metodo, uno strato parametrizzato da una matrice di pesi W è fattorizzato usando la SVD, risultando in $W = U\Sigma_t V^\top$, dove U e V sono matrici contenenti i vettori singolari sinistri e destri, e Σ_t è una matrice diagonale con i valori singolari.
   Questa tecnica riduce il numero di parametri e accelera significativamente la rete, soprattutto quando il numero di RoI è elevato.

Limitazioni e Miglioramenti Successivi

Nonostante i miglioramenti, il Fast R-CNN ha ancora alcune limitazioni:

- Dipendenza dalla qualità delle proposte di regioni.
- Tempi di inferenza relativamente lunghi dovuti alla necessità di processare multiple proposte per immagine.
- Complessità nell'addestramento se si utilizzano dataset molto grandi o si rilevano un gran numero di oggetti di differenti dimensioni.

3.8 Faster R-CNN

Faster R-CNN [6] aveva migliorato significativamente l'efficienza del modello R-CNN, ma la generazione di proposte di regioni rimaneva un collo di bottiglia significativo. Faster R-CNN ha affrontato direttamente questo problema integrando completamente la generazione di proposte di regioni con il processo di estrazione delle caratteristiche. Si è mostrato che calcolare le proposte con una rete neurale convoluzionale profonda, porta a una soluzione elegante ed efficace in cui il calcolo delle proposte è quasi privo di costi dato il calcolo della rete di rilevamento. A tal fine, si introducono nuove Reti di Proposta di Regioni (RPN) che condividono strati convoluzionali con reti di rilevamento oggetti all'avanguardia.
Il sistema di rilevamento oggetti di Faster R-CNN è composto da due moduli. Il primo modulo è una rete completamente convoluzionale che propone regioni, mentre il secondo modulo è il rilevatore Fast R-CNN che utilizza le regioni proposte. L'intero sistema è composto dall'unificazione di questi due moduli.

3.8.1 Region Proposal Network (RPN)

L'RPN è una rete convoluzionale che scorre su tutta l'immagine e trova le regioni che potrebbero contenere oggetti. Funziona in parallelo con la rete che estrae le caratteristiche per la classificazione e localizzazione degli oggetti, migliorando l'efficienza e riducendo i tempi di calcolo. Infatti, la stessa mappa di attivazione generata dalla CNN base viene utilizzata sia per la previsione delle proposte di regione tramite l'RPN sia per l'estrazione delle caratteristiche successive. Poiché il nostro obiettivo finale è condividere la computazione con una rete di rilevamento di oggetti Fast R-CNN, assumiamo che entrambe le reti condividano un insieme comune di strati convoluzionali. La RPN nel sistema Faster R-CNN funziona seguendo questi passaggi:

1. La RPN prende in input un'immagine di qualsiasi dimensione e scorre una piccola rete sulla mappa delle caratteristiche convoluzionali prodotte dall'ultimo strato convoluzionale condiviso.

2. Una finestra spaziale di dimensioni n x n (di solito 3 x 3) viene utilizzata per scorrere sulla mappa delle caratteristiche. Ogni finestra scorrevole viene mappata in una caratteristica di dimensione inferiore.

3. Questa caratteristica ridotta viene poi alimentata in due strati completamente connessi fratelli: uno per la regressione delle bounding box e uno per la classificazione delle bounding box.

4. In ogni posizione della finestra scorrevole, la RPN prevede più proposte di regioni, con il numero massimo di proposte per posizione indicato con k. Gli ancoraggi, bounding box di riferimento prefissati che sono centrate sulla finestra scorrevole in questione, vengono utilizzati per parametrizzare le proposte rispetto a k bounding box di riferimento. Sono utilizzati diversi rapporti di aspetto e scale per generare queste proposte.

5. Per ogni proposta di regione, vengono calcolati i punteggi di oggettività (probabilità di oggetto o non oggetto) e le coordinate delle bounding box. Lo strato di regressione ha 4k uscite che codificano le coordinate di k bounding box, e lo strato di classificazione produce 2k punteggi.

6. La RPN è progettata per essere invariante rispetto alla traslazione, sia per quanto riguarda le ancore che le funzioni che calcolano le proposte rispetto alle ancore.

7. Per addestrare la RPN, si minimizza una funzione di perdita che è una combinazione di perdita di classificazione e perdita di regressione, note come perdite multi-task.

   (a) La perdita di classificazione (Lcls) è calcolata come log loss su due classi (oggetto vs. non oggetto).

   (b) La perdita di regressione (Lreg) utilizza una funzione di perdita robusta (smooth L1) per confrontare le coordinate parametrizzate della bounding box predetta con quella di ground truth. È attivata solo per gli ancoraggi positive.

   Per assegnare le etichette alle ancore durante l'addestramento delle Reti di Proposta di Regioni (RPN), viene utilizzato il criterio di Intersection-over-Union (IoU). Un'etichetta positiva viene assegnata alle ancore che hanno l'IoU più alto con una bounding box di ground truth o un IoU superiore a 0.7 con qualsiasi bounding box di ground truth. Un'etichetta negativa viene assegnata a un'ancora non positiva se il suo rapporto IoU è inferiore a 0.3 per tutte le bounding box di ground truth.

3.8.2 Passaggio alla Fast R-CNN

Dopo che l'RPN ha generato proposte di regioni, queste vengono utilizzate dal Fast R-CNN per l'estrazione e la classificazione delle caratteristiche. Questa sinergia consente all'RPN di beneficiare dell'addestramento della rete CNN e viceversa, consentendo a entrambi di migliorare simultaneamente le loro prestazioni attraverso un approccio di apprendimento condiviso. L'introduzione dell'RPN ha significativamente migliorato il tempo di calcolo necessario per la generazione di regioni di interesse rispetto ai metodi precedenti come l'Selective Search usato in R-CNN e Fast R-CNN.

3.8.3 Considerazioni Finali

Faster R-CNN ha non solo migliorato l'accuratezza e l'efficienza ma ha anche fornito un framework flessibile che può essere adattato a nuovi problemi e set di dati. Le R-CNN e le loro evoluzioni continueranno a influenzare lo sviluppo di sistemi di visione artificiale più avanzati. Mentre le architetture più recenti come YOLO e SSD offrono approcci alternativi per la rilevazione di oggetti in tempo reale, Faster R-CNN rimane un modello di riferimento per le situazioni in cui la precisione e la robustezza sono di primaria importanza.

3.9 YOLO - You Only Look Once

YOLO (You Only Look Once) [7] ha rivoluzionato il rilevamento degli oggetti tramite un approccio unificato che si distingue nettamente dai metodi tradizionali. Invece di adattare i classificatori per il rilevamento, YOLO tratta il rilevamento degli oggetti come un problema di regressione, consentendo di prevedere direttamente dalle immagini intere sia i riquadri di delimitazione (bounding boxes) che le probabilità associate alle classi in un'unica valutazione. Questa innovativa architettura unisce tutti i componenti del rilevamento degli oggetti in una singola rete neurale.
Al contrario dei metodi precedenti, che analizzano parti dell'immagine separatamente, YOLO considera l'intera immagine per predire simultaneamente ogni riquadro di delimitazione e le probabilità di classe per ciascuna casella attraverso tutte le classi. Questo approccio consente alla rete di formulare una valutazione globale dell'intera immagine e di tutti gli oggetti presenti. Inoltre, ogni cella della griglia in cui l'immagine viene divisa è responsabile del rilevamento degli oggetti il cui centro cade all'interno di quella cella, permettendo così una localizzazione più accurata.

YOLO predice un numero definito di bounding boxes per ogni cella della griglia e assegna un punteggio di confidenza a ciascuna casella, che indica la probabilità che la casella contenga un oggetto e la precisione della previsione del riquadro di delimitazione.
Ogni riquadro di delimitazione è definito da cinque attributi: le coordinate (x, y) che rappresentano il centro del riquadro rispetto ai confini della cella della griglia, la larghezza e l'altezza relative all'intera immagine, e il punteggio di confidenza che riflette l'Intersection Over Union (IOU) tra il riquadro predetto e il riquadro di ground truth.
Inoltre, ogni cella della griglia prevede le probabilità condizionali di classe per gli oggetti, ulteriormente rafforzando l'accuratezza delle previsioni della rete.

3.9.1 Architettura

La rete si ispira al modello GoogLeNet per la classificazione delle immagini, con 24 strati convoluzionali seguiti da 2 strati completamente connessi. A differenza di GoogLeNet, YOLO utilizza strati di riduzione 1 × 1 seguiti da strati convoluzionali 3 × 3. Esiste anche una versione "Fast YOLO" progettata per essere ancora più veloce, con meno strati convoluzionali e filtri.

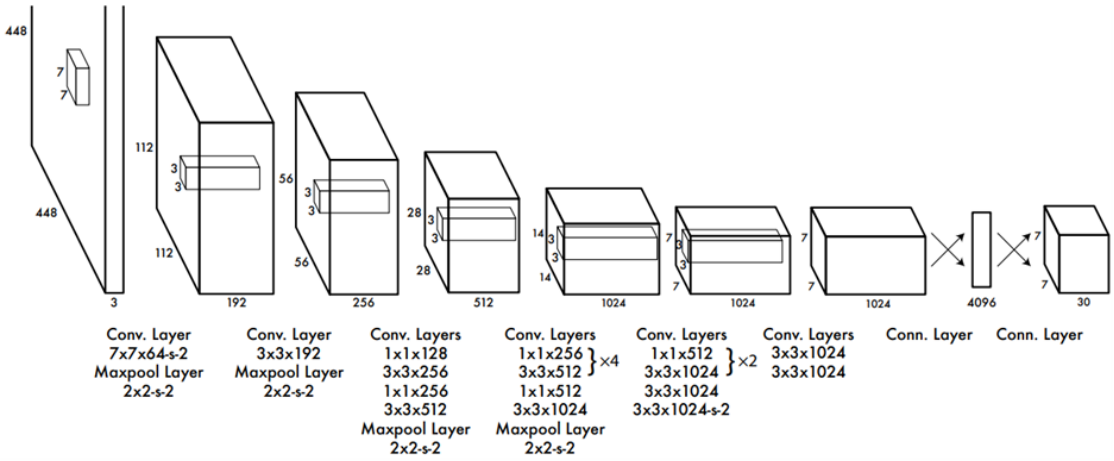

Figure 3.1: Architettura YOLO

3.9.2 Fase di training

- **Pre-Training**: Gli strati convoluzionali sono pre-addestrati sul dataset ImageNet. Vengono utilizzati i primi 20 strati convoluzionali seguiti da un livello di pooling medio e un livello completamente connesso. Questo passaggio permette alla rete di acquisire una conoscenza generale sulla classificazione delle immagini. Per tutte le operazioni di training e inferenza, YOLOv1 sfrutta Darknet, un framework open source per l'apprendimento profondo.
- **Modifica della Rete per il Rilevamento**: Successivamente, il modello viene convertito per eseguire il rilevamento. Questo implica l'aggiunta di ulteriori strati convoluzionali e completamente connessi con pesi inizializzati casualmente, migliorando così le prestazioni del rilevamento. La risoluzione di input della rete viene aumentata per una migliore cattura delle informazioni visive dettagliate. L'ultimo strato della rete prevede sia le probabilità delle classi sia le coordinate dei riquadri di delimitazione. Le dimensioni dei riquadri di

delimitazione vengono normalizzate rispetto alle dimensioni dell'immagine per garantire che rientrino in un intervallo tra 0 e 1. Anche le coordinate x e y dei riquadri di delimitazione sono parametrizzate come offset relativi alla posizione di una particolare cella della griglia. Per l'ultimo strato viene utilizzata una funzione di attivazione lineare, mentre tutti gli altri strati usano una versione leaky della funzione rettificata lineare.

$$\varphi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{altrimenti} \end{cases} \tag{3.1}$$

La rete ottimizza l'errore quadratico medio nell'output.

- **Sfide e Considerazioni nel Training**: La funzione di perdita di YOLOv1, l'errore quadratico medio, non discrimina tra errori di localizzazione e errori di classificazione. Questo potrebbe non essere l'ideale, dato che la rete dovrebbe idealmente concentrarsi di più sulla precisione delle localizzazioni rispetto alla classificazione per alcuni tipi di applicazioni. Molte celle della griglia in ogni immagine potrebbero non contenere oggetti, il che spinge i punteggi di "confidenza" di queste celle verso zero, sovrastando potenzialmente il gradiente dalle celle che contengono oggetti. Questo può portare a instabilità del modello e a problemi durante il training iniziale

3.9.3 Fase di testing

Durante il testing, YOLOv1 richiede una sola valutazione della rete per immagine. Questa caratteristica è fondamentale per la sua efficienza, distinguendola dai metodi basati sui classificatori che necessitano di molteplici valutazioni. In pratica, questo significa che una volta che l'immagine è alimentata nella rete, YOLOv1 è in grado di rilevare tutti gli oggetti presenti in quella singola passata, senza la necessità di ulteriori valutazioni o iterazioni. La rete prevede 98 riquadri di delimitazione per immagine, insieme alle probabilità di classe per ciascuna casella. Questo numero elevato di riquadri di delimitazione consente a YOLOv1 di coprire ampiamente l'immagine, aumentando le possibilità di catturare correttamente tutti gli oggetti presenti. Il design della griglia impiegato in YOLOv1 impone una diversità spaziale nelle previsioni dei riquadri di delimitazione. In generale, è chiaro in quale cella della griglia cada un oggetto, e la rete di solito prevede un solo riquadro per ciascun oggetto. Tuttavia, oggetti di grandi dimensioni o oggetti vicini ai confini di più celle possono essere localizzati correttamente da più celle. Per gestire i casi in cui più celle della griglia prevedono lo stesso oggetto (causando rilevamenti multipli), YOLOv1 utilizza la soppressione non massimale. Questo processo aiuta a filtrare le previsioni ridondanti o sovrapposte, mantenendo solo i riquadri di delimitazione più probabili.

3.9.4 Limitazioni

YOLO impone forti vincoli spaziali sulle previsioni dei riquadri di delimitazione, limitando il numero di oggetti vicini che il modello può prevedere. Lotta con piccoli oggetti che appaiono in gruppi e con oggetti in proporzioni o configurazioni nuove o insolite. Inoltre, a causa di più strati di riduzione dimensionale, utilizza caratteristiche relativamente grossolane per prevedere i riquadri di delimitazione.

3.10 SSD - Single Shot MultiBox Detector

Le Single Shot MultiBox Detector (SSD) [8] rappresentano una metodologia alternativa per il rilevamento di oggetti nelle immagini che si differenzia dalle architetture R-CNN per vari aspetti,

compresi il design e la prestazione complessiva.
Progettate per essere più veloci durante l'inferenza e semplificare la pipeline di rilevamento degli oggetti, le SSD sono particolarmente attraenti per applicazioni in tempo reale che richiedono la velocità oltre che la precisione. Utilizza un singolo passaggio attraverso la rete neurale per identificare le posizioni degli oggetti nelle immagini. Come per YOLO differisce dalle reti basate su R-CNN, che utilizzano due passaggi: uno per generare regioni di proposta e uno per classificare queste proposte. L'approccio SSD si basa su una rete feed-forward che produce una collezione di dimensioni fisse di bounding boxes e punteggi per la presenza di istanze di classe oggetto in quei box, seguita da un passo di soppressione non massima per produrre i rilevamenti finali. Per le ipotesi di Bounding Boxes, al contrario di altri approcci, non avviene il ricampionamento dei pixel o delle caratteristiche.
Questo approccio si traduce in un significativo miglioramento della velocità per il rilevamento di alta accuratezza (59 FPS con mAP 74.3% sul test VOC2007 contro 7 FPS con mAP 73.2% di Faster R-CNN o 45 FPS con mAP 63.4% di YOLOv1).

3.10.1 Funzionamento step-by-step

1. **Pre-elaborazione dell'Immagine**: Prima di essere processata dall'SSD, l'immagine di input è di solito ridimensionata a una dimensione fissa (e.g., 300x300 o 512x512 pixel). Questo ridimensionamento è necessario per assicurare che le caratteristiche estratte siano consistenti in termini di scala.

2. **Rete convoluzionale base**: L'architettura inizia con una base di rete convoluzionale pre-addestrata, come il VGG16 o il ResNet, che funge da estrattore di caratteristiche primarie. Questa base di rete sfrutta la capacità comprovata di tali architetture nel catturare caratteristiche complesse di immagini e viene solitamente modificata rimuovendo gli strati finali specifici per la classificazione.

3. **Mappe delle caratteristiche multi-scala**: Le SSD differiscono dai tradizionali metodi di rilevamento degli oggetti usando più strati convoluzionali per creare una serie di mappe di caratteristiche a scale diverse. Ogni mappa di caratteristiche successiva viene creata applicando strati convoluzionali con un passo (stride) maggiore, che riduce la dimensione spaziale della mappa. Questo approccio permette di catturare oggetti a diverse dimensioni e aspetti senza la necessità di analizzare molteplici immagini ridimensionate o di fare un uso intensivo di ancore (anchor boxes) a diverse scale e rapporti di aspetto.

4. **Predittori convoluzionali**: Una volta generata la gerarchia delle mappe di caratteristiche a diverse scale, la SSD fa uso di una serie di predittori convoluzionali. Questi predittori sono piccoli filtri convoluzionali applicati a ogni mappa di caratteristiche per produrre un set di score per la classificazione e le coordinate del bounding box per la localizzazione. Ciascuna posizione nella mappa di caratteristiche può produrre uno score per ciascuna classe di oggetti che l'algoritmo è in grado di rilevare, oltre a fornire i quattro valori necessari per definire la posizione e le dimensioni del bounding box dell'oggetto.

5. **Box di default e rapporti d'aspetto**: Per ogni posizione delle mappe di caratteristiche, la SSD definisce un insieme di "default boxes" (scatole di riferimento), che hanno diverse proporzioni e scale. Queste scatole sono analoghe alle ancore utilizzate nelle Faster R-CNN ma sono distribuite su molteplici mappe di caratteristiche piuttosto che essere concentrate in un'unica mappa.

6. **Calcolo della Loss**: Durante la fase di addestramento, la SSD calcola la perdita (loss) combinando la perdita per la localizzazione dei bounding box (di solito la Smooth L1 Loss) e la perdita per la classificazione (Cross Entropy Loss). La funzione di perdita combinata consente di ottimizzare contemporaneamente la rete per la precisione della localizzazione e per la correttezza della classificazione.

7. **Matching Strategy**: La SSD deve determinare quali default boxes corrispondano a un ground truth box in fase di allenamento. Ciò avviene mediante una strategia di matching che assegna i ground truth boxes alle default boxes più vicine basandosi sulla metrica IoU (Intersection over Union), assicurando che ciascun ground truth box abbia almeno una default box corrispondente.

8. **Hard Negative Mining**: Poiché la maggior parte delle default boxes non conterrà oggetti, questo potrebbe portare a un grande sbilanciamento tra le classi positive e negative durante la fase di allenamento. La SSD affronta questo problema attraverso una tecnica chiamata hard negative mining, che consiste nel selezionare default boxes negative che hanno la maggiore confidenza di errore (le predizioni sbagliate con alta confidenza) e addestrare la rete su di esse per migliorare la robustezza del modello.

9. **Augmentation dei dati**: Per rendere il modello più robusto a varie dimensioni e forme di oggetti in ingresso, ogni immagine di addestramento viene campionata casualmente con una delle seguenti opzioni:

   (a) Usare l'intera immagine di input originale.

   (b) Campionare una patch in modo tale che la sovrapposizione jaccard minima con gli oggetti sia 0,1, 0,3, 0,5, 0,7 o 0,9.

   (c) Campionare casualmente una patch.

10. **Non-Maxima Suppression**

3.10.2 Confronto con Faster R-CNN

Velocità

- **Faster R-CNN**: La generazione di proposte da parte dell'RPN è veloce ma non istantanea, e l'intero processo rimane relativamente computazionalmente oneroso.

- **SSD**: Elimina il bisogno di una fase di proposta, facendo tutto in un singolo passaggio, rendendolo significativamente più veloce.

Semplicità

- **Faster R-CNN**: Richiede l'allenamento di un RPN e poi l'allenamento di una rete separata per la classificazione e la regressione dei bounding box. Sebbene possa essere addestrato end-to-end, il processo è più complesso.

- **SSD**: Utilizza un'architettura più diretta senza bisogno di una rete separata per la proposta delle regioni, il che lo rende più semplice da addestrare e ottimizzare.

Precisione su Oggetti di Dimensioni Diverse

- **Faster R-CNN**: È molto preciso nel rilevamento di oggetti, soprattutto grazie all'uso di RPN che può essere sintonizzato per identificare oggetti con alta varianza in termini di dimensioni e forma.

- **SSD**: Utilizza strati multipli per rilevare oggetti a diverse scale, ma può non essere altrettanto accurato nel rilevamento di oggetti molto piccoli. Tuttavia, le varianti recenti di SSD hanno introdotto miglioramenti in questo aspetto.

Complessità Computazionale

- **Faster R-CNN**: Richiede maggiori risorse computazionali per l'analisi di immagini a causa del suo design in due fasi e del fatto che deve gestire due reti separate.

- **SSD**: È progettato per essere più leggero e richiede meno potenza computazionale per l'inferenza grazie al suo approccio di rilevamento in un solo passaggio.

Applicabilità

- **Faster R-CNN**: É spesso preferito per applicazioni dove la precisione del rilevamento è critica, come nei casi in cui è necessario rilevare oggetti piccoli o in scene congestionate.

- **SSD**: È più adatto per applicazioni in tempo reale, dove la velocità è più importante della precisione assoluta, come nel monitoraggio video o in sistemi di assistenza alla guida.

3.11 Versioni YOLO a confronto

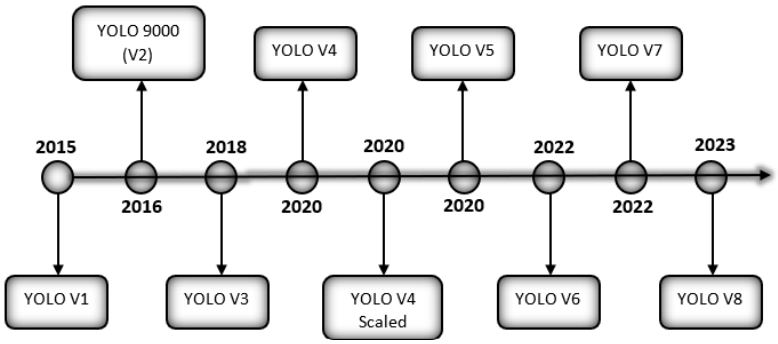

Figure 3.2: YOLO timeline

Dall'introduzione del primo modello YOLO (You Only Look Once) nel 2015, il panorama del Deep Learning si è evoluto in maniera significativa, testimoniando avanzamenti esponenziali sia nella teoria che nell'applicazione pratica. In poco più di un decennio, si è assistito al rilascio di una serie di versioni di YOLO, ognuna portatrice di innovazioni e miglioramenti rispetto alle precedenti.

3.11.1 YOLO 9000

YOLO9000, o YOLOv2 [9], è un avanzamento significativo nel rilevamento degli oggetti in tempo reale, capace di identificare oltre 9000 categorie di oggetti. Con l'introduzione di un metodo di training multi-scala, YOLO9000 offre una combinazione ottimale di velocità e precisione. Per esempio, a 67 FPS, ottiene un mAP del 76.8 su VOC 2007, e a 40 FPS raggiunge un mAP di 78.6, superando altri sistemi all'avanguardia come Faster R-CNN e SSD, pur rimanendo nettamente più veloce. Il suo algoritmo di training congiunto migliora l'apprendimento utilizzando sia dati di rilevamento che di classificazione, consentendo al modello di prevedere classi di oggetti anche senza dati di rilevamento etichettati, grazie all'uso combinato dei dataset di COCO e ImageNet.

- **Better**: É un avanzamento sostanziale rispetto al suo predecessore, che ottimizza la localizzazione e il recall pur mantenendo alta l'accuratezza di classificazione. Introduce la normalizzazione batch per una migliore convergenza e un training iniziale a risoluzione più elevata per adattarsi meglio all'input di rilevamento. L'adozione di anchor boxes con priors basati su k-means migliora la precisione delle previsioni e il "passthrough layer" per localizzare oggetti più piccoli.

- **Faster**: É stato progettato per essere estremamente veloce, superando i tradizionali modelli basati su VGG-16 grazie a una rete personalizzata ispirata a Googlenet, con meno operazioni richieste e un'accuratezza paragonabile. Introduce Darknet-19, una base di classificazione leggera e potente che integra tecniche da VGG e NIN, ottimizzata per la velocità con l'uso della normalizzazione batch. Darknet-19, addestrato su ImageNet, migliora significativamente l'accuratezza con il fine-tuning a risoluzione maggiore.

- **Stronger**: Mira a creare un modello veloce e capace di rilevare una vasta gamma di categorie. Utilizza un approccio unificato che combina il rilevamento e la classificazione, sfruttando dataset etichettati per entrambe le funzioni. Durante l'addestramento, miscela immagini da dataset di rilevamento e classificazione, applicando una funzione di perdita specifica per ciascuna. Introduce un metodo di classificazione gerarchica basato su WordNet, che permette di calcolare le probabilità assolute di un nodo attraverso un albero di concetti. Il modello Darknet-19 addestrato su WordTree, amplia il raggio d'azione da 1000 a 9418 classi.

3.11.2 YOLO V3

YOLOv3 [10] fa predizioni su tre diverse scale, usando un concetto simile alle reti piramidali di feature. Il modello estrae le feature da queste scale e utilizza più layer convoluzionali per elaborare le predizioni. Dalla nostra estrazione di caratteristiche di base aggiungiamo diversi strati convoluzionali. L'ultimo di questi predice un tensore 3D che codifica le predizioni di bounding box, oggettività e classe. YOLOv3 è significativamente più veloce e mantiene un'elevata accuratezza. Può eseguire inferenze in 22 ms con un'accuratezza di 28.2 mAP a 320x320 pixel, che è paragonabile all'accuratezza di SSD ma tre volte più veloce. Grazie alle predizioni su più scale, YOLOv3 gestisce meglio oggetti di diverse dimensioni, in particolare quelli più piccoli.
YOLOv3 utilizza una nuova rete per l'estrazione delle feature chiamata Darknet-53, che combina layer convoluzionali 3x3 e 1x1 con connessioni shortcut. Questo gli permette di essere più potente di Darknet-19 (usato in YOLOv2) e più efficiente rispetto si modelli ResNet.
Mentre YOLOv3 è efficace nel rilevare oggetti, ha difficoltà nel perfetto allineamento delle bounding boxes con gli oggetti. Sebbene migliori nel rilevamento di oggetti piccoli, YOLOv3 ha prestazioni leggermente inferiori per oggetti di medie e grandi dimensioni.

3.11.3 YOLO V4

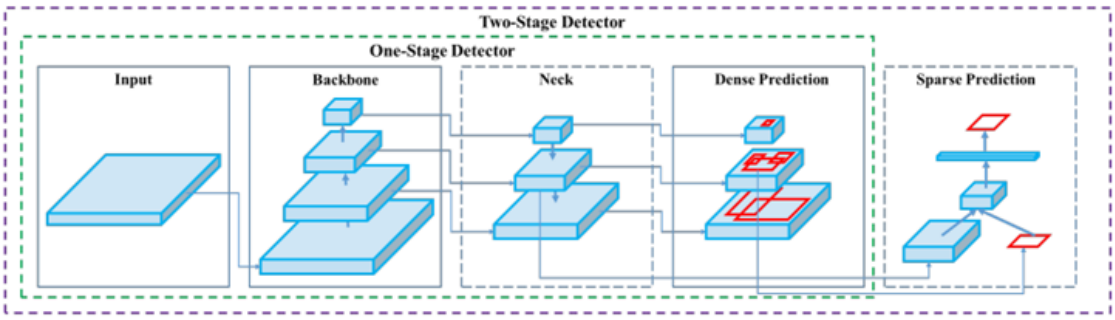

Figure 3.3: Architettura YOLO V4

YOLOv4 [11] mira a creare un rilevatore di oggetti che funzioni in tempo reale su una GPU convenzionale, adattandosi sia ai sistemi di raccomandazione che alla gestione dei processi autonoma, riducendo la necessità di input umani.
YOLOv4 è composto da tre parti principali:

- **Backbone**: Il nucleo principale per l'estrazione delle caratteristiche, pre-addestrato su ImageNet. YOLOv4 utilizza CSPDarknet53 come backbone.
- **Neck**: Una serie di strati che raccolgono mappe delle caratteristiche da diversi stadi del backbone. YOLOv4 impiega SPP (Spatial Pyramid Pooling) e PAN (Path Aggregation Network) come parte del suo Neck.
- **Head**: Utilizzato per la previsione delle classi e dei bounding box degli oggetti. YOLOv4 adotta la head di YOLOv3.

Bag of Freebies (BoF) e Bag of Specials (BoS)

Questi concetti sono centrali in YOLOv4.

- **BoF**: Tecniche che migliorano la precisione durante l'addestramento senza aumentare il costo computazionale durante l'inferenza. Esempi includono l'augmentation dei dati e la funzione obiettivo della regressione della "Bounding Box" IoU loss che tiene conto della copertura dell'area della BBox prevista e dell'area della BBox ground truth.
- **BoS**: Moduli plugin e metodi post elaborazione che aumentano leggermente il costo computazionale ma migliorano significativamente l'accuratezza. In generale, questi moduli plugin servono per potenziare determinati attributi in un modello, come l'ingrandimento del campo recettivo, l'introduzione di meccanismi di attenzione o il potenziamento della capacità di integrazione delle caratteristiche, ecc., mentre la post-elaborazione è un metodo per selezionare i risultati delle previsioni del modello. Includono funzioni di attivazione come Mish e metodi di post-processing come DIoU-NMS.

Metodologia e Selezione dell'Architettura

YOLOv4 è progettato per bilanciare vari fattori cruciali nella rilevazione di oggetti, quali la risoluzione di input, il numero di livelli convoluzionali, la quantità di parametri e i filtri, per

ottimizzare sia per la classificazione che per la rilevazione di oggetti. Mentre CSPResNext50 mostra prestazioni eccellenti nella classificazione su ImageNet grazie alla sua architettura, il CSPDarknet53 con un campo recettivo maggiore e più parametri si rivela superiore nella rilevazione di oggetti su MS COCO, specialmente per oggetti più piccoli e per l'identificazione di diversi oggetti in un'unica immagine.
La configurazione di YOLOv4 include il backbone CSPDarknet53, il modulo SPP per aumentare il campo recettivo, il neck PANet per l'aggregazione efficiente dei parametri, e la head basata su ancoraggio di YOLOv3.
Con l'obiettivo di migliorare l'accuratezza, si prevede di aggiungere nuove tecniche alla "Bag of Freebies" e di ottimizzare il rilevatore con data augmentation innovativa come Mosaic e Self-Adversarial Training, la selezione di iperparametri tramite algoritmi genetici, e miglioramenti ai metodi esistenti come:

- Modifica della Normalizzazione Batch (CmBN): Una versione modificata della normalizzazione batch (Cross mini-Batch Normalization), che raccoglie statistiche solo tra mini-batch all'interno di un singolo batch, piuttosto che sull'intero batch.
- Modifica del SAM (Spatial Attention Module): Il SAM è stato adattato da un'attenzione su scala spaziale a un'attenzione su scala punto.
- Modifica del PAN (Path Aggregation Network): Sostituzione della connessione shortcut con la concatenazione, per migliorare l'efficienza nella raccolta e aggregazione delle informazioni dai diversi livelli della rete.

3.11.4 YOLO V4 Scaled

YOLOv4 Scaled [12] impiega una strategia basata su CSP per costruire una rete scalabile. Questo permette di adattare il modello per reti di varie dimensioni, mantenendo alta velocità e precisione. Il CSPNet riduce parametri e calcoli, aumentando l'accuratezza e diminuendo il tempo di inferenza. Per i dispositivi meno potenti, si pone attenzione all'ottimizzazione delle risorse hardware, mentre per le GPU di fascia alta si mira a migliorare l'accuratezza senza compromettere la velocità. YOLOv4-CSP è ottimizzato per GPU generali, YOLOv4-tiny per dispositivi di fascia bassa e YOLOv4-large per GPU cloud, ognuno progettato per bilanciare specifiche esigenze di velocità, accuratezza e risorse computazionali.

3.11.5 YOLO V5

YOLOv5 [13] [14] è disponibile in quattro versioni principali: piccola (s), media (m), grande (l) ed extra grande (x), ognuna con tassi di precisione crescenti e diversi tempi di addestramento.
In seguito le modifiche architetturali apportate alla struttra di YOLOv5 rispetto al suo predecessore:

1. **Backbone e Neck**: Sebbene YOLOv4 e YOLOv5 condividano l'approccio di utilizzare CSP (Cross-Stage Partial connections) nel backbone e nel neck, YOLOv5 introduce una nuova struttura per il backbone basata su CSP-Darknet53, insieme a modifiche nel neck utilizzando strutture SPPF (Spatial Pyramid Pooling Fast) e una nuova versione di CSP-PAN. Questi aggiornamenti sono finalizzati a ottimizzare ulteriormente la raccolta e l'aggregazione delle caratteristiche.
2. **DenseNet**: YOLOv5 implementa il Bottleneck CSP (come in YOLOv4) e utilizza DenseNet come base per i modelli CSP, collegando i livelli nelle reti neurali convoluzionali. Questa

strategia migliora l'efficienza riducendo i gradienti duplicati e il numero di parametri e FLOPS.

Le altre parti della struttura come Backbone e Head rimangono pressochè invariate. Si continua a utilizzare componenti introdotti in YOLOv4 come CSP-Darknet53 e PA-Net.
YOLOv5 utilizza diverse **tecniche di augmentation** come augmentation con mosaico, copia-incolla, trasformazioni affini casuali, MixUp, albumentations, aumento HSV, riflessione orizzontale casuale.
YOLOv5 applica diverse **strategie di addestramento** come addestramento multiscala, AutoAnchor, programma di Riscaldamento e Cosine LR, EMA e addestramento a precisione mista, evoluzione degli iperparametri.
La **funzione di perdita** utilizzata è composta da:

1. Perdita per le Classi (BCE Loss).
2. Perdita per l'Oggettività (BCE Loss).
3. Perdita per la Localizzazione (CIoU Loss).

La funzione di perdita totale è una somma ponderata di queste tre perdite. Le perdite di oggettività dei tre strati di predizione (P3, P4, P5) vengono ponderate diversamente con pesi di $[4.0, 1.0, 0.4]$ rispettivamente.
Un aspetto importante di questa versione è stata la traduzione e il miglioramente del framework Darknet nel **framework PyTorch**.

3.11.6 YOLO V6

YOLOv6 [15] è una versione avanzata della serie YOLO, mirata a migliorare la velocità e l'accuratezza nella rilevazione di oggetti per applicazioni industriali. Il sistema è strutturato in diverse **componenti chiave**:

- **Backbone**: Introduce "EfficientRep", un backbone riparametrizzabile ed efficiente. Utilizza RepBlock per modelli piccoli, che si converte in pile di strati convoluzionali 3x3 con attivazione ReLU durante l'inferenza. Per modelli più grandi, impiega il CSPStackRep Block con connessione residua. Bilancia accuratezza e velocità, ottimizzando la densità computazionale e riducendo la latenza di inferenza.
- **Neck**: Utilizza una topologia PAN modificata, ribattezzata "Rep-PAN".
- **Head**: La head di rilevamento è "decoupled" ed efficiente, con un solo strato convoluzionale 3x3. Adotta un approccio "anchor-free" basato su punti di ancoraggio.
- **Assegnazione delle etichette**: YOLOv6 ha adottato TAL da TOOD, con una metrica unificata del punteggio di classificazione e della qualità prevista delle scatole.

YOLOv6 utilizza **funzioni di perdita** differenti:

1. **Perdita di Classificazione**: adozione di VariFocal Loss.
2. **Perdita di Regressione delle Bounding Box**: IoU-series Loss con varianti come GIoU e SIoU.
3. **Perdita di Probabilità**: utilizzo di Focal Loss per trattare le posizioni delle scatole.
4. **Perdita di Oggetti**: proposta in FCOS e utilizzata in YOLOX.

YOLOv6 estende il periodo di addestramento a 400 epoche, usa l'auto-distillazione usando la divergenza KL per ridurre le differenze tra le precisioni dei modelli e infine aggiunge un borgo grigio intorno alle immagini che aiuta a rilevare oggetti vicino al bordo dell'immagine.
Vengono implementate le tecniche di quantizzazione Post-Addestramento (PTQ) per accelerare il tempo di esecuzione, ottimizzazione di riparametrizzazione (RepOptimizer) per affrontare sfide nella quantizzazione nei modelli basati su riparametrizzazione, analisi di sensibilità per identificare le operazioni sensibili alla quantizzazione e infine quantizzazione consapevole dell'addestramento (QAT) con distillazione a livello di canale per migliorare ulteriormente le prestazioni della quantizzazione.

3.11.7 YOLO V7

L'architettura di YOLOv7 [16] è caratterizzata da un focus sulla progettazione di reti di aggregazione di layer efficienti estesi (E-ELAN) e sulla scalabilità del modello, specialmente per modelli basati sulla concatenazione:
YOLOv7 **ottimizza l'architettura** delle reti neurali enfatizzando l'efficienza dei parametri e la densità computazionale. Introduce la gestione avanzata del gradiente e dell'uso della memoria, con un'analisi approfondita del rapporto input/output e dell'impatto delle operazioni sugli elementi sulla velocità di inferenza. La struttura CSPVoVNet, derivata da VoVNet, migliora l'apprendimento e la velocità di inferenza.
L'**Extended-ELAN** (E-ELAN) viene introdotto per potenziare l'apprendimento senza alterare il percorso del gradiente, utilizzando convoluzioni di gruppo per espandere canali e cardinalità. La fusione di cardinalità, tramite permutazione e concatenazione, consente di mantenere invariato il numero di canali incrementando la fusione delle feature map.
Il modello YOLOv7 incorpora un **approccio di scalabilità** critico per adeguarsi alle varie necessità di velocità di inferenza, permettendo l'adattamento della larghezza, della profondità e della risoluzione del modello. L'analisi si concentra sull'effetto delle convoluzioni standard e di gruppo sul numero di parametri e calcoli durante il ridimensionamento. In architetture che utilizzano la concatenazione, il ridimensionamento della profondità altera la connettività dei layer di transizione, richiedendo un'analisi integrata dei fattori di scalabilità. Il metodo di scalabilità composita proposto adatta il fattore di larghezza nei layer di transizione in base alla variazione del canale di output durante lo scaling della profondità, preservando le caratteristiche originali del modello.
Tra i **"trainable bag-of-freebies"** (BoF) usati in YOLOv7 il Planned Re-parameterized Convolution (RepConv) che ottimizza l'architettura convoluzionale integrando convoluzioni 3x3, 1x1 e connessioni identiche, con la variante RepConvN che esclude connessioni identiche per preservare le connessioni residue e di concatenazione cruciali per la diversità dei gradienti.
Il metodo Coarse for Auxiliary and Fine for Lead Loss introduce una supervisione profonda e un innovativo sistema di assegnazione delle etichette per guidare l'apprendimento delle head ausiliarie e principali con etichette gerarchiche, ottimizzando la rilevazione. Ulteriori BoF includono l'implementazione della normalizzazione batch, l'integrazione della conoscenza implicita da YOLOR, e l'adozione del modello EMA per l'inferenza, arricchendo YOLOv7 con tecniche che ne aumentano l'efficacia senza gravare sulle risorse durante l'inferenza.

3.11.8 YOLO V8

YOLOv8 [17], implementato dalla società Ultralytics (come YOLOv5), rappresenta un'avanzamento significativo nell'ambito dei modelli di rilevamento degli oggetti dimostrando un'alta precisione nei benchmark COCO e Roboflow 100. Il modello medio YOLOv8m raggiunge un mAP (Mean

Average Precision) del 50,2% su COCO, superando le versioni precedenti come YOLOv5. Presenta un'interfaccia a riga di comando intuitiva e un pacchetto Python ben strutturato. Questo facilita l'addestramento del modello e offre un'esperienza di codifica più fluida. Una crescente comunità attorno a YOLOv8 offre supporto e orientamento, rendendo più agevole l'utilizzo del modello.

Architettura

A differenza delle versioni precedenti di YOLO, YOLOv8 predice direttamente il centro di un oggetto, riducendo il numero di previsioni di caselle. Questo migliora la velocità di Non-Maxima Suppression (NMS), una fase critica di post-elaborazione. Il primo layer di convoluzione 6x6 del blocco principale è sostituito da un 3x3, il blocco principale è stato modificato e C2f ha sostituito C3. Il modulo è riassunto nell'immagine qui sotto, dove "f" è il numero di feature, "e" è il tasso di espansione e CBS è un blocco composto da una Convoluzione, una BatchNorm e uno strato SiLU. In C2f, tutte le uscite del Bottleneck (un nome elegante per due convoluzioni 3x3 con connessioni residue) vengono concatenate, mentre in C3 viene utilizzata solo l'uscita dell'ultimo Bottleneck.

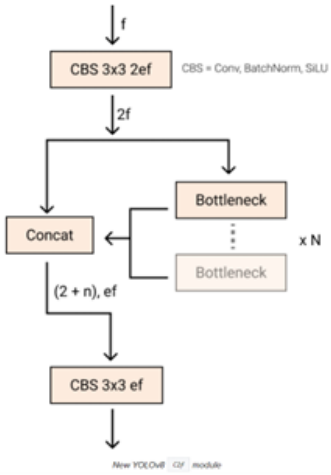

Figure 3.4: Architettura YOLO V8

Queste modifiche suggeriscono un ritorno verso l'architettura del Blocco ResNet definito nel 2015. Grazie alla concatenazione diretta delle feature senza uniformare le dimensioni dei canali, YOLOv8 riduce il conteggio dei parametri e le dimensioni complessive dei tensori. YOLOv8 utilizza una tecnica di aumentazione chiamata "mosaic augmentation. Questo comporta l'unione di quattro immagini, costringendo il modello a imparare oggetti in nuove posizioni, in parziale occlusione e contro diversi pixel circostanti. Tuttavia, è stato osservato che disattivarla nelle ultime dieci epoche di addestramento migliora le prestazioni.

Prestazioni

Dal punto di vista delle prestazioni, YOLOv8 ha dimostrato di essere superiore alle versioni precedenti, in particolare YOLOv5, nei compiti specifici del dominio, come evidenziato dal punteggio mAP del 80.2% sul benchmark Roboflow 100, rispetto al 73.5% di YOLOv5. Inoltre,

YOLOv8 offre varie dimensioni di modello per adattarsi a diverse esigenze di velocità e precisione, raggiungendo uno stato dell'arte nel benchmark COCO con un mAP che varia dal 37.3% al 53.9% a seconda della dimensione del modello.

Chapter 4

Face Recognition

4.1 Detection e Alignment con MTCNN

Rilevare e allineare i volti è fondamentale in applicazioni come il riconoscimento facciale e l'analisi delle espressioni. Le variazioni visive, come occlusioni, diverse posizioni e illuminazioni estreme, pongono notevoli sfide in queste attività in scenari reali. Si propone MTCNN [18], un framework basato su CNN a cascata per la rilevazione e l'allineamento congiunto dei volti e si progetta attentamente un'architettura leggera di CNN per prestazioni in tempo reale.

4.1.1 Pipeline

Il flusso di lavoro generale del nostro approccio è mostrato in Figura 4.1.

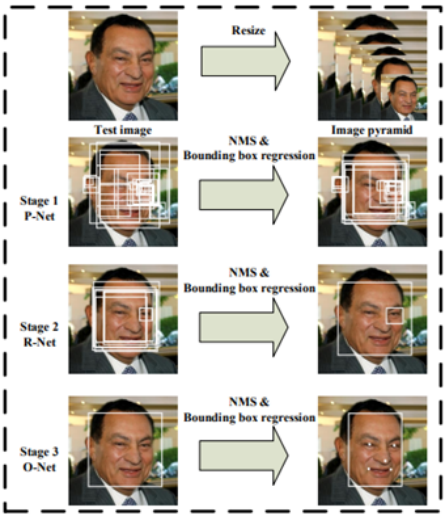

Figure 4.1: MTCNN pipeline

Dato un'immagine, la ridimensioniamo inizialmente a diverse scale per costruire una piramide di immagini, che è l'input delle seguenti tre fasi del framework a cascata:

1. Si utilizza una rete completamente convoluzionale, chiamata Proposal Network (P-Net), per ottenere le finestre facciali candidate e i loro vettori di regressione per il bounding box. Poi le candidature vengono calibrate in base ai vettori di regressione del bounding box stimati. Successivamente, utilizziamo la soppressione del non massimo (NMS) per unire candidati altamente sovrapposti.

2. Tutti i candidati vengono alimentati in un'altra CNN, chiamata Refine Network (R-Net), che ulteriormente scarta un gran numero di candidati falsi, esegue la calibrazione con la regressione del bounding box e esegue il NMS.

3. Questa fase è simile alla seconda fase, ma si mira ad identificare le regioni facciali con una maggiore supervisione. In particolare, la rete O-Net (Output Net) restituirà le posizioni dei cinque landmark facciali.

4.1.2 Architettura

Le architetture delle CNN utilizzate sono mostrate in Figura 4.2. Applichiamo PReLU come funzione di attivazione non lineare dopo i livelli di convoluzione e di connessione completamente connessi (tranne i livelli di output):

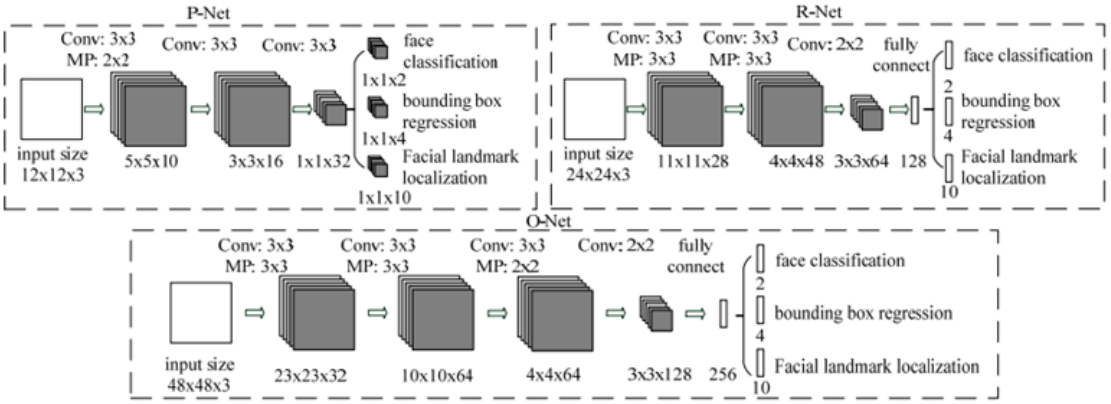

Figure 4.2: Architettura MTCNN

4.1.3 Addestramento

Sfruttiamo tre compiti per addestrare i nostri detector CNN: classificazione faccia/non-faccia, regressione del bounding box e localizzazione dei landmark facciali.

1. **Classificazione dei volti**: L'obiettivo dell'apprendimento è formulato come un problema di classificazione binaria. Per ciascun campione x_i, utilizziamo la cross-entropy loss:

$$L_{\mathrm{i}}^{det} = -y_{\mathrm{i}}^{det} \log(p_i) + (1 - y_{\mathrm{i}}^{det})(1 - \log(p_i)) \tag{4.1}$$

   dove p_i è la probabilità prodotta dalla rete che indica che il campione x_i sia un volto. La notazione $y_{\mathrm{i}}^{det} \in \{0, 1\}$ denota l'etichetta ground-truth.

2. **Regressione del bounding box**: Per ciascuna finestra candidata, prevediamo l'offset tra essa e il bounding box più vicino al ground truth (cioè, il bounding box sinistro, superiore,

altezza e larghezza). L'obiettivo dell'apprendimento è formulato come un problema di regressione, e utilizziamo la perdita euclidea per ciascun campione x_i:

$$L_{\mathrm{i}}^{box} = \|\hat{\mathbf{y}}_{\mathrm{i}}^{box} - \mathbf{y}_{\mathrm{i}}^{box}\|^2 \tag{4.2}$$

dove $\mathbf{y}_{\mathrm{i}}^{box}$ è il target di regressione ottenuto dalla rete dove $\mathbf{y}_{\mathrm{i}}^{box}$ è la coordinata ground-truth. Ci sono quattro coordinate, incluse sinistra superiore, altezza e larghezza, quindi $\mathbf{y}_{\mathrm{i}}^{box} \in \mathbb{R}^4$.

3. **Localizzazione dei landmark facciali**: Semplicemente al compito di regressione del bounding box, la rilevazione dei landmark facciali è formulata come un problema di regressione e minimizziamo la perdita euclidea:

$$L_{\mathrm{i}}^{landmark} = \|\mathbf{y}_{\mathrm{i}}^{landmark} - \hat{\mathbf{y}}_{\mathrm{i}}^{landmark}\|_2^2 \tag{4.3}$$

Dove $\mathbf{y}_{\mathrm{i}}^{landmark}$ sono le coordinate dei landmark facciali ottenute dalla rete e $\hat{\mathbf{y}}_{\mathrm{i}}^{landmark}$ è la coordinata ground-truth per il campione i. Ci sono cinque landmark facciali, inclusi l'occhio sinistro, l'occhio destro, il naso, l'angolo della bocca sinistra e l'angolo della bocca destra, quindi $\mathbf{y}_{\text{landmark}} \in \mathbb{R}^{10}$.

4. **Addestramento multi-sorgente**: Poiché impieghiamo compiti diversi in ciascuna CNN, ci sono diversi tipi di immagini di addestramento nel processo di apprendimento, come volti, non volti e volti parzialmente allineati. In questo caso, alcune delle funzioni di perdita (ossia Eq. (4.1) ... Eq. (4.3)) non vengono utilizzate. Ad esempio, per il campione della regione di sfondo, calcoliamo solo L_i^{det} e le altre due perdite vengono impostate a 0. Questo può essere implementato direttamente con un indicatore del tipo di campione. Quindi l'obiettivo di apprendimento complessivo può essere formulato come:

$$\min \sum_{i=1}^{N} \sum_{j \in \{det, box, landmark\}} \alpha_j \beta_i^j L_i^j \tag{4.4}$$

dove N è il numero di campioni di addestramento e α_j denota l'importanza del compito. Utilizziamo ($\alpha_{det} = 1, \alpha_{box} = 0.5, \alpha_{landmark} = 0.5$) in P-Net e R-Net, mentre ($\alpha_{det} = 1, \alpha_{box} = 0.5, \alpha_{landmark} = 1$) in O-Net per una localizzazione più accurata dei landmark facciali. β_i appartiene all'insieme $\{0, 1\}$ ed è l'indicatore del tipo di campione. In questo caso, è naturale impiegare la discesa gradiente stocastica per addestrare queste CNN.

5. **Estrazione online di campioni difficili**: A differenza della tradizionale estrazione di campioni difficili dopo che il classificatore originale è stato addestrato, conduciamo l'estrazione online di campioni difficili nel compito di classificazione faccia/non-faccia, che è adattativa al processo di addestramento. In particolare, in ciascun mini-batch, ordiniamo le perdite calcolate nella propagazione in avanti da tutti i campioni e selezioniamo il 70% superiore di esse come campioni difficili. Poi calcoliamo solo i gradienti da questi campioni difficili nella propagazione all'indietro. Ciò significa che ignoriamo i campioni facili che sono meno utili per rafforzare il detector durante l'addestramento. Gli esperimenti mostrano che questa strategia produce migliori prestazioni senza selezione manuale dei campioni.

4.2 AdaFace

4.2.1 Descrizione

Il riconoscimento facciale in dataset di volti di bassa qualità, come video di sorveglianza e riprese di droni, presenta sfide a causa dell'oscuramento e degrado degli attributi facciali. Uno

dei problemi con le immagini di volti di bassa qualità è che tendono ad essere irriconoscibili. Quando il degrado dell'immagine è troppo grande, le informazioni di identità rilevanti svaniscono dall'immagine, risultando in immagini non identificabili. Queste immagini non identificabili sono dannose per la procedura di addestramento poiché un modello cercherà di sfruttare altre caratteristiche visive, come il colore dell'abbigliamento o la risoluzione dell'immagine, per abbassare la perdita di addestramento. Se queste immagini dominano la distribuzione delle immagini di bassa qualità, è probabile che il modello si comporti male sui dataset di bassa qualità durante il test.
Per affrontare questo problema, AdaFace [19] introduce una funzione di perdita che assegna importanza diversa ai campioni in base alla loro difficoltà e qualità dell'immagine. Incorporando la qualità dell'immagine, si evita di enfatizzare le immagini non identificabili concentrandoci invece su campioni difficili ma riconoscibili.

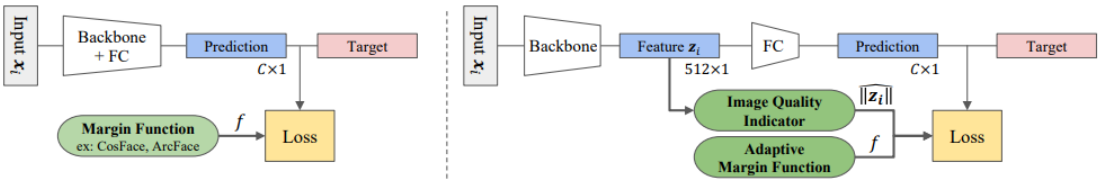

Figure 4.3: Architettura AdaFace

L'immagine confronta due approcci di funzione di perdita per l'addestramento nel riconoscimento facciale: uno tradizionale che enfatizza la riduzione della varianza all'interno delle stesse classi, e un metodo proposto che adatta la funzione di perdita basandosi sulla qualità dell'immagine per enfatizzare esempi facili o difficili di conseguenza.
Si mostra che il margine angolare scala il segnale di apprendimento (gradiente) in base alla difficoltà del campione di addestramento. Questa osservazione motiva a cambiare in modo adattivo la funzione di margine per enfatizzare campioni difficili se la qualità dell'immagine è alta e ignorare immagini molto difficili (immagini non identificabili) se la qualità dell'immagine è bassa.
Questo approccio si differenzia dall'apprendimento tradizionale basato sul curriculum, che non tiene conto della qualità dell'immagine. AdaFace utilizza la norma delle caratteristiche come proxy per la qualità dell'immagine, permettendo di regolare la funzione di margine in modo adattivo senza aggiungere complessità.

4.2.2 Confronti con lavori correlati

AdaFace adotta una funzione di perdita softmax basata sul margine, usata in modelli come CurricularFace, ArcFace, CosFace e SphereFace, per migliorare la discriminabilità nel riconoscimento facciale. Diversamente da altri, AdaFace adatta il margine alla qualità dell'immagine, ottimizzando l'apprendimento in base alla difficoltà e alla qualità dei campioni. Mentre approcci come CurricularFace aumentano il margine nel tempo per concentrarsi su campioni difficili, AdaFace e MagFace modulano il margine in base alla riconoscibilità. Altri metodi simulano dati di bassa qualità o utilizzano aumenti per migliorare la generalizzazione, ma AdaFace preferisce aumenti semplici, come ritagli e sfocature, per una maggiore applicabilità.

4.2.3 Approccio proposto

La loss function softmax cross-entropy è una componente fondamentale nell'addestramento di modelli di classificazione. È definita come:

$$L_{CE}(z_i) = -\log\left(\frac{\exp\left(w_{y_i}^T z_i + b_{y_i}\right)}{\sum_{j=1}^{C}\exp\left(w_j^T z_i + b_j\right)}\right) \tag{4.5}$$

dove z_i è l'embedding della feature di un'immagine x_i, w_{y_i} è il vettore dei pesi associato alla classe corretta y_i, b_{y_i} è il termine di bias per quella classe, e C è il numero totale di classi.
Per migliorare la discriminazione tra le classi, si possono utilizzare metriche basate sulla distanza coseno, modificando la softmax cross-entropy per includere il coseno dell'angolo tra l'embedding della feature e i pesi della classe, come segue:

$$L_{CE}(z_i) = -\log\left(\frac{\exp(s\cdot\cos(\theta_{y_i}))}{\sum_{j=1}^{C}\exp(s\cdot\cos(\theta_j))}\right) \tag{4.6}$$

dove θ_{y_i} è l'angolo tra z_i e w_{y_i}, e s è un fattore di scala.
Successivi sviluppi, come ArcFace e CosFace, hanno introdotto un margine all'interno di questa formulazione per ridurre ulteriormente le variazioni intra-classe e migliorare la separabilità delle feature. Questo si traduce in un apprendimento più robusto e in un riconoscimento facciale più accurato.

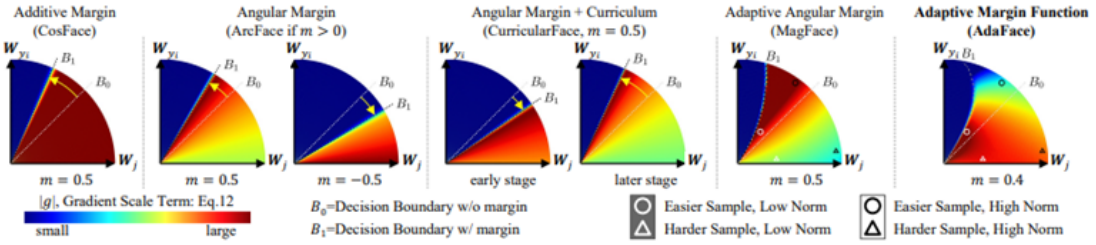

Figure 4.4: Funzioni di margine

La Figura 4.4 illustra diverse funzioni di margine e i relativi termini di scala del gradiente nello spazio delle caratteristiche. Le linee B0 e B1 rappresentano il confine decisionale senza e con il margine m rispettivamente. La freccia gialla mostra come il margine m sposti il confine decisionale. Un campione ben classificato sarà vicino, in termini angolari, al vettore di peso della classe corretta, w_{y_i}, mentre un campione mal classificato sarà più vicino a w_j, il vettore di peso della classe negativa. Il colore all'interno dell'arco indica l'ampiezza del termine di scala del gradiente g. I campioni nella regione di colore rosso scuro contribuiranno di più all'apprendimento.
La figura mostra che un margine additivo sposta il confine decisionale verso w_{y_i} senza modificare il termine di scala del gradiente. Al contrario, un margine angolare positivo non solo sposta il confine, ma aumenta anche la scala del gradiente vicino al confine e la riduce più ci si allontana dal confine. Questo comportamento riduce l'importanza dei campioni molto difficili. MagFace mostra un comportamento simile. Invece, un margine angolare negativo induce un comportamento opposto. CurricularFace adatta il confine decisionale in base alla fase di addestramento. L'approccio AdaFace cambia adattivamente le funzioni di margine basandosi sulla norma delle caratteristiche. Con una norma elevata, si enfatizzano i campioni lontani dal confine decisionale, mentre con una norma bassa si enfatizzano i campioni vicini al confine.

I cerchi e i triangoli rappresentati nell'arco forniscono esempi di scenari nel tracciato più a destra (AdaFace), mostrando come differenti campioni vengano enfatizzati in base alla loro distanza dal confine decisionale.
Durante la retropropagazione il cambiamento del gradiente dovuto al margine ha l'effetto di scalare l'importanza di un campione rispetto agli altri. In altre parole, il margine angolare può introdurre un termine aggiuntivo nell'equazione del gradiente che scala il segnale in base alla difficoltà del campione. Invece di utilizzare metodi computazionalmente onerosi per valutare la qualità dell'immagine durante l'addestramento, in questo lavoro si usa la norma delle caratteristiche come un indicatore della qualità dell'immagine.
È stato osservato che nelle reti addestrate con una perdita softmax basata su margine, la norma delle caratteristiche mostra una correlazione con la qualità dell'immagine. Progettiamo una funzione di margine in modo tale che:

1. se la qualità dell'immagine è alta, enfatizziamo i campioni difficili, e
2. se la qualità dell'immagine è bassa, de-enfatizziamo i campioni difficili.

Raggiungiamo ciò con due termini additivi g_{angle} e g_{add}, che si riferiscono rispettivamente ai margini angolari e additivi.

$$f(\boldsymbol{\theta}_j, m)_{\text{AdaFace}} = \begin{cases} s\cos(\theta_j + g_{\text{angle}}) - g_{\text{add}} & j = y_i, \\ s\cos(\theta_j) & j \neq y_i, \end{cases} \tag{4.7}$$

Dove g_{angle} e g_{add} sono le funzioni di $\|z_i\|$. Definiamo

$$g_{\text{angle}} = -m \cdot \frac{\|z_i\|}{1 + \|z_i\|}, \quad g_{\text{add}} = m \cdot \frac{1}{1 + \|z_i\|} \tag{4.8}$$

Si noti che quando $\|z_i\| = -1$, la funzione proposta diventa ArcFace. Quando $\|z_i\| = 0$, diventa CosFace. Quando $\|z_i\| = 1$, diventa un margine angolare negativo con uno spostamento. La Figura 3 mostra l'effetto della funzione adattativa sul gradiente. Le caratteristiche con norma alta riceveranno una scala di gradiente più alta, lontano dal confine decisionale, mentre le caratteristiche con norma bassa riceveranno una scala di gradiente più alta vicino al confine decisionale. Per le caratteristiche con norma bassa, i campioni più difficili lontani dal confine sono de-enfatizzati.

4.2.4 Esperimenti

Per l'addestramento di modelli di riconoscimento facciale sono stati utilizzati tre ampi **dataset**: MS1MV2 con 5,8 milioni di immagini, MS1MV3 con 5,1 milioni e WebFace4M con 4,2 milioni di immagini.
I **set di tes**t si dividono in tre livelli qualitativi: Alta Qualità (per esempio LFW e CFP-FP), con immagini nitide utilizzate come benchmark; Qualità Mista (come IJB-B e IJB-C), che mescolano immagini di alta e bassa qualità per testare la robustezza del modello; e Bassa Qualità (IJB-S e TinyFace), che includono immagini di bassa risoluzione o di sorveglianza, per valutare le performance del modello sotto condizioni sfavorevoli.
Per l'addestramento **pre-processiamo** il dataset ritagliando e allineando i volti con cinque landmark, come in ArcFace e MTCNN, ottenendo immagini 112×112.
Per il **backbone**, adottiamo ResNet come modificato in ArcFace [20]: implementa una nuova funzione di perdita ArcFace che mira a perfezionare l'efficienza del modello di riconoscimento facciale ottimizzando la distanza tra le rappresentazioni delle classi. Questa funzione introduce

un concetto innovativo: un margine angolare additivo che agisce direttamente sullo spazio degli embedding delle caratteristiche e il centro geometrico di ogni classe. L'obiettivo è rafforzare la distinzione tra le classi migliorando simultaneamente la coesione interna di ogni classe. Grazie a questo approccio, si ottiene una maggiore separazione tra le classi all'interno di uno spazio normalizzato a forma di ipersfera, facilitando così il compito del modello nel distinguere efficacemente tra soggetti diversi.
AdaFace utilizza lo stesso **ottimizzatore** e lo stesso **programma di apprendimento** di CurricularFace [21] (usa l'algoritmo SGD (Stochastic Gradient Descent) con un momentum pari a 0.9 e un weight decay di 5×10^{-4} per l'addestramento dei modelli), e lo addestra per 24 epoche. Il modello viene addestrato con **SGD** con un tasso di apprendimento iniziale di 0,1 e pianificazione degli step alle epoche 10, 18 e 22. Se il dataset contiene augmentations, aggiungiamo altre 2 epoche per la convergenza. Per il parametro di scala s, lo impostiamo su 64, seguendo la suggerimento di ArcFace e CosFace.
Il metodo di AdaFace si concentra sull'efficacia dell'addestramento con dati contenenti immagini non identificabili, implementando **augmentazioni** on-the-fly come ritagli, ridimensionamenti e alterazioni fotometriche. Queste tecniche generano dati aggiuntivi ma includono anche più immagini non identificabili, che la nostra funzione di loss è progettata per ignorare. Specifiche tecniche come il ritaglio casuale di aree, l'alterazione della tonalità, saturazione e luminosità, e il ridimensionamento per sfocatura, vengono applicate con una probabilità del 20%, mantenendo l'allineamento facciale cruciale per l'efficacia dell'addestramento.

Chapter 5

Progetto

Il progetto sviluppato consiste in un sistema che può essere rappresentato dalla seguente pipeline:

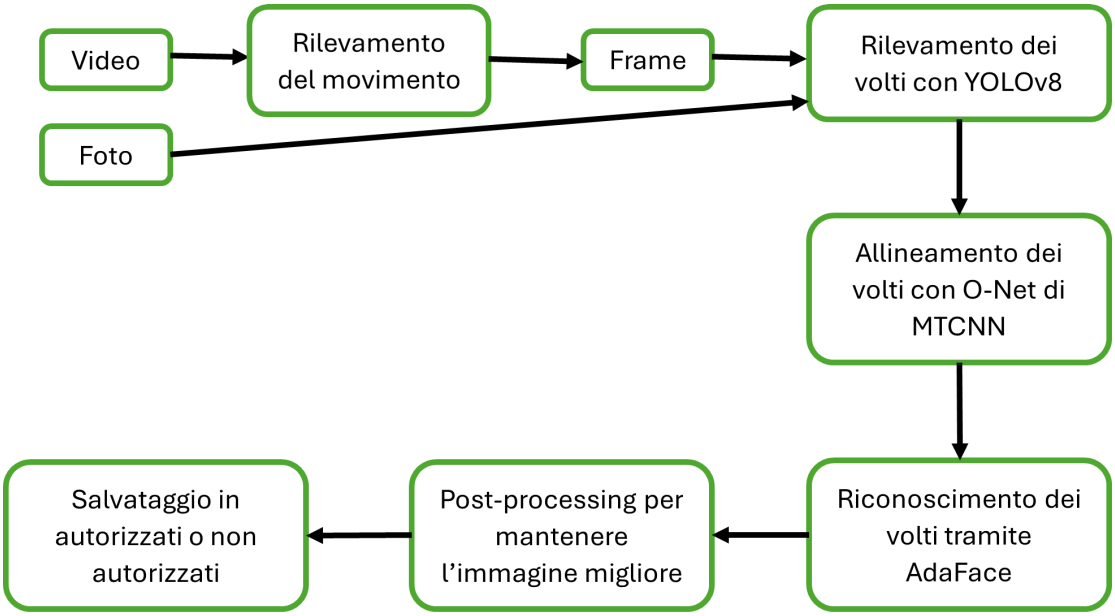

Figure 5.1: Progetto pipeline

Affronteremo fase per fase la pipeline mostrata nei seguenti capitoli.

5.1 Pre-elaborazione dell'input

Il sistema può ricevere come input sia file video sia immagini singole. In base al formato dell'input si sceglie se passarlo direttamente al componente successivo oppure effettuare una pre-elaborazione. Infatti, se l'input è una singola immagine, questa verrà passata al rilevatore di volti direttamente senza alcuna modifica. Se invece, l'input è un file video dovrà sottoporsi ad una elaborazione che consiste nel:

1. Estrarre dal video un frame alla volta.

2. Confrontare ogni frame con il precedente per rilevare il movimento.
3. Se il movimento è significativo, passare il frame al rilevatore di volti.

Analizziamo questi 3 passaggi step-by-step.

5.1.1 Estrazione di frame

Il video può essere sia un file caricato da memoria ma può essere anche un video in tempo reale che proviene da una fonte come una webcam oppure una telecamera installata. Nel primo caso, si sceglie il frame rate, ovvero quanti frame al secondo si vuole estrarre. In base a requisiti definiti secondo il nostro caso d'uso si è deciso di estrarre 1 frame al secondo.
Nel secondo caso si preleva dallo stream video un frame alla volta, si passa il frame ai componenti successivi e si preleva il frame successivo nel momento in cui il precedente è uscito dalla pipeline. Nell'eventualità in cui il frame abbia impiegato meno di 1 secondo per percorrere tutta la pipeline, si aspetta l'istante temporale per raggiungere il secondo.

5.1.2 Rilevamento del movimento

Questa fase serve per non immettere negli step successivi frame che avrebbero gli stessi risultati del frame precedente, ovvero rilevamenti delle stesse persone. Infatti, visto che l'obbiettivo è quello di rilevare tutte le persone, se non si rilevano movimenti significativi tra un frame e l'altro è dato per certo che non ci siano nuovi volti da acquisire.
Per fare ciò, si confronta ogni nuovo frame con il precedente e se la differenza dei frame supera una certa soglia, ciò indica un movimento significativo.
La differenza tra i due frame viene calcolata convertendo prima i frame in scala di grigi, poi calcolando la differenza assoluta tra questi. Successivamente, si applica una soglia per identificare i pixel che hanno cambiato intensità oltre un certo limite, indicando potenziale movimento. Il rapporto tra il numero di pixel che hanno superato la soglia e il numero totale di pixel nell'immagine determina il "movement ratio", che quantifica il movimento tra i due frame.

5.1.3 Passaggio al rilevatore

Se la "movement ratio" supera una soglia stabilita, decisa in base al caso d'uso, si può dedurre che tra i due frame è avvenuto un movimento significativo.
In caso contrario il frame verrà tenuto in cosiderazione per il successivo confronto ma non verrà consegnato al rilevatore di volti.
Visto che il primo frame non ha un frame precedente, verrà confrontato con un immagine totalmente nera in modo che superi il test del movimento e venga passato al rilevatore di volti.

5.2 Rilevamento dei volti con YOLOv8

Per il rilevamento dei volti si è scelto di utilizzare un modello di rilevamento di volti basato su YOLOv8n a cui è stato applicato transfer-learning per rilevare solo una classe di oggetti, ovvero i volti umani.
Il modello finale è stato ricavato dopo un addestramento su 89 immagini con volti di persone di età e sesso differenti, con e senza occhiali/mascherina chirurgica. Il dataset è stato suddiviso, dopo aver effettuato data augmentation, in 186 immagini di addestramento, 18 di validation e 9 di test. I risultati dell'addestramento sono stati sul set di addestramento del 96,4% di mAP (mean Average Precision), 92,8% di precision e un recall del 91.3%. Invece, sul set di test si è

raggiunto il 91% di mAP.
Per utilizzare il modello si è dovuto installare, con il package manager di python "pip", la libreria ultralytics e importare YOLO.
Caricato il modello, si usa il metodo predict passandogli il frame ricevuto dallo step precedente. Il risultato di questo metodo viene filtrato in modo da ricavare le coordinate delle bounding box dei volti (x1,y1,x2,y2) e il valore di confidenza compreso tra 0 e 1.

```
results = model.predict(source=frame)[0]
bboxes = results.boxes.data[:, :4].cpu().numpy().astype(int)
confs = results.boxes.conf.cpu().numpy().astype(float)
```

Figure 5.2: Codice per elaborazione risultati predict()

Successivamente viene effettuato un filtraggio in base al valore di confidenza e alle dimensioni delle bounding box. Infatti, vengono passate allo step successivo solo le bounding box che hanno dimensione maggiore uguale a 30x30 pixels e confidenza maggiore di 0,5.
Questi valori sono stati scelti in base al contesto, ovvero in base alle immagini di input (capitolo 5.7). Dopo diversi tentativi, si è raggiunto la coppia di valori che permette al riconoscitore di avere risultati ottimali. Infatti, l'obbiettivo è quello di scartare le bounding box che avrebbero creato errori nel riconoscimento poichè troppo piccole o perchè non inquadravano perfettamente un volto.

5.3 Allineamento dei volti con O-Net di MTCNN

Per effettuare l'allineamento dei volti rilevati nello step precedente si è scelto di utilizzare l'architettura di MTCNN formata da P-Net, R-Net e O-Net. A differenza del sistema appena citato, si è scelto di rimuovere la P-Net e la R-Net che avevano il compito principale di trovare le bounding box dei volti. Infatti, si utilizzano i rilevamenti dello step precedente, fornendoli direttamente alla O-Net. Questa scelta è stata fatta in quanto il rilevamento dei volti basato su YOLO è più veloce rispetto alla coppia di reti P-Net+R-Net.
La O-Net ci fornirà i 5 landmark facciali che verranno usati sia per l'allineamento in questa fase che nella fase di post-processing. Il metodo per l'allineamento dei volti funziona nel seguente modo:

1. **Rilevamento dei punti facciali**: Inizialmente, il metodo riceve i 5 landmark facciali dalla O-Net, che identificano le posizioni chiave del volto, come gli angoli degli occhi, la punta del naso e gli angoli della bocca.

2. **Definizione dei punti di riferimento**: Successivamente, si stabilisce un set di punti di riferimento predefinito che rappresenta la posizione ottimale dei landmark facciali dopo l'allineamento. Questi punti di riferimento mirano a posizionare i tratti facciali in modo standardizzato in tutte le immagini.

3. **Calcolo della trasformazione**: Utilizzando i landmark facciali rilevati e i punti di riferimento, il metodo calcola una matrice di trasformazione affine. Questa matrice dettaglia come muovere, ruotare e scalare l'immagine del volto per far coincidere i landmark facciali con i loro corrispondenti punti di riferimento.

4. **Applicazione della trasformazione**: La matrice di trasformazione viene applicata all'immagine sorgente, modificando l'orientamento, la posizione e la scala del volto nell'immagine per

allinearla ai punti di riferimento. Questo processo normalizza l'aspetto del volto per l'analisi successiva.

5. **Ritaglio dell'immagine**: Dopo aver allineato il volto, l'immagine viene ritagliata a una dimensione predefinita, rimuovendo le parti non necessarie. Questo assicura che le immagini dei volti allineati siano uniformi, facilitando il confronto e l'analisi ulteriori.

Questo processo di allineamento dei volti garantisce che le immagini siano normalizzate in termini di posizione, orientamento e scala dei volti, migliorando così l'efficacia dei sistemi di riconoscimento facciale e delle analisi successive.

5.4 Riconoscimento dei volti tramite AdaFace

Tra i modelli pre-addestrati forniti nella repository del progetto di AdaFace [22] si è scelto di utilizzare il modello con architettura ResNet-50, addestrato sul dataset MS1MV2. La scelta sulla profondità dell'architettura (erano presenti anche ResNet-18 e ResNet-100) è stata presa in base alla velocità e precisione dei modelli forniti. Infatti, dopo aver provato tutte e tre le architetture si è notato che ResNet-18 permetteva di lavorare in tempo reale ma risultava troppo impreciso nel riconoscimento. ResNet-100 non permetteva si lavorare in tempo reale e aveva comunque una precisione di riconoscimento molto vicino a ResNet-50.
Per ResNet-50 erano forniti modelli addestrati sui seguenti dataset:

- **CASIA-WebFace**: Uno dei primi ampi dataset utilizzati per il riconoscimento facciale, con circa 500.000 immagini di 10.000 soggetti. È stato utilizzato come punto di riferimento per lo sviluppo iniziale dei modelli di riconoscimento facciale.
- **WebFace4M**: É una versione estesa e migliorata, con circa 4 milioni di immagini, che fornisce una varietà più ampia di dati per l'addestramento di modelli più robusti.
- **MS1MV2 o MS-Celeb-1M versione 2**: É un set ancora più grande e pulito con immagini di qualità superiore rispetto alle versioni precedenti, contando milioni di immagini di decine di migliaia di soggetti. È noto per la sua ricchezza di variazioni e per la pulizia dei dati, rendendolo uno dei dataset di riconoscimento facciale più utilizzati e performanti.

Il modello AdaFace basato sull'architettura R50 e addestrato sul dataset MS1MV2 mostra eccellenti prestazioni nei test di validazione, raggiungendo punteggi elevati su vari benchmark. Nel test LFW (Labelled Faces in the Wild), che misura la capacità di riconoscere volti in condizioni non controllate, ha ottenuto un impressionante 99.82%. Nel CFPFP, che valuta la performance di fronte a cambiamenti di posa, il modello ha registrato un punteggio di 97.76%. In CPLFW e CALFW, che testano il riconoscimento su volti con variazioni di età e posa, i punteggi sono stati rispettivamente 92.83% e 96.07%. Infine, nel benchmark AGEDB, incentrato sull'età dei soggetti, il punteggio è stato del 97.85%. Complessivamente, il modello ha raggiunto un punteggio medio (AVG) del 96.88% attraverso questi benchmark, dimostrando la sua robustezza e affidabilità nel riconoscimento facciale. Questi risultati indicano che il modello R50 con MS1MV2 è particolarmente efficace nel gestire diverse sfide del riconoscimento facciale, inclusi cambiamenti di età, posa e qualità dell'immagine.

5.4.1 Caricamento e utilizzo del modello

Il modello viene caricato tramite la funzione load_pretrained_model() a cui viene passato il parametro che specifica quale architettura usare:

```python
def load_pretrained_model(architecture='ir_50'):
    assert architecture in adaface_models.keys()
    model = net.build_model(architecture)
    statedict = torch.load(adaface_models[architecture], map_location=torch.device('cpu'))['state_dict']
    model_statedict = {key[6:]: val for key, val in statedict.items() if key.startswith('model.')}
    model.load_state_dict(model_statedict)
    model.eval()
    return model
```

Figure 5.3: Funzione load_pretrained_model()

La funzione costruisce il modello layer per layer utilizzando la funzione build_model() fornita dalla libreria net, dopodiché carica i pesi preaddestrati dal percorso specificato nel dizionario adaface_models. "torch.load" indica che il modello preaddestrato deve essere caricato sulla CPU, anziché sulla GPU. Questo è utile quando si lavora su una macchina che non dispone di una GPU o quando si preferisce eseguire il modello sulla CPU per qualsiasi motivo. Infine, il modello viene messo in modalità di valutazione con model.eval(), che disabilita alcuni comportamenti specifici dell'addestramento, come il dropout, rendendolo pronto per effettuare inferenze. La funzione restituisce il modello pronto all'uso.
Per l'utilizzo del modello si passa come parametro un dato elaborato dalla funzione to_input():

```python
def to_input(pil_rgb_image):
    np_img = np.array(pil_rgb_image)
    brg_img = ((np_img[:, :, ::-1] / 255.) - 0.5) / 0.5
    np_transposed = np.array( object: [brg_img.transpose(2, 0, 1)], dtype=np.float32)
    tensor = torch.from_numpy(np_transposed)
    return tensor
```

Figure 5.4: Funzione to_input()

La funzione prende un'immagine PIL in formato RGB (fornito dal componente per l'allineamento dei volti), la converte in un array NumPy, e poi inverte gli indici dei canali per ottenere BGR poiché PyTorch [23] utilizza questo formato. L'immagine viene quindi normalizzata dividendola per 255 per avere valori tra 0 e 1, e quindi centrata e scalata con un fattore di 0.5 per avere un range di [-1, 1]. Dopo di chè, l'immagine viene riordinata per avere i canali come prima dimensione (come atteso da PyTorch) e infine viene convertita in un tensore di PyTorch. Questo tensore può essere utilizzato come input per modelli basati su PyTorch.

Il modello restituisce due valori:

- Magnitudine (lunghezza) del vettore di caratteristiche nel suo spazio di embedding, che viene utilizzata per valutare la qualità o la confidenza dell'embedding generato.

- Tensore contenente le caratteristiche (feature) estratte del volto, che rappresentano l'identità in uno spazio di embedding. In un modello di riconoscimento facciale, queste sono le caratteristiche che vengono confrontate per determinare se due immagini rappresentano la stessa persona.

5.4.2 Riconoscimento

La magnitude e il tensore di feature vengono passate alla funzione di riconoscimento insieme al vettore delle feature di tutti i volti delle persone autorizzate e le informazioni ricevute dalla funzione di allineamento ovvero volto allineato, bounding box e vettore dei 5 landmark.

```
def recognition(cropped_face, aligned_box, i_feature, mag, landmarks, cropped_features):
    authorized = False
    path_found = "Sconosciuto"
    try:    # cropped_features --> path, feature, mag, boxes, aligned_rgb_img, landmarks
        for i, c_feature in enumerate(cropped_features):
            similarity_score = i_feature @ c_feature[1].T
            if similarity_score.item() >= 0.25:
                authorized = True
                path_found = c_feature[0]
                if replace(c_feature[2], c_feature[5], c_feature[3], mag, landmarks, aligned_box):
                    if os.path.exists(c_feature[0]):
                        os.remove(c_feature[0])
                        face_rgb = cv2.cvtColor(np.array(cropped_face), cv2.COLOR_BGR2RGB)
                        cv2.imwrite(c_feature[0], face_rgb)
                    else:
                        print("Il file non esiste:", c_feature[0])
                    cropped_features[i] = (c_feature[0], i_feature, mag, aligned_box, cropped_face, landmarks)
                    print(f"{c_feature[0]}: autorizzato sostituito con immagine migliore ")
                else:
                    print(f"{c_feature[0]}: autorizzato, mantengo file vecchio")
    except Exception as e:
        print(f"Immagine rifiutata", e)
    return path_found, authorized, cropped_features
```

Figure 5.5: Funzione recognition()

La funzione scorre attraverso le caratteristiche pre-esistenti confrontando i_feature con ciascuna delle caratteristiche salvate per determinare una corrispondenza basata su un punteggio di similarità. Se il punteggio supera una soglia di 0.25 (decisa in base ai risultati riscontrati), il soggetto viene considerato autorizzato e il percorso del file corrispondente viene registrato.

5.5 Post-processing per mantenere l'immagine migliore

Successivamente al riconoscimento, viene utilizzato la funzione replace() che ha l'obbiettivo di verificare se l'immagine attuale è migliore dell'immagine salvata precedentemente negli autorizzati. La sostituzione avviene se la nuova immagine ha una magnitude maggiore e se ha un'indice di frontalità maggiore rispetto alla precedente. Per il calcolo dell'indice di frontalità si esegue l'estrazione e l'elaborazione dei landmark facciali per calcolare:

- La distanza normalizzata tra il centro degli occhi e il naso
- La simmetria degli occhi
- La simmetria della bocca rispetto alla larghezza della bounding box

Questi calcoli sono fondamentali per determinare quanto il volto sia rivolto verso la camera. I punteggi di simmetria ottenuti per occhi e bocca vengono poi ponderati in base alla loro distanza e larghezza relativa all'interno della bounding box, al fine di ottenere due misure di simmetria

ponderata. Queste misure sono infine combinate per calcolare lo score finale di frontalità, il quale è una media pesata che riflette quanto il volto sia orientato frontalmente. Lo score finale è già normalizzato per tenere conto delle dimensioni del viso, rendendo il punteggio indipendente dalle dimensioni assolute della bounding box o dei landmarks.
Un esempio in cui l'immagine viene sostituita da un immagine migliore, quindi con magnitude maggiore e indice di frontalità maggiore:

Figure 5.6: Esempio di immagine migliore

5.5.1 Non autorizzati

Utilizziamo lo stesso approccio sia per i volti delle persone autorizzate che per quelli non autorizzati. Per ogni individuo non autorizzato, memorizziamo le stesse informazioni che raccogliamo per le persone autorizzate. Questo assicura che, se il volto di una persona non autorizzata viene riconosciuto nuovamente, non verrà creato un duplicato nel sistema. Piuttosto, la nuova immagine verrà confrontata con quella esistente e, se migliore, la sostituirà.

5.6 Demo

La demo creata per mostrare la praticità del progetto è stata implementata utilizzando la libreria standard di Python "Tkinter" per la crezione di interfacce grafiche. Per semplicità si è scelto di creare un applicazione locale sull'applicativo PyCharm (IDE per applicazioni Python). Tutti i dati elaborati sono salvati, come immagini e/o video, nella memoria locale.

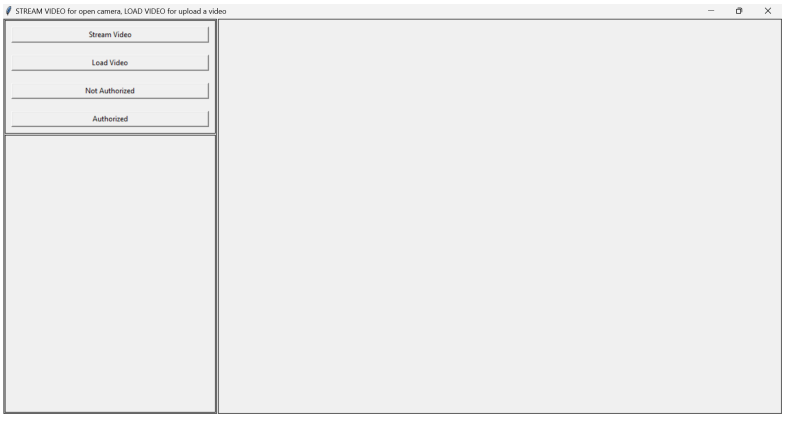

Figure 5.7: Demo

Quando l'applicazione viene eseguita, vengono immediatamente caricate le immagini dei volti degli autorizzati e dei non autorizzati già presenti in memoria. La schermata iniziale dell'applicazione mostra 4 bottoni che si comportano nel seguente modo:

- **Stream video**: Apre lo stream video della webcam del computer su cui si esegue l'analisi.
- **Load video**: Permette di scegliere un video formato .mp4 come input per l'analisi.
- **Authorized/Not Authorized**: Mostra nella parte destra della schermata tutte le immagini delle persone autorizzate/ non autorizzate rispettivamente.

5.6.1 Authorized/Not Authorized

Selezionando il bottone "Not Authorized" vengono mostrate nella regione a destra dell'applicazione i volti delle persone non autorizzate e tramite il bottone aggiungi si fornisce la possibilità di aggiungere l'immagine di una persona nuova.

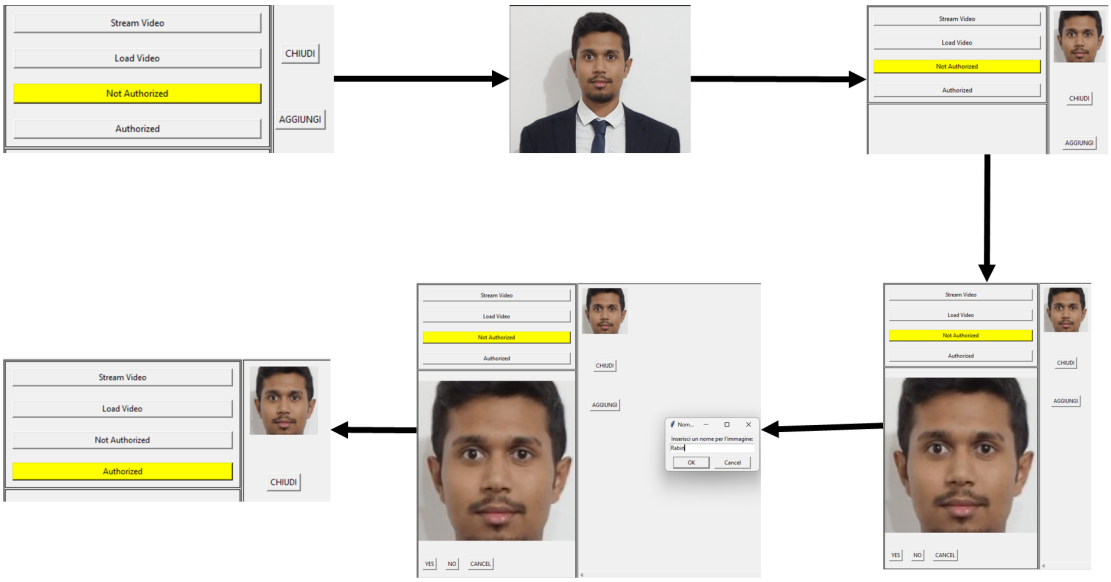

Figure 5.8: Pipeline caricamento manuale

Si può vedere come l'immagine inizialmente venga salvata come "Not Authorized" e successivamente si può renderla "authorized" prima cliccando sull'immagine e poi assegnandoli un nominativo. Infine, si può vedere come l'immagine venga spostata dalla cartella dei "Not Authorized" in quella degli "Authorized".

5.6.2 Stream video

Selezionato il bottone per lo stream video si apre una schermata come la seguente:

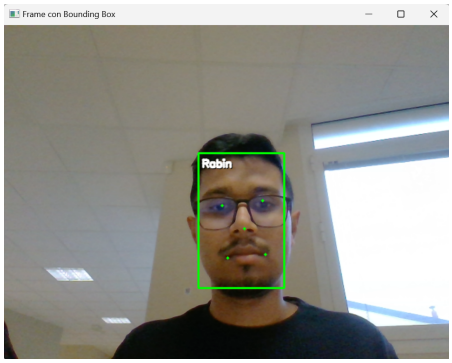

Figure 5.9: Riconoscimento in tempo reale su stream video

In questo caso, la persona inquadrata è riconosciuta ed etichettata con il nome in quanto presente nella cartella "Authorized".

5.6.3 Load video

Selezionando il bottone "Load video" si può caricare un video .mp4 ed analizzarlo:

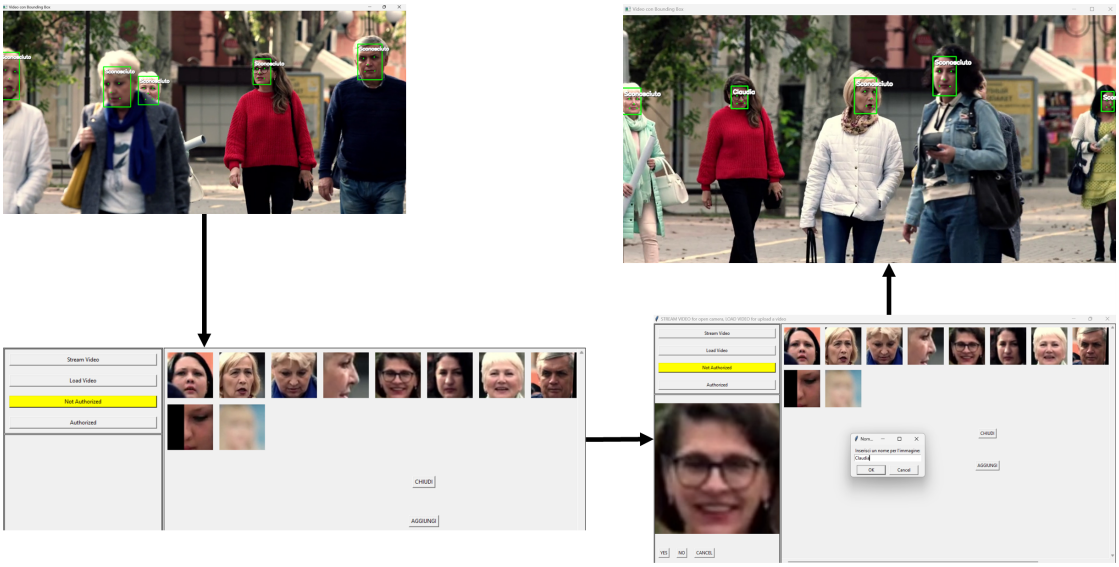

Figure 5.10: Caricamento di un video e riconoscimento

In questo caso si può notare come tutte le persone siano state riconosciute come "Not Authorized" e successivamente dopo aver reso "Authorized" la signora con gli occhiali, nominandola Claudia, e ri-analizzando il video si può vedere come venga riconosciuta correttamente.

5.7 Testing

Per poter valutare le prestazioni del sistema, quindi dell'intera pipeline, si è deciso di testarlo per semplicità in locale su un computer con precessore Intel Core i5-8250U con 4 core fisici, velocità del processore 1.60GHz e 8GB di RAM. I test sono eseguiti utilizzando l'applicazione su cui è stata creata la demo ovvero PyCharm.

5.7.1 Dataset di test

Il sistema è stato testato utilizzando due insiemi di immagini:

- Model: Composto da 168 immagini di volti di altrettante persone
- Scene: Composto da 7484 immagini di volti

Le 168 persone sono presenti in 1044 immagini (delle 7484). Le immagini di scena possono contenere sia immagini di singole persone sia immagini con un gruppo di persone. In totale, nelle 7484 immagini, ci sono 8621 volti.
Le immagini di modello sono composte solo da persone singole in modo da poterle utilizzare come immagini iniziali di riferimento.

Figure 5.11: Immagini modello

Le immagini modello e di scena sono stati selezionati dal dataset LFW - People (Face Recognition) [24]. Sono stati eliminate tutte le cartelle (ogni cartella rappresentava una persona) contenenti singole immagini visto che per poter eseguire un riconoscimento abbiamo bisogno di 1 immagine come modello e 1 immagine come scena. Successivamente si è separato le immagini tra modello e scena.

5.7.2 Testing del riconoscimento su dataset

Eseguendo l'applicazione con le immagini modello come persone "Authorized" e confrontandole con tutte le immagini di scena si hanno i seguenti risultati:

- Scarta per dimensione insufficiente o perchè i landmark sono occlusi il 5% (433 volti su 8621) dei volti analizzati
- Riconosce le persone autorizzate in modo corretto con una precisione del 94,25% (984 volti su 1044)
- Classifica in modo errato "Authorized" lo 0.61% (6 volti su 984)
- Su un totale di 8621 volti classifica correttamente "Authorized" e "Not Authorized" il 99.92% (8182 volti su 8188)

Questi risultati sono ottenuti utilizzando come parametro di confidenza della detection pari a 0.5, dimensioni minime dei volti 30x30pixel e similarità per considerare due volti uguali a 0.5.
Utilizzando come valore di similarità 0.75 si hanno meno errori nella classificazione (2 invece dei 6 precedenti) ma riconosce correttamente solo il 21.74% (227 volti su 1044) considerando erroneamente "Not Authorized" la maggior parte dei volti.
Invece, con similarità a 0.25 si hanno i numeri di riconoscimenti corretti pressochè uguali (983) ma gli errori si classificazione "Authorized" salgono a 17.4% (171 volti su 983).
Precisiamo che per questo test si è disattivato la funzionalità in cui si conserva in mememoria le feature dei soggetti "Not Authorized" in quanto saturava la memoria disponibile nel calcolatore. Disattivando la funzionalità che mantiene, anche per gli "Authorized", l'immagine migliore si ha una leggera degradazione dei risultati. Infatti, effettuando il test mantenendo i parametri precedenti con risultati migliori, si hanno i seguenti risultati:

- Riconosce le persone autorizzate in modo corretto con una precisione del 93,10% (972 volti su 1044)
- Classifica in modo errato "Authorized" lo 0.82% (8 volti su 972)
- Su un totale di 8621 volti classifica correttamente "Authorized" e "Not Authorized" il 99.90% (8180 volti su 8188)

5.7.3 Test della velocità di esecuzione su dataset

L'applicazione ha come obbiettivo quello di riuscire ad analizzare le immagini in tempo reale. Il raggiungimento di questo obbiettivo dipende da fattori come la velocità di inferenza delle componenti della pipeline (rilevatore, allineatore e riconoscitore), la velocità di accesso alla memoria per il salvataggio delle immagini e la velocità di acqusizione dell'immagine da memoria o da telecamera.
L'analisi effettuata in questo capitolo considera solo la velocità di inferenza delle componenti della pipeline. Ovviamente, anche questo dato varia in base al calcolatore che si usa per eseguire il programma. Nel nostro caso si fa riferimento al processore descritto all'inizio del capitolo.
Per la creazione del vettore di feature delle 168 immagini con un solo volto per immagine, il sistema ha impiegato 87.38982 secondi, ovvero 0.52sec per immagine.
Per confrontare 168 volti con 8621 volti impiega in media dopo 5 esperimenti:

- 1755,527 secondi per il rilevamento dei volti, ovvero 0.2 secondi per volto.
- 47,546 secondi per l'allineamento dei volti, ovvero 0.0055 secondi per volto.

- 1391,08 secondi per il riconoscimento dei volti, composto da inferenza del modello e confronto delle feature. Quindi, impiega 0.16 secondi per volto.

Quindi, i componenti della pipeline, considerando 168 persone presenti nel database degli autorizzati, impiegano in media 0.3655 secondi per volto.

5.7.4 Test su video

La differenza tra il caricamento di una serie di immagini e un video sta principalmente nella possibilità di scartare frame in cui non è presente un movimento significativo. Infatti, analizzando 5 video da mediamente 30 secondi ciascuno, in cui sono presenti persone che camminano per strada verso la telecamera ed estraendo 1 frame al secondo, si passano al rilevatore di volti:

- Con indice di movimento 0.1: 100% dei frame e un rilevamento corretto del 100% delle persone presenti
- Con indice di movimento 0.5: 60% dei frame e un rilevamento corretto del 100% delle persone presenti
- Con indice di movimento 0.75: 3% dei frame e un rilevamento corretto del 40% delle persone presenti

Da sottolineare che diminuendo i frame da analizzare, i volti salvati alla fine dell'analisi sono progressivamente peggiori ovvero meno frontali o con landmark facciali meno definiti e chiari.

5.7.5 Test su video in tempo reale

Per effettuare test su video in tempo reale si è usato una webcam a 30fps con risoluzione a 720p, rapporto di aspetto di 16:9 e reflesh rate a 60Hz. Si è cercato di ricreare i video usati per il capitolo 5.7.4 con l'unica differenza rappresentata dal minor numero di persone presenti (3,4,5). I test si basano principalmente su 4 video:

- Con 3,4 o 5 persone che entrano sequenzialmente nell'inquadratura si riesce ad eseguire un analisi in tempo reale con qualsiasi indice di movimento.
- Con 5 persone che entrano nell'inquadratura tutte insieme già nel primo frame e indice di movimento a 0.5, il primo frame impiega 1,63 secondi per essere acquisito, analizzato e memorizzato. I frame successivi vengono analizzati in tempo reale.

È importante chiarire che l'espressione "analisi in tempo reale" non implica necessariamente che l'elaborazione di ogni frame avvenga in meno di un secondo. Piuttosto, mediante la selezione e lo scarto dei frame privi di un indice di movimento significativo, il sistema è in grado di processare i frame rilevanti in maniera efficiente e senza latenza.
In tutti i casi, le persone presenti nel video vengono rilevate correttamente.

Chapter 6

AI Act

L'AI Act, o "Artificial Intelligence Act", è una proposta legislativa avanzata dall'Unione Europea con l'obiettivo di regolamentare l'uso e lo sviluppo dell'intelligenza artificiale (IA) all'interno dei suoi stati membri. Rappresenta uno dei primi tentativi su larga scala di stabilire un quadro giuridico globale per affrontare le sfide etiche, di sicurezza e di privacy legate alle tecnologie di intelligenza artificiale. Il focus principale dell'AI Act è su come l'IA può essere utilizzata in modo responsabile, garantendo al contempo che le innovazioni nel campo non vengano ostacolate da regolamentazioni eccessivamente restrittive.

6.1 Categorie di rischio

L'AI Act [25] classifica i sistemi di IA in quattro categorie di rischio: inaccettabile, alto, limitato e minimo. Questa classificazione determina il livello di regolamentazione e i requisiti di conformità che ciascun sistema di IA deve soddisfare prima di essere immesso sul mercato dell'UE. Le pratiche considerate ad alto rischio comprendono, ad esempio, i sistemi di IA utilizzati per la valutazione e la classificazione delle persone in contesti che possono influenzare significativamente le loro vite, come l'istruzione, l'impiego e l'applicazione della legge.
In particolare, l'AI Act vieta l'uso di sistemi di IA che possono manipolare gli individui oltre la loro consapevolezza o che sfruttano le vulnerabilità delle persone per causare danni, oltre a stabilire divieti specifici per pratiche quali la sorveglianza di massa e il punteggio sociale come in Cina. Il regolamento prevede anche meccanismi di trasparenza e di sorveglianza per i sistemi di IA ad alto rischio, richiedendo valutazioni di impatto e registrazioni adeguate.

6.2 Articolo 5: Pratiche di Intelligenza Artificiale Vietate

La gestione della videosorveglianza e dei dati biometrici nell'ambito dell'AI Act dell'Unione Europea si concentra in modo significativo sulla protezione dei diritti fondamentali dei cittadini, in particolare per quanto riguarda la privacy e la protezione dei dati personali. L'articolo 5 del regolamento stabilisce chiare linee guida e restrizioni sull'uso dell'intelligenza artificiale in relazione alla raccolta, elaborazione e utilizzo di dati biometrici attraverso sistemi di videosorveglianza.

6.2.1 Videosorveglianza in Tempo Reale e Identificazione Biometrica

Il regolamento (art.5 comma d) [26] impone restrizioni severe sull'uso di sistemi di intelligenza artificiale per l'identificazione biometrica in tempo reale in spazi pubblicamente accessibili. Questo utilizzo è consentito solo in circostanze eccezionali e per scopi specifici, come la ricerca di vittime di rapimenti, indificazione di sospettati di reati gravi o la prevenzione di minacce gravi alla sicurezza pubblica, inclusi attacchi terroristici. Ogni impiego di tali tecnologie deve essere strettamente necessario, proporzionato e limitato nel tempo, nello spazio e nei soggetti coinvolti.

6.2.2 Divieti Specifici

L'AI Act vieta sistemi di categorizzazione biometrica che deducono attributi sensibili (razza, opinioni politiche, appartenenza sindacale, convinzioni religiose o filosofiche, vita sessuale o orientamento sessuale), ad eccezione dell'etichettatura o del filtraggio di set di dati biometrici acquisiti legalmente o quando le forze dell'ordine classificano dati biometrici. Vieta anche la compilazione di database di riconoscimento facciale mediante raccolta non mirato di immagini facciali da Internet o filmati CCTV.

6.2.3 Obblighi e autorizzazioni

Le organizzazioni che implementano sistemi di videosorveglianza alimentati da IA per l'identificazione o la categorizzazione biometrica devono aderire a rigorosi obblighi di trasparenza, sicurezza dei dati e valutazione dell'impatto sulla protezione dei dati. È necessario ottenere previe autorizzazioni da autorità giudiziarie o amministrative indipendenti, che valuteranno la necessità e la proporzionalità dell'uso della tecnologia in questione. Inoltre, è richiesta una chiara informativa ai soggetti interessati, a meno che non siano applicabili specifiche esenzioni per motivi di sicurezza nazionale o ordine pubblico.

6.2.4 Entrata in vigore in Italia

L'AI Act è stato approvato dal Parlamento europeo e dal Consiglio il 6 dicembre 2023. Tuttavia, non è ancora in vigore. La data di entrata in vigore dell'AI Act è fissata per il 20 aprile 2025. A partire da questa data, i sistemi di intelligenza artificiale (IA) dovranno essere conformi ai requisiti del regolamento per poter essere immessi sul mercato europeo. L'Italia ha 2 anni di tempo dall'entrata in vigore dell'AI Act per recepire il regolamento nella propria legislazione nazionale. Questo significa che la legge italiana per l'IA conforme all'AI Act dovrà essere emanata entro il 20 aprile 2027.
Al momento, non è ancora stata definita una data precisa per il rilascio della legge italiana per l'IA conforme all'AI Act. Tuttavia, il Ministero dello Sviluppo Economico ha già avviato i lavori per la predisposizione del testo di legge.
Il processo di recepimento dell'AI Act in Italia prevede diverse fasi:

1. Analisi del regolamento europeo da parte del Ministero dello Sviluppo Economico.
2. Consultazione degli stakeholder (imprese, associazioni di consumatori, esperti di IA, ecc.).
3. Predisposizione del testo di legge da parte del Ministero dello Sviluppo Economico.
4. Approvazione del testo di legge da parte del Parlamento italiano.
5. Emanazione della legge da parte del Presidente della Repubblica.

È possibile rimanere aggiornati sui progressi del recepimento dell'AI Act in Italia consultando il sito web del Ministero dello Sviluppo Economico:
https://mimit.gov.it/.
Oppure, per eventuali modifiche sul sito ufficiale dell'unione europea:
https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

Conclusioni

La realizzazione di un sistema automatizzato per il rilevamento e il riconoscimento facciale di intrusi comporta la considerazione di molteplici fattori critici, quali l'affidabilità e la precisione dei risultati, la rapidità di elaborazione dei dati e la tutela della privacy degli individui rilevati. Il presente lavoro di tesi evidenzia come il rilevamento e il riconoscimento facciale siano campi di ricerca e sviluppo dinamici, capaci di fornire già oggi prestazioni notevoli. I capitoli 3 e 4 dimostrano come, grazie all'aumento esponenziale della disponibilità di dati, si sia assistito a un miglioramento costante della precisione dei modelli e dell'efficienza operativa. L'evoluzione del modello YOLO, che ha visto l'introduzione di otto versioni principali dal 2015 al 2023, senza considerare rilasci minori, ne è un esempio emblematico.

Il capitolo dell'analisi sperimentale ha dimostrato che il sistema è in grado di riconoscere volti con un'accuratezza quasi perfetta su un ampio spettro di soggetti. Tuttavia, la precisione assoluta non è garantita in ogni contesto, sottolineando la necessità di un controllo umano per la convalida dei risultati. Infatti, la sfida principale è l'individuazione e riconoscimento di volti in situazioni difficili o estreme come occlusioni, immagini degradate o a bassa risoluzione. L'adozione di tecnologie avanzate, quali YOLOv8 per il rilevamento e AdaFace per il riconoscimento, ha permesso di superare molte di queste difficoltà. Il nostro studio si è concentrato sull'impiego di immagini di sorveglianza a bassa risoluzione, raggiungendo tassi di rilevamento elevati.

Un ulteriore traguardo prefissato era l'analisi delle immagini in tempo reale, specificatamente un frame al secondo, adeguato al nostro caso d'uso, ovvero un luogo chiuso e poco affollato come può essere una banca, un ufficio postale o l'ingresso di un museo. I risultati ottenuti attestano che la capacità di elaborazione del processore è cruciale per il successo di questo obiettivo. Infatti, avere la possibilità di poter analizzare diversi frame al secondo permette di migliorare sia la precisione del sistema nel suo compito ma anche di poter visualizzare in tempo reale le immagini classificate senza scatti ma come un video in streaming.

In sintesi, i risultati conseguiti confermano il successo degli obiettivi stabiliti, evidenziando il raggiungimento di un equilibrio ottimale tra prestazioni elevate e requisiti hardware non eccessivamente onerosi come può essere l'adozione di una GPU dedicata.

Un aspetto cruciale per l'impiego di questa tecnologia è la conformità alle leggi sulla privacy relative ai soggetti raffigurati nelle scene. In quanto applicazione destinata ad ambienti pubblici, è imperativo considerare con attenzione la normativa vigente nel paese di utilizzo. Nel contesto del nostro studio, è necessario aderire all'AI Act, che disciplina l'uso di tali tecnologie limitandole a circostanze particolari, ad esempio per questioni di sicurezza pubblica. Pertanto, il caso d'uso esplorato in questa tesi, che include l'impiego in ambienti come banche, musei o altri luoghi pubblici non affollati, deve essere attentamente valutato in base alle restrizioni imposte dall'AI

Act, che potrebbe proibire l'uso di tecnologie di riconoscimento facciale in tali contesti. Anche se l'AI Act non è ancora entrato in vigore, è prudente considerarlo nell'attuale fase di sviluppo di nuove applicazioni e nell'adeguamento di quelle esistenti, dato che l'adozione della regolamentazione italiana corrispondente è imminente. Questo assicurerà che le applicazioni siano conformi ai futuri requisiti normativi, facilitando una transizione senza intoppi quando le leggi saranno ufficialmente applicate.

Bibliography

[1] S.Haykin. *Neural Networks and Learning Machines*. Pearson College Div, 2008.

[2] F. Rosenblatt. "The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain". In: *Psychological Review* 65.6 (1958), pp. 386–408.

[3] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[4] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV].

[5] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV].

[6] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[7] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

[8] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 21–37. ISBN: 9783319464480. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[9] Joseph Redmon and Ali Farhadi. *YOLO9000: Better, Faster, Stronger*. 2016. arXiv: 1612.08242 [cs.CV].

[10] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[11] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

[12] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *Scaled-YOLOv4: Scaling Cross Stage Partial Network*. 2021. arXiv: 2011.08036 [cs.CV].

[13] Jacob Solawetz. *What is YOLOv5? A Guide for Beginners*. June 2020. URL: https://blog.roboflow.com/yolov5-improvements-and-evaluation/.

[14] glenn-jocher and sergiuwaxmann. *Ultralytics YOLOv5 Architettura*. 2024. URL: https://docs.ultralytics.com/it/yolov5/tutorials/architecture_description/#44-build-targets.

[15] Chuyi Li et al. *YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications*. 2022. arXiv: 2209.02976 [cs.CV].

[16] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 2022. arXiv: 2207.02696 [cs.CV].

[17] Jacob Solawetz and Francesco. *What is YOLOv8? The Ultimate Guide*. Jan. 2023. URL: https://blog.roboflow.com/whats-new-in-yolov8/ (visited on 2024).

[18] Kaipeng Zhang et al. "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks". In: *IEEE Signal Processing Letters* 23.10 (Oct. 2016), pp. 1499–1503. ISSN: 1558-2361. DOI: 10.1109/lsp.2016.2603342. URL: http://dx.doi.org/10.1109/LSP.2016.2603342.

[19] Minchul Kim, Anil K. Jain, and Xiaoming Liu. *AdaFace: Quality Adaptive Margin for Face Recognition*. 2023. arXiv: 2204.00964 [cs.CV].

[20] Jiankang Deng et al. "ArcFace: Additive Angular Margin Loss for Deep Face Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.10 (Oct. 2022), pp. 5962–5979. ISSN: 1939-3539. DOI: 10.1109/tpami.2021.3087709. URL: http://dx.doi.org/10.1109/TPAMI.2021.3087709.

[21] Yuge Huang et al. *CurricularFace: Adaptive Curriculum Learning Loss for Deep Face Recognition*. 2020. arXiv: 2004.00288 [cs.CV].

[22] Minchul Kim. *Adaface github project*. 2023. URL: https://github.com/mk-minchul/AdaFace.

[23] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].

[24] Atul Anand. *LFW - People (Face Recognition*. 2020. URL: https://www.kaggle.com/datasets/atulanandjha/lfwpeople.

[25] *High-level summary of the AI Act*. 2024. URL: https://artificialintelligenceact.eu/high-level-summary/.

[26] *Article 5: Prohibited Artificial Intelligence Practices*. 2024. URL: https://artificialintelligenceact.eu/article/5/.

List of Figures