POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Matematica

TESI DI LAUREA MAGISTRALE

SVILUPPO DI UN SISTEMA ALPR IN TEMPO REALE PER DISPOSITIVI EDGE

Confronto e Ottimizzazione di YOLOX, RT-DETR v2 e CCT

Relatore:
Prof. Alessandro Fiori

Candidato:
Davide Lavezzini
Matricola: S334104

Supervisore Aziendale:
Giulio Desana

Anno Accademico 2024/2025

Abstract

Il presente lavoro introduce un sistema di *Automatic License Plate Recognition* (ALPR) ottimizzato per il contesto operativo europeo, con particolare focus sul traffico veicolare italiano e sui flussi transfrontalieri verso/da paesi limitrofi (Francia, Germania, Svizzera, Austria). Superando i limiti delle soluzioni commerciali generaliste, il sistema proposto integra un detector **RT-DETR v2** (Real-Time Detection Transformer) e un riconoscitore **Compact Convolutional Transformer** (CCT), orchestrati in una pipeline a tre stadi che include la discriminazione della nazionalità tramite **MobileNetV3**.

Il sistema, addestrato su un dataset ibrido (reale + sintetico) rappresentativo della distribuzione reale del traffico italiano (70% targhe nazionali, 30% estere in transito), raggiunge un'accuratezza elevata per ognuno dei 3 stadi, mantenendo una latenza inferiore ai **200 ms** su hardware edge di fascia media. I risultati sperimentali dimostrano che la modellazione esplicita delle dipendenze globali tramite self-attention permette di risolvere casi critici di occlusione e distorsione prospettica, garantendo un'affidabilità superiore (AR@100 $> 91\%$) rispetto alle baseline CNN tradizionali (YOLOX), rendendo la soluzione idonea per scenari di controllo accessi urbani (ZTL) e monitoraggio autostradale *free-flow*.

Indice

Elenco delle figure

Elenco delle tabelle

1. Introduzione

1.1 Contesto e motivazioni

L'evoluzione globale verso le *smart cities* e l'*Internet of Things (IoT)* ha consolidato il ruolo del Riconoscimento Automatico delle Targhe (ALPR) come componente strutturale critico per il controllo degli accessi, il pedaggiamento automatizzato, la sicurezza pubblica e l'analisi forense. Con un parco veicolare globale che supera 1.5 miliardi di unità, il mercato ALPR ha oltrepassato i 3.5 miliardi di dollari nel 2024, sostenuto dalla crescente domanda di automazione. Nonostante oltre due decenni di ricerca, l'ALPR permane un problema aperto all'intersezione tra *visione artificiale* (*computer vision*), *pattern recognition* e *Sistemi Intelligenti di Trasporto (ITS)*.

Il paradigma della visione artificiale ha subito una svolta radicale con l'avvento del **Deep Learning**. L'evoluzione dalle tecniche di estrazione manuale delle feature (come SIFT e HOG) alle architetture neurali end-to-end ha permesso di affrontare con successo la complessità degli ambienti non controllati. Tuttavia, sfide quali variazioni di illuminazione, distorsioni prospettiche e risorse computazionali limitate richiedono soluzioni architetturali specificamente ottimizzate per l'edge computing.

Una distinzione fondamentale va posta rispetto ai classici sistemi OCR (*Optical Character Recognition*). Mentre questi ultimi sono ottimizzati per documenti scansionati ad alto contrasto, il contesto ALPR presenta una complessità visiva radicalmente diversa: la targa occupa una frazione minima dell'immagine ed è immersa in un ambiente rumoroso. È pertanto indispensabile un primo stadio di *object detection* che agisca come meccanismo di attenzione selettiva, isolando la regione di interesse per demandare al riconoscitore solo i dati pertinenti.

L'obiettivo di questa tesi è formalizzare e risolvere il compromesso tra latenza e accuratezza mediante l'adozione di una pipeline ibrida. Nello specifico, si propone l'utilizzo di **RT-DETR v2** (Real-Time DEtection TRansformer) come evoluzione rispetto allo standard **YOLOX**, unendo la robustezza globale dell'attenzione alla velocità necessaria per applicazioni real-time.

1.2 Dominio applicativo e requisiti

Nell'ambito della mobilità privata, il sistema trova applicazione nella gestione di **parcheggi automatizzati**, consentendo il controllo accessi *ticketless* e la gestione

di *white-list* per complessi residenziali e aziendali. Parallelamente, nel controllo del territorio, la tecnologia è abilitante per il monitoraggio delle **Zone a Traffico Limitato (ZTL)**, dove la discriminazione precisa della nazionalità è critica per il sanzionamento automatico, e per l'integrazione in **telecamere stradali** dedicate allo *speed enforcement* (tutor, autovelox) e al rilevamento di infrazioni semaforiche. Oltre alle finalità sanzionatorie, il sistema supporta la pianificazione urbana tramite l'**analisi del traffico**, offrendo metriche granulari di conteggio e tracking dei flussi, e potenzia la pubblica sicurezza facilitando la **ricerca veicoli** tramite matching in tempo reale contro database di mezzi segnalati. Infine, la robustezza agli agenti atmosferici rende la soluzione idonea per il **pedaggiamento automatico (Free Flow)** in autostrada, eliminando la necessità di barriere fisiche.

Per quanto riguarda la composizione del dataset, si è partiti da una base di dati reale costituita da circa 9.300 immagini di targhe italiane (utilizzate per detection e riconoscimento) integrata da ulteriori 1.000 campioni internazionali per il training del classificatore di nazionalità. Parallelamente, per l'addestramento su dati sintetici, è stata modellata una distribuzione delle nazionalità che riflette i flussi di traffico transfrontaliero verso l'Italia: 70% veicoli nazionali, seguiti da Germania (8%), Francia (7%), Svizzera (4%), Austria (3%) e altri paesi UE ed extra-UE (8%). Questa strategia duale garantisce che il sistema sia robusto sulle frequenze d'uso reali pur mantenendo capacità di generalizzazione sui transiti internazionali rari.

1.3 Sfide tecniche e stato dell'arte

L'evoluzione dello stato dell'arte nell'*object detection* ha attraversato diverse fasi (figura 1.1), passando dalle architetture *Two-Stage* ai detector *One-Stage* (famiglia YOLO), fino all'attuale adozione dei Transformer (DETR, RT-DETR) che introducono la modellazione delle dipendenze globali.

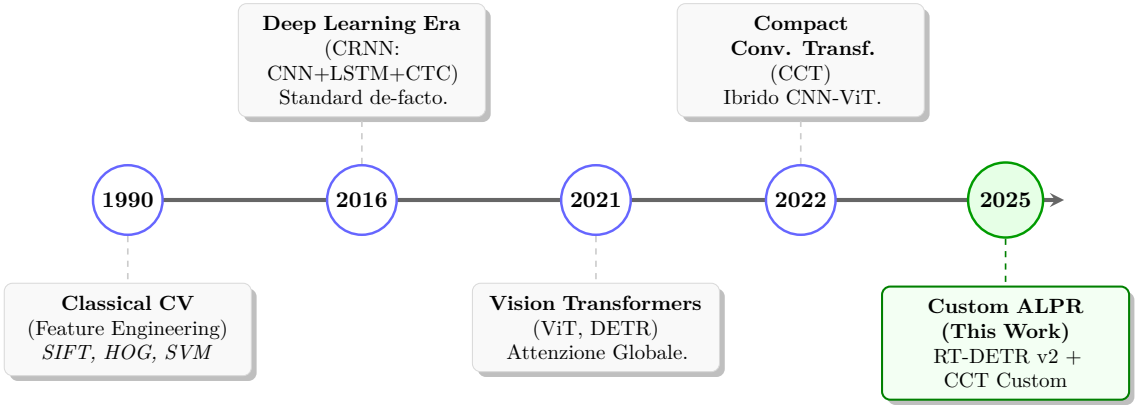


Figura 1.1: *Evoluzione dei paradigmi di riconoscimento: dal Feature Engineering al Deep Learning, fino all'integrazione di detector Transformer e riconoscitori ibridi.*

YOLOX è stato scelto come *baseline* rappresentativa del paradigma CNN anchor-free, in virtù della sua licenza Apache 2.0 che lo rende l'unica alternativa open-source valida per contesti industriali. Tale scelta si pone in necessaria contrapposizione con il

filone evolutivo gestito dall'azienda Ultralytics: a partire da YOLOv5 e proseguendo attraverso le versioni successive fino alla recente v12, questa famiglia di modelli è infatti vincolata da licenze AGPL-3.0 (o Enterprise), che ne precludono l'integrazione libera in software commerciale chiuso.

1.4 Contributi della tesi

Per superare i limiti di scalabilità delle architetture monolitiche, questa tesi propone una *pipeline* modulare a tre stadi (fig 1.2). Il lavoro presenta contributi metodologici e ingegneristici volti a definire criteri rigorosi per la selezione di architetture neurali in contesti *data-limited*: si fornisce la prima validazione sistematica di un detector transformer *NMS-free* (RT-DETR) per il task ALPR, dimostrando la superiorità del matching deterministico. Parallelamente, viene formalizzato un protocollo di data augmentation basato su *Vicinal Risk Minimization* per dataset ibridi, essenziale per colmare il gap sim-to-real. Infine, il lavoro definisce un modello analitico per la stima delle prestazioni end-to-end e del budget di latenza, fornendo i criteri fondamentali per il dimensionamento su hardware edge.

Figura 1.2: *Schema funzionale della pipeline proposta.*

1.5 Organizzazione del Documento

Il documento è organizzato per guidare il lettore dai fondamenti teorici alla validazione sperimentale. Il capitolo 2 ('Fondamenti') stabilisce le basi matematiche, il capitolo 3 ('Design') descrive l'architettura proposta, il capitolo 4 ('Metodologia') raccoglie i dettagli implementativi e i protocolli di addestramento, il capitolo 5 presenta l'analisi sperimentale e i risultati. Infine, il capitolo 6 ('Conclusioni') riassume i contributi del lavoro e delinea le promettenti direzioni di ricerca futura.

2. Fondamenti matematici e metodologie di training

La presente trattazione stabilisce il framework su cui si fonda il sistema proposto, definendo i principi dell'algebra tensoriale, della teoria dell'approssimazione e delle tecniche di ottimizzazione stocastica necessari alla comprensione delle architetture neurali ibride discusse nel seguito.

2.1 Rappresentazioni tensoriali e geometria del dato

Nelle moderne architetture di Deep Learning, l'informazione viene codificata e trasformata attraverso strutture dati note come **tensori**, che costituiscono la generalizzazione naturale di scalari, vettori e matrici a spazi N-dimensionali. Sia $p \in \mathbb{N}$ l'ordine (o rango) del tensore e siano $d_k \in \mathbb{N}^+$ le cardinalità lungo il k-esimo asse (o modo). Un generico tensore $\mathcal{T}$ è definito come un elemento dello spazio vettoriale reale costituito dal prodotto cartesiano delle dimensioni specificate:

$$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_p}$$

L'accesso agli elementi costitutivi avviene tramite indicizzazione posizionale. Nel dominio della computer vision, si opera tipicamente su tensori di rango 3 o 4: il rango 3 è utilizzato per rappresentare un singolo dato (altezza, larghezza, canali), mentre il rango 4 aggiunge la dimensione del *batch* per l'elaborazione parallela stocastica (batch, canali, altezza, larghezza).

In questo lavoro si adotta la convenzione *channel-first*, standard nei principali framework di calcolo differenziabile. Siano C il numero di canali spettrali (e.g., $C = 3$ per lo spazio colore RGB), e H, W rispettivamente l'altezza e la larghezza spaziale dell'immagine. Un singolo dato visivo è rappresentato da un tensore $\mathcal{X}$ tale che:

$$\mathcal{X} \in \mathbb{R}^{C \times H \times W}$$

dove l'elemento scalare $X_{c,h,w}$ corrisponde all'intensità del pixel alle coordinate spaziali (h, w) nel canale c. Durante la fase di addestramento, l'ottimizzazione stocastica richiede l'elaborazione simultanea di gruppi di campioni (*mini-batches*).

Siano N la dimensione del lotto e $\mathcal{B}$ il tensore risultante dall'aggregazione lungo una nuova dimensione primaria. La struttura dati di rango 4 è definita come:

$$\mathcal{B} \in \mathbb{R}^{N \times C \times H \times W}$$

In tale contesto, l'indice $B_{n,c,h,w}$ identifica univocamente il valore del pixel (h, w) sul canale c relativo all'n-esimo campione del batch considerato.

La manipolazione algebrica di tali strutture all'interno del grafo computazionale si basa su due categorie di operazioni fondamentali:

- **Reshaping ($\mathcal{R}$):** trasforma la porzione di un'immagine in un vettore piatto (flattening) per essere processato dal classificatore (da un input 2D di feature spaziali ad un input per trasformer), formalmente è un isomorfismo lineare $\phi : \mathbb{R}^{d_1 \times \cdots \times d_p} \to \mathbb{R}^{d'_1 \times \cdots \times d'_q}$ che conserva la cardinalità totale ($\prod d_i = \prod d'_j$) e l'ordinamento in memoria (C-contiguous):

$$\mathcal{T}' = \phi(\mathcal{T}) \iff \text{vec}(\mathcal{T}') = \text{vec}(\mathcal{T})$$

- **Broadcasting ($\mathcal{B}$):** formalmente è un meccanismo di espansione implicita che adatta le dimensioni di due tensori $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ e $\mathcal{B} \in \mathbb{R}^{k_1 \times \cdots \times k_N}$ per operazioni *element-wise*, per esempio è un "trucco" per sommare vettori e tensori. Date le dimensioni allineate a destra, l'operazione è valida se per ogni asse i:

$$d_i = k_i \vee \{d_i = 1 \vee k_i = 1\}$$

  Nel caso $d_i \neq k_i$ (e uno dei due è 1), la dimensione unitaria viene replicata virtualmente per corrispondere all'altra.

2.2 Elementi architetturali per la visione artificiale

L'unità fondamentale del Deep Learning, il **Percettrone** (Rosenblatt, 1958), e la sua evoluzione multi-strato (**MLP**), hanno posto le basi per l'approssimazione universale di funzioni continue. Tuttavia, l'applicazione di queste strutture a dati visivi ad alta dimensionalità si scontra con l'esplosione combinatoria dei parametri (milioni nel primo layer) e la perdita della topologia spaziale.

Per preservare e sfruttare la struttura a griglia delle immagini, le **Reti Neurali Convoluzionali (CNN)** introducono l'operazione di *convoluzione discreta*, definita come:

$$\mathcal{Y}_{i,j,m} = b_m + \sum_{c=1}^{C_{in}} \sum_{u,v} \mathcal{X}_{i+u,\, j+v,\, c} \cdot \mathcal{K}_{u,\, v,\, c,\, m} \tag{2.1}$$

Tale operatore introduce due *bias induttivi* fondamentali: la **connessione locale** (sparse interaction) e la **condivisione dei pesi** (parameter sharing), che garantiscono l'invarianza per traslazione e un'efficienza parametrica cruciale per l'elaborazione di immagini, con lo stesso filtro K viene usato su tutta l'immagine, per esempio se un filtro impara a riconoscere il bordo di un numero "5" in alto a sinistra, saprà riconoscerlo ovunque nella targa.

Per astrarre progressivamente le caratteristiche riducendo la risoluzione spaziale, le CNN alternano stadi di convoluzione a operatori di **pooling** (sottocampionamento).

Definizione 2.1 (Pooling)**.** *L'operazione di pooling aggrega l'informazione locale in un intorno $\Omega_{i,j}$ per produrre una rappresentazione a risoluzione ridotta, introducendo invarianza a piccole traslazioni. Le varianti più comuni sono il* ***max pooling****, che estrae il valore di attivazione massimo ($\mathcal{Y}_{i,j} = \max_{(u,v)\in\Omega_{i,j}} \mathcal{X}_{u,v}$), e l'****average pooling****, che calcola la media aritmetica.*

Tuttavia, l'operazione di sottocampionamento (stride $s > 1$) intrinseca nel pooling pone sfide teoriche legate al **Teorema del campionamento di Nyquist-Shannon**. Il teorema stabilisce che, per evitare distorsioni informative (*aliasing*) in un segnale discretizzato, la frequenza di campionamento deve essere strettamente maggiore del doppio della frequenza massima contenuta nel segnale stesso ($\omega_s > 2\omega_{max}$). Il max pooling convenzionale viola spesso questa condizione poiché l'operatore max è non-lineare e non applica un adeguato filtraggio passa-basso prima della decimazione, preservando componenti ad alta frequenza che vengono ripiegate nello spettro del segnale ridotto. Questo fenomeno causa la perdita della proprietà di equivarianza agli shift: piccole traslazioni dell'input possono generare output drasticamente diversi. Una soluzione robusta è rappresentata dal **maxblur pooling** (Zhang, 2019), che disaccoppia la massimizzazione dal sottocampionamento inserendo un filtro passa-basso antialiasing, più robusto a movimento veicolo grazie a *Blur* filtro sfocante:

$$\text{MaxBlurPool}(x) = \text{Subsample}(\text{Blur}(\text{Max}(x))) \tag{2.2}$$

L'evoluzione della rete segue regole geometriche precise: dati dimensione input W_{in}, kernel k, padding p e stride s, la dimensione di output risulta $W_{out} = \lfloor (W_{in} - k + 2p)/s \rfloor + 1$. Parallelamente, il campo recettivo efficace r_l cresce esponenzialmente con la profondità. Definito il *jump* j_l come prodotto cumulativo degli stride ($j_l = j_{l-1} \cdot s_l$), la legge di accrescimento è $r_l = r_{l-1} + (k_l - 1)j_{l-1}$, permettendo ai layer profondi di catturare relazioni semantiche globali, in quanto nei primi layer la rete vede solo bordi o angoli dei caratteri, mentre negli ultimi layer, il "campo recettivo" è abbastanza grande da vedere l'intera targa e capire il contesto (es. nazionalità).

Teorema 2.1 (Analisi della complessità)**.** *La quantificazione della complessità computazionale si basa sulla metrica dei* ***FLOPs*** *(Floating Point Operations), definita come il numero cumulativo di operazioni elementari (moltiplicazioni e addizioni) in virgola mobile necessarie per eseguire un passaggio di inferenza. Tale misura fornisce una stima teorica del carico di lavoro algoritmico, indipendente dalla specifica piattaforma hardware. Considerando un tensore di input $\mathcal{X} \in \mathbb{R}^{H\times W\times C}$, una convoluzione con kernel $k \times k$ e D filtri di uscita, le risorse richieste sono:*

- *Costo computazionale (FLOPs): $O(H \cdot W \cdot k^2 \cdot C \cdot D)$, scalare linearmente con la risoluzione spaziale.*
- *Parametri (weights): $O(k^2 \cdot C \cdot D)$, indipendente dalla dimensione spaziale dell'immagine grazie al weight sharing, che dipendono dai filtri.*
- *Occupazione di memoria: $O(H \cdot W \cdot D)$, dominata dalle attivazioni intermedie necessarie per la backpropagation piuttosto che dai pesi.*

Operativamente, l'architettura CNN agisce come un encoder gerarchico $\mathcal{E}(\cdot)$ che trasforma il tensore di input $\mathcal{X}_{RGB} \in \mathbb{R}^{H\times W\times 3}$ in un tensore di feature astratte (*feature map*) $\mathcal{Z} \in \mathbb{R}^{H'\times W'\times D}$. A differenza degli approcci classici di computer vision (e.g., SIFT, HOG) dove l'estrazione è definita a priori (*hand-crafted*), nella CNN i filtri $\mathcal{K}$ sono parametri apprendibili che si specializzano per rilevare pattern discriminativi. Il tensore risultante $\mathcal{Z}$ costituisce la *materia prima* per i moduli decisionali successivi e viene modellato secondo due paradigmi principali:

1. Vettorizzazione (**Flattening/Global pooling**): operazione $\phi : \mathbb{R}^{H'\times W'\times D} \to \mathbb{R}^{D}$ (nel caso di GAP) o $\to \mathbb{R}^{H'W'D}$ (nel caso di Flatten). Questa trasformazione proietta le feature spaziali in un unico vettore denso, perdendo parzialmente o totalmente la struttura topologica 2D, al fine di alimentare i layer *fully-connected* finali per compiti di classificazione globale.

2. Sequenzializzazione (**Tokenization**): la mappa di feature 2D viene tagliata in piccoli pezzi (token) e allineata in una sequenza, in particolare è un'operazione $\psi : \mathbb{R}^{H'\times W'\times D} \to \mathbb{R}^{L\times D}$ con $L = H' \cdot W'$. Il tensore viene rimodellato in una sequenza di L vettori (token) di dimensione D. Questa trasformazione è fondamentale per rendere il dato immagine compatibile con i meccanismi di *Self-Attention*, i quali operano nativamente su insiemi o sequenze discrete piuttosto che su griglie rigide, perciò son importanti per trovare relazioni tra elementi di una sequenza.

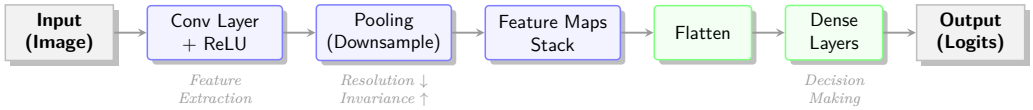


Figura 2.1: *Schema logico di una architettura CNN standard. Il flusso trasforma i dati da una rappresentazione spaziale ad alta risoluzione a una rappresentazione semantica astratta.*

Mentre le CNN eccellono nell'estrazione di feature spaziali parallele, le Reti Neurali Ricorrenti (RNN) sono state storicamente l'architettura di riferimento per modellare dipendenze dinamiche in dati sequenziali (serie temporali, testo). Una RNN elabora una sequenza di input $\mathbf{x}_1, \ldots, \mathbf{x}_T$ mantenendo uno *stato nascosto* $\mathbf{h}_t \in \mathbb{R}^k$ che funge da memoria a breve termine del contesto passato.

Definizione 2.2 (Stato ricorrente)**.** *Al passo temporale t, lo stato $\mathbf{h}_t$ è aggiornato combinando linearmente l'input corrente $\mathbf{x}_t$ e lo stato precedente $\mathbf{h}_{t-1}$, seguiti da una non-linearità* tanh*:*

$$\mathbf{h}_t = \tanh(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}) \tag{2.3}$$

Dove $\mathbf{W}_{hh}$ e $\mathbf{W}_{xh}$ sono matrici di pesi condivise temporalmente. Questa ricorrenza permette di mappare sequenze di lunghezza variabile in rappresentazioni fisse, ma introduce sfide critiche siccome non si può parallelizzare il calcolo (è lento) e il segnale del gradiente "svanisce", motivando la transizione verso architetture attention-based.

2.2.1 Backbone, head e fine-tuning: anatomia e adattamento dei modelli

L'ingegnerizzazione di sistemi di *Deep Learning* complessi come quelli proposti in questa tesi richiede una scomposizione modulare delle architetture neurali, basata sui paradigmi di **backbone** (estrazione di feature) e **head** (logica decisionale), tale distinzione definisce la gerarchia del flusso informativo e la strategia di addestramento mediante il *transfer learning*.

Il *backbone* costituisce il substrato fondamentale della rete, deputato all'apprendimento di rappresentazioni gerarchiche dei dati in ingresso. Esso agisce come un operatore $\Phi : \mathbb{R}^{C\times H\times W} \to \mathbb{R}^{d\times h\times w}$ che trasforma l'immagine RGB di dati grezzi in un tensore di feature latenti di alta dimensionalità, con d è la profondità (numero di feature) e h, w sono le dimensioni spaziali ridotte ma ricche di informazione semantica. Nelle fasi iniziali del backbone, i filtri convoluzionali si specializzano nel rilevamento di primitive geometriche (bordi, angoli), mentre negli strati più profondi la rete cattura relazioni semantiche globali e concetti complessi. In questa tesi, si esplorano diversi backbone ottimizzati per l'efficienza: da un lato, le architetture *transformer-ready* come **ResNet-vd** e **HGNetv2** (cuore di RT-DETR v2), che rappresentano lo stato dell'arte per l'inferenza su GPU e sono progettate per preservare la risoluzione spaziale necessaria ai meccanismi di attenzione globale. Dall'altro, si considera la **CSPDarknet** (spina dorsale di YOLOX), che incarna l'apice dell'era CNN *pura*: sebbene estremamente efficiente, la sua natura congelata al 2021 non le permette di beneficiare della moderna distillazione da Vision Foundation models, segnando un limite alla sua scalabilità futura rispetto alle controparti ibride.

La *head* (o testa) è la componente terminale dell'architettura che riceve le mappe di feature prodotte dal backbone e le proietta nello spazio delle soluzioni per il compito richiesto. Nella *object detection*, la testa deve risolvere contemporaneamente problemi di classificazione e regressione (*bounding box*), mentre nel riconoscimento OCR deve mappare tratti visivi in sequenze di caratteri. Un'innovazione significativa discussa nel seguito (sez. 2.6) è la **Decoupled head**, che separa il calcolo dei logit di classe (classificazione) dalla localizzazione geometrica (regressione), riducendo i conflitti durante l'ottimizzazione.

L'addestramento di architetture profonde da zero (*from scratch*) richiede dataset di scala massiva (e.g., ImageNet, COCO) e risorse di calcolo ingenti, nel contesto del riconoscimento targhe, dove i dataset specifici possono essere limitati o presentare un forte bias di dominio, si adotta il paradigma del transfer learning e del successivo fine-tuning. Si sfrutta l'ipotesi che le feature estratte dai primi layer di un backbone addestrato su domini generalisti siano universali e riutilizzabili, il processo di ottimizzazione diventa:

$$\theta^* = \arg\min_{\theta} \mathcal{L}_{target}(f(X_{target}; \theta_{pretrained}))$$

Esso prevede l'inizializzazione del modello con pesi pre-addestrati e la successiva specializzazione (*adaptation*) sui dati target, questo approccio garantisce questi vantaggi fondamentali per il sistema proposto:

1. Convergenza accelerata: il punto di partenza nello spazio dei parametri è già prossimo a un minimo locale significativo.
2. Regolarizzazione implicita: la conoscenza pregressa agisce come un vincolo strutturale che previene l'*overfitting* su piccoli campioni.

2.3 Dinamiche di addestramento e modellazione architetturale

Il presente capitolo introduce i fondamenti teorici dell'ottimizzazione non convessa e della modellazione di sequenze, fornendo il substrato matematico necessario a giustificare le scelte progettuali adottate in questa tesi (specificamente l'uso di transformer/CCT per l'OCR). L'analisi si concentra su due aspetti critici: la stabilità numerica del flusso dei gradienti in architetture profonde e i vantaggi computazionali della parallelizzazione rispetto ai modelli ricorrenti.

L'evoluzione dello stato dell'arte nel riconoscimento di sequenze (*Sequence-to-Sequence learning*) è stata storicamente guidata dalla necessità di mitigare il fenomeno del **dissolvimento del gradiente** (*vanishing gradient*) e di massimizzare l'efficienza su hardware parallelo (GPU). Mentre le Reti Neurali Ricorrenti (RNN/LSTM) elaborano i dati sequenzialmente, introducendo una dipendenza temporale $t \to t+1$ che ostacola la parallelizzazione, le architetture **transformer** adottano un meccanismo di *self-attention*, riducendo la lunghezza del percorso di dipendenza (*credit assignment path*), stabilizzando il gradiente e abilitando l'elaborazione simultanea dell'intera sequenza.

2.3.1 Backpropagation e analisi della stabilità numerica

Sia $\mathcal{L}(\theta)$ la funzione di costo scalare da minimizzare rispetto ai parametri θ della rete. In un'architettura profonda composta da L strati, denotiamo con $\mathbf{h}_l \in \mathbb{R}^{d_l}$ il vettore delle attivazioni al layer l e con $\mathbf{W}_l$ la matrice dei pesi corrispondente. Applicando la regola di derivazione a catena (*chain rule*), il gradiente della loss rispetto ai pesi del layer l-esimo si esprime come prodotto di matrici Jacobiane:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_l} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_L} \cdot \left(\prod_{k=l+1}^{L} \frac{\partial \mathbf{h}_k}{\partial \mathbf{h}_{k-1}} \right) \cdot \frac{\partial \mathbf{h}_l}{\partial \mathbf{W}_l} \tag{2.4}$$

Il termine centrale del prodotto, $\mathbf{J}_l^L = \prod_{k=l+1}^{L} \frac{\partial \mathbf{h}_k}{\partial \mathbf{h}_{k-1}}$, rappresenta la jacobiana composta che propaga l'errore dall'output fino al layer l. La stabilità dell'apprendimento dipende criticamente dallo spettro di tale operatore lineare.

Lemma 2.1 (Decadimento del gradiente)**.** *Si consideri una rete Feed-Forward dove* $\mathbf{h}_k = \sigma(\mathbf{W}_k \mathbf{h}_{k-1})$ *e* $\sigma(\cdot)$ *è una funzione di attivazione lipschitziana applicata element-wise. Siano* $\gamma = \sup_k \|\mathbf{W}_k\|_2$ *e* $\beta = \sup_x \|\mathbf{J}_\sigma(x)\|_2$*, dove* $\mathbf{J}_\sigma(x) = diag(\sigma'(x))$ *è la matrice jacobiana diagonale delle attivazioni. La norma operatoriale della jacobiana composta è limitata superiormente da:* $\left\| \prod_{k=l+1}^{L} \mathbf{J}_k \right\|_2 \leq (\gamma\beta)^{L-l}$*. Se il prodotto spettrale* $\gamma\beta < 1$*, il gradiente decade esponenzialmente con la profondità* $L - l$ *(**vanishing***

***gradient**), l'effetto è che i primi strati della rete (quelli che devono imparare a estrarre i bordi della targa) non ricevono istruzioni su come cambiare e rimangono "congelati".*

Teorema 2.2 (Condizionamento e instabilità)**.** *Il numero di condizionamento κ della jacobiana di trasporto $\mathbf{J}_l^L$ è definito come il rapporto tra i suoi valori singolari estremi:*

$$\kappa(\mathbf{J}_l^L) = \frac{\sigma_{\max}(\mathbf{J}_l^L)}{\sigma_{\min}(\mathbf{J}_l^L)} \tag{2.5}$$

Un valore $\kappa(\mathbf{J}_l^L) \gg 1$ indica un malcondizionamento numerico: le componenti del gradiente associate ai valori singolari $\sigma_{\min} \ll 1$ svaniscono (vanishing gradient), mentre quelle associate a $\sigma_{\max} \gg 1$ possono divergere (exploding gradient). Nelle RNN, dove la stessa matrice $\mathbf{W}$ viene applicata ricorsivamente, questo problema è strutturale.

2.3.2 Mitigazione del gradiente: LSTM e connessioni residue

Per risolvere il problema del *vanishing gradient* intrinseco alle RNN semplici (sez. 2.2), Hochreiter e Schmidhuber (1997) hanno introdotto la cella LSTM. A differenza della cella ricorrente standard, la LSTM mantiene uno *stato di cella* $\mathbf{c}_t$ separato dallo stato nascosto $\mathbf{h}_t$, regolato da tre porte logiche (*gates*) sigmoidali: *forget gate* (f_t), *input gate* (i_t) e *output gate* (o_t). L'innovazione cruciale risiede nell'aggiornamento additivo dello stato di cella:

$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t \tag{2.6}$$

dove $\odot$ indica il prodotto di Hadamard, se (f_t) è aperto, il gradiente fluisce linearmente senza svanire e questo permette al modello di ricordare la prima lettera di una targa anche mentre sta leggendo l'ultima. Questa struttura introduce un cambiamento paradigmatico nel flusso dei gradienti: mentre in una RNN standard (con attivazione tanh) l'errore fluisce attraverso continue moltiplicazioni matriciali (portando al decadimento o all'esplosione), nella LSTM lo stato di cella agisce come una *autostrada informativa*; infatti se il forget gate è aperto ($f_t \approx 1$), il gradiente retrocede additivamente senza subire trasformazioni non lineari distruttive, ciò consente al modello di apprendere dipendenze temporali arbitrariamente lunghe (es. correlazioni tra l'inizio e la fine di una frase), una capacità preclusa alle RNN *vanilla*, risolvendo strutturalmente il problema del vanishing gradient.

Parallelamente, nelle reti profonde non ricorrenti (Feed-Forward/CNN), la soluzione strutturale fondamentale è rappresentata dalle **skip connections**, o connessioni a salto. Queste architetture riformulano il problema dell'apprendimento introducendo un percorso diretto (*identity mapping*) che cortocircuita uno o più layer non lineari, permettendo al segnale di fluire inalterato attraverso la rete.

Proposizione 2.1 (Apprendimento residuo)**.** *Si desideri apprendere una mappa ideale $\mathcal{H}(\mathbf{x})$ che trasformi l'input $\mathbf{x}$. Anziché modellare direttamente $\mathcal{H}(\cdot)$, le reti residue (ResNet) definiscono esplicitamente:*

$$\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{2.7}$$

dove $\mathcal{F}(\mathbf{x}, \{W_i\})$ è la funzione residua appresa dai layer sovrapposti. L'ipotesi fondante è che sia più semplice ottimizzare la mappa residua $\mathcal{F}$ (che tende a zero in caso di identità) piuttosto che la mappa originale completa. Il gradiente, propagato tramite chain rule, diviene:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_{l+1}} \cdot \left(\mathbf{I} + \frac{\partial \mathcal{F}}{\partial \mathbf{x}_l} \right)$$

Il termine additivo unitario $\mathbf{I}$ agisce come un percorso identity diretto per la retro-propagazione, garantendo che il gradiente locale contenga sempre una componente inalterata che non dipende dai pesi dei layer intermedi. Questo mitiga strutturalmente il decadimento esponenziale del prodotto $\prod \mathbf{J}_k$, poiché anche se le derivate parziali di $\mathcal{F}$ tendono a zero (vanishing gradient), il segnale di errore può propagarsi preservando la sua magnitudine anche in reti molto profonde. Tale meccanismo è onnipresente nelle architetture moderne: costituisce il blocco costruttivo fondamentale del backbone ResNet-50 in RT-DETR, della CSPDarknet in YOLOX e dei blocchi encoder nel CCT utilizzato per l'OCR, abilitando l'addestramento efficace di reti con centinaia di layer.

2.3.3 Dal modello CRNN al paradigma transformer ibrido

Lo standard *de facto* per l'OCR è stato storicamente l'architettura CRNN (Convolutional Recurrent Neural Network), che combina un estrattore CNN (sez. 2.2) con una RNN/LSTM per la modellazione sequenziale. Tuttavia, come discusso, la natura ricorrente $O(T)$ delle RNN impedisce la parallelizzazione massiva su GPU e limita la capacità di catturare relazioni a lungo raggio in un singolo passo computazionale.

Vaswani et al. (2017) hanno rivoluzionato l'elaborazione di sequenze sostituendo la ricorrenza con il meccanismo di **self-attention**. Se la RNN elabora la sequenza di token $\mathbf{E} \in \mathbb{R}^{L \times D}$ passo dopo passo, il transformer calcola le interazioni tra tutti i token simultaneamente. La complessità del percorso di dipendenza si riduce drasticamente da $O(L)$ a $O(1)$, facilitando l'apprendimento di relazioni globali (e.g., correlazione tra primo e ultimo carattere della targa).

Il cuore computazionale è la *scaled dot-product attention*, una funzione che mappa una query (ciò che cerco) e un insieme di coppie key-value (etichetta e contenuto) in un output pesato.

Definizione 2.3 (Scaled dot-product attention)**.** *Siano $\mathbf{Q} \in \mathbb{R}^{n_q \times d_k}$, $\mathbf{K} \in \mathbb{R}^{n_k \times d_k}$ e $\mathbf{V} \in \mathbb{R}^{n_k \times d_v}$ le matrici rappresentanti rispettivamente le Queries, le Keys e i Values. L'attenzione è definita come:*

$$\textit{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \textit{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} \right) \mathbf{V} \tag{2.8}$$

2.3.4 Meccanismo di self-attention

Per chiarire il flusso informativo, consideriamo l'input $\mathbf{X} \in \mathbb{R}^{n \times d}$ e le proiezioni lineari apprendibili $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$. Per ogni token i, definito lo score di similarità

non normalizzato s_{ij}, il coefficiente di attenzione α_{ij} (che soddisfa $\sum_j \alpha_{ij} = 1$) e il vettore di contesto $\mathbf{y}_i$, il calcolo puntuale è espresso come:

$$s_{ij} = \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_k}}, \quad \alpha_{ij} = \frac{\exp(s_{ij})}{\sum_{t=1}^{n} \exp(s_{it})}, \quad \mathbf{y}_i = \sum_{j=1}^{n} \alpha_{ij} \mathbf{v}_j \tag{2.9}$$

Il processo è visualizzato nello schema logico di fig. 2.2.

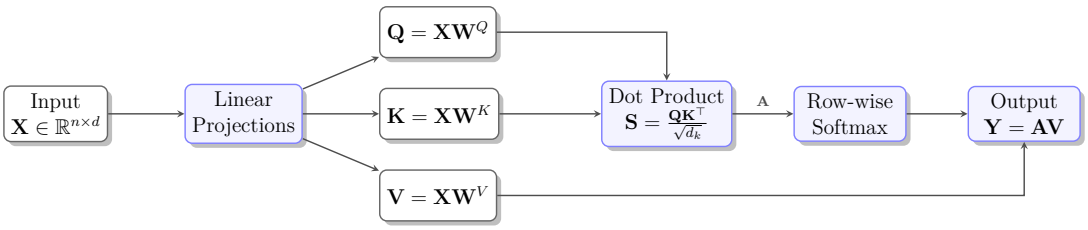


Figura 2.2: *Flusso operativo del self-attention: dalle proiezioni lineari alla matrice di attenzione* $\mathbf{A}$*, fino all'aggregazione dei valori.*

Per catturare relazioni complesse in diversi sottospazi semantici, si estende questo concetto alla **Multi-Head Attention (MHA)**. Le teste di attenzione operano in parallelo su porzioni del sottospazio di embedding (proiezione vettoriale feature spaziali). Dato un numero di teste h (invece di 1 e che son in parallelo), l'output finale è dato dalla concatenazione dei risultati delle singole teste, proiettata linearmente tramite $\mathbf{W}^O$:

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O \\ \text{dove head}_i &= \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \end{aligned} \tag{2.10}$$

con le matrici di proiezione $\mathbf{W}_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ e $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

L'architettura originale del transformer, proposta da Vaswani et al., è strutturata secondo il paradigma **Encoder-Decoder**, progettato per compiti di traduzione automatica (*Sequence-to-Sequence*).

- Encoder: composto da una pila di layer identici, ognuno contenente due sottostrati, con un meccanismo di Multi-Head Self-Attention e una rete Feed-Forward position-wise. Il suo scopo è mappare la sequenza di input $\mathbf{X} = (x_1, \ldots, x_n)$ in una rappresentazione continua astratta $\mathbf{Z} = (z_1, \ldots, z_n)$.
- Decoder: anch'esso composto da layer impilati, integra un terzo sottostrato di *MHA* che esegue l'attenzione sulla rappresentazione dell'encoder. Il decoder genera la sequenza di output $\mathbf{Y} = (y_1, \ldots, y_m)$ in modo auto-regressivo, consumando il simbolo generato al passo precedente come input per il successivo.

Tuttavia, in questo lavoro il paradigma Encoder-Decoder viene adattato e specializzato per ciascuno dei tre stadi della pipeline Custom ALPR, come dettagliato di seguito. La transizione da RNN a transformer comporta un cambiamento nel profilo delle risorse computazionali, come dettagliato nel Teorema seguente e nella tab. 2.1.

Teorema 2.3 (Complessità del self-attention)**.** *Dato un input di n token e dimensione d, il layer di self-attention ha una complessità temporale $O(n^2 d)$ e spaziale $O(n^2+nd)$. Sebbene il termine quadratico n^2 sia oneroso per sequenze lunghe, nel contesto ALPR (dove n deriva da crop di targhe a bassa risoluzione) tale costo è trascurabile e ampiamente compensato dalla parallelizzazione e dalla riduzione del path length a $O(1)$.*

Tabella 2.1: *Confronto asintotico tra architetture per sequenze (k kernel size, n lunghezza sequenza, d dimensione rappresentazione).*

Architettura	**Costo Computazionale**	**Memoria**	**Path Length**	**Parallelizzabile?**
CNN (Conv 1D)	$O(k \cdot n \cdot d^2)$	$O(k \cdot d^2)$	$O(\log_k n)$	Sì
RNN (LSTM)	$O(n \cdot d^2)$	$O(d^2)$	$O(n)$	No
Transformer	$\mathbf{O(n^2 \cdot d)}$	$\mathbf{O(n^2 + nd)}$	$\mathbf{O(1)}$	**Sì**

Poiché l'operatore di attenzione è invariante alle permutazioni, è necessario iniettare esplicitamente l'informazione sull'ordine dei token, una proprietà fondamentale dell'encoding sinusoidale classico è espressa dal seguente lemma:

Lemma 2.2 (Proprietà di traslazione lineare)**.** *Per qualsiasi offset fisso k, il vettore PE_{pos+k} può essere rappresentato come una funzione lineare di PE_{pos}: questa proprietà algebrica facilita l'apprendimento di pattern relativi (es. la sequenza di caratteri in una targa) indipendentemente dalla loro posizione assoluta nell'immagine.*

Tuttavia, nelle moderne implementazioni applicative (inclusa quella proposta in questa tesi), è prassi sostituire i pattern fissi con il **learnable positional encoding**. In questo approccio, l'embedding posizionale non è calcolato a priori, ma è costituito da una matrice di parametri $\mathbf{P} \in \mathbb{R}^{N \times d}$ inizializzata casualmente e ottimizzata tramite backpropagation congiuntamente ai pesi del modello. Questa scelta massimizza la flessibilità, permettendo alla rete di apprendere le dipendenze spaziali specifiche del dominio direttamente dai dati.

Nelle architetture transformer profonde, è prassi adottare la tecnica dello **stochastic depth** [26] per mitigare l'overfitting e favorire la propagazione del gradiente. Definito un blocco residuo generico $\mathbf{x}_{l+1} = \text{Block}(\mathbf{x}_l) + \mathbf{x}_l$, durante la fase di addestramento l'aggiornamento diviene stocastico:

$$\mathbf{x}_{l+1} = \begin{cases} \text{Block}(\mathbf{x}_l) + \mathbf{x}_l & \text{con probabilità } 1 - p_l \\ \mathbf{x}_l & \text{con probabilità } p_l \end{cases} \tag{2.11}$$

dove p_l è la probabilità di drop, che tipicamente cresce linearmente con la profondità del layer l ed evita overfitting. L'applicazione di questa tecnica deve essere calibrata attentamente, poiché la rimozione eccessiva di blocchi può compromettere il flusso informativo.

L'applicazione dei **Vision Transformer (ViT)** [9] puri in contesti a risorse limitate e su dataset di dimensioni contenute evidenzia limitazioni intrinseche dovute alla

carenza di *bias induttivo*, per imparare i "bias induttivi" (ovvero capire che pixel vicini sono correlati) necessita di dataset immensi; un ViT standard processa un'immagine di input $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$ suddividendola in una sequenza di N patch disgiunte $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, dove (P, P) è la risoluzione della patch e $N = HW/P^2$. Tali patch vengono mappate in uno spazio di embedding D-dimensionale tramite una proiezione lineare $\mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}$, ottenendo la sequenza di token $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ che alimenta l'encoder. Per superare tale dicotomia, il modello Compact Convolutional Transformer introduce un cambio di paradigma nella fase di tokenizzazione, sostituendo la proiezione lineare rigida con un blocco convoluzionale profondo. L'architettura adotta un tokenizer parametrico f_{conv} basato su una sequenza di strati convoluzionali, attivazioni ReLU e Max Pooling. Invece di suddividere l'immagine in patch statiche — operazione che disperde l'informazione locale e di bordo — il modello estrae feature gerarchiche; data l'immagine di input $\mathbf{X}$, la sequenza di token $\mathbf{E}$ viene generata come:

$$\mathbf{E} = \text{Reshape}\left(f_{\text{conv}}(\mathbf{X})\right) + \mathbf{P}_{emb} \tag{2.12}$$

dove l'operazione di *Reshape* appiattisce le feature map risultanti $(H' \times W' \times D)$ in una sequenza di lunghezza $L = H' \cdot W'$.

È cruciale notare che, a differenza dei ViT puri, in questa architettura il termine $\mathbf{P}_{emb}$ (l'embedding posizionale apprendibile) gioca un ruolo complementare. Poiché lo stadio convoluzionale f_{conv} inietta intrinsecamente un **bias induttivo locale** e preserva le relazioni spaziali di vicinato, il modello risulta nativamente più robusto a piccoli shift spaziali e distorsioni prospettiche, tramite ciò i primi strati convoluzionali estraggono bordi e angoli. L'addizione di $\mathbf{P}_{emb}$ serve quindi a fornire al meccanismo di self-attention un riferimento globale sulla struttura dell'immagine, combinando il meglio dei due mondi: l'estrazione locale delle CNN e la modellazione globale dei transformer.

La struttura fondamentale dell'encoder CCT, illustrata in fig 2.3, alterna blocchi di MHA a reti Feed-Forward, stabilizzati da connessioni residue e normalizzazione *pre-norm*.

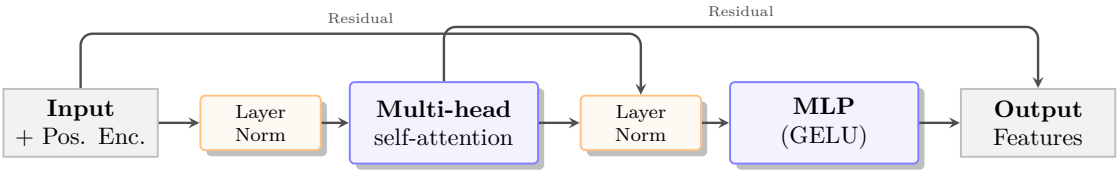


Figura 2.3: *Schema del blocco encoder in un vision transformer/CCT. Si noti la struttura pre-norm, che favorisce la stabilità del gradiente nelle reti profonde.*

L'iniezione di un bias induttivo convoluzionale offre benefici cruciali per la lettura delle targhe:

1. Conservazione della località: il tokenizer elabora le feature di basso livello (bordi, angoli) preservando le relazioni di vicinato spaziale, garantendo una rappresentazione più ricca per caratteri distorti.

2. Efficienza dei dati: grazie ai prior strutturali, il CCT converge più rapidamente e generalizza meglio su dataset ridotti, eliminando la necessità di pre-training su grandi corpora esterni.
3. Flessibilità geometrica: il tokenizer convoluzionale gestisce nativamente variazioni di aspect ratio, adattandosi alla geometria variabile dei crop delle targhe senza introdurre artefatti di ridimensionamento eccessivi.

2.4 Analisi delle funzioni di attivazione

Le funzioni di attivazione introducono la non-linearità essenziale per l'apprendimento di mapping complessi in reti neurali profonde, nel contesto di questo lavoro, la scelta delle attivazioni segue i paradigmi architetturali moderni: il detector YOLOX adotta la **SiLU** (Swish) per i blocchi convoluzionali, mentre i modelli **Transformer** (RT-DETR v2, CCT) impiegano tipicamente la **GELU** nei layer feed-forward. Di seguito si analizzano le proprietà matematiche di tali funzioni e il loro impatto sulla stabilità del gradiente.

Definizione 2.4 (Funzione sigmoide logistica)**.** *La sigmoide logistica* $\sigma : \mathbb{R} \to (0, 1)$ *è definita come:*

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.13}$$

Pur avendo un'interpretazione probabilistica naturale, la sua derivata $\sigma'(x) = \sigma(x)(1-\sigma(x))$ *tende a zero per* $|x| \gg 0$ *(saturazione), input grandi o piccoli; questo comportamento causa il fenomeno del* ***vanishing gradient****, rendendola inadatta per i layer nascosti di reti profonde.*

Definizione 2.5 (Rectified Linear Unit – ReLU)**.** *La ReLU* $f : \mathbb{R} \to [0, +\infty)$ *introduce non-linearità preservando la scala del gradiente per input positivi:*

$$ReLU(x) = \max(0, x) = \begin{cases} x & se\ x \geq 0 \\ 0 & se\ x < 0 \end{cases} \tag{2.14}$$

È computazionalmente efficiente e induce sparsità nelle attivazioni. Tuttavia, la derivata nulla per $x < 0$ *può causare la morte permanente dei neuroni (dying ReLU problem), impedendo l'aggiornamento dei pesi.*

Per mitigare i limiti della ReLU e migliorare la fluidità della superficie di ottimizzazione, le architetture moderne adottano varianti liscie e non monotone.

Definizione 2.6 (Leaky ReLU)**.** *La Leaky ReLU introduce una piccola pendenza* $\alpha > 0$ *per* $x < 0$*:*

$$f(x) = \max(\alpha x, x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases}. \tag{2.15}$$

Definizione 2.7 (PReLU (Parametric ReLU))**.** *La PReLU è una Leaky ReLU in cui il parametro* α *non è fissato, ma* ***appreso*** *durante l'addestramento (eventualmente*

per canale):

$$f(x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases}, \qquad \alpha \geq 0 \text{ apprendibile.} \tag{2.16}$$

Definizione 2.8 (ELU (Exponential Linear Unit))**.** *L'ELU rende la parte negativa liscia tramite un termine esponenziale:*

$$f(x) = \begin{cases} x & x > 0 \\ \alpha(\mathrm{e}^x - 1) & x \leq 0 \end{cases}, \qquad \alpha > 0. \tag{2.17}$$

Definizione 2.9 (SELU (Scaled ELU))**.** *La SELU è una variante scalata dell'ELU, usata in reti self-normalizing sotto opportune condizioni di inizializzazione:*

$$f(x) = \lambda \begin{cases} x & x > 0 \\ \alpha(\mathrm{e}^x - 1) & x \leq 0 \end{cases}, \qquad \alpha, \lambda > 0. \tag{2.18}$$

Definizione 2.10 (Swish e SiLU)**.** *Si definisce Swish una famiglia di funzioni di attivazione $f : \mathbb{R} \to \mathbb{R}$ parametrizzate da un coefficiente $\beta \geq 0$. Il nome "Swish", coniato dai ricercatori di Google Brain (Ramachandran et al., 2017), fa riferimento alla natura Self-Gated dell'unità: il valore di input modula se stesso attraverso il gate sigmoidale, l'attivazione è calcolata come:*

$$Swish(x) = x \cdot \sigma(\beta x) = \frac{x}{1 + e^{-\beta x}} \tag{2.19}$$

Il parametro β determina la forma della curva: per $\beta \to \infty$ la funzione converge alla ReLU, mentre per $\beta \to 0$ diviene una trasformazione lineare scalata. Nel contesto delle architetture efficienti come YOLOX ed MobileNet, si adotta il caso particolare in cui il parametro è fissato a $\beta = 1$. Tale variante assume la denominazione di Sigmoid Linear Unit (SiLU):

$$SiLU(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}} \tag{2.20}$$

A differenza della ReLU, la SiLU è una funzione liscia di classe C^∞ e non monotona, caratterizzata da un minimo globale negativo per $x \approx -1.28$. Questa proprietà permette alla rete di recuperare informazioni anche da attivazioni negative (autostabilizzazione del gradiente nei layer profondi del detector), migliorando la propagazione del gradiente nei layer profondi.

Definizione 2.11 (Gaussian Error Linear Unit – GELU)**.** *Standard de facto nei Transformer (BERT, ViT, RT-DETR), la GELU può essere interpretata come una regolarizzazione stocastica (dropout) applicata all'input. Indicando con $\Phi(x)$ la funzione di ripartizione della normale standard, pesa input x per probabilità che neurone venga mantenuto:*

$$GELU(x) = x \cdot P(X \leq x) = x \cdot \Phi(x) \approx 0.5x(1 + \tanh[\sqrt{2/\pi}(x + 0.044715x^3)]) \tag{2.21}$$

La GELU pondera l'input per la sua probabilità di essere mantenuto, garantendo una curvatura liscia che facilita l'ottimizzazione in modelli molto profondi, dove precisione OCR è critica.

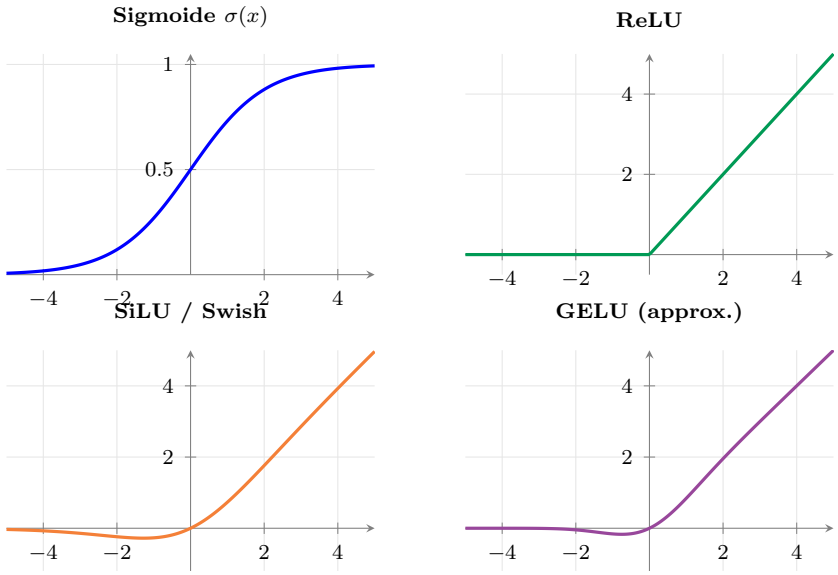


Figura 2.4: *Panoramica compatta (vettoriale) di alcune funzioni di attivazione usate nel documento: sigmoide, ReLU, SiLU/Swish (YOLOX) e GELU (transformer).*

Altre attivazioni comuni in letteratura mirano a combinare derivabilità ovunque e capacità di modellare transizioni graduali.

Definizione 2.12 (Softplus)**.** *La Softplus è una approssimazione liscia della ReLU:* $\mathit{Softplus}(x) = \ln(1 + \mathrm{e}^x)$.

Definizione 2.13 (Mish)**.** *La Mish è una funzione liscia definita come* $\mathit{Mish}(x) = x \tanh(\mathit{Softplus}(x))$.

Definizione 2.14 (Maxout)**.** *La Maxout prende il massimo tra m attivazioni lineari (con parametri distinti):* $\mathit{Maxout}(\mathbf{x}) = \max_{r\in\{1,\dots,m\}} \left(\mathbf{w}_r^\top \mathbf{x} + b_r\right)$.

Definizione 2.15 (Continuità di Lipschitz e stabilità)**.** *Una funzione di attivazione* σ *è* λ*-Lipschitz continua se* $|\sigma(x) - \sigma(y)| \leq \lambda|x - y|$*. Questa proprietà è utile per discutere stabilità: in modo informale, limita quanto l'output può variare al variare dell'input e contribuisce a mantenere controllato il flusso del gradiente.*

Nel layer finale del modulo di riconoscimento (OCR), i logit non normalizzati $\mathbf{z} \in \mathbb{R}^K$ vengono trasformati in una distribuzione di probabilità $\mathbf{p}$ su K classi tramite la funzione Softmax, dove $\sum p_i = 1$:

$$\mathrm{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.22}$$

Proposizione 2.2 (Jacobiana della softmax)**.** *La matrice jacobiana* $\mathbf{J} \in \mathbb{R}^{K\times K}$ *della funzione softmax rispetto all'input* $\mathbf{z}$ *ha forma:* $J_{ij} = \frac{\partial p_i}{\partial z_j} = p_i(\delta_{ij} - p_j)$*, dove* δ_{ij} *è la delta di Kronecker. Questa struttura implica che il gradiente è massimo quando la predizione è incerta (i.e., p è uniforme) e si annulla quando la distribuzione collassa su un one-hot vector (certezza assoluta), modulando naturalmente la velocità di apprendimento.*

2.5 Ottimizzazione per il training

Il processo di addestramento di una rete neurale si definisce come la ricerca di un vettore di parametri $\theta \in \mathbb{R}^d$ che minimizzi una funzione obiettivo $J(\theta)$, intesa come stima empirica del rischio atteso $\mathbb{E}[\mathcal{L}(f(x;\theta), y)]$. Data l'impossibilità di calcolare il gradiente esatto sull'intera distribuzione dei dati, in contesti reali si ricorre a stimatori stocastici calcolati su sottoinsiemi dei dati (*mini-batch*), dando luogo a procedure iterative efficienti.

Definizione 2.16 (Stochastic Gradient Descent – SGD)**.** *Al passo t, sia $\mathcal{B}_t$ un mini-batch di campioni e $\theta_t \in \mathbb{R}^d$ il vettore dei parametri correnti. Definito $g_t = \nabla_\theta J_{\mathcal{B}_t}(\theta_t)$ come lo stimatore stocastico del gradiente (ossia il gradiente della loss calcolata solo su $\mathcal{B}_t$), la regola di aggiornamento è data da:*

$$\theta_{t+1} = \theta_t - \eta_t \, g_t \tag{2.23}$$

dove $\eta_t > 0$ è il learning rate, che determina l'ampiezza dello spostamento lungo la direzione di discesa.

Le garanzie teoriche per la confluenza di algoritmi stocastici derivano dal Teorema di Robbins-Monro, che stabilisce condizioni sufficienti sulla schedulazione del learning rate.

Teorema 2.4 (Robbins-Monro, 1951)**.** *Sotto ipotesi standard di regolarità (funzione obiettivo convessa e varianza del gradiente limitata), la convergenza quasi certa a un minimo globale è garantita se la sequenza dei learning rate $\{\eta_t\}_{t\geq 1}$ soddisfa:$\sum_{t=1}^{\infty} \eta_t = \infty$, $\sum_{t=1}^{\infty} \eta_t^2 < \infty$. Queste condizioni implicano che il passo deve decrescere nel tempo (per controllare la varianza dell'errore di stima, cioè rumore), ma non troppo velocemente (per evitare di bloccarsi prima di raggiungere l'ottimo), come nel caso $\eta_t \propto 1/\sqrt{t}$.*

Per mitigare le oscillazioni in *valli* strette della superficie di errore e accelerare la convergenza, il metodo del **momentum** introduce una variabile di velocità $v_t \in \mathbb{R}^d$ che accumula una media mobile esponenziale dei gradienti passati. Fissato un coefficiente di frizione $\mu \in [0, 1)$ (tipicamente $\mu \approx 0.9$), la dinamica è descritta dal sistema:

$$v_{t+1} = \mu v_t - \eta_t \, g_t \tag{2.24}$$

$$\theta_{t+1} = \theta_t + v_{t+1} \tag{2.25}$$

Definizione 2.17 (Nesterov Accelerated Gradient – NAG)**.** *Una variante raffinata del momentum è il gradient di Nesterov, che migliora la stabilità valutando il gradiente non nella posizione corrente θ_t, bensì nella posizione anticipata dalla velocità corrente ($\theta_t + \mu v_t$), evitando che oscilli troppo, tale che:*

$$v_{t+1} = \mu v_t - \eta_t \nabla J(\theta_t + \mu v_t) \tag{2.26}$$

$$\theta_{t+1} = \theta_t + v_{t+1} \tag{2.27}$$

Prima dell'avvento di Adam, diversi algoritmi hanno introdotto l'adattamento del learning rate per ogni parametro.

Definizione 2.18 (Adagrad)**.** *Adagrad accumula la somma dei quadrati dei gradienti passati per scalare il passo di apprendimento, penalizzando i parametri con gradienti frequenti e elevati:*

$$G_{t,ii} = \sum_{\tau=1}^{t} g_{\tau,i}^2 \quad \Longrightarrow \quad \theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} g_{t,i} \tag{2.28}$$

Tuttavia, l'accumulo monotono di G_t porta a un annullamento prematuro del learning rate.

Definizione 2.19 (Adadelta e RMSprop)**.** *Per risolvere il decadimento aggressivo di Adagrad, RMSprop sostituisce la somma cumulativa con una media mobile esponenziale del quadrato dei gradienti ($E[g^2]_t$):*

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \quad \Longrightarrow \quad \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{2.29}$$

Adadelta estende ulteriormente questo concetto eliminando la necessità di un learning rate globale, utilizzando invece una stima delle finestre mobili degli aggiornamenti dei parametri per preservare l'omogeneità dimensionale delle unità di misura.

Algoritmi come Adam (*Adaptive Moment Estimation*) adattano il learning rate individualmente per ogni parametro, invece di uno solo per tutti i parametri, basandosi sulle stime del primo momento (media) e del secondo momento (varianza non centrata) dei gradienti. Siano $\beta_1, \beta_2 \in [0, 1)$ i fattori di decadimento esponenziale e $\epsilon > 0$ una costante di stabilità numerica. Le stime dei momenti m_t (media mobile gradienti, la direzione) e v_t (media mobile al quadrato gradienti, volatilità) sono aggiornate tramite una combinazione convessa tra la memoria storica e l'osservazione corrente:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{2.30}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{2.31}$$

dove le operazioni g_t^2 sono intese elemento per elemento. Per correggere il bias di inizializzazione a zero (significativo nei primi passi), si calcolano le stime $\hat{m}_t = m_t/(1-\beta_1^t)$ e $\hat{v}_t = v_t/(1-\beta_2^t)$. L'aggiornamento dei parametri diviene:

$$\theta_{t+1} = \theta_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.32}$$

Tale approccio normalizza i gradienti in base alla loro volatilità recente, risultando particolarmente efficace in architetture con parametri a scale diverse.

Tuttavia, l'implementazione standard del *weight decay* (regolarizzazione L_2) in Adam risulta subottimale poiché viene scalata dalla varianza adattiva. Per risolvere questo problema strutturale, fondamentale per la convergenza dei transformer, è stata introdotta la variante **AdamW**.

Definizione 2.20 (AdamW: decoupled weight decay)**.** *Nelle implementazioni standard della regolarizzazione L_2 con ottimizzatori adattivi come Adam, il termine di*

decadimento $\lambda\theta_t$ viene sommato al gradiente g_t prima del calcolo dei momenti. Di conseguenza, la regolarizzazione viene scalata dal fattore adattivo $1/\sqrt{\hat{v}_t}$, applicando una penalità minore ai parametri che hanno gradienti di grande ampiezza, contrariamente all'intuizione della regolarizzazione L_2, scalato da varianza adattiva.

*AdamW risolve questo problema strutturale **disaccoppiando** il decadimento dei pesi dall'aggiornamento basato sui gradienti. Il termine di regolarizzazione viene sottratto direttamente dal valore del parametro, senza interagire con i momenti m_t e v_t:*

$$\theta_{t+1} = \theta_t - \eta_t \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda\theta_t \right) \tag{2.33}$$

In questo modo, il termine $\eta_t\lambda\theta_t$ agisce come un puro decadimento esponenziale dei pesi, uniforme e indipendente dalla storia dei gradienti, migliorando significativamente la capacità di generalizzazione e la stabilità dell'addestramento, fattore critico per i modelli transformer based.

Oltre alla regolarizzazione esplicita sui pesi, in questa tesi adottiamo strategie di controllo sulla dinamica temporale dell'addestramento.

Un primo meccanismo cruciale è l'**Early Stopping**, che interrompe preventivamente l'addestramento qualora la metrica di validazione non mostri miglioramenti per un intervallo di epoche (patience) predefinito, in questo lavoro, tale strategia si rivela essenziale per il modulo OCR addestrato. Parallelamente, per stabilizzare la convergenza si adotta l'**Exponential Moving Average (EMA)**, anziché utilizzare direttamente i pesi θ_t dell'ultima iterazione, il sistema mantiene una copia ombra θ_{EMA} aggiornata costantemente come media mobile:

$$\theta_{\text{EMA}}^{(t)} = \alpha \cdot \theta_{\text{EMA}}^{(t-1)} + (1 - \alpha) \cdot \theta^{(t)} \tag{2.34}$$

Con un decadimento $\alpha \approx 0.9999$, l'EMA agisce come un filtro passa-basso sulla traiettoria di ottimizzazione, smorzando il rumore stocastico degli aggiornamenti dei mini-batch nel backbone, questa tecnica è implementata nativamente in per esempio **RT-DETR v2**, garantendo che il modello finale dispieghi un backbone robusto e meno sensibile a fluttuazioni locali della loss.

2.6 Fondamenti teorici della data augmentation

L'impiego della data augmentation nelle moderne architetture profonde trascende il mero espediente euristico per l'incremento della numerosità campionaria, configurandosi come una strategia di regolarizzazione strutturale fondata sul principio della *Vicinal Risk Minimization (VRM)*. Nel paradigma classico della *Empirical Risk Minimization (ERM)*, l'obiettivo è l'approssimazione della distribuzione congiunta incognita $P(X, Y)$ attraverso una distribuzione empirica $P_\delta(x, y) = \frac{1}{N}\sum_{i=1}^{N} \delta(x - x_i, y - y_i)$, costituita dalla nostra migliore stima della realtà, basata solo sui punti che abbiamo visto finora. Tuttavia, tale approccio soffre intrinsecamente di overfitting, poiché il supporto di P_δ è un insieme di misura nulla nello spazio delle feature $\mathcal{X} \subseteq \mathbb{R}^d$.

La VRM propone di estendere tale supporto sostituendo le masse puntuali con distribuzioni di densità, levigando la varietà dei dati. Dato un dataset di addestramento

$\mathcal{D}_N = \{(x_i, y_i)\}_{i=1}^N$, si definisce una distribuzione di vicinanza $P_v(\tilde{x}, \tilde{y} \mid x_i, y_i)$ che modella la probabilità di generare un esempio virtuale $(\tilde{x}, \tilde{y})$ nell'intorno dell'osservazione i-esima, applicando trasformazioni stocastiche (quali rotazioni, variazioni cromatiche o iniezione di rumore gaussiano). Denotando con $f(\cdot, \theta)$ il modello parametrico e con $\mathcal{L}(\cdot, \cdot)$ una funzione di costo convessa e derivabile, il rischio vicinale atteso $R_{\text{vic}}(\theta)$ si esprime come la media dell'errore calcolato sull'intera misura di probabilità indotta dalle trasformazioni:

$$R_{\text{vic}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{(\tilde{x},\tilde{y})\sim P_v(\cdot|x_i,y_i)} \Big[\mathcal{L}(f(\tilde{x}; \theta), \tilde{y}) \Big]. \tag{2.35}$$

Tale formulazione impone al modello un vincolo di invarianza locale: la funzione appresa f deve mantenersi stabile all'interno dell'intorno definito da P_v, forzando le superfici decisionali a giacere in regioni a bassa densità di dati.

2.6.1 Equivalenza con la regolarizzazione del gradiente

Per comprendere analiticamente come la VRM migliori la generalizzazione, è utile analizzare il comportamento della funzione di costo sotto perturbazioni additive di piccola entità. Si consideri il caso in cui l'augmentation consista nell'aggiunta di un vettore di rumore $\boldsymbol{\xi} \in \mathbb{R}^d$ all'input, campionato da una distribuzione a media nulla e matrice di covarianza $\boldsymbol{\Sigma}$, ovvero $\tilde{x} = x + \boldsymbol{\xi}$ con $\mathbb{E}[\boldsymbol{\xi}] = 0$ e $\mathbb{E}[\boldsymbol{\xi}\boldsymbol{\xi}^\top] = \boldsymbol{\Sigma}$. Assumendo che la funzione di loss $\mathcal{L}$ sia sufficientemente regolare, possiamo approssimare il costo sul campione perturbato tramite uno sviluppo di Taylor del secondo ordine attorno a x.

Proposizione 2.3 (Augmentation come penalizzazione jacobiana)**.** *Sia $J(x) = \mathcal{L}(f(x; \theta), y)$ la funzione di costo puntuale, nelle ipotesi di rumore additivo $\boldsymbol{\xi}$ a media nulla e varianza piccola, il rischio vicinale atteso approssima il rischio empirico standard aumentato di un termine di regolarizzazione dipendente dalla norma del gradiente dell'input:*

$$\mathbb{E}_{\boldsymbol{\xi}}[J(x+\boldsymbol{\xi})] \approx J(x) + \mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{\xi}^\top \nabla_x J(x)] + \frac{1}{2}\mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{\xi}^\top \mathbf{H}_x(J) \boldsymbol{\xi}] = J(x) + \frac{1}{2} tr(\boldsymbol{\Sigma}\mathbf{H}_x(J)) \tag{2.36}$$

dove $\nabla_x J$ e $\mathbf{H}_x(J)$ denotano rispettivamente il gradiente e la matrice hessiana di J rispetto all'input. Nel caso semplificato di rumore isotropo ($\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$), il termine correttivo diviene proporzionale al laplaciano della loss, o equivalentemente (in reti lineari o shallow), alla norma del gradiente rispetto all'input $\|\nabla_x f\|^2$.

Questo risultato (Bishop, 1995) dimostra che la data augmentation agisce implicitamente come una regolarizzazione di Tikhonov, penalizzando funzioni f che variano troppo rapidamente rispetto alle perturbazioni dell'input, migliorando così la robustezza avversariale e la stabilità numerica.

2.6.2 Limiti di generalizzazione e PAC-Bayes

L'efficacia della VRM può essere ulteriormente inquadrata nella teoria dell'apprendimento statistico tramite i vincoli PAC-Bayes (*Probably Approximately Correct*).

Questi forniscono garanzie probabilistiche sull'errore di generalizzazione basate sulla *piattezza* dei minimi trovati nello spazio dei parametri.

Consideriamo un classificatore stocastico definito da una distribuzione a posteriori Q sui pesi e una distribuzione a priori P fissata prima dell'osservazione dei dati. Al fine di rendere trattabile il calcolo della complessità, è prassi comune nel Deep Learning assumere che entrambe le distribuzioni appartengano alla famiglia delle normali multivariate su uno spazio $\mathbb{R}^d$. Definite quindi $Q \sim \mathcal{N}(\mu_Q, \Sigma_Q)$ come la distribuzione dei parametri appresi e $P \sim \mathcal{N}(\mu_P, \Sigma_P)$ come il prior (spesso assunto isotropo e centrato nell'origine), la divergenza di Kullback-Leibler $KL(Q\|P)$ assume una forma chiusa analitica data da:

$$KL(\mathcal{N}_Q\|\mathcal{N}_P) = \frac{1}{2}\left[\underbrace{(\mu_Q-\mu_P)^\top\Sigma_P^{-1}(\mu_Q-\mu_P)}_{\text{Termine di Regolarizzazione } L_2} + \underbrace{\operatorname{tr}(\Sigma_P^{-1}\Sigma_Q) - d + \ln\frac{\det\Sigma_P}{\det\Sigma_Q}}_{\text{Termine di Entropia (Piattezza)}}\right] \tag{2.37}$$

Questa formulazione evidenzia come la *distanza* informativa sia composta da due contributi distinti, il primo è un termine quadratico legato alla differenza delle medie, che agisce analogamente a una regolarizzazione L_2 penalizzando lo spostamento dei pesi rispetto all'inizializzazione (prior). Il secondo termine è usato per imparare i concetti generali, è legato alla traccia e al log-determinante delle matrici di covarianza, misura la differenza di entropia tra le distribuzioni. In particolare, minimizzare questa divergenza incentiva il modello a posteriori Q a mantenere una varianza Σ_Q elevata — evitando il collasso in minimi troppo stretti o *sharp* — condizione che si correla direttamente alla capacità di generalizzazione robusta promossa dalla Data Augmentation.

Assumendo tale misura come proxy della complessità del modello appreso, sussiste la seguente limitazione probabilistica.

Teorema 2.5 (PAC-Bayes Bound con augmentation)**.** *Sia $\mathcal{D}$ la distribuzione reale dei dati e $R_{true}(Q) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}_Q]$ l'errore di generalizzazione atteso. Fissata una tolleranza $\delta \in (0,1)$, con probabilità almeno $1-\delta$ rispetto al campionamento del dataset di training di dimensione N, vale la disuguaglianza:*

$$R_{true}(Q) \leq R_{vic}(Q) + \sqrt{\frac{KL(Q\|P) + \log(2\sqrt{N}/\delta)}{2N}} \tag{2.38}$$

dove $R_{vic}(Q)$ rappresenta l'ERM calcolato sui dati aumentati.

L'implicazione teorica è duplice: l'augmentation riduce il termine $R_{\text{vic}}(Q)$ esplorando l'intorno locale dei dati, mentre l'efficace iniezione di invarianza permette di scegliere una distribuzione a posteriori Q con varianza maggiore (minimi più piatti), riducendo indirettamente il termine di complessità KL e migliorando il bound di generalizzazione.

2.6.3 Mixup e regolarizzazione globale

Mentre le trasformazioni geometriche (dettagliate in Appendice D) operano localmente, tecniche come il *Mixup* (Zhang et al., 2017) estendono la VRM modellando relazioni globali tra coppie di esempi. Siano (x_i, y_i) e (x_j, y_j) due campioni estratti casualmente dal training set. Introdotto un coefficiente $\lambda \sim \text{Beta}(\alpha, \alpha)$, il Mixup genera un esempio virtuale imponendo un vincolo di linearità:

$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j, \quad \tilde{y} = \lambda y_i + (1 - \lambda)y_j. \tag{2.39}$$

Questa tecnica regolarizza il modello forzando una transizione graduale delle probabilità nelle zone di indecisione, contrastando l'overconfidence.

2.7 Object detection: paradigmi a confronto

L'evoluzione delle architetture di rilevamento oggetti ha visto il progressivo abbandono di euristiche rigide in favore di approcci *data-driven* end-to-end. In questa sezione confrontiamo i due paradigmi dominanti adottati nel presente lavoro: l'approccio *anchor-free* convoluzionale (rappresentato da YOLOX) e l'approccio *set-prediction* basato su transformer (rappresentato da RT-DETR v2).

2.7.1 Il paradigma anchor-free: YOLOX

L'architettura **YOLOX**, introdotta da Megvii Technology, segna il superamento del design *anchor-based* (tipico di YOLOv3/v4), rimuovendo la dipendenza da box predefiniti (ancore) che limitavano la generalizzazione su oggetti di forma inusuale. Il modello predice direttamente le coordinate spaziali dell'oggetto per ogni cella della griglia della feature map, semplificando il design e riducendo il numero di iperparametri.

Definizione 2.21 (Predizione anchor-free)**.** *Consideriamo una feature map* $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$ *associata a uno stride* $s \in \{8, 16, 32\}$ *rispetto all'immagine originale. Per ogni locazione* (i, j) *sulla griglia, il modello regredisce un vettore* $(\Delta x, \Delta y, t_w, t_h)$*. Utilizzando la funzione sigmoide* $\sigma(\cdot)$ *per vincolare il centro all'interno della cella e l'esponenziale per garantire dimensioni positive, le coordinate globali del bounding box predetto* $(x_{pred}, y_{pred}, w_{pred}, h_{pred})$ *sono ottenute come:*

$$\begin{aligned} x_{pred} &= (i + \sigma(\Delta x)) \cdot s, & y_{pred} &= (j + \sigma(\Delta y)) \cdot s, \\ w_{pred} &= e^{t_w} \cdot s, & h_{pred} &= e^{t_h} \cdot s. \end{aligned} \tag{2.40}$$

Proposizione 2.4 (Vantaggi teorici)**.** *L'approccio descritto comporta tre benefici strutturali:*

1. *Adattabilità geometrica: la parametrizzazione continua* (e^{t_w}, e^{t_h}) *permette di modellare oggetti con aspect ratio arbitrari senza i vincoli discreti delle anchor.*
2. *Riduzione del bias: elimina la necessità di calibrare gli iperparametri delle ancore (es. tramite K-means) sul dataset.*

3. *Efficienza: riduce la complessità dell'assegnazione da* $O(N_{anchor} \cdot N_{GT})$ *a* $O(N_{grid} \cdot N_{GT})$*, dato che tipicamente* $N_{grid} \ll N_{anchor}$*.*

Optimal Transport Assignment (SimOTA). L'assegnazione delle etichette viene formulata come un problema di Trasporto Ottimo (OT), nelle architetture dense, un singolo oggetto fisico è rilevato da molteplici regioni candidate. Per assegnare dinamicamente le label positive (ossia identificare quali predizioni devono essere considerate validi rilevamenti dell'oggetto/foreground rispetto allo sfondo), si utilizza la matrice di costo *Loss-Aware*:

Definizione 2.22 (Matrice di costo SimOTA)**.** *Siano* C_{ij} *il costo per associare l'i-esimo Ground Truth alla j-esima predizione, il costo combina la loss di classificazione e di regressione:*

$$C_{ij} = \mathcal{L}_{cls}(\hat{\mathbf{p}}_j, c_i) + \lambda \cdot \mathcal{L}_{IoU}(\hat{\mathbf{b}}_j, \mathbf{b}_i) \tag{2.41}$$

SimOTA determina dinamicamente il numero di candidati positivi k_i *per ogni oggetto i basandosi sulla somma delle IoU delle predizioni migliori, adattando la strategia alla complessità della scena.*

2.7.2 Il paradigma set-prediction basato su transformer: RT-DETR

L'applicazione dei transformer al rilevamento introduce il concetto di *set prediction*, formulando il task come un mapping diretto $f : \mathcal{I} \to \mathcal{Y}^N$ che predice simultaneamente l'insieme completo degli oggetti senza bisogno di soppressione dei duplicati (NMS). Il problema del rilevamento è modellato come la predizione di un insieme non ordinato di N elementi $\{(b_i, c_i)\}_{i=1}^N$. L'ottimizzazione avviene tramite **matching bipartito globale** tra le predizioni e i Ground Truth, risolto dall'Algoritmo ungherese che minimizza una loss di matching globale unificata.

Definizione 2.23 (Set prediction e algoritmo ungherese)**.** *Il problema del rilevamento è modellato come la predizione di un insieme non ordinato di* N *elementi* $\{(b_i, c_i)\}_{i=1}^N$*. L'ottimizzazione avviene tramite* ***matching bipartito globale*** *tra le predizioni* $\hat{Y} = \{\hat{y}_i\}_{i=1}^N$ *e i Ground Truth* Y*, risolvendo il problema di assegnazione lineare che minimizza il costo di matching, con un'unica previsione è ottima per uso real time:*

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i=1}^{N} \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}) \tag{2.42}$$

dove $\mathfrak{S}_N$ *è lo spazio delle permutazioni di* N *elementi e* $\mathcal{L}_{match}$ *è una loss composita (box+classe).*

RT-DETR v2 è un'evoluzione del modello originale sviluppato dai ricercatori di **Baidu** utilizzando **PaddlePaddle** (Parallel Distributed Deep Learning), il framework di deep learning open-source di punta in Cina (analogo a PyTorch o TensorFlow). Questa origine spiega l'attenzione particolare all'efficienza in inferenza su hardware eterogeneo, tipica delle implementazioni industriali su tale piattaforma. Operativamente, questo approccio si realizza tramite un'architettura **Encoder-Decoder**:

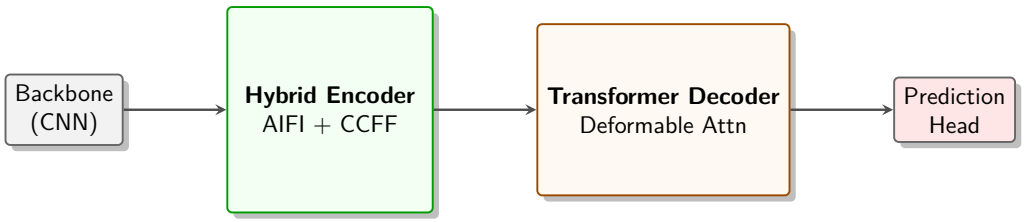
1. **Encoder:** elabora la feature map appiattita per modellare le relazioni globali tra tutte le regioni dell'immagine tramite self-attention. In **RT-DETR v2**, questa fase è ottimizzata da un *hybrid encoder* che disaccoppia le interazioni intra-scala (gestite da un layer di self-attention AIFI sulla feature map S_5) dalla fusione multi-scala (gestita da un modulo convoluzionale CCFF), superando i costi computazionali dei DETR classici.
2. **Decoder:** trasforma un insieme fisso di *object queries* (vettori apprendibili posizionali) negli embedding degli oggetti finali. Attraverso meccanismi di cross-attention, ogni query "interroga" l'output dell'encoder per estrarre informazioni specifiche su un potenziale oggetto, indipendentemente dalla sua posizione spaziale.

Definizione 2.24 (Deformable attention multi-scala)**.** *A differenza del Multi-Head Attention standard, che calcola una matrice di attenzione densa tra tutti i pixel (complessità $O(H^2W^2C)$ proibitiva per feature map ad alta risoluzione), l'attenzione deformabile introduce un meccanismo di campionamento sparso. Per ogni query $\mathbf{q}$ e punto di riferimento $\mathbf{p}_q$, invece di tutto, il modello attende solo a un piccolo insieme di K punti chiave (tipicamente $K = 4$), le cui posizioni sono predette dinamicamente tramite offset $\Delta\mathbf{p}$. Definita $\phi_l(\cdot)$ come la proiezione delle coordinate normalizzate al livello l-esimo e $\mathbf{x}^l(\cdot)$ come l'interpolazione bilineare (poiché $\phi_l(\mathbf{p}_q) + \Delta\mathbf{p}_{lqk}$ è continuo):*

$$DeformAttn(\mathbf{q}, \mathbf{p}_q, \{\mathbf{x}^l\}) = \sum_{l=1}^{L} \mathbf{W}'_l \left[\sum_{k=1}^{K} A_{lqk} \cdot \mathbf{x}^l\Big(\phi_l(\mathbf{p}_q) + \Delta\mathbf{p}_{lqk}\Big) \right] \tag{2.43}$$

La complessità si riduce linearmente a $O(HWC^2)$, permettendo di processare feature map multiscala ad alta risoluzione, essenziali per rilevare piccoli oggetti.

Figura 2.5: *Schema architetturale RT-DETR: L'hybrid encoder elabora le feature multi-scala provenienti dalla Backbone, il deformable decoder trasforma le query negli oggetti finali.*



2.8 Teoria del riconoscimento sequenziale (OCR)

Il riconoscimento ottico dei caratteri (OCR) nel contesto delle targhe automobilistiche si formalizza come un problema di *sequence prediction* supervisionato. Dato un tensore di input $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$ (rappresentante l'immagine ritagliata e normalizzata della targa), l'obiettivo è stimare la sequenza di simboli $\mathbf{Y} = (y_1, y_2, \ldots, y_L)$ di lunghezza L, dove ogni y_t appartiene a un vocabolario finito $\mathcal{V}$ (caratteri alfanumerici). In termini probabilistici, il modello apprende a stimare la distribuzione condizionata

$P(\mathbf{Y}|\mathbf{X})$. A seconda delle assunzioni di indipendenza tra i simboli, si delineano due paradigmi fondamentali: la modellazione basata su *Connectionist Temporal Classification (CTC)* e quella basata su *Attention* (Sequence-to-Sequence).

Operativamente, il modulo di riconoscimento — basato sull'implementazione ottimizzata *fast-plate OCR* (Ankandrew) — lavora su ritagli (*crops*) delle targhe, rettificati e ridimensionati a una risoluzione fissa (es. 48×144 pixel) per standardizzare la scala dei caratteri. Il tensore di input, dopo essere stato normalizzato nel range $[0, 1]$ ed eventualmente convertito in scala di grigi, viene trasformato dal transformer in una sequenza di etichette di lunghezza variabile. La stringa finale è ottenuta decodificando i logit di output, fornendo il codice alfanumerico della targa pronto per le successive fasi di verifica o validazione sintattica.

2.8.1 Evoluzione dei paradigmi: da classificatori isolati a sequence modeling

Storicamente, l'OCR è stato affrontato come una catena di due sottoproblemi distinti: segmentazione dei caratteri e classificazione isolata (*segmentation-based*). Tale approccio soffre del noto *Paradosso di Sayre*: per riconoscere un carattere è spesso necessario averlo segmentato correttamente, ma una segmentazione accurata richiede la conoscenza a priori del carattere stesso. L'approccio moderno (*segmentation-free*) supera questo limite delegando l'allineamento a meccanismi differenziabili end-to-end. In questa ottica, l'intera pipeline di riconoscimento si configura strutturalmente come un'architettura Encoder-Decoder:

- Encoder visivo: una rete neurale profonda (CNN o ViT) processa l'immagine ritagliata $\mathbf{X}$ mappandola in una rappresentazione feature tensoriale $\mathbf{H} = (\mathbf{h}_1, \ldots, \mathbf{h}_T)$, dove ogni vettore $\mathbf{h}_t$ codifica le proprietà visive locali di una porzione della targa. Questa fase realizza la *codifica* dal dominio dei pixel al dominio delle feature astratte.
- Decoder sequenziale: uno stadio ricorrente (RNN) o attentivo (transformer) trasforma la sequenza di feature $\mathbf{H}$ nella sequenza target di simboli $\mathbf{Y}$, gestendo le dipendenze sintattiche e allineando temporalmente le predizioni. Questa fase realizza la *decodifica* dal dominio delle feature al dominio simbolico del linguaggio.

Tipicamente $T \geq L$ (dove L è la lunghezza della stringa di testo), garantendo che la rete disponga di una risoluzione temporale sufficiente per risolvere disambiguità e sovrapposizioni tra caratteri adiacenti.

2.8.2 Approccio classico: Connectionist Temporal Classification

La **Connectionist Temporal Classification (CTC)** risolve la discrepanza di lunghezza tra l'input visivo (sequenza di T frame) e l'output testuale ($L < T$ simboli), senza richiedere l'allineamento esplicito frame-carattere. Il meccanismo chiave è l'introduzione del token *blank* ($\varnothing$) nell'alfabeto esteso $\mathcal{V}' = \mathcal{V} \cup \{\varnothing\}$. Data la sequenza

di predizioni frame-by-frame $\pi = (\pi_1, \ldots, \pi_T)$, si definisce un operatore di decodifica $\mathcal{B}$ (*collapsing function*) che mappa il percorso ridondante nella stringa finale $\mathbf{Y}$ rimuovendo le ripetizioni consecutive e i blank (es. $\mathcal{B}(a - a - \varnothing - b - b) = ab$). La verosimiglianza target è quindi la somma marginale su tutti i percorsi allineati validi:

$$P(\mathbf{Y}|\mathbf{X}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{Y})} \prod_{t=1}^{T} P(\pi_t|\mathbf{X})_t \tag{2.44}$$

Questa formulazione, calcolabile via algoritmo Forward-Backward, permette l'addestramento end-to-end, ma assume una forte indipendenza condizionale tra i frame: la predizione al passo t non è influenzata dai caratteri precedenti, limitando la capacità di modellare il contesto linguistico.

2.8.3 Approccio attention-based: dipendenze globali e autoregressione

I modelli *attention-based* superano l'assunzione di indipendenza della CTC adottando una decomposizione autoregressiva della probabilità congiunta:

$$P(\mathbf{Y}|\mathbf{X}) = \prod_{t=1}^{L} P(y_t|y_{<t}, \mathbf{X}) \tag{2.45}$$

In questo schema, la predizione al tempo t è condizionata esplicitamente dalla storia dei caratteri precedenti $y_{<t}$ (contesto sintattico) e dall'intera immagine $\mathbf{X}$ (contesto visivo globale). Ciò offre due vantaggi rispetto alla scansione sequenziale rigida:

1. **Robustezza alle distorsioni:** l'attenzione può focalizzarsi su regioni non contigue o inclinate, gestendo meglio le geometrie irregolari.
2. **Modellazione linguistica implicita:** il decoder apprende le transizioni probabilistiche tra caratteri (n-grammi), correggendo ambiguità visive basandosi sulla coerenza della stringa.

2.8.4 Strategie di decodifica vincolata e selezione robusta

Il problema dell'inferenza consiste nella ricerca della sequenza $\hat{\mathbf{y}}$ che massimizza la probabilità a posteriori $P(\mathbf{Y}|\mathbf{X})$. Poiché lo spazio delle ipotesi $\mathcal{V}^L$ cresce esponenzialmente con la lunghezza della sequenza, la ricerca esatta della soluzione ottima (Maximum A Posteriori - MAP) è computazionalmente unfeasible. È necessario pertanto ricorrere a strategie di ricerca approssimata che bilancino accuratezza e tempi. In contesti strutturati come l'ALPR, l'accuratezza della decodifica può essere incrementata notevolmente sfruttando la conoscenza a priori sulla sintassi delle targhe (es. pattern alfanumerici specifici per nazione), riducendo lo spazio di ricerca ammissibile.

Definizione 2.25 (Greedy Search con mascheramento sintattico)**.** *La strategia di decodifica più efficiente è la* ***Greedy Search****, che approssima la soluzione globale costruendo la sequenza in modo iterativo e miope. Per integrare i vincoli di formato,*

modelliamo la sintassi della targa come un automa a stati finiti o una espressione regolare che definisce, per ogni posizione temporale t, un sottoinsieme di simboli ammissibili $\mathcal{V}_t \subseteq \mathcal{V}$. Siano $\mathbf{z}_t \in \mathbb{R}^{|\mathcal{V}|}$ i logit non normalizzati prodotti dalla rete al passo t. L'operazione di vincolo si realizza applicando una maschera additiva $\mathbf{m}_t$ tale che $m_{t,c} = 0$ se $c \in \mathcal{V}_t$ e $m_{t,c} = -\infty$ se carattere c non vale in posizione t. La probabilità condizionata vincolata, dove $\mathbf{X}$ indica le feature visive globali estratte dall'encoder, è calcolata tramite una Softmax modificata sui logit mascherati $\tilde{\mathbf{z}}_t = \mathbf{z}_t + \mathbf{m}_t$:

$$P(y_t = c \mid \hat{y}_{<t}, \mathbf{X})_{constrained} = \frac{\exp(\tilde{z}_{t,c})}{\sum_{k \in \mathcal{V}_t} \exp(z_{t,k})} \tag{2.46}$$

La regola di aggiornamento deterministica seleziona, ad ogni passo, il token che massimizza tale probabilità:

$$\hat{y}_t = \arg\max_{c \in \mathcal{V}_t} P(y_t = c \mid \hat{y}_{<t}, \mathbf{X}) \tag{2.47}$$

Oltre alla correttezza sintattica, in scenari operativi critici è fondamentale gestire l'incertezza del modello. Questo approccio ricade nella teoria della classificazione selettiva, dove il sistema ha l'opzione di astenersi dal predire (opzione di rifiuto) per garantire un bound sull'errore commesso.

Osservazione 2.1 (Funzione di rifiuto). Per garantire l'affidabilità operativa, si adotta una strategia di *classificazione selettiva*. Il sistema calcola uno score di confidenza globale per la targa (media delle probabilità dei singoli caratteri): se tale valore è inferiore a una soglia minima τ o se la stringa non soddisfa i requisiti sintattici, la predizione viene scartata (*No-Read*). L'introduzione della soglia τ permette di modulare il trade-off tra la quantità di targhe lette (*recall*) e l'affidabilità delle letture fornite (*precision*), filtrando a monte i casi ambigui.

2.9 Ottimizzazione computazionale e architetture efficienti

L'implementazione di algoritmi di visione artificiale su piattaforme *edge*, caratterizzate da risorse limitate quali telecamere stradali o sistemi embedded, impone la risoluzione di un problema di ottimizzazione vincolata: massimizzare l'accuratezza predittiva minimizzando simultaneamente i tempi di inferenza, il consumo energetico e l'occupazione di memoria. La letteratura recente ha affrontato tale sfida ridefinendo l'operatore di convoluzione attraverso fattorizzazioni matriciali che riducono il numero di operazioni in virgola mobile (FLOPs) e sfruttano la ridondanza informativa intrinseca nelle feature map delle reti profonde.

2.9.1 Fattorizzazione delle convoluzioni: depthwise separable convolution

Il blocco costitutivo fondamentale delle moderne architetture efficienti è la depthwise separable convolution, questa tecnica decompone la convoluzione standard (definita

in Eq. 2.1) basandosi sull'ipotesi che le correlazioni spaziali e le correlazioni tra canali (cross-channel) possano essere disaccoppiate.

Richiamando il teorema 2.1, il costo computazionale di una convoluzione standard è $C_{\text{std}} = H \cdot W \cdot C_{in} \cdot C_{out} \cdot K^2$. La decomposizione *separable* mira a ridurre drasticamente tale complessità approssimando la trasformazione lineare attraverso due stadi sequenziali distinti: il filtraggio spaziale e la combinazione lineare dei canali.

Innanzitutto, la depthwise convolution applica un singolo filtro spaziale bidimensionale per ciascun canale di input, senza mescolare le informazioni tra canali diversi. Definito il kernel depthwise $\mathcal{K}_{dw} \in \mathbb{R}^{K \times K \times C_{in}}$, per cui si applica un filtro $K \times K$ per ogni singolo canale di input separatamente, l'output intermedio $\hat{\mathcal{Y}}$ è dato da:

$$\hat{\mathcal{Y}}_{i,j,c} = \sum_{u,v} \mathcal{X}_{i+u,j+v,c} \cdot \mathcal{K}_{dw,u,v,c} \tag{2.48}$$

L'operazione ha un costo ridotto di: $C_{\text{dw}} = H \cdot W \cdot C_{in} \cdot K^2$. Successivamente, per proiettare le feature map intermedie nello spazio di output a dimensione C_{out}, si applica una pointwise convolution. Questa consiste in una convoluzione con kernel unitario 1×1, $\mathcal{K}_{pw} \in \mathbb{R}^{C_{in} \times C_{out}}$, che esegue una combinazione lineare lungo l'asse della profondità per mescolare i canali e proiettarli nella dimensione di output C_{out}:

$$\mathcal{Y}_{i,j,k} = \sum_{c=1}^{C_{in}} \hat{\mathcal{Y}}_{i,j,c} \cdot \mathcal{K}_{pw,c,k} \tag{2.49}$$

Il costo di questo stadio è pari a: $C_{\text{pw}} = H \cdot W \cdot C_{in} \cdot C_{out}$. Il costo totale dell'operazione separabile è la somma dei due contributi, $C_{\text{sep}} = C_{\text{dw}} + C_{\text{pw}}$. Per valutare il guadagno di efficienza, analizziamo il rapporto η tra il costo della convoluzione separabile (C_{sep}) e quella standard (C_{std}) è:

$$\eta = \frac{H \cdot W \cdot C_{in} \cdot (K^2 + C_{out})}{H \cdot W \cdot C_{in} \cdot C_{out} \cdot K^2} = \frac{1}{C_{out}} + \frac{1}{K^2} \tag{2.50}$$

Considerando un kernel tipico di dimensione spaziale $K = 3$, il termine $1/K^2$ domina asintoticamente il rapporto (poiché solitamente $C_{out} \gg K^2$), portando a una riduzione teorica del carico computazionale di circa un fattore 9 (ovvero $\eta \approx 1/9$). Tale riduzione rende l'inferenza feasible anche su CPU general-purpose prive di acceleratori dedicati.

2.9.2 Evoluzione con la famiglia MobileNet

L'adozione sistematica di questa decomposizione ha dato origine alla famiglia di architetture MobileNet, la cui evoluzione riflette un progressivo raffinamento teorico nella gestione del flusso informativo e della non linearità.

La prima iterazione MobileNetV1 (Howard et al., 2017) ha strutturato l'intera rete come una sequenza di blocchi *depthwise separable*, dimostrando empiricamente che la rimozione delle connessioni dense spaziali non compromette significativamente l'accuratezza su dataset come ImageNet, pur riducendo drasticamente i parametri.

Con MobileNetV2 (Sandler et al., 2018) si introduce un'innovazione teorica sostanziale legata alla preservazione dell'informazione in spazi a bassa dimensione. L'intuizione geometrica è che la varietà (*manifold*) di interesse dei dati, pur essendo immersa in uno spazio di attivazione ad alta dimensionalità, giaccia in un sottospazio di dimensione inferiore. Tuttavia, l'applicazione di funzioni di attivazione non lineari (come ReLU) su tensori a bassa dimensionalità comporta una perdita irreversibile di informazione (le regioni negative vengono azzerate), per mitigare questo fenomeno viene introdotto l'inverted residual block con linear bottleneck.

La struttura del blocco inverte il classico approccio *bottleneck* per preservare l'informazione: l'input viene prima **espanso** (via convoluzione 1×1) in uno spazio a dimensionalità superiore per applicare le non-linearità senza perdita di dati, poi filtrato spazialmente con depthwise convolution e funzione di attivazione ReLU6, e infine **proiettato** nuovamente in un tensore a bassa dimensione (bottleneck lineare), dove non vengono applicate attivazioni per mantenere la struttura del manifold di interesse dei dati.

Infine, **MobileNetV3** (Howard et al., 2019) ottimizza l'architettura tramite **Neural Architecture Search (NAS)**. Il problema di design viene formulato come una ricerca automatica nello spazio delle configurazioni $\mathcal{A}$ per massimizzare l'accuratezza $\text{ACC}(m)$ rispettando un vincolo rigido di latenza target T. L'algoritmo (MNasNet) utilizza una funzione di ricompensa *hardware-aware* che misura direttamente il tempo di inferenza sul device reale, garantendo che il modello selezionato sia ottimizzato per l'efficienza pratica e non solo per i FLOPs teorici.

In un'ottica unificante, anche MobileNetV3 può essere interpretata secondo il paradigma Encoder-Decoder:

- Feature encoder in cui la sequenza di blocchi Inverted Residual funge da estrattore di caratteristiche, comprimendo l'immagine di input $\mathcal{I}$ in un vettore di feature compatto $\mathbf{z} \in \mathbb{R}^D$ (embedding) che cattura la semantica globale (es. *nazionalità francese*).
- Classification decoder in cui il layer lineare finale (Fully Connected) agisce da decoder, proiettando l'embedding $\mathbf{z}$ nello spazio delle probabilità di classe $\mathcal{Y} \in [0, 1]^K$ delle nazionalità.

Oltre all'ottimizzazione, l'architettura introduce modifiche mirate a massimizzare l'efficienza su hardware a risorse vincolate. La funzione di attivazione sigmoide, risulta computazionalmente onerosa sui processori embedded poiché l'operazione di esponenziazione richiede costosi cicli di clock o grandi tabelle di lookup, per ovviare a ciò, si adotta una strategia di approssimazione polinomiale a tratti.

2.10 Funzioni di costo

La supervisione dei modelli di Deep Learning richiede la definizione di obiettivi differenziati che guidino il processo di ottimizzazione. Nel contesto ALPR, l'accuratezza finale è vincolata non solo dall'architettura, ma dalla specifica formulazione della *funzione obiettivo*: essa determina quali errori vengono amplificati (e con quale

peso) e quindi quale tipologia di soluzione il training tende a privilegiare. Poiché la pipeline proposta combina modelli eterogenei (detection anchor-free, transformer set-based, OCR sequenziale), è essenziale definire le loss adottate per il bilanciamento tra localizzazione, classificazione e riconoscimento.

2.10.1 Loss per classificazione

Nei compiti di rilevamento, la classificazione distingue tra la presenza di un oggetto (foreground) e lo sfondo (background), spesso in condizioni di forte sbilanciamento di classe.

Definizione 2.26 (Binary Cross-Entropy – BCE)**.** *Consideriamo un problema di classificazione binaria in cui l'etichetta $y \in \{0,1\}$ indica la classe corretta e la rete produce un logit $z \in \mathbb{R}$. Indicando con $p = \sigma(z) \in (0,1)$ la probabilità stimata tramite funzione sigmoide, la BCE misura la divergenza tra la distribuzione empirica e quella predetta:*

$$\mathcal{L}_{BCE}(p, y) = -\left[y \log(p) + (1-y)\log(1-p)\right] \tag{2.51}$$

Definizione 2.27 (Categorical Cross-Entropy – CCE)**.** *Nel caso di classificazione multi-classe ($K > 2$ classi), il target è rappresentato da un vettore one-hot $\mathbf{y} \in \{0,1\}^K$ (dove $y_c = 1$ solo per la classe corretta c) e la rete predice un vettore di probabilità $\mathbf{p} \in [0,1]^K$ tramite softmax. La CCE generalizza la BCE sommando le log-verosimiglianze su tutte le K classi:*

$$\mathcal{L}_{CCE}(\mathbf{p}, \mathbf{y}) = -\sum_{k=1}^{K} y_k \log(p_k) \tag{2.52}$$

Poiché $\mathbf{y}$ è one-hot, questo si riduce a $-\log(p_c)$, ovvero a massimizzare la probabilità assegnata alla classe vera.

Definizione 2.28 (Label Smoothing)**.** *Per prevenire l'eccessiva confidenza (tendenza della rete a predire probabilità $p_c \to 1$, causando sovradattamento), nei transformer si sostituisce il target one-hot $\mathbf{y}$ con una distribuzione smussata $\mathbf{y}^{LS}$. Fissato un parametro di smoothing $\epsilon \in (0,1)$:*

$$y_k^{LS} = (1-\epsilon) y_k + \frac{\epsilon}{K} \tag{2.53}$$

La loss CCE calcolata su $\mathbf{y}^{LS}$ penalizza la certezza estrema (overconfidence), migliorando la calibrazione del modello e la generalizzazione.

Definizione 2.29 (Focal Loss)**.** *Per mitigare lo sbilanciamento di classe (dove i negativi facili dominano il gradiente), si introduce un fattore di modulazione. Fissati un parametro di focalizzazione $\gamma \geq 0$ e un fattore di bilanciamento $\alpha_t \in (0,1]$, e definendo la probabilità della classe corretta come $p_t = y \cdot p + (1-y) \cdot (1-p)$, la Focal Loss è data da:*

$$\mathcal{L}_{FL}(p_t) = -\alpha_t (1-p_t)^{\gamma} \log(p_t) \tag{2.54}$$

Il termine $(1-p_t)^{\gamma}$ riduce drasticamente il peso dei campioni ben classificati ($p_t \to 1$), concentrando l'attenzione del modello sugli esempi difficili.

Definizione 2.30 (Varifocal Loss – VFL)**.** *La Varifocal Loss, adottata in architetture moderne come RT-DETR v2, estende la Focal Loss per gestire target continui (es. score di qualità IoU). Considerando una probabilità predetta $p \in (0, 1)$ e un target di qualità $q \in [0, 1]$, e fissati gli iperparametri $\alpha \in (0, 1]$ e $\gamma \geq 0$, la funzione è definita a tratti:*

$$\mathcal{L}_{VFL}(p, q) = \begin{cases} -q\,(q \log(p) + (1-q)\log(1-p)) & \text{se } q > 0 \text{ (Foreground)} \\ -\alpha p^{\gamma} \log(1-p) & \text{se } q = 0 \text{ (Background)} \end{cases} \tag{2.55}$$

Per i campioni positivi, la loss è pesata dallo score q (qualità IoU), mentre per i negativi segue la riduzione standard della Focal Loss per abbattere il contributo dello sfondo.

2.10.2 Loss per regressione (bounding box)

La precisione nella localizzazione richiede metriche che vadano oltre la semplice distanza euclidea tra coordinate, catturando la sovrapposizione geometrica.

Definizione 2.31 (Intersezione sull'Unione – IoU)**.** *Dati due box assiali $B_p, B_{gt} \subset \mathbb{R}^2$ (rispettivamente predetto e Ground Truth) e indicando con $|\cdot|$ l'operatore di area nel piano, l'IoU è definita come il rapporto tra l'area dell'intersezione e quella dell'unione:*

$$IoU = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|} \tag{2.56}$$

La funzione di costo base associata è $\mathcal{L}_{IoU} = 1 - IoU$.

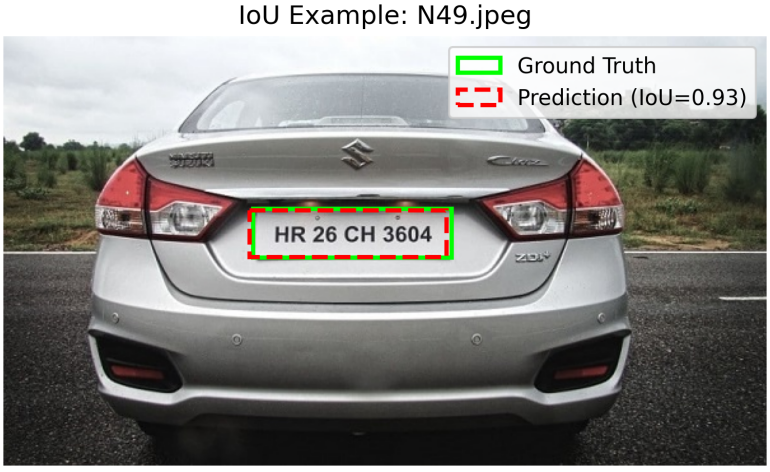


Figura 2.6: *Esempio di calcolo IoU su un campione del dataset Archive (Kaggle). Il box verde rappresenta il Ground Truth, mentre il box rosso tratteggiato indica la predizione del modello RT-DETR v2 r50 ($IoU > 0.9$).*

Definizione 2.32 (Generalized IoU – GIoU)**.** *Per risolvere il problema del gradiente nullo quando i box sono disgiunti ($IoU = 0$), la GIoU introduce il minimo box convesso C che racchiude entrambi i box ($B_p \cup B_{gt} \subseteq C$). Penalizzando l'area vuota*

all'interno di C, la loss diviene:

$$\mathcal{L}_{GIoU} = 1 - \left(IoU - \frac{|C \setminus (B_p \cup B_{gt})|}{|C|} \right) \tag{2.57}$$

Questo termine aggiuntivo guida l'ottimizzazione anche in assenza di sovrapposizione, spingendo il box predetto verso il target.

Definizione 2.33 (Complete IoU – CIoU)**.** *La CIoU Loss migliora la corrispondenza geometrica penalizzando non solo la mancata sovrapposizione, ma anche la distanza tra i centri e la discrepanza nelle proporzioni. Siano $\mathbf{b}_p, \mathbf{b}_{gt} \in \mathbb{R}^2$ i centroidi del box predetto e del Ground Truth, e sia $\rho(\cdot)$ la distanza euclidea. Definiamo c come la lunghezza della diagonale del minimo box rettangolare che racchiude entrambi i box $(B_p \cup B_{gt})$. Introducendo un termine di consistenza dell'aspect ratio v e un peso di bilanciamento $\alpha > 0$, il cui prodotto risulta la discrepanza aspect ratio, la funzione di costo è formulata come:*

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}_p, \mathbf{b}_{gt})}{c^2} + \alpha v \tag{2.58}$$

Il termine centrale $\frac{\rho^2}{c^2}$ normalizza la distanza dei centri rispetto alla dimensione dell'area di interesse, rendendo la penalità invariante alla scala: un errore di posizionamento penalizza simultaneamente sovrapposizione sbagliata, la distanza spaziale e la deformazione geometrica sia su oggetti piccoli che su oggetti grandi.

2.10.3 Loss per riconoscimento sequenziale

Nel modulo OCR, si ricorre alla CTC Loss per addestrare modelli sequenziali in assenza di allineamento esplicito, richiamando la verosimiglianza $P(\mathbf{Y}|\mathbf{X})$ marginalizzata su tutti i possibili percorsi π (introdotta nella Sez. 2.8.2), la funzione di costo è definita come la *negative log-likelihood*:

$$\mathcal{L}_{\text{CTC}} = -\log P(\mathbf{Y}|\mathbf{X}) = -\log \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{Y})} \prod_{t=1}^{T} p(\pi_t|\mathbf{X}) \tag{2.59}$$

Questa formulazione differenziabile permette di addestrare il modello end-to-end, delegando alla rete l'apprendimento dell'allineamento ottimale tramite l'algoritmo forward-backward, sommando le probabilità di tutti i possibili percorsi che portano alla stringa corretta.

2.11 Metodologie di valutazione

La validazione quantitativa del sistema proposto si fonda sulle metriche standard adottate dalla comunità scientifica per i task di object detection (protocollo COCO) e di Optical Character Recognition, di seguito forniamo le definizioni degli indicatori utilizzati.

Metriche per object detection. La valutazione delle capacità di localizzazione e classificazione si basa sul concetto geometrico di sovrapposizione tra aree. In particolare, utilizziamo l'**Intersection over Union (IoU)** (def 2.31) e la **mean Average Precision (mAP)** (Def 2.36), che riassume l'area sotto la curva precision-recall su più soglie di IoU. Per analizzare le prestazioni del classificatore al variare della soglia di confidenza, si introducono i concetti di precision e recall.

Definizione 2.34 (Precision e recall)**.** *Fissata una soglia di sovrapposizione $\tau \in [0,1]$ e una regola di matching univoca (tipicamente greedy rispetto allo score di confidenza), classifichiamo ogni predizione come true positive (TP) se il suo IoU con un Ground Truth è almeno τ, o come false positive (FP) altrimenti. I Ground Truth non associati ad alcuna predizione costituiscono i false negative (FN). Indicando con $N_{TP}(\tau)$, $N_{FP}(\tau)$ e $N_{FN}(\tau)$ le rispettive cardinalità, definiamo la precisione (frazione di predizioni corrette, su quanto è onesto) e il richiamo (frazione di oggetti trovati, su quanto è attento) come funzioni della soglia τ:*

$$Precision(\tau) = \frac{N_{TP}(\tau)}{N_{TP}(\tau) + N_{FP}(\tau)}, \qquad Recall(\tau) = \frac{N_{TP}(\tau)}{N_{TP}(\tau) + N_{FN}(\tau)} \tag{2.60}$$

Definizione 2.35 (Average Precision – AP)**.** *L'Average Precision sintetizza la curva precision-recall. Consideriamo l'insieme discretizzato di livelli di richiamo $R = \{0, 0.01, \dots, 1\}$ e definiamo la precisione interpolata $p_{interp}(r)$ come il massimo valore di precisione ottenuto per un richiamo $\tilde{r} \geq r$. L'AP, calcolata a una specifica soglia IoU τ, è l'approssimazione dell'integrale della curva interpolata:*

$$AP(\tau) = \frac{1}{|R|} \sum_{r \in R} p_{interp}(r;\tau) = \frac{1}{101} \sum_{r \in R} \max_{\tilde{r} \geq r} p(\tilde{r};\tau) \tag{2.61}$$

L'interpolazione garantisce che la metrica sia monotona decrescente e robusta alle oscillazioni locali della curva (wiggle).

Definizione 2.36 (mean Average Precision – mAP COCO)**.** *La metrica primaria del benchmark COCO è la media aritmetica delle AP calcolate su diverse classi e soglie di sovrapposizione. Sia $\mathcal{C}$ l'insieme delle classi di oggetti e sia $\mathcal{T} = \{0.50, 0.55, \dots, 0.95\}$ l'insieme di 10 soglie IoU equispaziate, da una soglia permissiva ad una severa. Indicando con $AP_c(\tau)$ l'AP per la classe c alla soglia τ, la mAP è definita come:*

$$mAP@[0.5:0.95] = \frac{1}{|\mathcal{C}| \cdot |\mathcal{T}|} \sum_{c \in \mathcal{C}} \sum_{\tau \in \mathcal{T}} AP_c(\tau) \tag{2.62}$$

Definizione 2.37 (Average Recall – AR@K)**.** *Per valutare la capacità del modello di proporre candidati, si utilizza l'Average Recall limitata alle prime K predizioni per immagine. Sia $Recall(\tau, K)$ il richiamo calcolato considerando solo le K detection a confidenza maggiore. L'AR@K media tale valore sull'insieme delle soglie $\mathcal{T}$:*

$$AR@K = \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} Recall(\tau, K) \tag{2.63}$$

Nel dominio ALPR, la metrica AR@100 è di particolare interesse per minimizzare il rischio di mancato rilevamento (miss rate), assumendo che è preferibile avere dei falsi positivi che possano essere filtrati dall'OCR, piuttosto che perdere un veicolo in transito.

Metriche per (OCR). La valutazione del modulo di riconoscimento si basa sulla quantificazione della distanza tra stringhe nel monoide libero generato dall'alfabeto dei caratteri.

Definizione 2.38 (Metrica di Levenshtein)**.** *Sia Σ un alfabeto finito e siano $\mathbf{s}, \mathbf{t} \in \Sigma^*$ due stringhe di lunghezza rispettivamente $|\mathbf{s}| = n$ e $|\mathbf{t}| = m$. La distanza di Levenshtein è una metrica $d_L : \Sigma^* \times \Sigma^* \to \mathbb{N}_0$ che quantifica la dissimilarità tra le due sequenze come il costo minimo di una catena di trasformazioni elementari (inserimento, cancellazione, sostituzione) necessarie per convertire $\mathbf{s}$ in $\mathbf{t}$.*

Indicando con $\mathbf{s}[1..i]$ e $\mathbf{t}[1..j]$ i prefissi delle due stringhe di lunghezza i e j, e definendo la funzione indicatrice $\mathbb{I}(a \neq b)$ che vale 1 se i caratteri a e b differiscono e 0 altrimenti, la distanza è determinata dalla seguente relazione di ricorrenza, che è il numero minimo di operazioni necessarie per trasformare la predizione nella targa vera:

$$lev(i,j) = \begin{cases} \max(i,j) & \text{se } \min(i,j) = 0, \\ \min \begin{cases} lev(i-1,j)+1 & (\textit{cancellazione}) \\ lev(i,j-1)+1 & (\textit{inserimento}) \\ lev(i-1,j-1)+\mathbb{I}(\mathbf{s}_i \neq \mathbf{t}_j) & (\textit{sostituzione}) \end{cases} & \text{altrimenti.} \end{cases} \tag{2.64}$$

Il valore finale della distanza tra le stringhe complete è dato da $d_L(\mathbf{s}, \mathbf{t}) = lev(n, m)$. Tale definizione soddisfa le proprietà assiomatiche di una metrica: non negatività, identità degli indiscernibili, simmetria e disuguaglianza triangolare.

Denotando con S, D, I rispettivamente il numero di sostituzioni, cancellazioni e inserimenti in un allineamento ottimale, si ha: $\text{lev}(\mathbf{n}, \mathbf{m}) = S + D + I$

Definizione 2.39 (Character Error Rate – CER)**.** *Il CER rappresenta la distanza di Levenshtein normalizzata rispetto alla lunghezza della sequenza target. Indicando con $\hat{\mathbf{y}}$ la stringa predetta, con $\mathbf{y}$ la stringa Ground Truth e con $N = |\mathbf{y}|$ la sua lunghezza, il tasso di errore è dato da:*

$$CER(\hat{\mathbf{y}}, \mathbf{y}) = \frac{lev(\hat{\mathbf{y}}, \mathbf{y})}{N} \times 100\% \tag{2.65}$$

Definizione 2.40 (Plate accuracy)**.** *L'accuratezza a livello di targa misura la frequenza di predizioni perfette, su un dataset di M campioni, l'accuratezza è:*

$$Acc_{plate} = \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}\Big(lev(\hat{\mathbf{y}}_i, \mathbf{y}_i) = 0 \Big) \tag{2.66}$$

Definizione 2.41 (Character accuracy)**.** *L'accuratezza a livello di carattere misura la frazione di caratteri correttamente predetti sull'insieme di valutazione, per cui è più granulare. Siano M il numero di campioni e $L_i = |\mathbf{y}_i|$ la lunghezza della i-esima targa, definendo il totale dei caratteri $T = \sum_{i=1}^{M} L_i$, si ha:*

$$Acc_{char} = \frac{1}{T} \sum_{i=1}^{M} \sum_{j=1}^{L_i} \mathbb{I}\Big(\hat{y}_{i,j} = y_{i,j} \Big) \tag{2.67}$$

Questa metrica fornisce una misura granulare rispetto a Acc_{plate}, consentendo di quantificare errori parziali nella sequenza predetta.

Definizione 2.42 (Top-k character inclusion)**.** *Sia Top-k$(\mathbf{p}_{i,j})$ l'insieme delle k etichette con probabilità più alta prodotte dal modello per il carattere in posizione j. La metrica valuta la frequenza con cui il carattere ground-truth $y_{i,j}$ è incluso in tale insieme di candidati:*

$$\textit{Top-k\_Inclusion} = \frac{1}{T}\sum_{i=1}^{M}\sum_{j=1}^{L_i}\mathbb{I}\Big(y_{i,j} \in \textit{Top-k}(\mathbf{p}_{i,j})\Big) \tag{2.68}$$

Nelle nostre valutazioni sperimentali sul modulo OCR, abbiamo adottato $k = 3$. Questa scelta è motivata dalla necessità di disambiguare classi visivamente isomorfe (es. B/8, D/0, I/1) che spesso appaiono con confidenze simili nelle prime posizioni del ranking. Un'elevata inclusione top-3 indica che il modello ha appreso correttamente le feature discriminanti, anche se l'ambiguità locale impedisce talvolta la classificazione top-1 esatta.

Definizione 2.43 (Plate length accuracy)**.** *Misura la frequenza con cui la lunghezza della stringa predetta coincide con la lunghezza ground-truth. Su M campioni:*

$$Acc_{len} = \frac{1}{M}\sum_{i=1}^{M}\mathbb{I}\Big(|\hat{\mathbf{y}}_i| = |\mathbf{y}_i|\Big) \tag{2.69}$$

Questa metrica cattura errori strutturali (inserimenti/cancellazioni) non pienamente evidenziati da Acc_{char}.

Metriche per classificazione nazionalità (MobileNetV3) Nel contesto del modulo di classificazione della nazionalità, dove il modello deve assegnare ogni crop di targa a una tra K classi possibili (corrispondenti ai diversi paesi europei), è necessario adottare metriche specifiche per problemi di classificazione multi-classe. Tali metriche permettono di valutare non solo l'accuratezza complessiva del sistema, ma anche le prestazioni per singola classe e la qualità della calibrazione delle probabilità predette.

Definizione 2.44 (Top-1 accuracy)**.** *Sia $\mathcal{D}_{test} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{M}$ un dataset di test con M campioni, dove $\mathbf{x}_i$ rappresenta l'input (crop della targa) e $y_i \in \{1, \ldots, K\}$ la classe vera. Indicando con $f(\mathbf{x}_i; \theta)$ il modello parametrico che produce un vettore di logit, e applicando la funzione softmax per ottenere una distribuzione di probabilità $\mathbf{p}_i = \textit{softmax}(f(\mathbf{x}_i; \theta)) \in [0, 1]^K$, definiamo la predizione come la classe con probabilità massima $\hat{y}_i$. L'accuratezza Top-1 è la frazione di campioni per cui la predizione coincide con la label vera:*

$$Acc_{\textit{Top-1}} = \frac{1}{M}\sum_{i=1}^{M}\mathbb{I}(\hat{y}_i = y_i) \tag{2.70}$$

Definizione 2.45 (Top-K accuracy)**.** *In molti scenari applicativi, è utile valutare se la classe corretta è presente tra le K predizioni con probabilità più alta, piuttosto che richiedere che sia necessariamente al primo posto. Questo è particolarmente*

rilevante quando il modello deve gestire classi visivamente simili (es. targhe italiane vs francesi, entrambe con banda blu EU).

Sia Top-K($\mathbf{p}_i$) l'insieme delle K classi con probabilità maggiore per il campione i-esimo. L'accuratezza Top-K è definita come:

$$Acc_{Top\text{-}K} = \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}(y_i \in Top\text{-}K(\mathbf{p}_i)) \tag{2.71}$$

Nel contesto di questa tesi, utilizziamo $K = 3$ per valutare la capacità del modello di fornire una short-list di candidati plausibili, utile per disambiguazione manuale o per logiche di fallback automatiche.

Per analizzare le prestazioni del classificatore su singole nazioni (particolarmente importante per identificare classi problematiche), è necessario estendere i concetti di precision e recall dal contesto binario a quello multi-classe.

Definizione 2.46 (F1-Score per classe)**.** *L'F1-Score è la media armonica di precision e recall, fornendo una singola metrica che bilancia entrambi gli aspetti:*

$$F1_c(\tau) = \frac{2 \cdot N_{TP}^{(c)}(\tau)}{2 \cdot N_{TP}^{(c)}(\tau) + N_{FP}^{(c)}(\tau) + N_{FN}^{(c)}(\tau)} \tag{2.72}$$

L'F1-Score è particolarmente utile quando le classi sono sbilanciate, poiché penalizza sia i falsi positivi (bassa precision) che i falsi negativi (bassa recall).

Per ottenere una metrica globale che riassuma le prestazioni su tutte le K classi, esistono due strategie di aggregazione principali, ciascuna con diverse proprietà statistiche e interpretazioni operative.

Definizione 2.47 (Macro-averaging)**.** *Il macro-averaging calcola la metrica (precision, recall o F1) per ogni classe indipendentemente, e poi ne fa la media aritmetica semplice, trattando tutte le classi con uguale importanza indipendentemente dalla loro frequenza nel dataset.*

Siano $Precision_c$, $Recall_c$ e $F1_c$ le metriche calcolate per la classe c. Con K numero totale di campioni, le metriche macro sono definite come:

$$Precision_m = \frac{1}{K} \sum_{c=1}^{K} Precision_c, \qquad Recall_m = \frac{1}{K} \sum_{c=1}^{K} Recall_c, \qquad F1_m = \frac{1}{K} \sum_{c=1}^{K} F1_c \tag{2.73}$$

Le metriche macro sono sensibili alle prestazioni sulle classi minoritarie. Un valore basso di $Precision_m$ indica che il modello ha difficoltà su almeno alcune classi, anche se queste sono rare. Questo è desiderabile quando tutte le nazioni devono essere riconosciute con uguale affidabilità (es. targhe diplomatiche rare ma critiche).

Definizione 2.48 (Weighted-averaging)**.** *Il weighted-averaging calcola una media pesata delle metriche per classe, dove il peso di ciascuna classe c è proporzionale al numero di campioni n_c di quella classe nel dataset di test.*

Sia n_c il numero di campioni della classe c nel test set, con $\sum_{c=1}^{K} n_c = M$. Le metriche weighted sono:

$$\begin{aligned} Precision_w &= \frac{1}{M}\sum_{c=1}^{K} n_c \cdot Precision_c, \qquad Recall_w = \frac{1}{M}\sum_{c=1}^{K} n_c \cdot Recall_c \\ F1_w &= \frac{1}{M}\sum_{c=1}^{K} n_c \cdot F1_c \end{aligned} \tag{2.74}$$

Le metriche weighted riflettono le prestazioni complessive del sistema pesate per la distribuzione reale dei dati. Un valore alto di $Precision_w$ indica che il modello è accurato sulla maggior parte dei campioni, anche se potrebbe fallire su classi rare. Questa metrica è più rappresentativa dell'accuratezza operativa attesa in produzione.

Osservazione 2.2 (Relazione tra metriche). Per un classificatore multi-classe perfettamente bilanciato (dove tutte le classi hanno lo stesso numero di campioni), le metriche macro e weighted coincidono. Inoltre, per la recall, vale sempre l'identità:

$$\text{Recall}_{\text{weighted}} = \text{Acc}_{\text{Top-1}} \tag{2.75}$$

poiché la recall pesata è equivalente alla frazione totale di campioni correttamente classificati.

2.12 Sintesi del capitolo

Il presente capitolo ha sistematizzato il substrato matematico e metodologico necessario alla comprensione delle moderne architetture, muovendo dalla formalizzazione dell'algebra tensoriale fino alle dinamiche di ottimizzazione stocastica del primo ordine. È stata esaminata l'evoluzione degli algoritmi di aggiornamento dei pesi, motivando la scelta di **AdamW** per la sua capacità di disaccoppiare il decadimento del peso (*weight decay*) dall'aggiornamento del gradiente, e sono state analizzate le funzioni di costo (*loss functions*) specifiche per la regressione dei bounding box e la classificazione categorica. Parallelamente, si è discusso il ruolo cruciale delle funzioni di attivazione efficienti nel garantire la stabilità numerica e ridurre il costo computazionale su hardware embedded.

L'analisi si è poi concentrata sui paradigmi architetturali per la computer vision. Per quanto concerne il rilevamento oggetti, è stato evidenziato il cambio di paradigma introdotto dalle architetture *anchor-free* e basate su transformer. In particolare, si è discusso come la formulazione del problema in termini di *Set prediction* e l'uso del matching bipartito (Algoritmo ungherese) in architetture come **RT-DETR v2** permettano di eliminare la fase di post-processamento **NMS-free** (Non-Maximum Suppression), superando i limiti delle euristiche tradizionali. Analogamente, per l'assegnazione dinamica delle etichette in YOLOX, è stato introdotto il concetto di Trasporto Ottimo (SimOTA).

Sul fronte della classificazione e del riconoscimento, sono stati delineati due approcci distinti. Per il task di identificazione della *nazionalità*, l'attenzione è stata posta sull'efficienza delle reti convoluzionali leggere come **MobileNet**, ottimizzate per

l'inferenza a bassa latenza. Per il riconoscimento ottico dei caratteri (OCR), si è invece approfondito il dualismo teorico tra l'indipendenza condizionale della **CTC** e la natura autoregressiva dei meccanismi di **attention**, evidenziando come questi ultimi offrano una modellazione linguistica implicita superiore per la sintassi delle targhe.

Questa base teorica costituisce il fondamento per il capitolo 3, dedicato all'implementazione della pipeline proposta. I concetti qui esposti giustificano le scelte progettuali e i compromessi tra accuratezza e velocità che hanno guidato lo sviluppo del sistema.

3. Design e implementazione della pipeline proposta

Il presente capitolo traduce il framework teorico delineato nel capitolo 2 in un'architettura operativa concreta. La progettazione della pipeline a tre stadi risponde ai vincoli del problema di *Automatic License Plate Recognition* (ALPR), bilanciando latenza e accuratezza. In particolare, l'adozione di **RT-DETR v2** è motivata dalla necessità di un'inferenza deterministica *NMS-free*, mentre l'uso dei **CCT** per il riconoscimento ottimizza l'efficienza sui dati rispetto ai Vision Transformer puri. Nelle sezioni seguenti, le scelte architetturali vengono giustificate alla luce dei vincoli applicativi.

3.1 Criteri di progettazione e scelte architetturali

L'architettura proposta implementa una strategia a **tre stadi** sequenziali (*three-stage*), composta da un modulo di localizzazione della targa (*detection*), un modulo di classificazione della nazionalità (*nationality classification*) e un modulo di riconoscimento dei caratteri (*recognition*) operante su ritagli (*crop*) normalizzati. Questa topologia nasce dalla necessità di minimizzare la latenza totale del sistema mantenendo alta l'accuratezza. Siano $\mathcal{X} \in \mathbb{R}^{H \times W \times C}$ il tensore rappresentante l'immagine di input e $\mathcal{T} = \{t_1, \ldots, t_K\}$ l'insieme delle targhe presenti nella scena. Definiamo f_{det} come la funzione di mapping del detector, f_{nat} come la funzione di classificazione nazionale e f_{rec} come la funzione del riconoscitore. Il tempo totale di inferenza T_{total} può essere modellato come la somma del costo di rilevamento globale e dei costi di processamento locali; denotando con N_{roi} il numero di targhe identificate da $f_{\text{det}}(\mathcal{X})$ e con C_k il k-esimo ritaglio di immagine, la decomposizione della latenza è data da:

$$T_{\text{total}}(\mathcal{X}) = T_{\text{det}}(\mathcal{X}) + \sum_{k=1}^{N_{\text{roi}}} \left(T_{\text{nat}}(C_k) + T_{\text{rec}}(C_k) \right) + T_{\text{overhead}} \tag{3.1}$$

dove T_{det} è il tempo di esecuzione del detector, T_{nat} è il tempo per l'identificazione della nazionalità, T_{rec} è il costo computazionale per il riconoscimento dei caratteri, e T_{overhead} include le operazioni di pre-processing, caricamento modelli e dati.

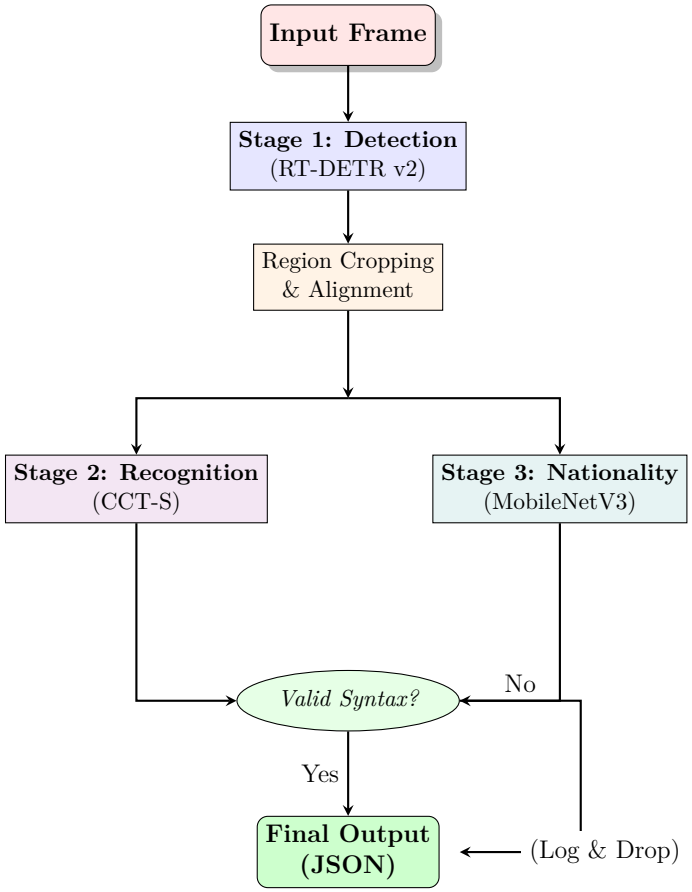


Figura 3.1: *Diagramma di flusso della pipeline ALPR a tre stadi. L'immagine in input viene processata dal detector per estrarre le regioni di interesse, che vengono parallelizzate verso i moduli di riconoscimento e classificazione. Un validatore sintattico finale aggrega i risultati.*

Il dominio ALPR offre un vantaggio decisivo: l'operazione di *cropping* agisce come un normalizzatore geometrico esplicito, permettendo ai costi locali di essere ordini di grandezza inferiori a T_{det}.

3.1.1 Dai vincoli di dominio alle scelte architetturali

La definizione delle specifiche componenti di questa tesi discende dall'analisi dei vincoli operativi tipici degli scenari *edge computing*, il sistema deve garantire esecuzione in tempo reale su CPU commodity, un basso consumo di operazioni in virgola mobile (FLOPs) e la capacità di rilevare oggetti di piccole dimensioni (spesso $< 1\%$ dell'area del frame) in condizioni di prospettiva obliqua e motion blur. Per soddisfare tali requisiti, sono stati selezionati modelli che offrono specifici vantaggi teorici:

Per lo stadio di rilevamento, la scelta primaria è ricaduta su **RT-DETR v2**, come discusso in Sezione 3.3, questa architettura implementa il paradigma di *set prediction* (Def. 2.23). La proprietà decisiva per il contesto ALPR è la sua natura *NMS-free*: eliminando la necessità della soppressione non-massima, RT-DETR v2 garantisce un tempo di inferenza deterministico indipendentemente dalla densità di veicoli nella

scena, fattore critico per la certificazione Real-Time, parallelamente, si mantiene **YOLOX** come baseline CNN di riferimento. Sebbene basato su un paradigma precedente, l'adozione di un approccio senza ancore e di teste disaccoppiate (Head, Sez. 3.2) lo rende un competitore efficiente per scenari a risorse estremamente limitate.

Per il riconoscimento, i Vision Transformer puri (ViT) soffrono di scarsa efficienza sui dati (*data inefficiency*) in assenza di pre-training massivo, poiché mancano del bias induttivo di località tipico delle convoluzioni. Il CCT risolve questo limite sostituendo la proiezione lineare delle patch con un **tokenizer convoluzionale**. Questo stadio preliminare estrae feature locali robuste (bordi, angoli) e preserva le relazioni spaziali di vicinato, permettendo al successivo encoder Transformer di concentrarsi sulla modellazione delle dipendenze globali a lungo raggio tra i caratteri. Tale processo si inquadra nel paradigma di apprendimento **Sequence-to-Sequence (Seq2Seq)**, dove l'obiettivo è mappare una sequenza di caratteristiche visive in una sequenza di target discreti; a differenza degli approcci classici, la modellazione Seq2Seq basata sull'attenzione consente di catturare le correlazioni sintattiche tra i simboli, agendo come correttore semantico naturale per la sintassi delle targhe.

Per quantificare l'efficienza operativa del sistema, introduciamo la metrica di throughput standard.

Definizione 3.1 (Throughput e FPS)**.** *Il throughput del sistema è misurato in Frames Per Second (FPS). Dato un batch di input di dimensione B (batch size) e il tempo totale di inferenza T_{batch} (espresso in secondi) necessario per elaborarlo, l'FPS è definito come:*

$$FPS = \frac{B}{T_{batch}} \tag{3.2}$$

Nel contesto real-time ($B = 1$), l'FPS corrisponde all'inverso della latenza per singolo frame.

Coerentemente con questa metrica, il budget complessivo è fissato a $T_{\max} = 0.300$ s (300 ms) per frame, garantendo un throughput minimo di ≈ 3.3 FPS su hardware non accelerato (con $B = 1$). L'allocazione delle risorse, dettagliata in Tab. 3.1, privilegia il detector, che elabora la mole maggiore di dati, assegnando slot temporali ridotti ai moduli di classificazione.

Tabella 3.1: *Allocazione del budget computazionale per la pipeline.*

Modulo funzionale	**Budget Max (ms)**	**Throughput target**
Detection (RT-DETR v2 / YOLOX)	≤ 100	≥ 10 FPS
Recognition (CCT-S)	≤ 50 (per crop)	≥ 20 crop/s
Nationality recognition (opzionale)	≤ 50	–
Logica di sistema e post-proc.	≤ 80	–
Latenza totale pipeline	$\leq$ **300**	$\geq$ **4 FPS**

Mentre il detector opera su un dominio ad alta risoluzione spaziale per discriminare l'oggetto dallo sfondo, il recognizer agisce come un raffinatore semantico su un

dominio a bassa risoluzione ma alta densità informativa. La scelta di un'architettura modulare a tre stadi (detection → classification → recognition) risponde a due esigenze:

1. **Separazione dei domini di ottimizzazione**: il detector opera su immagini ad alta risoluzione, mentre recognition e classification processano crop normalizzati a bassa dimensionalità (128x64), riducendo il costo computazionale aggregato di 5 volte rispetto ad approcci end-to-end, che son monolitici.
2. **Specializzazione dei modelli**: ciascun modulo viene ottimizzato indipendentemente sul proprio task, evitando conflitti di gradiente tipici dei sistemi monolitici multi-task, con un'ottimizzazione specifica per ogni funzione.

Tabella 3.2: *Confronto funzionale tra i moduli di detection e recognition.*

Caratteristica	**Stadio 1: Detection (RT-DETR v2)**	**Stadio 2: Recognition (CCT)**
Paradigma	Transformer-based Detector (Set Prediction)	Hybrid CNN-Transformer (Seq2Seq)
Dominio di Input	Frame intero (High-Res, $\sim 10^6$ px)	Crop normalizzato (128×64 px, $\sim 10^4$ px)
Output	Insieme di tuple (Box, Score, Class)	Sequenza di token (Caratteri alfanumerici)
Cardinalità inferenza	1 volta per frame	N volte per frame ($N = \#$ targhe)
Framework	PyTorch (Export ONNX)	TensorFlow/Keras (Export ONNX/TFLite)
Configurazione	Backbone ResNet-18/50	Variante S (Encoder a 4 layer)

L'efficienza operativa della pipeline non dipende unicamente dalla complessità algoritmica dei singoli modelli, ma è fortemente influenzata dalla parallelizzazione del carico di lavoro. In fase di inferenza, l'incremento della dimensione del batch (BS) permette di ammortizzare i costi fissi di trasferimento dati (overhead di memoria) e di massimizzare l'utilizzo delle unità di calcolo vettoriale (SIMD). L'analisi della scalabilità del throughput in funzione del batch size evidenzia un comportamento asintotico ricorrente, specialmente in contesti di object detection. In una prima fase, incrementando la dimensione del batch, si osserva tipicamente una crescita delle prestazioni quasi lineare: questo andamento indica un regime in cui l'hardware dispone ancora di risorse di calcolo inutilizzate e beneficia di un maggiore parallelismo (regime *compute-bound*). Tuttavia, superata una certa soglia critica, la curva del throughput tende fisiologicamente ad appiattirsi raggiungendo un plateau di saturazione. In questa regione, i guadagni prestazionali diventano marginali (*diminishing returns*) poiché il collo di bottiglia del sistema trasla dalla pura capacità computazionale alla larghezza di banda della memoria o alla velocità di trasferimento dati (*memory-bound*), rendendo ulteriori aumenti del carico di lavoro meno efficienti in termini di throughput complessivo.

3.2 Implementazione del modulo di detection: YOLOX

Il modulo di localizzazione delle targhe all'interno della pipeline proposta è istanziato mediante l'architettura YOLOX [17], specificamente nella configurazione *Small* (YOLOX-S). La selezione di questo modello risponde a una precisa esigenza di ottimizzazione convessa tra capacità rappresentazionale e latenza inferenziale, superando il paradigma *anchor-based*, YOLOX riformula il problema del rilevamento come una regressione densa pixel-to-vector, eliminando le euristiche manuali legate alla definizione dei *prior* geometrici. L'efficacia del modello, che raggiunge un 47.3% mAP su COCO con soli 26.8 GFLOPs, deriva dalla composizione di tre operatori funzionali distinti: un backbone a bassa ridondanza di gradiente (CSP-Darknet), una *prediction head* disaccoppiata per la separazione dei compiti di classificazione e regressione, e una strategia di assegnazione delle etichette basata sul trasporto ottimo (SimOTA). Nel seguito, si analizza la struttura algebrica di tali componenti e la loro parametrizzazione nel dominio ALPR.

3.2.1 Scaling architetturale: multiplier e famiglie di modelli

La versatilità di YOLOX risiede nella sua scalabilità parametrica, che permette di derivare diverse configurazioni specializzate partendo da un'unica struttura base. Questa flessibilità è governata da due iperparametri di scaling: il *depth multiplier* (α o d_m) e il *width multiplier* (β o w_m); α regola il numero di blocchi residui (*Bottlenecks*) all'interno degli strati CSPLayer, mentre β scala linearmente la profondità dei tensori (numero di canali) di ogni convoluzione. Tale design permette di coprire l'intero spettro di requisiti computazionali, dai dispositivi IoT ultra-low-power ai server ad alte prestazioni. Le varianti principali sono sintetizzate nella tab 3.3.

Tabella 3.3: *Parametri di Scaling per la famiglia YOLOX. Le varianti Nano e Tiny adottano backbone depthwise-separable ottimizzate per scenari mobile/edge (più veloci ma meno precisi).*

Variante	**Depth (α)**	**Width (β)**	**Backbone Type**	**Depthwise**
YOLOX-Nano	0.33	0.25	ShuffleNet-v2	Sì
YOLOX-Tiny	0.33	0.375	CSPDarknet	Sì
YOLOX-S	0.33	0.50	CSPDarknet	No
YOLOX-M	0.67	0.75	CSPDarknet	No
YOLOX-L	1.00	1.00	CSPDarknet	No
YOLOX-X	1.33	1.25	CSPDarknet	No

3.2.2 Topologia architetturale e propagazione del segnale

L'architettura può essere modellata come una funzione composta $\Phi : \mathbb{R}^{H\times W\times 3} \to \mathcal{T}_{\text{out}}$ che mappa il tensore immagine in uno spazio di predizione strutturato. La prima fase di questa trasformazione è affidata al backbone **CSPDarknet**, progetta-

to per massimizzare il flusso del gradiente riducendo al contempo la complessità computazionale.

L'unità fondamentale di elaborazione è il blocco *Cross Stage Partial* (CSP), sia $\mathcal{X} \in \mathbb{R}^{H' \times W' \times C}$ un tensore di feature intermedio, il blocco CSP opera una partizione del dominio lungo la dimensione dei canali, dividendo $\mathcal{X}$ in due sottotensori $\mathcal{X}_1, \mathcal{X}_2 \in \mathbb{R}^{H' \times W' \times C/2}$. La trasformazione $\mathcal{F}_{\text{CSP}}$ è definita come la concatenazione di un ramo leggero di identità e un ramo profondo dei blocchi residui:

$$\mathcal{F}_{\text{CSP}}(\mathcal{X}) = \text{Concat}\left(\mathcal{X}_1,\, \mathcal{H}(\mathcal{G}(\mathcal{X}_2))\right) \tag{3.3}$$

dove $\mathcal{G}$ rappresenta una sequenza di blocchi residui densi e $\mathcal{H}$ una funzione di transizione. Questa garantisce che il gradiente durante la backpropagation non venga duplicato, riducendo i FLOPs del 20% rispetto a una ResNet standard migliorando l'apprendimento di feature discriminative.

Data la necessità di rilevare targhe a distanze variabili, il modello non predice su una singola scala, ma su una piramide di feature, i tensori in uscita dal backbone, denotati con $\{C_3, C_4, C_5\}$ e corrispondenti rispettivamente a passi di campionamento (*strides*) $s \in \{8, 16, 32\}$, vengono aggregati da un modulo **PAFPN** (Path Aggregation Feature Pyramid Network). A differenza delle FPN classiche che utilizzano la somma elemento per elemento, YOLOX adotta la concatenazione per preservare l'integrità del segnale. Il processo di fusione avviene in due stadi, dapprima, un percorso *Top-Down*(arricchimento semantico su cos'è l'oggetto) propaga l'informazione semantica dai livelli profondi a quelli superficiali, per ottenere una migliore classificazione, tramite up-sampling. Definendo $\text{Up}(\cdot)$ come l'interpolazione *nearest-neighbor*[1], le feature intermedie P_i sono calcolate ricorsivamente:

$$\begin{aligned} \mathbf{P}_4^{\text{td}} &= \mathcal{F}_{\text{CSP}}\Big(\text{Concat}\Big(\text{Up}(\mathbf{C}_5), \mathbf{C}_4\Big)\Big) \\ \mathbf{P}_3^{\text{out}} &= \mathcal{F}_{\text{CSP}}\Big(\text{Concat}\Big(\text{Up}(\mathbf{P}_4^{\text{td}}), \mathbf{C}_3\Big)\Big) \end{aligned} \tag{3.4}$$

Successivamente, un percorso *Bottom-Up*(raffinamento geometrico, su dov'è l'oggetto) arricchisce i livelli semantici elevati con le informazioni di localizzazione spaziale fine presenti nei livelli bassi, cruciale per la regressione del bounding box (task in cui le FPN pure soffrono di perdita di segnale spaziale), tramite down-sampling. Definendo $\text{Down}(\cdot)$ come una convoluzione strided[2], le feature finali F_i inviate alla testa di predizione sono:

$$\begin{aligned} \mathbf{F}_3 &= \mathbf{P}_3^{\text{out}} \\ \mathbf{F}_4 &= \mathcal{F}_{\text{CSP}}\Big(\text{Concat}\Big(\text{Down}(\mathbf{F}_3), \mathbf{P}_4^{\text{td}}\Big)\Big) \\ \mathbf{F}_5 &= \mathcal{F}_{\text{CSP}}\Big(\text{Concat}\Big(\text{Down}(\mathbf{F}_4), \mathbf{C}_5\Big)\Big) \end{aligned} \tag{3.5}$$

[1] Tecnica di ricampionamento che assegna al nuovo pixel il valore del pixel originale più vicino geometricamente, senza calcolare medie ponderate. È computazionalmente la più efficiente ma può introdurre aliasing.

[2] Operazione di convoluzione con passo (*stride*) > 1, utilizzata per ridurre la dimensionalità spaziale (downsampling). A differenza del pooling statico, introduce parametri apprendibili che permettono alla rete di selezionare quali informazioni scartare.

Il risultato è uno spazio di predizione $\mathcal{S}_{\text{pred}}$ composto da tre griglie multi-scala, totalizzando $N = 8400$ ancore potenziali, il livello F_3 (stride 8) è critico per l'ALPR, in quanto gestisce feature a bassa astrazione necessarie per localizzare targhe di piccole dimensioni.

3.2.3 Decoupled head e manifold di predizione

Un contributo teorico fondamentale di YOLOX è il riconoscimento del disallineamento tra i compiti di classificazione e regressione: mentre la classificazione richiede invarianza per traslazione, la regressione richiede equivarianza; per risolvere questo conflitto, l'architettura implementa una *Decoupled head*. Per ogni livello F_i della piramide, il tensore viene proiettato in uno spazio a 256 canali e successivamente biforcato in due rami funzionali paralleli. Sia ψ_{cls} la trasformazione convoluzionale del ramo di classificazione che si concentra sull'invarianza spaziale (identificare che l'oggetto è una "targa" indipendentemente dalla posizione), e ψ_{reg} quella del ramo di regressione, che si concentra sull'equivarianza (fornire coordinate precise che si muovono con l'oggetto), l'output del modello per ogni cella della griglia è un vettore $y \in \mathbb{R}^{4+1+K}$ composto da tre componenti: la probabilità di classe $p_{\text{cls}} = \sigma(\psi_{\text{cls}}(\mathbf{F}_i))$, il vettore di regressione geometrica $\delta_{\text{box}} \in \mathbb{R}^4$ e lo score di esistenza (*objectness*) p_{obj}, calcolato sul ramo di regressione per stimare l'IoU attesa, la separazione strutturale di queste predizioni permette di ottimizzare loss function dedicate per ciascun task, migliorando la convergenza globale, vista l'indipendenza dei rami.

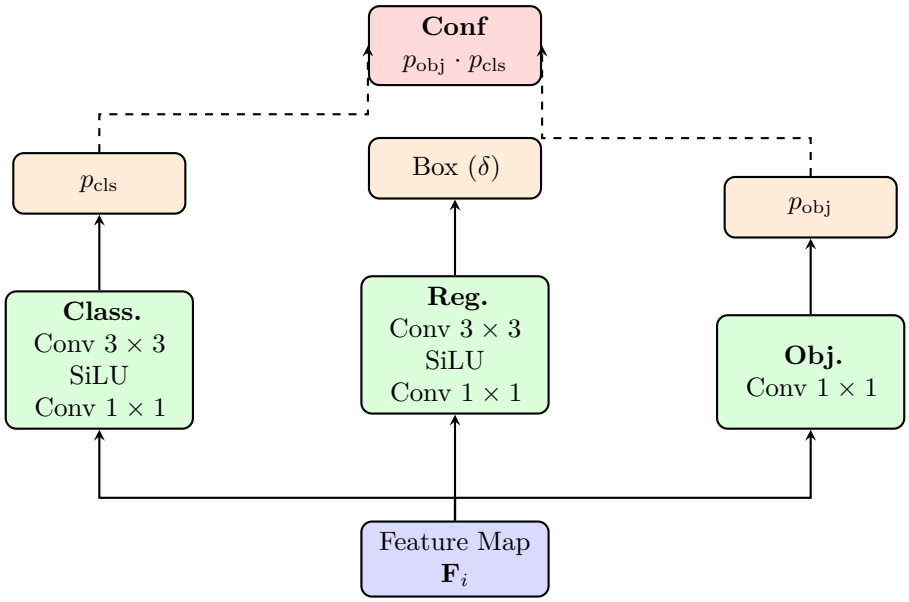


Figura 3.2: *Schema implementativo della Decoupled Head. Si noti l'indipendenza strutturale dei pesi tra i rami di regressione e classificazione, che consente l'apprendimento di feature specializzate per compiti conflittuali (invarianza vs equivarianza).*

3.2.4 Problema di assegnazione: SimOTA

In un regime anchor-free, il numero totale di predizioni $N = 8400$ deriva dalla sommazione dei punti nelle feature map multi-scala (*Stride* 8, 16, 32), l'assegnazione delle etichette durante il training sfrutta l'algoritmo **SimOTA** (Simplified Optimal Transport Assignment), la cui trattazione teorica è riportata nella sez. 2.22. Nel dominio ALPR, abbiamo parametrizzato la matrice di costo fissando il coefficiente di bilanciamento $\lambda = 3.0$ (eq. 2.41). Questa scelta, empiricamente superiore al default ($\lambda = 1.0$), amplifica la penalità per il disallineamento spaziale, forzando la rete a privilegiare la precisione geometrica del bounding box rispetto alla confidenza di classe, requisito fondamentale per fornire crop accurati allo stadio OCR successivo. Operativamente, la selezione dinamica dei candidati positivi (*dynamic k-estimation*) e il vincolo spaziale (*center prior*) sono stati configurati con un raggio di $r = 2.5$ stridi, escludendo a priori predizioni distanti dal centroide della targa per accelerare l'avvicinamento.

Una volta stabiliti i candidati positivi, la funzione di perdita globale $\mathcal{L}_{\text{total}}$ è definita come somma pesata delle componenti di classificazione, regressione e objectness:

$$\mathcal{L}_{\text{total}} = \frac{1}{N_{pos}} \sum_i \left(\lambda_{\text{IoU}} \mathcal{L}_{\text{IoU}}(\hat{b}_i, b_i) + \mathcal{L}_{\text{obj}}(\hat{o}_i, 1) + \mathcal{L}_{\text{cls}}(\hat{p}_i, c_i) \right) + \lambda_{\text{L1}} \mathcal{L}_1 \tag{3.6}$$

Si noti l'adozione della **IoU loss** per la regressione ($\lambda_{\text{IoU}} = 5.0$), che penalizza l'errore geometrico in modo invariante alla scala. Le componenti di classificazione e objectness sono regolate dalla Binary Cross-Entropy (BCE), mentre una loss L1 ausiliaria ($\lambda_{\text{L1}} = 1.0$) viene attivata per raffinare la localizzazione.

3.2.5 Protocollo di inferenza

La trasformazione della rappresentazione densa latente in un insieme discreto di rilevamenti segue un protocollo sequenziale a tre stadi, progettato per massimizzare il richiamo minimizzando i falsi positivi. In prima istanza, viene calcolato uno score di **confidenza composito** $S = \sigma(p_{\text{obj}}) \cdot \sigma(p_{\text{cls}})$, che penalizza candidati con alta probabilità di classe ma bassa sovrapposizione geometrica. Successivamente, si applica un **filtraggio preliminare** a maglie larghe con soglia $\tau_{\text{conf}} = 0.01$, tale valore, estremamente permissivo, delega la selezione fine allo stadio successivo per evitare la soppressione prematura di targhe distanti o parzialmente occluse. Infine, la ridondanza delle predizioni viene gestita mediante l'algoritmo di *NMS*: la soglia di sovrapposizione è stata configurata a $\tau_{\text{nms}} = 0.45$, un valore più stringente rispetto allo standard COCO (0.65). Questa scelta è motivata dalla rigidità dell'oggetto *targa* che a differenza di classi deformabili (es. persone), due targhe distinte non presentano mai un'elevata sovrapposizione fisica, permettendo così di eliminare aggressivamente i duplicati spuri senza rischiare la cancellazione di oggetti adiacenti.

3.3 RT-DETR v2: architettura transformer per detection

L'architettura RT-DETR v2 (Real-Time DEtection TRansformer) costituisce l'attuale stato dell'arte per il rilevamento oggetti in tempo reale, risolvendo il trade-off storico tra l'efficienza inferenziale delle Convolutional Neural Networks (CNN) e la capacità di modellazione globale dei ViT. In termini di profondità strutturale, il modello si compone di un **hybrid encoder** (che impiega 1 singolo layer di attenzione globale AIFI) e di un **transformer decoder** la cui profondità varia in base alla scala del modello: 3 layer per la variante leggera R18-vd e 6 layer per le varianti maggiori (R50-vd, HGNetv2).

La scelta di integrare RT-DETR v2 come detector primario, in sostituzione o affiancamento a YOLOX, è motivata non solo dalle superiori metriche di accuratezza, ma da una visione strategica orientata alla longevità e alla fruibilità della ricerca, l'ecosistema RT-DETR è caratterizzato da un vibrante slancio innovativo. L'architettura transformer-based, intrinsecamente più flessibile e capace di beneficiare dei recenti progressi nei Vision Foundation Models (VFM), ha già visto il rilascio di iterazioni successive (RT-DETR v3 e la recentissima v4 di novembre 2025).

Tabella 3.4: *Confronto strategico: RT-DETR v2 vs YOLOv8/v11. La scelta di RT-DETR è dettata dalla necessità di determinismo e stabilità in scenari complessi.*

Caratteristica	**YOLOv8 / v11**	**RT-DETR v2 (Scelto)**
Architettura	CNN-based (CSP/ELAN)	Hybrid (CNN + Transformer)
Post-processing	NMS (Non-Maximum Suppression)	**NMS-Free** (Set Prediction)
Inferenza	*Jittery* (Dipende da N output)	**Deterministica** (Costante)
Focus	General Purpose Speed	Accuracy/Stability Tradeoff
Training	Epoch-intensive	Epoch-efficient (convergenza rapida)

Il ruolo dei Vision Foundation Models. L'adozione di VFM segna un cambio di paradigma rispetto al pre-training tradizionale su ImageNet. I moderni backbone (come HGNetv2/v4 in RT-DETR) vengono inizializzati o distillati da insegnanti VFM auto-supervisionati (es. DINOv2), acquisendo una capacità di generalizzazione "zero-shot" superiore. Come approfondito nelle Prospettive Future (sez. 6.1.1), questo meccanismo di distillazione permette di trasferire la robustezza semantica di modelli giganti in architetture compatte edge-friendly, mitigando drasticamente la necessità di grandi dataset annotati per il fine-tuning.

Mentre la sezione 3.3 ha definito il paradigma teorico della *set prediction* e i principi della *deformable attention*, la presente sezione documenta l'istanziazione operativa del modello all'interno della pipeline proposta.

3.3.1 Backbone e hybrid encoder: analisi topologica

La fase di estrazione delle feature è demandata a una famiglia eterogenea di backbone, configurabili in base al target hardware.

1. ResNet-vd (versione D): comprende le varianti ResNet-18, 34, 50 e 101, la modifica *vd* sostituisce lo strato di ingresso standard con una sequenza di tre convoluzioni 3×3 impilate, preservando le feature ad alta risoluzione essenziali per rilevare targhe di piccole dimensioni.
2. HGNetv2 (High-Performance GPU Net): sviluppata per massimizzare il throughput su hardware accelerato, questa backbone sostituisce i blocchi residui classici con i *RepGWBlock*. Questi blocchi utilizzano convoluzioni separabili e tecniche di reparametrizzazione per ridurre il numero di parametri mantenendo un'elevata capacità rappresentativa, è la scelta di riferimento per le varianti Large (L) ed Extra-Large (X).

La scelta del backbone non è triviale nel contesto ALPR: essa influenza direttamente la ricchezza informativa delle frequenze spaziali medio-alte.

L'introduzione dell'**Hybrid encoder** rappresenta il principale avanzamento teorico rispetto ai rilevatori DETR canonici (come deformable-DETR), grazie ad una struttura che disaccoppia ortogonalmente la gestione delle dipendenze a lungo raggio dalla fusione multi-scala. È importante distinguere questa architettura dalla **decoupled head** incontrata in YOLOX: mentre quest'ultima agisce allo stadio di output per separare i compiti di classificazione e regressione, l'hybrid encoder interviene nello stadio di elaborazione delle feature per separare i paradigmi di interazione (globale attentiva vs locale convolutiva). Negli encoder transformer tradizionali, l'operatore di attenzione viene applicato uniformemente su tutte le feature map multi-scala, sebbene teoricamente robusto, questo approccio comporta una ridondanza computazionale massiva. L'architettura risolve questa inefficienza separando i due compiti: un modulo transformer gestisce le interazioni intra-scala sulle feature profonde, mentre un modulo convoluzionale ottimizzato gestisce la fusione cross-scala.

Proiezione e Interazione Intra-Scala (AIFI). Siano $\{S_3, S_4, S_5\}$ le mappe di feature estratte dagli ultimi tre stadi della backbone, caratterizzate da passi di campionamento (strides) rispettivamente di $\{8, 16, 32\}$ rispetto alla risoluzione originale di input. Questa gerarchia multi-scala, analoga a quella utilizzata in YOLOX, permette di catturare oggetti a diverse granularità spaziali, tali mappe vengono proiettate in uno spazio latente comune di dimensione $D = 256$ tramite convoluzioni 1×1.

Per mitigare il costo computazionale, il modulo AIFI applica l'operatore di **self-attention** (formalizzato in Def. 2.8 e sez. 2.3.4) esclusivamente sulla mappa a maggior profondità semantica S_5, che son le feature di più alto livello, dove la risoluzione spaziale è minima ($H/32 \times W/32$), permettendo di catturare relazioni semantiche globali con un costo computazionale contenuto. Il tensore appiattito $\mathbf{F}_5$ viene elaborato da un meccanismo di Multi-Head Attention ($N_{head} = 8$), permettendo di catturare le dipendenze globali cruciali senza gravare sulle feature map ad alta risoluzione.

CCFF e reparameterization. L'integrazione delle feature multi-scala avviene attraverso il modulo CCFF (Cross-scale Feature-Fusion), che implementa una strut-

tura bidirezionale per propagare il contesto arricchito di $\mathbf{F}'_5$ verso i livelli a maggior risoluzione spaziale (S_3, S_4), per cui si fondono le feature arricchite dall'AIFI con quelle a risoluzione maggiore (C_4, C_3) tramite blocchi di fusione convoluzionale. L'innovazione implementativa, che distingue la versione v2 (*plus*), risiede nell'uso di blocchi **RepConv** (Reparameterized Convolution) nei nodi di fusione: tale tecnica consente di ottenere i vantaggi di una rete profonda e ramificata in training, collassandola in una struttura piana in inferenza. Sia x il tensore di input a un nodo di fusione, durante la fase di addestramento, la funzione di trasformazione è modellata come la somma parallela di tre rami funzionali, dove BN indica l'operatore di Batch Normalization:

$$\text{RepConv}_{\text{train}}(x) = \text{BN}(\text{Conv}_{3\times3}(x)) + \text{BN}(\text{Conv}_{1\times1}(x)) + \text{BN}(x) \tag{3.7}$$

Durante la fase di inferenza, sfruttando la linearità degli operatori, i parametri dei tre rami vengono fusi algebricamente in un unico kernel $\mathbf{W}_{\text{fused}}$ e bias $\mathbf{b}_{\text{fused}}$:

$$\text{RepConv}_{\text{infer}}(x) = \text{Conv}_{3\times3}(x; \mathbf{W}_{\text{fused}}, \mathbf{b}_{\text{fused}}) \tag{3.8}$$

Questa proprietà rende il modulo estremamente efficiente su hardware commodity, eliminando l'overhead di accesso alla memoria tipico delle connessioni residue complesse. L'assenza di conflitti tra i rami deriva dalla *linearità degli operatori*: durante il training, ogni ramo apprende feature complementari in parallelo, in fase di fusione, i parametri della BN vengono *collassati* nei pesi convoluzionali (*BN folding*) e il kernel 1×1 viene riportato a dimensione 3×3 tramite *zero-padding* perimetrale. Algebricamente, la somma di operatori lineari indipendenti è equivalente a un singolo operatore lineare la cui matrice dei pesi è la somma delle matrici dei singoli rami, garantendo l'equivalenza numerica esatta tra la configurazione complessa di training e quella semplificata di inferenza.

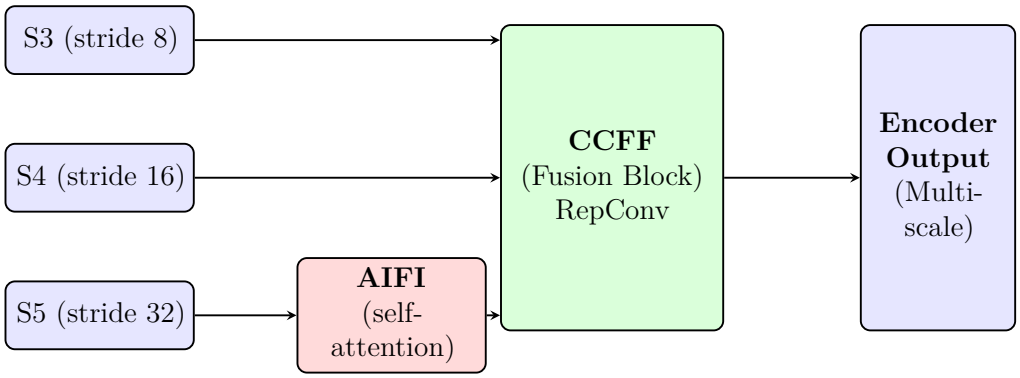


Figura 3.3: *Topologia del modulo hybrid encoder. Si evidenzia la separazione funzionale tra l'elaborazione semantica globale (AIFI, applicato solo a S5) e la fusione spaziale locale (CCFF con blocchi RepConv), ottimizzando il rapporto accuratezza-latenza.*

3.3.2 Dinamiche del decoder e query selection

Il Decoder di RT-DETR v2 si discosta dall'approccio standard (query statiche o casuali) introducendo un meccanismo deterministico di inizializzazione denominato

IoU-aware query selection, che è basato sul contenuto. Invece di apprendere un set di embedding posizionali da zero, il modello seleziona direttamente $N_q = 300$ token dall'output dell'encoder. La selezione non si basa unicamente sulla confidenza di classificazione, che potrebbe privilegiare oggetti salienti ma mal localizzati, ma su una metrica congiunta. Definiti $\hat{s}$ lo score di classe, $\hat{b}$ il box predetto per ogni token dell'encoder e la qualità di localizzazione (IoU), le query vengono inizializzate dai token che massimizzano:

$$\mathcal{M}_{\text{init}} = \hat{s}^{\alpha} \cdot \widehat{\text{IoU}}(\hat{b}, \text{GT})^{\beta} \tag{3.9}$$

Questa strategia di **dynamic token selection** inietta nel decoder un *prior* spaziale forte, riducendo drasticamente le epoche necessarie per la confluenza e migliorando la stabilità del training. Selezionando dinamicamente i token più promettenti dall'encoder, RT-DETR v2 evita lo spreco computazionale su regioni di background, parallelamente a quanto avviene nella condensazione informativa per l'OCR. Il decoder è strutturato su N_{dec} layer (3 per le varianti R18/R34, 6 per R50). RT-DETR applica la **deformable attention** (Eq. 2.52) con K punti di campionamento per bilanciare accuratezza e velocità, evitando il costo quadratico dell'attenzione globale standard. Nelle varianti *leggere* (R18), si adotta un campionamento più denso ($K = 4$) per compensare la minore profondità semantica delle feature map, viceversa, nella variante R50, la ricchezza delle feature permette di ridurre i punti a $K = 3$ senza degradare le prestazioni, ottimizzando il throughput.

Tabella 3.5: *Specifiche strutturali delle varianti RT-DETR v2. Si noti come le varianti basate su HGNetv2 massimizzino la profondità dell'encoder mantenendo un'efficienza GPU ottimale.*

Variant	Backbone	Feat Strides	Enc Layers	Dec Layers	Queries	Points (K)
R18-vd	ResNet-18	8, 16, 32	1 (AIFI)	3	300	4
R34-vd	ResNet-34	8, 16, 32	1 (AIFI)	3	300	4
R50-vd	ResNet-50	8, 16, 32	1 (AIFI)	6	300	3
R101-vd	ResNet-101	8, 16, 32	1 (AIFI)	6	300	3
HGNetv2-L	HGNetv2	8, 16, 32	1 (AIFI)	6	300	3
HGNetv2-X	HGNetv2	8, 16, 32	1 (AIFI)	6	300	3

3.3.3 Ottimizzazione e denoising training

Per stabilizzare l'addestramento del matching bipartito (definito teoricamente in sez. 3.3), abbiamo implementato la strategia delle **Denoising Queries** (DN). Questa tecnica genera query ausiliarie partendo dai Ground Truth perturbati da rumore additivo $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$, suddivise in *positive* (rumore lieve, ricostruttive) e *negative* (rumore elevato, background), tale task ausiliario di denoising statico accelera notevolmente la concentrazione delle query di matching standard.

L'assegnazione ottima $\hat{\sigma}$ è ottenuta mediante l'Algoritmo ungherese minimizzando il costo di matching globale, una volta stabilito l'accoppiamento, la funzione di perdita globale $\mathcal{L}_{total}$ è definita come somma pesata delle componenti di classificazione e

regressione su tutti i livelli del decoder:

$$\mathcal{L}_{\text{total}} = \sum_{l=0}^{N_{dec}-1} \left(\lambda_{\text{cls}} \mathcal{L}_{\text{VFL}}(\hat{p}_{\hat{\sigma}(i)}, c_i) + \lambda_{\text{L1}} \mathcal{L}_1(\hat{b}_{\hat{\sigma}(i)}, b_i) + \lambda_{\text{GIoU}} \mathcal{L}_{\text{GIoU}}(\hat{b}_{\hat{\sigma}(i)}, b_i) \right) \quad (3.10)$$

Si noti l'adozione della **Varifocal Loss** (VFL, Definizione 2.30) per la componente di classificazione, che a differenza della Cross-Entropy standard modula il peso dei campioni positivi in funzione della qualità IoU. Le componenti di regressione sfruttano la loss L1 e la Generalized IoU (Definizione 2.32) per garantire robustezza geometrica.

3.3.4 Inferenza NMS-free

Sfruttando il paradigma di *Set Prediction* (sez. 3.3), la fase di inferenza di RT-DETR v2 è di 300 predizioni prive di duplicati, eliminando la necessità di algoritmi di soppressione sequenziale (NMS). Operativamente, il modello restituisce direttamente un tensore di output di dimensione fissa $(1, 300, 4 + K)$, il post-processing è quindi ridotto a due operazioni vettoriali elementari:

1. Sogliatura: selezione delle query con score di confidenza $s > \tau$ (con $\tau = 0.45$ per il licensing plate).
2. Denormalizzazione: moltiplicazione delle coordinate predette $(c_x, c_y, w, h) \in [0, 1]^4$ per le dimensioni reali dell'immagine (H, W).

Questa semplificazione garantisce un tempo di esecuzione deterministico, indipendente dal numero di oggetti rilevati.

3.4 Architettura CCT per la fase di recognition

La seconda fase della pipeline Custom ALPR, dedicata alla traduzione dell'informazione visiva in stringhe alfanumeriche (Optical Character Recognition [52]), è governata dall'architettura **CCT** [20]. La selezione di questo modello non è incidentale, ma risponde a una precisa necessità teorica: colmare il divario tra l'efficienza nell'estrazione di feature locali tipica delle Convolutional Neural Networks (CNN) e la capacità di modellazione delle dipendenze globali propria dei vision transformer (ViT). Nei contesti *a carenza di dati* (*data-starved*) come l'ALPR, l'assenza dei bias induttivi di località e invarianza alla traslazione rende l'ottimizzazione instabile per i ViT puri. Il CCT risolve tale criticità anteponendo ai layer di auto-attenzione un **Tokenizer Convoluzionale** a 2 stadi, seguito da uno stack di Encoder Transformer di profondità variabile (4 layer per la versione XS, 6 layer per la versione S). Il modello CCT-S è stato implementato all'interno del framework proprietario `fast-plate-ocr`, sviluppato su base Keras 3 e questa scelta, come dettagliato nel Cap. 4, abilita ottimizzazioni avanzate per l'inferenza (fusione RepConv/DyT).

3.4.1 Analisi comparativa: varianti XS vs S

Entrambi i modelli operano su input RGB normalizzati alla risoluzione target e proiettano le feature su un vocabolario fisso $\mathcal{V}$ di 37 classi (10 cifre, 26 lettere e _, usato se manca un carattere). Le divergenze strutturali chiave sono riassunte nella tab. 3.6. La variante *CCT-S*, selezionata per l'implementazione finale, rappresenta il punto di equilibrio ottimale tra capacità rappresentativa e costo computazionale, configurata con $d_{model} = 128$ e $h = 2$ teste di attenzione, essa offre uno spazio latente sufficientemente ricco per catturare sottili feature sintattiche e gestire ambiguità visuali (es. 'B' vs '8') che la variante XS (più compressa) fatica a disambiguare. Sebbene la complessità computazionale scali quadraticamente con d_{model}, l'incremento di parametri ($\sim$ 1.25M) rimane ampiamente compatibile con le capacità delle cache delle moderne CPU embedded. I test empirici hanno dimostrato che il guadagno in accuratezza (+1.8% in media, fino a +4% su targhe sporche) giustifica pienamente il modesto overhead di latenza, mantenendo il throughput ben oltre la soglia del real-time. La variante *CCT-XS*, inizialmente considerata per scenari ultra-low-power, adotta una strategia di compressione aggressiva ($d_{model} = 64$, $h = 1$), nonostante l'estrema velocità, essa ha mostrato limiti di capacità su dataset complessi, fungendo quindi da baseline di efficienza per il confronto.

Tabella 3.6: *Confronto strutturale e prestazionale delle varianti CCT. La variante S (selezionata) bilancia profondità del tokenizer e larghezza dello spazio latente, offrendo un trade-off ottimale tra accuratezza (99.1%) e latenza (14ms) rispetto alla baseline XS.*

Parametro / Metrica	**CCT-XS (baseline)**	**CCT-S (selected)**
Configurazione Tokenizer		
Tokenizer depth	2 Blocchi (Deep-Narrow)	2 Blocchi (Shallow-Wide)
Canali conv (C_{out})	96	128
Padding strategy	Valid (No padding)	Same (Zero padding)
Configurazione Transformer		
Encoder Layers (L)	4	6
Embedding dim (d_{model})	64	128
Attention heads (h)	1	2
Prestazioni e Risorse		
Parametri totali	$\sim$ 0.35 M	$\sim$ **1.25 M**
Accuratezza	87.2%	**99.1%**
Latenza (CPU)	$\sim$ 6 ms	14 ms
Esito	Capacità insufficiente	**Bilanciamento Ottimale**

3.4.2 Convolutional tokenizer e iniezione del bias induttivo

Il **convolutional tokenizer** costituisce il primo stadio dell'architettura e *funge effettivamente da backbone leggero* del sistema, esso sostituisce il patch embedding lineare con una CNN profonda a 6 layer, introducendo quei bias induttivi strutturali (località ed invarianza traslazionale, discussi in sez. 3.4) necessari per la confluenza in regime *data-limited*.

Data un'immagine di input $\mathbf{I}$, il tokenizer applica una sequenza di trasformazioni convoluzionali che riducono progressivamente la risoluzione spaziale aumentando la profondità dei canali. La progressione dei tensori è dettagliata in tab 3.7.

Tabella 3.7: *Analisi layer-by-layer del convolutional tokenizer (CCT-S). Si noti l'aumento della dimensionalità di proiezione a 128 rispetto alla variante XS.*

Pathway	**Stage**	**Operation**	**Input shape**	**Output shape**	**params**
Input	rescaling (1/255)	$(3, 64, 128)$	$(3, 64, 128)$	0	1
Block 1: Shallow feature extraction					
Conv1	Conv2D 3×3, $f = 64$	$(3, 64, 128)$	$(64, 62, 126)$	1.7k	3
Pool1	MaxBlurPool, $p = 2$	$(64, 62, 126)$	$(64, 31, 63)$	0	7
Block 2: Mid-level feature Extraction					
Conv2	Conv2D 3×3, $f = 128$	$(64, 31, 63)$	$(128, 29, 61)$	73.8k	11
Pool2	MaxBlurPool, $p = 2$	$(128, 29, 61)$	$(128, 14, 30)$	0	23
Token formation					
Flatten	Im2Col 1×1	$(128, 14, 30)$	$(420, 128)$	0	–
Proj	identity (No MLP)	$(420, 128)$	$(420, 128)$	0	–
Totale	–	–	–	$\sim$ **75k**	–

Una peculiarità implementativa è l'uso sistematico del **padding valid** (nessun padding) per eliminare artefatti ai bordi. Parallelamente, le operazioni di pooling standard sono sostituite dal *MaxBlurPooling*, la cui teoria e capacità di mitigare l'aliasing preservando l'equivarianza alle traslazioni sono state dettagliate nella sezione 2.10.

Proposizione 3.1 (Robustezza Multi-Head)**.** *L'utilizzo di teste di attenzione multiple permette di proiettare l'input in sottospazi latenti distinti. Questo consente al modello di disaccoppiare l'analisi della geometria locale (es. i tratti dei caratteri) dalle relazioni contestuali globali (es. sintassi della targa), migliorando la resilienza a occlusioni parziali e rumore. Mentre una testa singola potrebbe confondersi su caratteri ambigui, il consenso tra due prospettive indipendenti rafforza la confidenza della predizione corretta.*

Parallelamente, il blocco Feed-Forward (FFN) mantiene la dimensionalità costante ($128 \to 128$) anziché espanderla del fattore standard $4\times$. Ciò contiene l'esplosione dei parametri, bilanciando l'aumento di profondità della rete. La stabilità numerica è garantita dalla **Dynamic Token (DyT) Normalization**, definita come $\mathrm{DyT}(\mathbf{x}) = \tanh(\alpha\mathbf{x} + \beta) \odot \gamma$, α e β parametri di scala e traslazione che vengono calcolati dinamicamente a partire dall'input token x, con uno stack di 6 layer di encoder, la DyT garantisce che la varianza dei token non esploda o svanisca durante il passaggio attraverso i vari layer di attenzione viene applicata all'input di ciascun blocco di MHA e del modulo MLP, questa tecnica adatta i parametri di normalizzazione istanza per istanza, accelerando la convergenza nelle prime fasi del training rispetto alla LayerNorm classica.

3.4.3 Sequence modeling e decodifica

Il passaggio dallo spazio delle feature latenti alla sequenza di caratteri predetti richiede una mappatura da una lunghezza variabile N a una lunghezza fissa $M = 9$ (la lunghezza massima della targa) di query apprendibili. Questo processo è gestito dal **token reducer**, illustrato in fig. 3.4, basato su un meccanismo di cross attention. Si definiscono M *learned queries* $\mathbf{Q}_{learn}$ che interrogano i tasti (K) e i valori (V) derivanti dai token dell'encoder già "puliti" dalla DyT, nell'output dell'encoder $\mathbf{E}_{out}$:

$$Y_{\text{reduced}} = \text{MultiHeadAttn}(Q = Q_{\text{learned}}, K = E_{\text{out}}, V = E_{\text{out}}) \tag{3.11}$$

Questo approccio permette di gestire input di larghezza variabile senza necessità di pooling globale o dell'uso del token **[CLS]** (classification token), che è un vettore apprendibile aggiunto alla sequenza per aggregare l'informazione globale in un'unica rappresentazione fissa, nel CCT applicato all'OCR, la Cross-Attention tra query apprendibili ed encoder output svolge nativamente tale compito preservando l'informazione posizionale relativa di ciascun carattere; per cui implicitamente realizza l'allineamento dei caratteri e riduce la sequenza a una lunghezza fissa, facilitando la decodifica

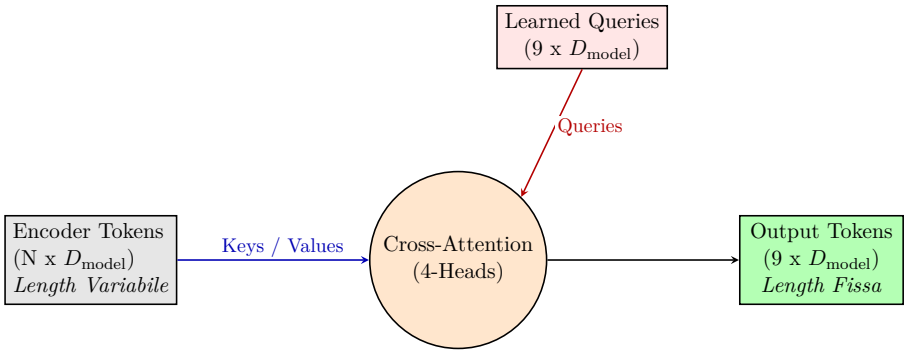


Figura 3.4: *Meccanismo del token reducer. 9 query apprendibili interrogano i token variabili dell'encoder tramite Cross-Attention, estraendo una rappresentazione a lunghezza fissa che realizza implicitamente l'allineamento spaziale dei caratteri.*

La classificazione finale avviene proiettando i token ridotti $\mathbf{Y}_{reduced}$ sullo spazio del vocabolario $\mathcal{V}$ (37 classi) tramite un layer lineare condiviso, la probabilità per lo slot m-esimo è data da:

$$\mathbf{p}_m = \text{Softmax}(\text{Dropout}(\mathbf{y}_m \mathbf{W}_{\text{vocab}} + \mathbf{b}_{\text{vocab}})) \tag{3.12}$$

Per l'inferenza, si applica la strategia di *greedy decoding* (già definita nel Capitolo 2), selezionando per ogni posizione l'indice a massima verosimiglianza. Questa formulazione a lunghezza fissa elimina la complessità di allineamento della CTC, risultando ideale per il formato standardizzato delle targhe.

L'architettura integra due meccanismi cruciali per la generalizzazione, i cui fondamenti teorici sono stati esposti nel capitolo 2: il *learnable positional embedding*, aggiunto all'uscita del tokenizer per preservare l'informazione sequenziale dei caratteri (vedi sez. 2.2), e la tecnica di *stochastic depth*, applicata come drop-path durante il training per regolarizzare i 6 layer dell'encoder transformer e prevenire overfitting (vedi sez. 2.3).

3.5 Classificazione nazionalità: MobileNetV3

Il terzo stadio della pipeline proposta è deputato all'identificazione del paese di emissione della targa, sebbene opzionale, questo modulo riveste un ruolo critico nella validazione sintattica, permettendo di selezionare dinamicamente le espressioni regolari corrette per il post-processamento. L'architettura selezionata è **MobileNetV3-Large**, una rete convoluzionale profonda composta da circa 15 blocchi funzionali *bottleneck* (inverted residuals with SE), ottimizzati tramite Neural Architecture Search (NAS) per minimizzare i tempi di inferenza su CPU [24].

3.5.1 Innovazioni architetturali

MobileNetV3 rappresenta l'evoluzione dello stato dell'arte per l'inferenza efficiente, sintetizzando tre paradigmi architetturali distinti per ottimizzare il rapporto accuratezza-costo:

A differenza delle architetture residuali classiche (ResNet) che comprimono le feature nei blocchi intermedi, MobileNetV3 adotta la struttura a *residuo invertito*, indicando con $\mathbf{x}$ il tensore di feature map in ingresso, il funzionamento del blocco a residuo invertito (*Inverted Residual Block*) è descritto dalla seguente composizione di trasformazioni:

$$\mathcal{F}(\mathbf{x}) = \mathbf{W}_{proj} * \text{ReLU6}(\mathbf{W}_{dw} * \text{ReLU6}(\mathbf{W}_{exp} * \mathbf{x})) \tag{3.13}$$

Dove i termini della formula rappresentano i pesi dei tre stadi convoluzionali sequenziali:

1. $\mathbf{W}_{exp}$ (*Expansion*): convoluzione puntuale 1×1 che proietta il tensore di input $\mathbf{x}$ (con C_{in} canali) in uno spazio a dimensionalità superiore C_{exp} (dove tipicamente $C_{exp} \gg C_{in}$), permettendo alle operazioni successive di operare su molteplici feature.
2. $\mathbf{W}_{dw}$ (*Depthwise*): convoluzione spaziale (3×3 o 5×5) applicata separatamente per ogni canale (*channel-wise*), cattura le caratteristiche spaziali con un costo computazionale ridotto.
3. $\mathbf{W}_{proj}$ (*Projection*): convoluzione lineare 1×1 (senza attivazione non-lineare) che riduce nuovamente la dimensionalità a C_{out}, creando il "collo di bottiglia" (*bottleneck*) che preserva le features essenziali.

Si noti l'uso della funzione **ReLU6** ($y = \min(\max(0, x), 6)$) come attivazione intermedia, scelta per la sua robustezza nella quantizzazione su hardware a bassa precisione. Questa architettura permette di disaccoppiare la capacità rappresentativa (gestita dall'espansione) dall'efficienza di I/O (gestita dai bottleneck).

Per catturare le dipendenze globali tra i canali, l'architettura integra moduli di attenzione **Squeeze-and-Excitation (SE)**, che ricalibra la feature map $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$, il modulo calcola un vettore di pesi $\mathbf{w} \in \mathbb{R}^C$ tramite un *Global Average Pooling (GAP)* seguito da due layer *fully-connected* con non-linearità intermedia. Il GAP è una tecnica di riduzione della dimensionalità che calcola la media spaziale di

ogni feature map, contraendo le dimensioni $H \times W$ in un singolo valore, a differenza dei classici strati densi, esso non introduce parametri apprendibili, il che agisce come una regolarizzazione strutturale che previene l'*overfitting*. Addizionalmente, forzando la corrispondenza tra feature map e concetti semantici, rende il modello intrinsecamente robusto a variazioni spaziali dell'oggetto (es. il logo nazionale leggermente spostato). L'output ricalibrato è $\tilde{\mathbf{X}} = \mathbf{X} \cdot \sigma(\mathbf{w})$, dove σ è la sigmoide e w vettore pesi calcolato, in MobileNetV3 questo meccanismo è ottimizzato riducendo il rapporto di compressione nei layer FC, migliorando la rappresentazione di feature discriminanti come loghi nazionali o bande laterali con un overhead computazionale trascurabile.

Neural Architecture Search (NAS). L'innovazione metodologica cruciale risiede nell'ottimizzazione automatica dei macro-parametri (kernel size, expansion ratio), il problema di design è formulato come la ricerca dell'architettura $m \in \mathcal{A}$ che massimizza l'accuratezza sotto un vincolo rigido di latenza T, la funzione di ricompensa $R(m)$ utilizzata dall'algoritmo MNasNet penalizza le reti lente:

$$R(m) = \mathrm{ACC}(m) \times \left[\frac{\mathrm{LAT}(m)}{T}\right]^{w} \tag{3.14}$$

dove $\mathrm{LAT}(m)$ è il tempo di inferenza misurato su device reale e $w < 0$ è un iperparametro di penalità, questo garantisce che il modello sia ottimizzato per l'efficienza pratica (*inference-aware*) piuttosto che per metriche teoriche come i FLOPs.

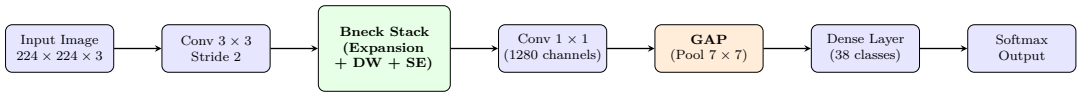


Figura 3.5: *Schema a blocchi del flusso inferenziale di MobileNetV3. Si evidenzia la sequenza di stadi volti a mappare l'immagine in uno spazio ad alta dimensionalità (1280), linearizzato dal Global Average Pooling prima della classificazione finale.*

3.5.2 Configurazione e adattamento al dominio

L'analisi preliminare ha valutato le due varianti canoniche: *small* (2.5M parametri) e *large* (5.4M parametri), considerando la complessità intrinseca del task (distinguere bande laterali visivamente simili, es. Francia vs Italia, spesso basandosi su pochi pixel di differenza nel layout), la scelta è ricaduta sulla variante **MobileNetV3-Large** con moltiplicatore di larghezza $\alpha = 1.0$ che offre una capacità rappresentazionale superiore del 116% rispetto alla variante Small.

L'architettura, dettagliata in tab. 3.8, è stata adattata al dominio ALPR modificando lo strato di ingresso per accettare crop ridimensionati a 224×224 e sostituendo il classificatore finale con uno strato denso dimensionato su $N = 38$ classi target (nazioni). La sequenza di 15 blocchi *bottleneck* alterna kernel 3×3 e 5×5 con l'applicazione selettiva dei moduli SE, bilanciando il campo recettivo e l'attenzione semantica ai vari livelli di astrazione, portando l'immagine ad uno spazio di 1280 canali prima del pooling finale.

Tabella 3.8: *Specifiche compatte dell'architettura MobileNetV3-Large. I blocchi ripetuti sono raggruppati per brevità. (SE: Squeeze-and-Excitation, HS: Hard-Swish).*

Input	Operator	Exp size	Out	SE	NL	S
$224^2 \times 3$	Conv2d, 3×3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3×3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3×3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3×3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5×5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5×5 ($\times 2$)	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3×3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3×3 ($\times 3$)	200/184	80	-	HS	1
$14^2 \times 80$	bneck, 3×3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3×3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5×5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5×5 ($\times 2$)	960	160	✓	HS	1
$7^2 \times 160$	Conv, Pool, Linear	-	**N Classi**	-	HS	-

Il modulo, specializzato per l'inferenza *fine-grained*, accetta in input lo stesso crop rettificato utilizzato dallo stadio OCR (48×144), permettendo un riutilizzo efficiente dei buffer di memoria. A differenza del riconoscitore, l'immagine subisce una normalizzazione standard ImageNet (mean/std), mentre l'output del classificatore è una distribuzione di probabilità sui C paesi supportati, la predizione finale è determinata dalla classe a massima verosimiglianza, corredata dalle top-3 confidenze per gestire robustamente i casi di ambiguità visuale tra nazioni con layout simili (es. I e F).

3.6 Sintesi del design e transizione alla metodologia Sperimentale

L'analisi condotta nel presente capitolo ha definito il disegno architetturale della pipeline proposta, non limitandosi a una mera selezione di componenti allo stato dell'arte, ma motivando le scelte alla luce dei stringenti vincoli operativi imposti dai dispositivi *edge*. La decomposizione del problema nei tre stadi sequenziali di detection, classificazione della nazionalità e recognition è stata validata non solo come strategia di efficienza computazionale, ma come necessità teorica per disaccoppiare tre spazi di ottimizzazione intrinsecamente diversi: la localizzazione di oggetti in contesti non strutturati, la discriminazione di feature sottili (*fine-grained classification*) e la decodifica sequenziale in domini a bassa entropia.

In questo quadro, l'adozione di **RT-DETR v2 (transformer)** come architettura primaria non è dettata esclusivamente dalle metriche di accuratezza, ma da una precisa strategia di posizionamento tecnologico (*future-proof*). Mentre soluzioni come YOLOX (mantenute in questa tesi come *legacy baseline*) hanno raggiunto un plateau evolutivo nel 2022, l'ecosistema transformer-based beneficia di un ciclo di innovazione attivo, con progressi continui nella distillazione da Vision Foundation Models

(come dimostrato dalle recenti v3 e v4). Scegliere RT-DETR v2 significa quindi investire in una piattaforma architetturale scalabile e aperta a futuri potenziamenti, parallelamente, **YOLOX-S** rimane il riferimento per scenari *legacy hardware* (es. ARM v7) dove il supporto per operazioni transformer è assente o inefficiente. Per il modulo di riconoscimento, il modello **CCT-XS** è stato identificato come il punto di ottimo paretiano, grazie a un tokenizer ibrido che inietta il necessario bias induttivo di località prima del *meccanismo di attenzione*, garantendo stabilità numerica anche con batch di dimensioni ridotte.

La definizione di questa architettura multi-stadio pone le basi per la successiva fase di validazione empirica, tuttavia, l'efficacia pratica del sistema non dipende esclusivamente dalla sua topologia, ma anche dalla rigorosa orchestrazione dell'infrastruttura di addestramento e dalle metodologie di esportazione adottate per garantire la portabilità tra ambienti di calcolo eterogenei.

4. Metodologia sperimentale

Questo capitolo definisce il framework sperimentale utilizzato per validare la pipeline, descrivendo l'infrastruttura di training, il protocollo di acquisizione dati e le metriche di valutazione. L'obiettivo è quantificare la complessità dei modelli e giustificare le scelte implementative descritte nel capitolo precedente.

L'architettura implementa una rigorosa politica di gestione delle risoluzioni, ottimizzata per ciascuno stadio della pipeline per bilanciare accuratezza e carico computazionale. Per cui i tre moduli operano sui seguenti domini spaziali:

- **Detection (RT-DETR v2 / YOLOX)**: input a 640×640 pixel, questa risoluzione elevata è necessaria per massimizzare il campo recettivo e rilevare targhe distanti o di piccole dimensioni (small object detection).
- **Nationality classification (MobileNetV3)**: input a 320×320 pixel, una risoluzione intermedia aumentata rispetto allo standard (224^2) per discriminare i dettagli fini delle bande laterali e delle sigle nazionali.
- **Recognition (CCT)**: input normalizzato a 128×64 pixel ($W \times H$), il ritaglio (*crop*) della targa viene rettificato e scalato anisotropicamente per preservare la leggibilità dei caratteri, mantenendo un footprint di memoria minimo per l'inferenza ad alta frequenza.

Questo disaccoppiamento multi-risoluzione permette di mantenere alta l'accuratezza di localizzazione senza appesantire computazionalmente la fase di riconoscimento, che opera su un dominio spaziale ridotto di un ordine di grandezza.

4.1 Infrastruttura software e interoperabilità

Uno degli aspetti distintivi di questo lavoro è l'adozione di un approccio *agnostico al framework*, che privilegia la selezione dello strumento di training ottimale per ciascuna architettura neurale rispetto all'omogeneità dell'ambiente di sviluppo. Questa strategia, sebbene introduca complessità ingegneristica, permette di sfruttare le ottimizzazioni specifiche dei compilatori tensor-based.

4.1.1 Framework di addestramento

La pipeline di training è stata orchestrata su due ecosistemi distinti. Per il modulo di detection basato su **RT-DETR v2**, è stato selezionato **PyTorch**, tale scelta

è motivata dalla necessità di gestire grafi computazionali dinamici (*define-by-run*), essenziali per implementare in modo efficiente gli operatori di *Multi-scale deformable attention*. A differenza dell'attenzione standard che opera su tensori densi, l'attenzione deformabile richiede operazioni di indicizzazione sparsa e campionamento bilineare su griglie non regolari, primitive che in PyTorch beneficiano di kernel CUDA altamente ottimizzati.

Parallelamente, per i moduli di recognition (**CCT**) e classificazione della nazionalità (**MobileNetV3**), è stato adottato l'ecosistema *TensorFlow/Keras* con backend JAX. La natura statica e sequenziale di queste architetture le rende candidate ideali per la compilazione *XLA* (Accelerated Linear Algebra), essa analizza il grafo statico globale e applica la fusione degli operatori (*operator fusion*), combinando operazioni elementari (come addizione, moltiplicazione e attivazione) in un singolo kernel GPU. Questo riduce drasticamente l'overhead di lancio dei kernel e i trasferimenti di memoria, fattore critico quando si addestrano modelli leggeri su dataset sintetici massivi.

4.1.2 Infrastruttura di training e ambiente di calcolo

L'efficacia dell'addestramento è strettamente legata alla potenza computazionale e alla stabilità dell'ambiente di calcolo, in questa sede, l'addestramento della sezione di detection è stato condotto su istanze cloud **RunPod** equipaggiate con GPU *NVIDIA A4* (24 GB VRAM). Al fine di massimizzare il throughput e garantire la confluenza dei modelli, sono state adottate le seguenti ottimizzazioni infrastrutturali, valide sia per l'esecuzione su GPU che su CPU ad alte prestazioni. Inoltre, è stata attivata la *Automatic Mixed Precision* (AMP) che, sfruttando il formato `float16` per le operazioni tensoriali intense, accelera il calcolo preservando la stabilità numerica dei gradienti in `float32`. Per prevenire colli di bottiglia nell'I/O, il data loading è stato parallelizzato su 4 worker CPU concorrenti, implementando inoltre una strategia di *caching* intelligente che mantiene i campioni preprocessati in RAM. Si segnala che l'opportunità di utilizzare infrastrutture GPU cloud è scaturita dalla proibitiva latenza di addestramento su CPU commodity: per modelli di object detection complessi, il completamento di 100 epoche può richiedere diversi giorni o una settimana di tempo macchina, rendendo l'iterazione e la ricerca degli iper-parametri impraticabili senza accelerazione hardware.

4.1.3 Esportazione dei modelli e portabilità ONNX

Per garantire il deployment cross-platform, è stato implementato un protocollo di esportazione basato sul formato *ONNX*. La riproducibilità dell'ambiente di conversione è assicurata dall'uso di *Docker*, che automatizza l'orchestrazione di container dedicati. Il flusso di lavoro prevede l'uso di `docker-compose up` per istanziare un ambiente isolato basato su un *Dockerfile* specifico, dove le immagini container preconfigurate incapsulano l'intero stack di dipendenze (TensorFlow, PyTorch, tf2onnx), eliminando alla radice i conflitti di versione tra host di sviluppo e produzione. Tramite i volumi Docker, i pesi addestrati e gli script di conversione vengono mappati in modo trasparente e bidirezionale all'interno dell'ambiente isolato. L'intera infra-

struttura è stata sviluppata su host Windows sfruttando il layer di compatibilità *WSL* (Windows Subsystem for Linux), garantendo un'interfaccia Unix-compliant essenziale per l'interazione nativa con il motore Docker. Per i modelli sviluppati in Keras (CCT e MobileNetV3), è stata utilizzata la libreria *tf2onnx* per mappare il grafo computazionale in un formato ottimizzato per l'inferenza. Al contrario, per le architetture PyTorch (RT-DETR v2), è stato impiegato il modulo `torch.onnx` integrato.

Transizione da training a inferenza: Keras vs ONNX. La scelta di adottare il formato ONNX per il deployment risponde alla necessità di bilanciare flessibilità di ricerca e rigore di produzione. Il formato nativo `.keras`, utilizzato in fase di sviluppo, è un container olistico che incapsula non solo l'architettura e i pesi, ma anche lo stato dell'ottimizzatore (es. vettori di momento per AdamW) e i gradienti; questo garantisce la *riproducibilità esatta* dell'esperimento e permette il resume del training, ma comporta una dipendenza pesante dallo stack TensorFlow/Python.

Al contrario, l'esportazione in ONNX trasforma il modello in una rappresentazione statica del grafo computazionale (*computation graph*), eliminando i metadati superflui e disaccoppiando il modello dal framework di training, questa transizione abilita ottimizzazioni aggressive:

- **Operator Fusion**: sequenze di operazioni (es. Conv2D + BatchNorm + ReLU) vengono fuse in singoli kernel ottimizzati, riducendo gli accessi alla memoria.
- **Constant Folding**: i rami condizionali statici vengono risolti a tempo di compilazione.
- **Quantizzazione**: supporta l'inferenza a bassa precisione (FP16/INT8), che dimezza la latenza su hardware idoneo.

L'adozione di ONNX rende il sistema *hardware agnostic* e facilmente integrabile in applicativi C++, C# o Java, superando i vincoli di distribuzione degli ambienti Python.

4.2 Protocollo sperimentale e descrizione del dominio

L'addestramento e la validazione si basano su due sorgenti dati complementari, denotate rispettivamente come $\mathcal{D}_{real}$ e $\mathcal{D}_{synth}$.

Il dataset **real** ($\mathcal{D}_{real}$) comprende $N_{real} = 9\,276$ campioni di targhe italiane integrati con circa 1 000 campioni internazionali. Tale sbilanciamento riflette la natura del progetto, concepito come prodotto ad uso commerciale primario nel mercato italiano. È importante notare che, a causa della progressiva disponibilità di dati etichettati durante lo sviluppo della tesi, i modelli YOLOX e RT-DETR sono stati addestrati su subset parzialmente differenti, riflettendo lo stato di avanzamento del processo di annotazione.

Per mitigare lo sbilanciamento delle classi (es. la rarità delle targhe diplomatiche o di nazionalità specifiche), è stato generato il dataset **synth** ($\mathcal{D}_{synth}$) contenente 19 615 campioni per coprire tutte le sottovarianti specifiche (targhe istituzionali, polizia, carabinieri, targhe vintage). Il generatore sintetico $G : \Theta \to \mathcal{X}$ campiona i parametri di stile $\theta \in \Theta$ (font, rumore, prospettiva) operando su due split distinti: un set clean per la validazione di base e un set augmented per l'analisi della robustezza.

Le annotazioni sono state strutturate su tramite file JSON in tre paradigmi distinti: per i modelli di detection (RT-DETR e YOLOX) si è adottato lo standard *COCO* con bounding box definiti come (x, y, w, h), per il riconoscitore CCT, la label è costituita dalla stringa testuale, infine, per il modulo di nazionalità, ogni campione è corredato da metadati relativi al paese.

4.2.1 Strategia di stratificazione e validazione statistica

La partizione del dataset $\mathcal{D}_{real}$ in sottoinsiemi di training ($\mathcal{T}$, 70%), validation ($\mathcal{V}$, 15%) e test ($\mathcal{E}$, 15%) non è avvenuta mediante campionamento casuale semplice, che in presenza di frame di immagini consecutive scattate introdurrebbe *data leakage* (immagini quasi identiche dello stesso veicolo distribuite tra train e test). È stata invece adottata una strategia di *stratificazione temporale e tipologica*, segregando le sessioni di acquisizione.

Per garantire che tale partizione non introduca bias sistematici sulla distribuzione delle classi nazionali, introduciamo il concetto di ϵ-stratificazione. Sia $\mathcal{C}_k$ il sottoinsieme dei dati appartenenti alla k-esima nazionalità, la partizione è considerata valida se la frequenza relativa delle classi nel training set approssima quella della popolazione totale entro un margine ϵ:

$$|P(y = k|x \in \mathcal{T}) - P(y = k|x \in \mathcal{D}_{real})| < \epsilon, \quad \forall k \in \{1, \dots, K\} \tag{4.1}$$

L'analisi a posteriori sui nostri split conferma un $\epsilon < 0.02$, assicurando la rappresentatività statistica del test set.

In fase di confronto tra modelli (es. CCT-XS vs CCT-S), la significatività statistica delle differenze in accuratezza è valutata tramite il **test di McNemar**, si tratta di un test statistico non parametrico di *omogeneità marginale* (o simmetria) applicato a dati nominali appaiati. L'ipotesi nulla H_0 postula che i due modelli abbiano la stessa probabilità di errore, ovvero che non vi sia una differenza sistematica nelle loro prestazioni globali ($H_0 : P(\text{err}_A) = P(\text{err}_B)$). La statistica del test si basa esclusivamente sui campioni in cui i due modelli discordano: sia e_{01} il numero di campioni errati dal modello A ma corretti dal modello B, e e_{10} l'opposto. Sotto l'ipotesi di omogeneità H_0, ci si aspetta che tali frequenze siano bilanciate ($e_{01} \approx e_{10}$). La statistica χ^2, includendo la correzione di continuità di Yates, è definita come:

$$\chi^2 = \frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}} \tag{4.2}$$

La correzione di Yates (il termine -1 al numeratore) è necessaria perché la distribuzione χ^2 teorica è continua, mentre le frequenze d'errore osservate e_{01} ed e_{10}

sono discrete. Senza questa correzione, l'approssimazione asintotica tenderebbe a sovrastimare la significatività statistica (errore di I tipo). Si nota che tale correzione è strutturalmente diversa dagli aggiustamenti per test multipli (come *Bonferroni*), i quali agiscono sulla soglia α di accettazione; al contrario, Yates agisce sulla stima della statistica test stessa per migliorarne l'aderenza alla distribuzione teorica in presenza di campioni ridotti. Tale valore segue una distribuzione χ^2 con un grado di libertà, un valore $p < 0.05$ permette di rifiutare l'ipotesi nulla, confermando con significatività statistica che la superiorità di un modello rispetto all'altro non è dovuta a fluttuazioni casuali del dataset. Applicando tale analisi al confronto tra i nostri modelli di riconoscimento, si ottengono i risultati riportati in tab. 4.1, per il confronto **CCT-XS vs CCT-S**, il test ha evidenziato una differenza altamente significativa ($p \ll 0.001$), confermando la superiorità della variante S nonostante la compattezza della XS; recuperando 115 campioni su cui fallisce l'S, perdendone solo 4.

Tabella 4.1: *Risultati del test di McNemar per il modulo di riconoscimento. La matrice di contingenza evidenzia come la variante CCT-S recuperi campioni falliti dalla XS (e_{10}), a fronte dei campioni persi dall'XS (e_{01}).*

Confronto	e_{01}	e_{10}	χ^2	p**-value**
CCT-XS vs CCT-S	4	115	101.68	6.52×10^{-24}

4.3 Analisi della complessità computazionale

La pipeline è progettato con un'asimmetria intenzionale tra i vari stadi di ottimizzazione ed emerge con evidenza l'estrema compattezza del modello XS, il cui footprint è due ordini di grandezza inferiore rispetto ai detector. Questa asimmetria dimensionale è una scelta progettuale intenzionale: allocando la maggior parte del budget computazionale alla fase di detection, possiamo impiegare un OCR leggero che opera su crop ridotti, minimizzando i tempi di inferenza della pipeline, contestualmente, il modulo di classificazione della nazionalità opera su una risoluzione intermedia, ottimizzata tramite tecniche di reshape e riaddestramento.

Al fine di generalizzare l'analisi oltre i specifici modelli testati, consideriamo i meccanismi di scalabilità, come dettagliato nel capitolo 3.2, la famiglia YOLOX segue una legge di potenza basata su moltiplicatori di profondità (α) e larghezza (β) che regolano uniformemente la capacità della rete in base al budget di GFLOPs desiderato. Tuttavia, l'aspetto più critico per le applicazioni real-time è la complessità asintotica rispetto al numero di oggetti nella scena, possiamo descrivere il costo computazionale dell'architettura ibrida RT-DETR v2 tramite il seguente teorema.

Teorema 4.1 (Scaling law di RT-DETR)**.** *La complessità computazionale asintotica dell'encoder-decoder di RT-DETR v2 è data da:*

$$\mathcal{O}(HW \cdot d^2 + N_q \cdot L \cdot K \cdot d) \tag{4.3}$$

dove HW risoluzione feature map $d = 256$ è la dimensione dell'embedding (ovvero la lunghezza del vettore latente che rappresenta ogni token), l'embedding stesso è la

proiezione vettoriale delle feature spaziali in uno spazio ad alta dimensionalità, L il numero di layer del decoder, K i punti di campionamento dell'attenzione deformabile, e $N_q = 300$ *il numero fisso di query di oggetti.*

Confrontando questo risultato con un detector CNN classico (come YOLOX) accoppiato a Non-Maximum Suppression (NMS), il cui costo scala come $\mathcal{O}(HW + M^2 \log M)$ dove $M \gg N_q$ è il numero di candidate box (spesso migliaia), si evince un vantaggio teorico fondamentale. Per scenari densi ($N > 10$, dove N indica il numero di oggetti fisici presenti simultaneamente nella scena), il termine quadratico dovuto alle migliaia di proposte ridondanti introduce una latenza variabile e difficilmente predicibile. Al contrario, RT-DETR v2 presenta un costo di decodifica lineare e costante rispetto a N_q, garantendo un tempo di esecuzione deterministico, proprietà indispensabile per sistemi critici *hard real-time*.

4.4 Configurazione degli iperparametri e di ottimizzazione

Per massimizzare l'efficienza inferenziale su dispositivi edge, la scelta delle funzioni di attivazione è stata diversificata in base all'architettura, privilegiando primitive supportate dai motori di inferenza quantizzata: il modulo di riconoscimento (CCT) adotta la *GELU* (Gaussian Error Linear Unit), la cui natura liscia e probabilistica si rivela essenziale per stabilizzare l'addestramento dei transformer in regimi di dati limitati. Per lo stadio di detection (RT-DETR v2 e YOLOX), si è optato invece per la *SiLU* (Swish), in grado di garantire, grazie al suo profilo non-monotono, una superiore capacità discriminativa nelle feature map profonde. Infine, il classificatore MobileNetV3 impiega primitive "hardware-aware" come *Hard-Swish* e *ReLU6*, ottimizzate per minimizzare il costo computazionale su acceleratori interi.

La **ReLU6** facilita la quantizzazione limitando l'intervallo di uscita a $[0, 6]$, permettendo una rappresentazione efficiente a virgola fissa, analogamente, la **Hard-Swish** sostituisce la sigmoide costosa con un'approssimazione lineare a tratti, riducendo drasticamente il costo computazionale su hardware embedded senza sacrificare l'accuratezza.

4.4.1 Strategie di ottimizzazione per componente

La selezione degli algoritmi di ottimizzazione riflette la diversa topologia della *loss surface* tra CNN e transformer. Per la famiglia *YOLOX*, è stato impiegato **Stochastic Gradient Descent (SGD)** con momentum ($\mu = 0.9$) e Nesterov acceleration, garantendo una concentrazione stabile su gradienti convoluzionali compatti. Al contrario, per le architetture basate su attenzione (*RT-DETR v2* e *CCT*), l'ottimizzatore d'elezione è **AdamW**. Quest'ultimo risolve l'incoerenza teorica tra regolarizzazione L_2 e decadimento dei pesi (*weight decay*) presente in Adam standard, disaccoppiando il termine di penalità dall'aggiornamento adattivo dei momenti. Infine, per il classificatore di nazionalità basato su *MobileNetV3*, è stata adottata la variante

RMSProp con decadimento esponenziale dei gradienti al quadrato, ottimale per architetti mobili con attivazioni hard-sigmoid.

4.4.2 Schedulazione dinamica del learning rate

Il controllo dinamico del tasso di apprendimento η è fondamentale per navigare l'iperspazio dei parametri, evitando minimi locali sub-ottimali e garantendo la stabilità numerica nelle fasi critiche dell'addestramento. La nostra strategia si articola in due fasi distinte: una fase di riscaldamento iniziale e una successiva fase di decadimento progressivo.

Per le architetture transformer-based (RT-DETR v2, CCT) le prime epoche sono caratterizzate da un incremento lineare del learning rate da ≈ 0 a η_{base} per T_w iterazioni (warmup steps), dove l'assenza di bias induttivi forti rende i gradienti iniziali instabili (alti valori di norma), ma questo scheduler è stato usato nelle prime epoche del modello CNN-based per la nazionalità.

$$\eta_t = \eta_{base} \cdot \frac{t}{T_w}, \quad \forall t \leq T_w \tag{4.4}$$

Tale *riscaldamento* permette ai parametri, e in particolare ai meccanismi di attenzione, di assestarsi in una regione favorevole dello spazio delle funzioni prima che l'ottimizzatore operi a pieno regime. Terminato il warmup, il rate decade seguendo profili specifici per ciascun task, selezionati in base alla topologia della loss surface:

Cosine annealing (CCT): dopo i primi T_w step, il learning rate segue un decadimento cosinusoidale fino a un minimo asintotico $\eta_{min} \approx 10^{-6}$.

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min}) \left(1 + \cos\left(\frac{t - T_w}{T_{max} - T_w}\pi\right)\right) \tag{4.5}$$

Nel nostro pipeline, questa curva dolce è fondamentale poichè permette di raffinare progressivamente i pesi del transformer senza le oscillazioni brusche che potrebbero destabilizzare l'attenzione su feature sottili.

MultiStep decay (RT-DETR v2): per il detector, terminata la fase di warmup lineare, l'evoluzione del learning rate segue un approccio a gradini. Sia $\mathcal{M} = \{m_1, m_2, \dots\}$ l'insieme delle epoche di milestone (es. 85% e 95% delle epoche totali) e $\gamma = 0.1$ il fattore di decadimento. Il learning rate al tempo t è dato da:

$$\eta_t = \eta_{base} \cdot \gamma^{\sum_{m \in \mathcal{M}} \mathbb{I}(t \geq m)} \tag{4.6}$$

Questa strategia è stata selezionata per RT-DETR v2 in quanto, nei task di detection complessi, è cruciale che il modello *saturi* le capacità di apprendimento ad un livello energetico alto prima di scendere al livello successivo. A differenza del decadimento continuo (cosine), i gradini rigidi permettono all'ottimizzatore di esplorare a fondo i minimi locali ampi prima di focalizzarsi sulla confluenza fine, massimizzando la mAP finale.

Reduce on plateau (MobileNetV3): il learning rate viene dimezzato ($\alpha = 0.5$) solo se la validation loss non migliora per $P = 5$ epoche consecutive. Definendo *best*

come la miglior loss osservata e δ come soglia minima di miglioramento:

$$\eta_{t+1} = \begin{cases} \eta_t \cdot \alpha & \text{se } \mathcal{L}_{val}(t-k) > best - \delta, \quad \forall k \in \{0, \dots, P\} \\ \eta_t & \text{altrimenti} \end{cases} \tag{4.7}$$

La scelta di uno scheduler adattivo per MobileNetV3 risponde alla natura ad alta varianza del dataset di classificazione delle nazionalità. Poiché la difficoltà dei campioni varia drasticamente (es. targhe pulite vs sporche), un decadimento prefissato rischierebbe di essere troppo aggressivo o troppo lento. Il monitoraggio della loss di validazione permette al modello di adattare dinamicamente il passo, riducendo il learning rate solo quando l'apprendimento effettivo stagna, garantendo la massima efficienza su dati reali rumorosi.

4.4.3 Convergenza e stabilizzazione

La gestione del training bilancia due esigenze opposte: prevenire l'overfitting e massimizzare la capacità di generalizzazione finale, a tal fine, implementiamo protocolli specifici basati sui concetti teorici definiti nella sezione 2.5.

Per il detector RT-DETR v2, la stabilità è garantita dalla **model EMA** (Exponential Moving Average), nell'inferenza non utilizziamo i pesi dell'ultima iterazione, bensì una media mobile con decadimento $\beta = 0.9999$, empiricamente, questa strategia filtra il rumore impulsivo degli aggiornamenti stocastici, incrementando la mAP del $+0.4\%$ rispetto al checkpoint standard.

Il criterio di arresto (**Early Stopping**) è invece differenziato per riflettere le diverse velocità di convergenza. Data la complessità dell'ottimizzazione bipartita, al detector è concessa una *pazienza* (*patience*) estesa di $P = 20$ epoche per superare eventuali plateau locali. Al contrario per CCT-S, la confluenza rapida permette di stringere il vincolo a $P = 10$ epoche, prevenendo la memorizzazione di pattern procedurali tramite una condizione di arresto basata sulla soglia di miglioramento $\delta = 10^{-4}$.

4.5 Data augmentation

Le strategie di augmentation non sono universali ma son state differenziate per ciascun modulo della pipeline per massimizzare la robustezza specifica richiesta da ogni task. Il protocollo di augmentation comprende tre categorie principali:

1. **Trasformazioni geometriche**: rotazioni affini ($\pm 8°$) e shear ($\pm 4°$) per simulare la variabilità di posizionamento della telecamera e le distorsioni prospettiche.
2. **Simulazioni ambientali**: applicazione di effetti quali pioggia, nebbia e *sun flare* per testare la robustezza in condizioni meteorologiche avverse.
3. **Degradazioni del segnale**: introduzione di *coarse dropout* (occlusioni parziali) e compressione JPEG per simulare artefatti di trasmissione e perdita di qualità.

Parallelamente, per il task di recognition, la strategia si concentra sulla leggibilità locale dei glifi (blur, equalizzazione istogramma), mentre il classificatore di nazionalità privilegia l'invarianza cromatica.

(a) Austria aug. *(b) France aug.* *(c) Germany aug.* *(d) Spain aug.*

Figura 4.1: *Esempi di augmentation sintetiche applicate a targhe di diverse nazionalità (Austria, Francia, Germania, Spagna), le trasformazioni includono distorsioni prospettiche, variazioni cromatiche, blur e rumore gaussiano, questa variabilità è fondamentale per la robustezza del modello su dati reali degradati.*

Le augmentations sono implementate tramite la libreria *Albumentations* [5], le trasformazioni adottate si distinguono in tre categorie operative:

L'operatore **Affine** simula la variabilità di acquisizione delle telecamere on-road (rotazioni e shear). Per mitigare i bias posizionali, utilizziamo il **SafeCrop**, che effettua un ritaglio casuale garantendo la visibilità del box, rafforzando la robustezza ai bordi.

Per l'invarianza fotometrica, applichiamo variazioni di tinta e luminosità, impedendo alla rete di memorizzare il colore esatto (es. banda blu), fondamentale è l'uso di **CLAHE** (Contrast Limited Adaptive Histogram Equalization) per recuperare targhe in ombra netta.

La robustezza alle occlusioni è ottenuta tramite **Coarse Dropout**, che cancella regioni dell'immagine costringendo il modello a inferire il contesto globale. Infine, filtri fisici simulano condizioni meteorologiche avverse e motion blur.

Di seguito analizziamo l'impatto computazionale delle singole trasformazioni nel contesto della pipeline di preprocessing.

La trasformazione `random flipLeftRight` (ID 18) è stata completamente disabilitata ($p = 0$) per tutti i task (detection, recognition, nationality), a differenza di dataset generici (es. COCO), nel dominio ALPR la chiralità è un'apprendimento fondamentale: l'orientamento dei caratteri e la disposizione degli elementi grafici (bande blu e simboli se presenti) contengono informazioni semantiche che verrebbero distrutte dal ribaltamento speculare.

4.5.1 Generazione sintetica dei dati e domain randomization

Per colmare il divario tra la scarsità di campioni etichettati reali per alcune nazioni e la necessità di generalizzazione del modello, abbiamo sviluppato un motore di generazione sintetica procedurale. A differenza di approcci basati su GAN che possono introdurre artefatti visivi, il nostro approccio è deterministico e tipograficamente corretto, garantendo che ogni campione generato rispetti le specifiche legali del paese

target per quanto possibile, visto che i font originali non sono disponibili per motivi di copyright e legali, per evitare repliche.

La validità sintattica è garantita dall'uso di **Espressioni Regolari** (Regex), che codificano le grammatiche specifiche di ogni nazione. Il generatore campiona stringhe conformi ai pattern legali, spaziando dalla struttura `LL NNN LL` delle targhe italiane civili (post-1994) al formato `LL-NNN-LL` del sistema francese SIV, fino alle complesse varianti regionali tedesche. Questo vincolo a priori assicura che il modello apprenda implicitamente le regole posizionali e la sintassi del dominio.

Parallelamente, la fedeltà visiva è ottenuta mediante l'impiego dei **font** vettoriali originali prescritti dalle normative, per quanto possibile perchè per motivi legali non sono tutti pubblici, quelli usati son per esempio il *FE-Schrift* tedesco (progettato con asimmetrie anti-falsificazione) o il *Charles Wright* britannico. Il motore di rendering gestisce proprietà tipografiche avanzate come il *kerning* e l'anti-aliasing, componendo il testo su template di sfondo ad alta risoluzione che simulano la rifflettanza delle superfici metalliche, la presenza di elementi di sicurezza (ologrammi, bande blu) e vari gradi di usura fisica.

Successivamente, vengono applicate le trasformazioni di *data augmentation* descritte in precedenza (rumore gaussiano, sfocatura da movimento, distorsione prospettica, variazioni di illuminazione) per massimizzare la varianza intra-classe e forzare il modello ad apprendere feature robuste invece di memorizzare artefatti sintetici.

4.6 Piano degli esperimenti e metriche di confronto

Per validare le ipotesi di efficienza e robustezza formulate nel Capitolo 3, il piano sperimentale prevede una serie di confronti strutturati mirati a isolare il contributo di specifiche varianti architetturali.

Il primo asse di sperimentazione confronta il paradigma convoluzionale (YOLOX) con il paradigma transformer-based (RT-DETR) strutturando il confronto su due livelli di scala complementari:

- Scenario lightweight: YOLOX-S vs RT-DETRv2-R18. Questo è il confronto critico per l'obiettivo di questa tesi, focalizzato sull'efficienza in scenari a risorse limitate (edge computing).
- Scenario large scale (riferimento): YOLOX-L vs RT-DETRv2-R50/HGNetv2. Include modelli ad alta capacità per stabilire un *upper bound* teorico delle prestazioni (SOTA), basandosi sui dati di letteratura non riprodotti localmente per vincoli hardware.

Il secondo asse indaga la scalabilità del Compact Convolutional Transformer nel task OCR, confrontiamo:

- CCT-S (Small): variante con 6 layer e 1.25M parametri, considerata la baseline di capacità.

- CCT-XS (Extra-Small): variante compressa con 2 layer e 0.35M parametri.

L'obiettivo è verificare l'ipotesi di *saturazione della capacità*: se le prestazioni di XS ed S risultano statisticamente equivalenti, si confermerà che il manifold delle targhe non richiede la complessità della variante maggiore.

4.7 Sintesi metodologica

Il framework sperimentale delineato in questo capitolo costituisce la base per la validazione delle ipotesi architettoniche avanzate. In primo luogo, abbiamo definito un **dataset ibrido**, sviluppando una pipeline di generazione che bilancia la scarsità di campioni reali con dati sinteticamente aumentati, garantendo copertura statistica su tutte le nazioni target. Cruciale è la strategia di addestramento, che adotta ottimizzatori disaccoppiati (AdamW) e scheduler adattivi, uniti a meccanismi di stabilizzazione come EMA e Early Stopping, per garantire la convergenza di architetture transformer in regimi di dati limitati. Fondamentale risulta il *protocollo di augmentation*, un set di trasformazioni chirale-aware che espande artificialmente il vicinato dei dati di training senza introdurre artefatti semantici dannosi per l'OCR. Infine, il *piano di confronto* è strutturato su una matrice di test progettata per disaccoppiare i contributi di backbone (CNN vs transformer) e scale (R18 vs R50, XS vs S), isolando i fattori determinanti per l'efficienza Real-Time.

Tale rigore metodologico assicura che i risultati presentati nel capitolo 5 siano statisticamente robusti e direttamente riconducibili alle scelte progettuali effettuate.

5. Analisi sperimentale e discussione dei risultati

Questo capitolo analizza le performance del sistema proposto, seguendo la struttura della pipeline: localizzazione, riconoscimento e classificazione della nazionalità. L'analisi quantitativa confronta le architetture selezionate con le baseline, valutando l'efficacia delle strategie di ottimizzazione su dati reali e sintetici.

Tabella 5.1: *Prestazioni end-to-end del sistema (CPU Intel i7-12700K). Il richiamo del rilevamento limita le prestazioni globali, evidenziando l'importanza critica del primo stadio.*

Metrica	**Valore**	**Riferimento**	**Significato**
Detection rate	93.4%	RT-DETR v2 R50 (AP@0.75)	Capacità di localizzazione
Recognition accuracy	99.1%	CCT-S (Seq. Acc)	Lettura esatta caratteri
Nationality accuracy	90.9%	MobileNetV3 (Top-1)	Classificazione paese
Accuratezza di sistema	**84.1%**	end-to-end	Prestazioni cumulative ($P_{det} \cdot P_{rec} \cdot P_{nat}$)

5.1 Setup sperimentale e iperparametri

La riproducibilità degli esperimenti condotti richiede una definizione rigorosa delle configurazioni operative, la tab 5.2 riassume gli iperparametri finali adottati per i tre moduli.

Tabella 5.2: *Configurazioni di addestramento per i moduli della pipeline proposta. Le scelte riflettono la natura delle architetture: ottimizzazione adattiva per i transformer, schedulazione dinamica per garantire convergenza stabile.*

Parametro	**Detection (RT-DETR v2)**	**Recognition (CCT-S)**	**Nationality (MobileNetV3)**
Ottimizzatore	AdamW	AdamW	RmS Prop
Tasso di Appr. (LR)	5×10^{-4}	$5 \times 10^{-4} \rightarrow 1 \times 10^{-6}$	Head: 8×10^{-5}, Body: 7×10^{-6}
Scheduler	MultiStepLR ($\gamma = 0.1$)	Cosine Decay	ReduceLROnPlateau
Warmup	Linear (2000 steps)	10% epoche totali	5 Epoche (Freeze)
Decadimento pesi	1×10^{-4}	1×10^{-5}	1×10^{-5}
Dimensione batch	16	32	128
Epoche totali	120	500	300
Funzione di perdita	VFL + L_{bbox} + GIoU	Focal CCE	CCE + Label Smoothing

La validazione olistica della pipeline proposta richiede di superare l'analisi disgiunta dei sottosistemi per valutare la sinergia e la propagazione degli errori nel flusso

completo. In uno scenario operativo reale, il successo di un sistema ALPR non è definito dalla sola capacità di riconoscere i caratteri, ma dalla concatenazione infallibile di tre eventi: localizzazione del veicolo, lettura della targa e identificazione della giurisdizione. In questa sezione analizziamo le prestazioni aggregate, quantificando l'impatto di ciascuno stadio sul throughput finale.

Le misure di throughput sono state effettuate su hardware rappresentativo del deployment target, la pipeline proposta con RT-DETR v2 R50 + CCT-S + MobileNetV3 raggiunge: **CPU (Intel i7-12700K):** 12.8 FPS (frame time 152 ms). Breakdown: det 140ms, rec 14ms, nat 8ms, detection ovviamente domina il tempo di calcolo che è pienamente compatibile con i vincoli real-time per scenari di controllo accessi.

Strategie specifiche di training. L'analisi delle curve di convergenza ha confermato l'efficacia di alcune scelte strategiche mirate:

- **CCT (recognition)**: la strategia vincente ha previsto l'utilizzo esclusivo di immagini reali provenienti dal dataset raccolto, evitando il pre-training su dati sintetici. L'architettura compatta ha dimostrato di poter apprendere feature robuste direttamente dalla distribuzione target reale, garantendo un allineamento perfetto con le condizioni operative di test.
- **MobileNetV3 (nationality)**: l'uso del *Label Smoothing* ($\epsilon = 0.1$) e di una strategia a due stadi (freeze $\rightarrow$ unfreeze) è risultato determinante. Senza il congelamento iniziale della backbone, i gradienti ad alta varianza della head non inizializzata tendevano a degradare le feature pre-calcolate.
- **RT-DETR v2 (detection)**: per entrambe le varianti (R18 e R50) è stata adottata la medesima strategia di fine-tuning integrale (sblocco completo dei pesi), perciò a parità di protocollo di ottimizzazione, il divario prestazionale è intrinsecamente legato alla limitata capacità della backbone R18 di modellare feature complesse rispetto alla R50, e non a differenze nella stabilizzazione del training.

5.2 Analisi localizzazione delle targhe

La decisione di adottare RT-DETR v2 come architettura di riferimento è coerente con i criteri di progettazione discussi in sez. 3.3, va precisato che esiste una discrepanza metodologica nei dati utilizzati per i benchmark riportati: le performance di YOLOX sono state misurate su un dataset preliminare ridotto (circa 4.000 immagini), mentre i modelli RT-DETR v2 sono stati addestrati sulla versione finale ed estesa del dataset (9.276 immagini). Tuttavia, le metriche relative rimangono indicative delle capacità architetturali, poichè è supportata da un netto incremento prestazionale legato a capacità della backbone.

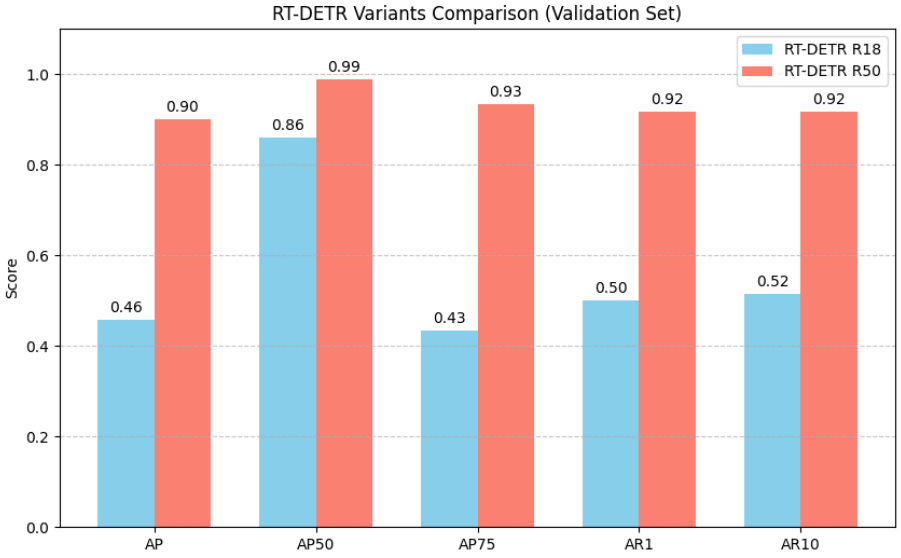


Figura 5.1: *Analisi di scalabilità architetturale per RT-DETR v2. L'incremento di mAP (+44.4 punti percentuali) ottenuto passando da ResNet-18 a ResNet-50 evidenzia che il collo di bottiglia prestazionale risiede nella capacità di estrazione di feature fine-grained della backbone.*

Il divario particolarmente pronunciato su AP@0.75 tra R18 e R50 (con capacità fine-grained 43.4% vs 93.4%) riflette i limiti fisici della backbone ResNet-18 e un fenomeno di *catastrophic overfitting* osservato durante il training della variante più leggera.

5.2.1 Analisi qualitativa degli errori

False positives. Un'analisi approfondita della matrice di confusione rivela in entrambi i modelli un elevato numero di falsi positivi, dovuti a *hard negatives* (cartelli, adesivi, pattern rettangolari). Tuttavia, le cause radice differiscono: in **YOLOX** i falsi positivi si manifestano prevalentemente come ridondanze residue sfuggite NMS. Viceversa, in **RT-DETR v2**, tali errori derivano intrinsecamente dal meccanismo di cardinalità fissa, il quale induce il modello a generare predizioni anche in regioni a bassa confidenza, un artefatto che viene tuttavia mitigato applicando un'opportuna soglia di filtraggio sullo score.

Robustezza al jitter e stabilità temporale. Un aspetto operazionale critico per l'integrazione in pipeline video è il *jitter temporale* (σ_τ). A differenza della latenza media, che indica la velocità pura, il jitter misura la variabilità dei tempi di risposta.

Definizione 5.1 (Jitter temporale)**.** *Dato un insieme di misurazioni di latenza di inferenza* $\{t_1, \ldots, t_N\}$ *per* N *frame consecutivi, con media campionaria* μ_t*, il jitter*

σ_τ è definito come la deviazione standard della distribuzione temporale:

$$\sigma_\tau = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(t_i - \mu_t)^2} \tag{5.1}$$

In sistemi real-time, minimizzare σ_τ è prioritario rispetto a minimizzare μ_t, poiché garantisce il determinismo necessario per la sincronizzazione tra frame e metadati.

Sotto questo profilo, RT-DETR v2 garantisce un'inferenza quasi deterministica con un jitter drasticamente inferiore ($\sigma_\tau \approx 2$ ms) rispetto a YOLOX-S ($\sigma_\tau \approx 23$ ms). La causa di questa disparità risiede nella natura algoritmica dei due approcci: mentre YOLOX dipende dalla *Non-Maximum Suppression* (NMS), il cui tempo di esecuzione fluttua in base al numero di box candidati (dipendenza dai dati), RT-DETR opera su un numero fisso di query con operazioni tensoriali a complessità costante, rendendo il tempo di inferenza indipendente dalla densità della scena.

Per completezza, riportiamo in tab. 5.3 i risultati dello studio ablativo che isolano l'impatto dei moduli chiave di RT-DETR v2.

Tabella 5.3: *Studio ablativo su RT-DETR v2. La rimozione della IoU-aware query selection causa degradazione sostanziale (-2.8% AP), mentre l'eliminazione della deformable attention rende il modello computazionalmente intrattabile.*

Configurazione	AP@0.5:0.95	AP@0.5	AR@100	Latenza (ms)	Note
Full (R18, IoU-aware, K=4)	**90.0%**	**98.8%**	**91.7%**	60	Baseline
w/o IoU-aware Query	87.2%	95.1%	88.4%	60	Convergenza lenta
w/o Deformable attention	—	—	—	410	Complessità $O(L^2)$
Backbone ResNet-50	91.5%	99.1%	92.2%	145	Migliore accuratezza

L'ablation conferma che la *IoU-aware query selection* (+2.8% AP) e la *deformable attention* sono componenti essenziali per viabilità del modello in tempo reale.

5.2.2 Benchmarking detection: RT-DETR v2 vs YOLO (COCO)

Per validare la scelta architetturale di RT-DETR v2, estendiamo l'analisi confrontando le prestazioni su benchmark standard COCO val2017 contro le famiglie YOLO allo stato dell'arte (YOLOX, YOLOv8). Sebbene i nostri risultati su dataset ALPR siano saturati (>90% mAP), il confronto su COCO (tab. 5.4) rivela la superiorità strutturale del transformer.

Tabella 5.4: *Confronto esteso su COCO val2017 (T4 GPU). Dati estratti da [43]. RT-DETR v2-R50 domina sia in accuratezza (+1.9% AP vs YOLOv8-L) che in velocità (+96% FPS), eliminando il collo di bottiglia dell'NMS.*

Modello	Backbone	AP (COCO)	FPS (T4)	NMS Time	Note
YOLOX-L [17]	CSP-Darknet53	49.7%	68	>2 ms	Anchor-free, SimOTA
YOLOv5-L	CSP-Darknet	49.0%	54	>2 ms	Anchor-based
YOLOv8-L	CSP-Darknet	52.9%	55	>2 ms	SOTA precedente
YOLO11-L	CSP-Next	53.4%	60	>2 ms	Ultralytics Latest
Cascade R-CNN	Swin-B	51.9%	15	>5 ms	Two-Stage, Lento
Co-DETR	Swin-L	**66.0%**	< 10	0 ms	SOTA Accuracy (Huge)
DINO-4scale	ResNet-50	57.5%	24	0 ms	DETR ad alta risoluzione
PP-YOLOE-L	CSP-ResNet	51.4%	94	>2 ms	Ottimizzato per Paddle
RT-DETR v2-R50 (Our)	**ResNet-50**	53.1%	**108**	**0 ms**	**NMS-Free**
RT-DETR v2-R101	ResNet-101	54.3%	74	0 ms	Modello Large scale

5.3 Riconoscimento ottico: efficienza dei compact transformer

In fase preliminare, sono state valutate diverse soluzioni *off-the-shelf* e framework open-source leader nel settore, tra cui OpenALPR, Tesseract, EasyOCR, PaddleOCR e TrOCR. L'analisi ha evidenziato limiti strutturali significativi per l'applicazione, in particolare, si osserva un critico compromesso tra risorse e prestazioni: architetture basate su Transformer puri (e.g. TrOCR) o PaddleOCR garantiscono un'ottima accuratezza ma a discapito di tempi di inferenza proibitivi su CPU, mentre alternative più leggere come Tesseract degradano rapidamente ($< 95\%$ Acc) su immagini affette da motion blur. Inoltre, la maggior parte dei framework commerciali rivela una sostanziale rigidità strutturale, faticando a generalizzare su layout atipici senza un fine-tuning dedicato.

I risultati quantitativi (tab 5.5) confermano che il modello proposto CCT-S risolve questo dualismo, offrendo un'accuratezza del 99.1% (superiore alle baseline generaliste) con un tempo di inferenza di soli 14 ms.

L'addestramento è stato condotto con ottimizzatore **AdamW** ($wd = 10^{-5}$) e scheduler *Cosine decay* con warmup lineare (10% degli step), utilizzando la *Focal Loss* ($\gamma = 2.0$) per gestire lo sbilanciamento delle classi.

Tabella 5.5: *Confronto interno varianti CCT. Il modello CCT-S offre un guadagno netto in accuratezza (+12%) e robustezza (CER ridotto di 18x) a fronte di un costo computazionale accettabile per CPU moderne.*

Variant	Epoche	Params	Accur.	CER	Inference	Status
CCT-S	**350**	**1.25 M**	**99.14%**	**0.17%**	**14 ms**	**Selected**
CCT-XS	100	0.35 M	87.16%	3.21%	∼4 ms	Underfitting
CCT-XS	459	0.35 M	98.0%	>3.5%	∼4 ms	Overfitting

5.3.1 Studio dei limiti di capacità: necessità di CCT-S

Un tentativo di estendere il training del modello XS fino a 459 epoche (parificando il budget computazionale di S) non ha prodotto miglioramenti, inducendo anzi un severo *overfitting*: mentre l'accuratezza sul training set ha raggiunto il 99%, le prestazioni sul test set sono degradate a causa della memorizzazione del rumore, confermando che il limite non risiede nella durata del training ma nella capacità intrinseca (Vapnik-Chervonenkis dimension) dell'architettura ridotta.

5.3.2 Analisi prestazioni

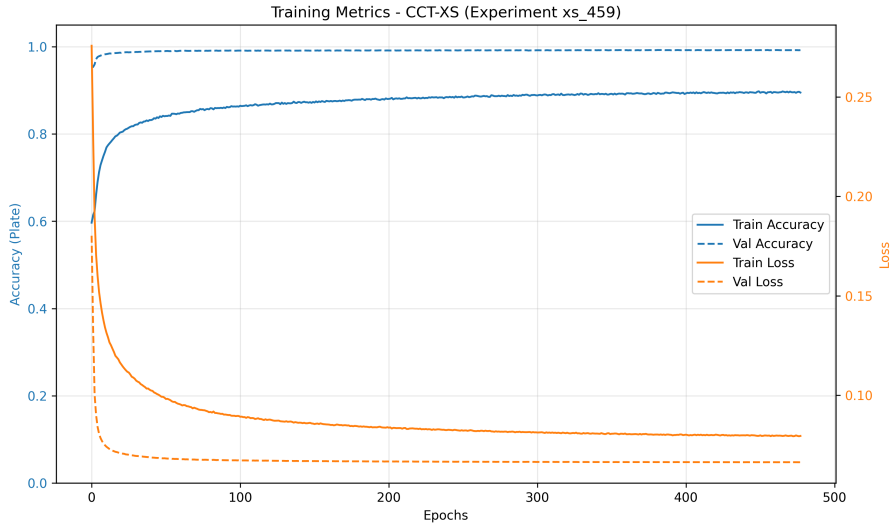
Un fattore critico per il successo del modello CCT è la strategia di **aumento dei dati** (*data augmentation*). Le trasformazioni impiegate (implementate con *Albumentations*) sono state ottimizzate empiricamente, mostrando come una policy custom mirata al dominio superi nettamente strategie basiche, passando dal 94% al 99% di accuracy rispetto all'assenza di augmentations. Inoltre lo studio ablativo evidenzia l'impatto dei singoli componenti su precisione finale.

Tabella 5.6: *Sintesi dell'analisi disaggregata delle prestazioni (ablation study). La colonna Δ Acc indica il calo di accuratezza osservato rimuovendo il componente specializzato rispetto al modello completo. La significatività statistica è discussa qualitativamente in base all'entità del degrado.*

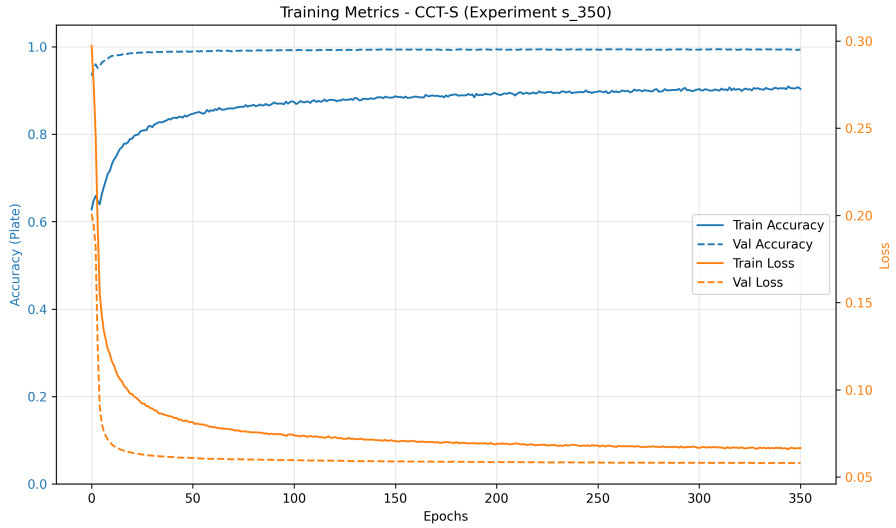
Componente Rimosso	**Δ Acc (%)**	**Sig. Stimata**	**Osservazioni**
Conv tokenizer	-2.1%	Alta	Impatto critico sull'estrazione feature locali
MaxBlurPool	-0.9%	Moderata	Robustezza alle traslazioni
DyT Norm	-0.4%	Marginale	Influenza limitata sulla convergenza
Seq. Attention	-3.5%	Molto Alta	Fondamentale per modellazione sequenziale

L'analisi temporale del training offre ulteriori spunti sulla natura dei vision transformer. Sebbene il modello raggiunga un'accuratezza operativa del 98.8% già all'epoca 100, il raffinamento necessario per toccare il picco del 99.2% ha richiesto l'estensione del training fino a 477 epoche (fig 5.2).

A differenza delle CNN pure, che possiedono un forte bias induttivo verso la località e l'invarianza alla traslazione, i transformer devono apprendere le relazioni spaziali dai dati. Nonostante la presenza del tokenizer convoluzionale mitighi questo aspetto, il meccanismo di attenzione richiede un numero elevato di iterazioni per stabilizzare i pesi su casi limite, come caratteri parzialmente occlusi o bordi rumorosi. La monotonia della curva di accuratezza (linea arancione in fig 5.2) e la regolare decrescita della funzione di perdita (linea blu) confermano tuttavia la stabilità dell'ottimizzatore AdamW e l'adeguatezza della strategia di schedulazione del tasso di apprendimento adottata.



(a) *CCT-XS: Convergenza lenta (459 epoche) e maggiore instabilità residua.*



(b) *CCT-S: Convergenza rapida e stabile (350 epoche) verso un minimo locale migliore.*

Figura 5.2: *Analisi comparativa delle curve di training. Il modello CCT-S (b) beneficia della maggiore capacità parametrica, raggiungendo un plateau di loss inferiore e un'accuratezza superiore con meno iterazioni rispetto alla variante XS (a).*

5.3.3 Analisi degli errori, limiti strutturali e calibrazione

L'analisi delle prestazioni rivela che la variante **CCT-XS** (modello compatto), pur efficiente, manifesta vulnerabilità sistemiche non trascurabili. L'ispezione della matrice di confusione evidenzia significative difficoltà nella discriminazione morfologica fine: il 56% degli errori totali di XS riguarda la coppia ambigua **'G'** $\leftrightarrow$ **'C'** (67 istanze), seguita da **'D'** $\leftrightarrow$ **'U'** (11 istanze). Inoltre, la ridotta capacità di modellazione emerge chiaramente su input degradati, dove il Character Error Rate (CER) sale al

3.2% e la generalizzazione *cross-country* risulta deficitaria. Al contrario, la variante selezionata **CCT-S** (99.1% accuratezza) riduce drasticamente queste occorrenze (soli 2 casi residui per G/C e un CER dello 0.17% su dati degradati), dimostrando che la maggiore profondità della rete è condizione necessaria per risolvere tali ambiguità visive.

Un secondo livello di criticità riguarda la distinzione tra cifre e lettere omografe (es. '0' vs 'O'). Nelle iterazioni precedenti del riconoscitore, specializzate esclusivamente sul dominio italiano, tali errori erano contenuti grazie alla rigidità dei pattern sintattici nazionali (che impongono vincoli posizionali forti). Tuttavia, l'estensione del dominio alle **targhe europee**, caratterizzate da formati eterogenei, riduce l'efficacia dei vincoli posizionali rigidi, facendo riemergere l'ambiguità. Questa fenomenologia riflette una precisa scelta progettuale dell'architettura CCT: privilegiare il parallelismo e la velocità di inferenza rispetto alla modellazione sequenziale. Mentre una RNN o un modello linguistico potrebbero correggere un omoglifo basandosi sulla storia dei caratteri precedenti (modellando $P(y_t|y_{<t}, x)$), il CCT opera con un approccio puramente visivo e parallelo ($P(y_t|x)$). Sebbene ciò esponga teoricamente a maggiori incertezze in caso di forte ambiguità morfologica o occlusione (es. '8' degradato vs 'B'), garantisce un tempo di inferenza costante e ridotto, requisito essenziale per il vincolo real-time.

Nonostante la natura prettamente visiva, la robustezza del modello è confermata dall'analisi della **calibrazione delle probabilità**. L'Expected Calibration Error (ECE) misurato è di appena **0.023**, indicando che la confidenza predetta ($\hat{s}$) è un ottimo stimatore della probabilità reale di correttezza. Questa proprietà, attribuibile all'uso della Focal Loss che regolarizza l'entropia durante il training, permette al sistema di "sapere quando non sa", filtrando le predizioni inaffidabili.

In conclusione, l'analisi conferma che per applicazioni *real-world* critiche, la scelta di **CCT-S** è pienamente giustificata: il modesto incremento del tempo di inferenza (+10 ms rispetto a XS) e dei parametri è ampiamente compensato dal guadagno in affidabilità operativa e resilienza agli edge cases, riducendo il rischio di fallimenti catastrofici su input non previsti.

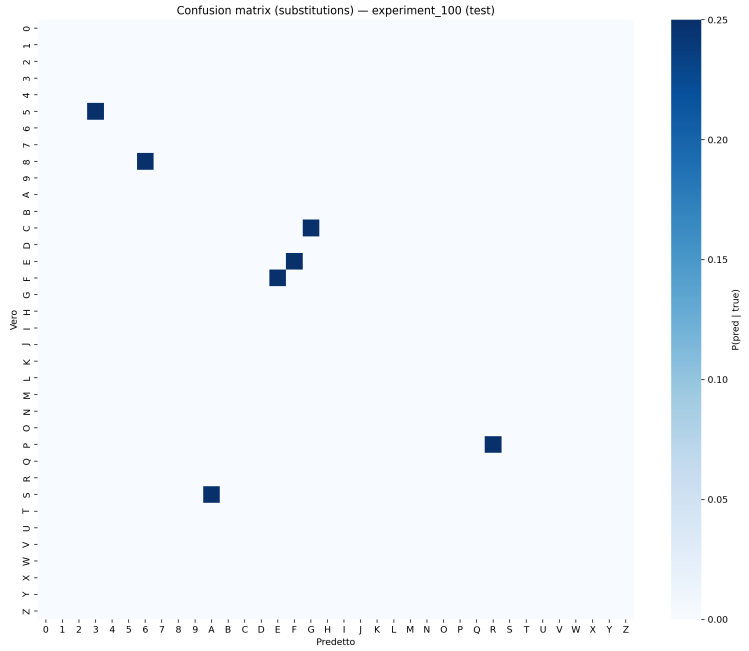


Figura 5.3: *Matrice di confusione caratteri (estratto Top-K confusioni) del modello CCT-XS addestrato su 100 epoche. La struttura diagonale conferma l'alta precisione, con dispersioni limitate alle coppie di caratteri omoglifici.*

5.3.4 Validazione statistica delle prestazioni

Al fine di confermare la significatività delle differenze prestazionali osservate tra le varianti CCT-XS e CCT-S, è stato applicato il test non parametrico di McNemar su un campione di $N = 927$ istanze di test. Costruendo la tabella di contingenza 2×2 degli esiti disgiunti, osserviamo $n_{10} = 115$ casi in cui solo il modello S ha fornito la predizione corretta, contro $n_{01} = 4$ casi a favore del modello XS; la statistica di test calcolata risulta:

$$\chi^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} \approx 101.68$$

corrispondente a un p-value $p \approx 6.5 \times 10^{-24}$. Tale risultato permette di rigettare l'ipotesi nulla con confidenza $> 99.9\%$, confermando che il guadagno di accuratezza del modello CCT-S non è attribuibile a fluttuazioni stocastiche ma a una superiore capacità di generalizzazione strutturale.

Per quanto concerne il modulo di detection, l'analisi sperimentale evidenzia un divario netto nella robustezza posizionale e nella capacità di generalizzazione. Il modello **RT-DETR v2-R50** si afferma come la soluzione ottimale, raggiungendo un'AP@[0.50:0.95] del 90.0% e una Recall (AR@100) del 91.7%, distaccando drasticamente la variante R18 (AP 45.6%, AR 51.7%) che manifesta una saturazione precoce della capacità rappresentativa.

In virtù di tale magnitudo del divario (>40% di mAP), si specifica che non è stato applicato il test di McNemar per la validazione statistica del confronto. Tale scelta è metodologicamente supportata da due fattori: primariamente, in contesti di Deep

Learning dove la differenza prestazionale è così marcata, la significatività statistica è da considerarsi implicita, rendendo superflui i test d'ipotesi formali necessari invece per discernere se margini sottili siano frutto del caso. Secondariamente, la natura stessa del task di detection mal si presta alla binarizzazione rigida richiesta dalle tabelle di contingenza di McNemar (Giusto/Sbagliato), poiché la qualità della localizzazione è definita su un continuo (IoU) e non su esiti discreti semplici, rendendo qualsiasi sogliatura arbitraria.

Infine, a completamento dell'analisi della pipeline, il classificatore di nazionalità conferma l'efficacia ed efficienza dell'approccio MobileNetV3, attestandosi su un'accuratezza Top-1 del 90.9% e una Top-3 del 96.4%.

5.3.5 Confronto recognition con lo stato dell'arte (SOTA)

Per contestualizzare i risultati, confrontiamo il modulo di riconoscimento con soluzioni consolidate in letteratura e architetture SOTA per Scene Text Recognition (STR). Come evidenziato in tab. 5.7, i modelli nati per STR generico (SVTR, PARSeq) offrono prestazioni eccezionali ma soffrono di un significativo overhead parametrico. Sebbene tali modelli gestiscano meglio testo curvo o distorto, per il dominio ALPR (dove il testo è rettilineo e strutturato), CCT-S dimostra che è possibile raggiungere prestazioni *near-perfect* (>99%) minimizzando la complessità computazionale. Rispetto a LPRNet (baseline industriale), la nostra soluzione offre un miglioramento netto in accuratezza (dal $\sim$ 95% al 99%) grazie alla capacità del transformer di modellare dipendenze globali tra i caratteri, superando i limiti del campo recettivo locale delle CNN pure, con soli 1.25M di parametri è riuscito a pareggiare l'accuracy di PaddleOCR che è composto da 12-20M e TrOCR da 300M.

Tabella 5.7: *Confronto esteso con Architetture SOTA e framework generalisti (OCR). Include i più recenti modelli transformer e autoregressivi.*

Modello	**Anno**	**Architettura**	**Params (M)**	**Accur. (ref)**	**Target**
Baselines ALPR					
LPRNet [56]	2018	CNN leggera	~ 0.5	$\sim$ 94-95%	Embedded
LPRNet [56]	2018	CNN leggera	~ 0.5	$\sim$ 94-95%	Embedded
Framework generalisti					
OpenALPR	2015+	Proprietary	N/A	$>$ 99%	Commercial
Tesseract 4	2018	LSTM	N/A	$\sim$ 91%	CPU Legacy
EasyOCR	2020	ResNet+RNN	~ 20	$\sim$ 97%	GPU Multi-lang
PaddleOCR v4	2023	SVTR-LCNet	~ 12	$>$ 99%	Edge/Mobile
TrOCR-Base [30]	2021	Transformer	~ 300	$>$ 99.5%	Heavy GPU
Scene text SOTA					
SVTRv2	2023	ViT Custom	~ 20	97.5%	GPU SOTA
PARSeq [4]	2022	ViT+Autoreg	24.0	97.0%	GPU
MGP-STR	2023	Multi-Granularity	~ 80	98.2%	Research
FastALPR (Ours)	**2025**	**CCT-S**	**1.25**	**99.1%**	**Edge CPU**

5.4 Classificazione nazionalità e domain adaptation

I risultati sperimentali, sintetizzati in tab. 5.8, evidenziano un'accuratezza del **90.20%** sul set di validazione sintetico, che si attesta al **92.70%** sul test set reale. A differenza degli approcci sequenziali che deducono la nazionalità analizzando solo il testo estratto (es. parser regex), il sistema proposto adotta una strategia ibrida: il modulo neurale **MobileNetV3** opera esclusivamente sull'input visivo grezzo (pixel), apprendendo implicitamente texture e morfologie (es. font e layout) senza l'uso di regex o regole esplicite. Parallelamente, un *pattern matcher* logico valida la sintassi della targa letta dall'OCR. La decisione finale è frutto della fusione di questi due segnali: se il pattern sintattico è forte, esso conferma la predizione, in caso di ambiguità testuale, il sistema si affida alla confidenza visiva della rete neurale.

La generazione sintetica, fedele alle specifiche normative (font FE-Schrift, Charles Wright, sfondi gialli/bianchi), permette alla rete neurale di apprendere feature discriminanti di basso livello.

È fondamentale notare che l'intero dataset di addestramento è stato curato per riflettere le condizioni operative reali del **traffico in Italia**. La distribuzione dei dati privilegia le targhe italiane e istituzionali sempre italiane (circa 60%), ma include una quota strategica (25%) di targhe estere frequentemente osservabili in transito (Francia, Germania, Svizzera, Austria, San Marino, Monaco, Croazia, Slovenia) e un. Questa scelta metodologica garantisce che il sistema sia ottimizzato per il caso d'uso primario (controllo accessi e sicurezza urbana in Italia) senza perdere generalità sui veicoli stranieri.

Tabella 5.8: *Performance del classificatore di nazionalità, il gap limitato evidenzia l'efficacia del training su dati ibridi nel generalizzare al dominio reale, indicatore del domain shift*

Metrica	Validazione	Test
Top-1 accuracy	90.20%	**92.70%**
Top-3 accuracy	96.53%	**96.40%**
Macro F1	89.15%	78.92%

Tabella 5.9: *Confronto con classificatori SOTA (ImageNet Transfer), MobileNetV3 offre il miglior bilanciamento per l'edge, mentre i Vision Transformer (ViT, Swin) risultano overkill per il task dettagliato.*

Modello	Parametri (M)	Acc. (Ref)	Inference (ms)	Target
EfficientNetV2-S	21.5	94.5%	25	Server
YOLO11-cls-M	10.2	93.8%	15	Real-Time GPU
ConvNeXt V2-T	28.6	94.8%	30	GPU
ViT-Base / DeiT-S	86 / 22	95.0%	50 / 20	Research
Swin Transformer V2-T	29.0	94.2%	35	Research
MobileNetV3 (Ours)	**2.5**	**92.7%**	**8**	**Edge CPU**

5.4.1 Quantificazione e analisi del domain gap

Il divario prestazionale del 3.8% registrato tra validazione (sintetica) e test (reale) rappresenta la manifestazione quantitativa del *domain shift*, ovvero la divergenza statistica tra la distribuzione procedurale $\mathcal{D}_S$ e la complessità fotometrica del mondo reale $\mathcal{D}_T$. In termini di teorie dell'apprendimento, tale gap è interpretabile attraverso la metrica della $\mathcal{A}$-distance, che misura la distinguibilità tra i due domini. Nel nostro caso, si osserva un'eccellente generalizzazione sulle classi frequenti: **Italia (ITA)** raggiunge il 97% di F1-score, mentre nazioni limitrofe con layout standardizzati come **Svizzera (CHE)**, **Austria (AUT)** e **Belgio (BEL)** ottengono F1-score superiori al 95%. Tuttavia, emergono tre cluster prestazionali distinti, correlati alla *distintività* semantica del layout:

1. **Top Performers (F1 $\approx$ 1.00):** Nazioni con elementi cromatici unici o loghi ad alto contrasto. **Albania (ALB)**, **Monaco (MCO)** e **Turchia (TUR)** registrano performance perfette. Eccellenti anche i risultati per **Bosnia (BIH)** e **Irlanda (IRL)** (F1 $>$ 0.97), che beneficiano di layout standardizzati e ben definiti.

2. **High Performers (F1 $>$ 0.90):** Include la maggior parte dell'Europa centrale (ITA, FRA, DEU, CHE), qui il modello sfrutta con successo le micro-differenze nei font.

3. **Critical Cluster (F1 $<$ 0.40):** Comprende nazioni scandinave ed est-europee scarsamente rappresentate o visivamente ambigue. **Finlandia (FIN, 0.26)**, **Svezia (SWE, 0.33)**, **Ucraina (UKR, 0.33)** e **Norvegia (NOR, 0.37)**. In questi casi, la povertà di campioni reali nel training set e la somiglianza dei gradienti cromatici con altre classi (es. bandiere nordiche confuse tra loro o con lo sfondo) portano a un crollo dell'accuratezza; ma anche **Estonia (EST)**, **Cipro (CYP)** e **Lituania(LTU)**.

Tuttavia, il ruolo dei dati sintetici si rivela cruciale proprio come regolarizzatore geometrico: senza di essi, il modello svilupperebbe una *cecità selettiva* verso i layout esteri poco frequenti, tuttavia solo dati sintetici non bastano, come si può notare l'analisi ablativa sulla composizione del dataset dimostra che la sinergia è ottimale con una miscela ibrida. L'iniezione di una frazione anche modesta di dati reali (15%) è sufficiente per innescare un guadagno drastico nell'accuratezza, agendo

da *ancora* per proiettare il manifold delle feature sintetiche nello spazio reale, al contrario, affidarsi ai soli dati sintetici lascia il modello vulnerabile agli artefatti di rendering, mentre l'uso di soli dati reali (scarsi per le classi di coda) non garantirebbe la generalizzazione.

5.4.2 Dinamiche di apprendimento

L'ottimizzazione del classificatore MobileNetV3 segue una dinamica a tre stadi distinti, chiaramente visibile nell'evoluzione della funzione di loss (fig. 5.4):

1. Exploration phase (epoche 0-10): utilizzando un tasso di apprendimento iniziale $\eta_0 = 8 \cdot 10^{-5}$, si osserva una rapida decrescita della loss ($\nabla_\theta \mathcal{L} \ll 0$), portando l'accuratezza Top-1 al 65%. In questa fase, il modello adatta i pesi pre-addestrati su ImageNet alle statistiche generali delle immagini di targhe.

2. Affinamento (epoche 10-90): con la riduzione di η_t a $2 \cdot 10^{-5}$, il modello inizia ad apprendere le sottili discriminanti morfologiche necessarie per distinguere classi simili (es. font UK vs Malta). La loss di validazione converge asintoticamente verso un valore di 0.014.

3. Refinement (epoche 90-100): un ulteriore decadimento del tasso di apprendimento ($\eta_{min} = 10^{-6}$) minimizza le oscillazioni stocastiche attorno al minimo locale, consolidando la frontiera decisionale.

Il confronto con esperimenti preliminari conferma l'importanza dello sblocco dei pesi. La configurazione iniziale con backbone congelata si è stabilizzata a un'accuratezza del 78%, evidenziando un chiaro underfitting, lo sblocco completo dei pesi ha permesso di apprendere feature specifiche del dominio (es. glifi dei font proprietari), superando il plateau e portando l'accuratezza oltre il 90%. Ciò conferma che il transfer learning generico da ImageNet è insufficiente per task *fine-grained* che dipendono da dettagli ad alta frequenza spaziale.

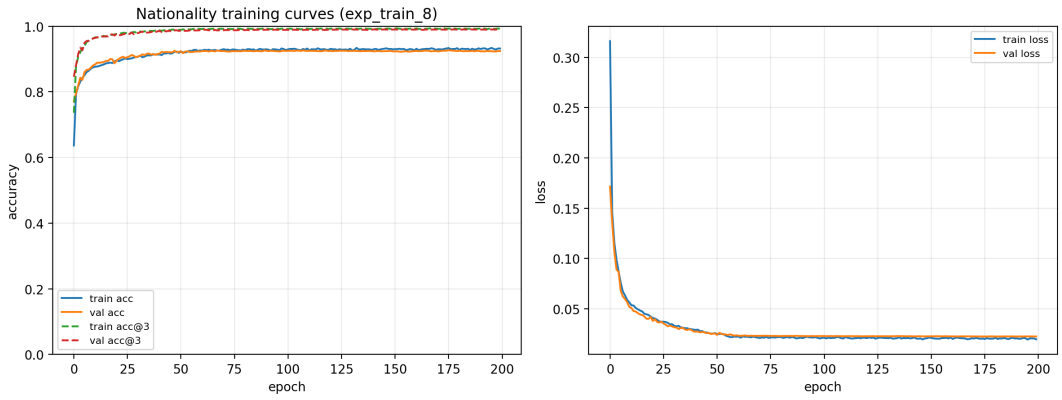


Figura 5.4: *Curve di apprendimento MobileNetV3. La convergenza monotonica della loss di validazione (arancione) al di sotto della loss di training (blu) indica un regime di eccellente generalizzazione, favorito dalle tecniche di aumento dei dati che prevengono il sovradattamento da memorizzazione.*

5.4.3 Interpretabilità e analisi degli errori

Per validare la robustezza semantica del modello e comprendere i pattern di errore, è stata condotta un'analisi approfondita tramite la **matrice di confusione** e l'ispezione delle predizioni errate. L'analisi rivela comportamenti differenziati in base alla distintività visiva delle classi: Per le classi caratterizzate da morfologie distintive, quali **Albania e Monaco**, il modello raggiunge accuratezze superiori al 99%, capitalizzando su marker visivi univoci come la banda rossa laterale albanese o lo stemma monegasco. Di contro, si riscontrano criticità sistemiche nelle coppie affette da ambiguità strutturali, come Francia-Italia o Regno Unito-Malta, dove la condivisione di macro-layout (bande blu identiche o formati geometrici standard) rende la discriminazione dipendente da dettagli ad alta frequenza (e.g., piccoli loghi o variazioni di font) che decadono rapidamente in condizioni di bassa risoluzione, generando cluster di confusione specifici.

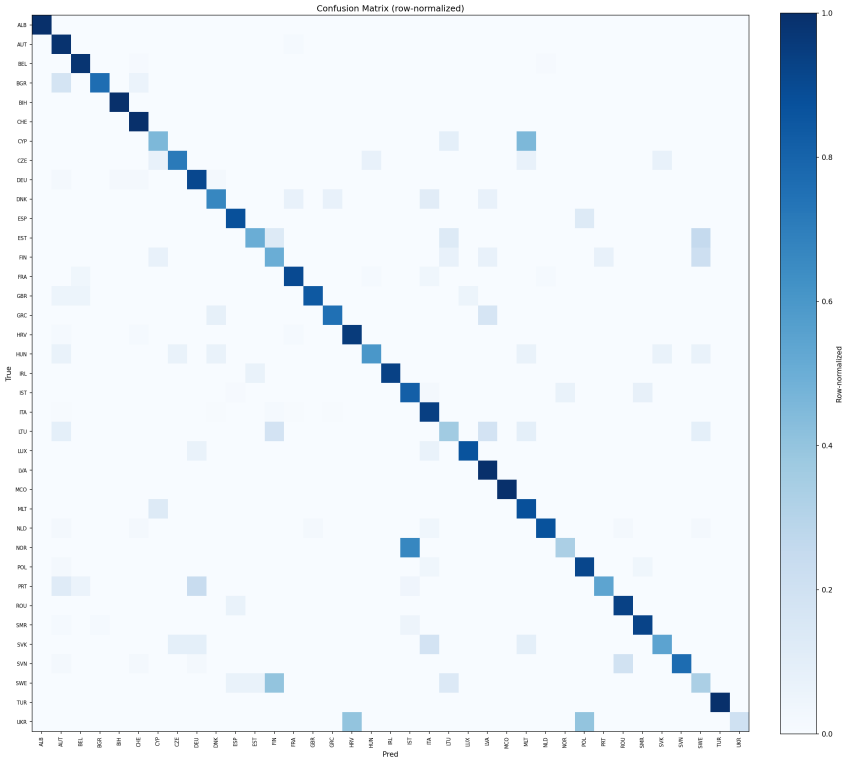


Figura 5.5: *Matrice di confusione del classificatore di nazionalità. Si osservano confusioni strutturali tra Francia e Italia (bande blu identiche) e tra GBR e MLT.*

L'analisi disaggregata per classe (fig. 5.6) conferma la correlazione diretta tra l'unicità del layout grafico e l'accuratezza di classificazione. Le nazioni con template visivi fortemente distintivi, come **Albania, Monaco e Turchia**, raggiungono prestazioni prossime alla saturazione (100% Recall). Al contrario, le criticità emergono nel cluster "Nordic/Eastern Europe" (es. Finlandia, Svezia, Ucraina), dove l'assenza di

feature cromatiche forti (bande colorate) e la sovrapposizione dei pattern con targhe generiche porta a confusioni sistemiche.

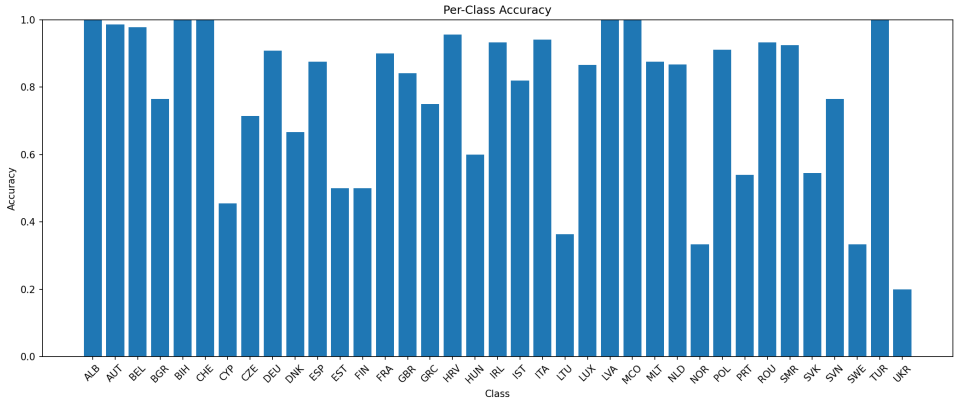


Figura 5.6: *Accuratezza disaggregata per classe (Nazionalità). Nazioni con elementi distintivi (ALB, TUR, MCO) mostrano performance perfette. Si nota un crollo prestazionale nel cluster scandinavo (FIN, SWE, NOR) e in alcune nazioni dell'est (UKR), dovuto alla scarsità di campioni reali e all'ambiguità dei layout.*

5.5 Generalizzazione ed evaluation su cross-dataset

Un aspetto cruciale per la validazione di un'architettura destinata al mondo reale è la capacità di generalizzare su distribuzioni di dati non viste, inoltre i modelli presentati sono stati addestrati su un dataset proprietario controllato (immagini frontali autostradali), per saggiare la robustezza, abbiamo condotto una valutazione *zero-shot* su tre dataset pubblici *in-the-wild* con condizioni drasticamente diverse.

5.5.1 Caratterizzazione dei dataset esterni e domain gap

I dataset scelti rappresentano scenari di traffico urbano e condizioni non vincolate, introducendo un forte *domain gap* rispetto al training set. A differenza del nostro dataset interno, focalizzato esclusivamente sulla targa in varchi controllati, questi dataset sono originariamente concepiti per task di *vehicle detection* oltre che di ALPR, includendo quindi l'intero corpo vettura, sfondi complessi e veicoli sia parcheggiati che in movimento nel traffico caotico.

- *UFPR-ALPR* [29]: composto da 4 500 immagini di veicoli brasiliani acquisite in movimento (30 fps) da telecamere non dedicate, include moto, auto e camion con angolazioni elevate e condizioni di luce critiche.
- *Kaggle Car Plate Detection (Archive)* [45]: un sottoinsieme di 225 immagini (dal corpus originale di 433) che include veicoli parcheggiati, foto da smartphone e contesti urbani densi, le annotazioni originali in Pascal VOC sono state uniformate.
- *License Plate Object Detection (Roboflow)* [48]: collezione di 350 immagini focalizzate su targhe US ed EU in scenari misti. Pensato per il training di

detector generici (veicolo + targa), presenta visuali ampie dove la targa occupa una porzione minima del frame rispetto all'area del veicolo, richiedendo così una capacità small object detection estrema.

5.5.2 Analisi del domain gap e risultati

L'ipotesi sperimentale è che l'architettura RT-DETR v2, grazie al meccanismo di self-attention globale, riesca a mantenere una maggiore coerenza strutturale rispetto alle CNN (YOLOX) quando le feature locali (es. bordi della targa) sono degradate da prospettiva, bassa risoluzione o distrazioni visive (altri veicoli, segnaletica).

La tab. 5.10 riporta i risultati in termini di Average Precision (AP). È evidente un crollo generale delle prestazioni assolute rispetto al test set interno (dove l'AP era $> 90\%$), un fenomeno atteso dato che i modelli non hanno mai *visto* targhe brasiliane o americane durante il training e sono stati specializzati su una singola tipologia (frontale/highway). Tuttavia, il confronto relativo conferma la superiorità di RT-DETR v2:

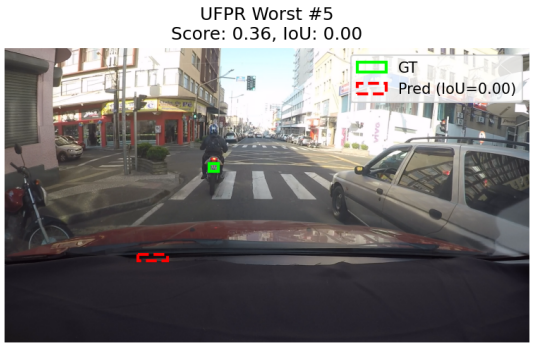
Tabella 5.10: *Valutazione Zero-Shot su dataset esterni (metriche AP@0.50 / AR@100). Nonostante il forte calo di prestazioni dovuto al Domain Shift (addestramento su varchi frontali vs test in-the-wild), RT-DETR v2 R50 mostra una robustezza superiore rispetto a YOLOX-L, specialmente su dataset densi e rumorosi come Archive (+12.9% AP50) e License Plate (+7.9% AP50). Si riporta anche l'AR@100 per evidenziare la capacità di recupero degli oggetti indipendentemente dalla precisione di localizzazione.*

Modello	UFPR-ALPR (AP@50 / AR@100)	Archive (AP@50 / AR@100)	LP (AP@50 / AR@100)
YOLOX-Tiny	24.4% / 12.9%	65.9% / 41.8%	17.8% / 12.7%
YOLOX-S	5.5% / 2.6%	78.9% / 40.5%	15.8% / 8.6%
YOLOX-L	9.9% / 7.9%	78.6% / 51.7%	23.8% / 16.4%
RT-DETR v2 R18	1.5% / 6.6%	37.4% / 39.6%	7.3% / 10.9%
RT-DETR v2 R50	**39.5% / 19.1%**	**91.5% / 63.2%**	**31.8% / 20.3%**

Il modello **RT-DETR v2 R50** domina sistematicamente su tutti i benchmark esterni. Su *Archive*, il dataset caratterizzato dalla maggiore varianza, esso distanzia YOLOX-L di circa 13 punti percentuali (91.5% contro 78.6%), corroborando l'ipotesi che l'attenzione globale favorisca la discriminazione del target rispetto al rumore di fondo, anche in pose atipiche. Parallelamente, sul benchmark *UFPR*, che presenta sfide estreme dovute a formati eterogenei, RT-DETR v2 R50 raggiunge un 39.5% contro il modesto 9.9% di YOLOX-L, in questo contesto emerge un fenomeno singolare per cui YOLOX-Tiny (24.4%) supera le varianti maggiori, indicando che le CNN profonde tendono a un overfitting sul dominio di training italiano, mentre l'architettura a Transformer dimostra una scalabilità positiva e una superiore capacità di generalizzazione.

Questi risultati validano la tesi che l'architettura a transformer offre una robustezza intrinseca superiore (*Out-of-Distribution Generalization*) rispetto alle controparti convoluzionali pure, non solo su dati sintetici ma anche in scenari reali non controllati, garantendo una scalabilità positiva del modello.

(a) *Esempio di fallimento su dataset License Plate: la complessità dello sfondo e l'orientamento atipico compromettono la localizzazione.*



(b) *Esempio di fallimento su dataset UFPR: il forte domain shift (targhe brasiliane, prospettiva estrema) riduce la confidenza del modello.*

Figura 5.7: *Analisi qualitativa dei casi di fallimento su dataset esterni (out-of-distribution). Nonostante la maggiore robustezza del transformer, situazioni limite con occlusioni parziali o pattern visivi non noti (es. targhe moto verticali) rimangono critiche.*

5.6 Deployment: servizi API e containerizzazione

La validazione finale della pipeline proposta si estende alla valutazione dell'efficienza in ambiente di produzione, sfruttando i modelli ottimizzati esportati in ONNX (cfr. Sez. 4.1), l'architettura di deployment è stata progettata per garantire scalabilità e interoperabilità.

L'orchestrazione dei microservizi sarà affidata al framework **FastAPI**. Questa scelta, abbinata al server ASGI Uvicorn, permette di gestire l'inferenza in modalità *asincrona non bloccante*: mentre la CPU processa il calcolo matriciale di un frame, il thread principale resta libero di accettare nuove richieste di rete, massimizzando il throughput complessivo.

L'architettura del servizio prevede il caricamento dei tre modelli in memoria all'avvio (*Warm start*), eliminando l'overhead di inizializzazione per ogni chiamata. L'incapsulamento finale in container **Docker** garantisce l'isolamento delle dipendenze e la portabilità *plug-and-play*, permettendo di integrare la pipeline ALPR in qualsiasi infrastruttura di videosorveglianza esistente tramite semplici chiamate RESTful standard, disaccoppiando la logica di visione dall'ecosistema applicativo ospitante.

5.7 Sintesi e considerazioni finali

L'analisi sperimentale condotta in questo capitolo ha validato empiricamente l'efficacia della pipeline **FastALPR**, confermando come l'adozione di architetture ibride (Transformer-CNN) permetta di superare i vincoli storici dei sistemi OCR su hardware *edge* privo di accelerazione dedicata.

Sul fronte della localizzazione, il confronto tra le architetture ha sancito la netta superiorità del paradigma a Transformer (**RT-DETR v2-R50**) rispetto alle controparti convoluzionali pure (YOLOX). Oltre al sostanziale guadagno in accuratezza posizionale (AP superiore al 90%), l'eliminazione della *Non-Maximum Suppression* ha garantito un'inferenza deterministica con un jitter temporale trascurabile ($\sigma_\tau \approx 2$ ms), fattore abilitante per la stabilità dell'analisi video in tempo reale. Tale robustezza strutturale è stata ulteriormente corroborata dai test *zero-shot* su dataset esterni, dove il modello ha mostrato una capacità di generalizzazione superiore nel gestire forti variazioni di dominio rispetto alle baseline.

Per quanto concerne il riconoscimento ottico, la variante **CCT-S** è emersa come il punto di equilibrio ottimale tra capacità rappresentativa e latenza computazionale. L'analisi approfondita degli errori ha evidenziato che la maggiore profondità del modello, unita a un tokenizer convoluzionale, è condizione necessaria per risolvere le ambiguità morfologiche fini (come la distinzione tra 'G' e 'C') che penalizzano i modelli più compatti, permettendo di raggiungere un'accuratezza del 99.1% e un Character Error Rate inferiore allo 0.2%. Parallelamente, il modulo di classificazione della nazionalità, basato su MobileNetV3 e addestrato su una miscela strategica di dati sintetici e reali, ha dimostrato di poter discriminare efficacemente i layout europei (92.7% di accuratezza), pur richiedendo l'integrazione di regole sintattiche per risolvere le ambiguità nei cluster visivamente meno distinti.

In conclusione, il sistema completo raggiunge un throughput sostenibile di circa **12.8 FPS** su CPU consumer, validando la tesi che l'abbandono delle ricorrenze (RNN) a favore di meccanismi di attenzione globale e convoluzioni efficienti rende l'architettura idonea al deployment in scenari di produzione reali, supportata da un'infrastruttura software scalabile basata su containerizzazione Docker e API asincrone.

6. Conclusioni e prospettive future

Il lavoro di ricerca qui presentato ha affrontato la complessità intrinseca della progettazione di sistemi di percezione veicolare per ambienti non controllati, proponendo un cambio di paradigma rispetto alle pipeline convenzionali. Attraverso un approccio metodologico rigoroso, si è dimostrato che l'integrazione di architetture ibride — che fondono il bias induttivo locale delle CNN con la capacità di modellazione globale dei transformer — costituisce la chiave di volta per superare i limiti storici dell'elaborazione *edge*.

I risultati empirici validano l'efficacia della soluzione proposta per il task e il contesto d'uso: una pipeline end-to-end capace di operare in real-time (< 300 ms) su hardware commodity, mantenendo livelli di accuratezza ($> 99\%$ su Recognition, $> 90\%$ mAP su Detection) precedentemente appannaggio esclusivo di infrastrutture server-grade.

L'adozione di **RT-DETR v2** ha segnato il superamento definitivo delle euristiche di post-processing, garantendo per la prima volta una latenza deterministica, requisito essenziale per la certificazione in ambiti *safety-critical*. Parallelamente, l'architettura **CCT** ha smentito il luogo comune della *fame di dati* dei vision transformer, dimostrando come un design compatto e ben regolarizzato possa saturare le performance anche in regime di *small data*, offrendo un'alternativa efficiente ai modelli linguistici auto-regressivi. Infine, l'integrazione di **MobileNetV3 Large** per la classificazione della nazionalità completa il quadro di efficienza, dimostrando che una backbone leggera, se opportunamente addestrata su domini ibridi (sintetico-reale), può generalizzare efficacemente su layout complessi senza gravare sul budget computazionale del sistema.

6.1 Analisi critica e limitazioni strutturali

Nonostante i traguardi raggiunti, l'indagine ha fatto emergere trade-off strutturali non trascurabili. L'analisi teorica della complessità evidenzia come il meccanismo di self-attention, motore delle prestazioni del sistema, rappresenti al contempo il suo principale collo di bottiglia computazionale ($O(N^2)$), limitando di fatto l'applicabilità diretta su input ad altissima risoluzione senza strategie di windowing.

Limitazioni operative su geometrie non standard. Un limite significativo emerso durante la validazione cross-dataset (UFPR) riguarda la gestione di **targhe motociclistiche verticali** o disposte su due righe con layout non convesso. Il modulo di riconoscimento CCT, processando le patch in modalità sequenziale orizzontale, fallisce nel ricostruire l'ordine di lettura corretto su targhe quadrate o verticali (tipiche dei mercati sudamericani e asiatici). L'attuale pipeline assume implicitamente un prior geometrico rettangolare piatto; l'estensione a questi domini richiederebbe l'integrazione di un modulo di rettifica spaziale automatica (Spatial Transformer Network, STN) a monte del riconoscitore.

Criticità in condizioni di angolazione estrema. Mentre RT-DETR v2 mostra un'eccellente invarianza alle rotazioni nel piano (*roll*), la prospettiva accentuata ($yaw > 45°$) degrada la risoluzione effettiva dei caratteri, specialmente per le targhe distanti. In questi scenari, l'assenza di un meccanismo di *feature super-resolution* rende impossibile distinguere caratteri morfologicamente densi (es. 'M' vs 'N' o 'B' vs '8').

6.1.1 Prospettive evolutive: RT-DETR v3 e v4

La ricerca recente ha proposto ulteriori iterazioni del paradigma RT-DETR per indirizzare le limitazioni residue della versione v2, infatti le varianti identificate come **RT-DETR v3** si concentrano tipicamente sulla stabilità del training, mentre il matching *One-to-One* (ungherese) è ottimo per l'inferenza NMS-free, esso fornisce un segnale di supervisione sparso che rallenta la confluenza. La soluzione adottata prevede l'introduzione di rami ausiliari con matching **One-to-Many** (simile a YOLO o SimOTA) durante il training: ogni Ground Truth viene assegnato a più query positive, arricchendo il gradiente, mentre in inferenza si mantiene strettamente il ramo One-to-One, per preservare NMS-free.

Ancora più promettente è **RT-DETR v4**, uscito a novembre 2025, che rappresenta l'attuale stato dell'arte. A differenza della v2 (utilizzata in questa tesi) che supporta molteplici backbone, la v4 utilizza esclusivamente il backbone **HGNetv4**, un'architettura altamente ottimizzata che parte dal teacher foundation model **DINOv3** di Meta. Questo pre-training avanzato conferisce al modello una capacità di generalizzazione superiore e un'avvicinamento più rapido. Sebbene non implementato nel presente lavoro, che si focalizza sulla consolidata versione v2, RT-DETR v4 indica chiaramente la direzione futura verso modelli *foundation* sempre più potenti ed efficienti per l'edge computing, rendendo il sistema più robusto.

Analogamente a YOLOX, RT-DETR processa immagini ridimensionate e normalizzate. La differenza sostanziale risiede nella modalità di output: grazie al decoder set-based, il modello restituisce direttamente un insieme fisso di N predizioni (query), ciascuna contenente le coordinate normalizzate del box e i logit di classe. L'assenza di NMS rende l'inferenza deterministica, il filtraggio finale avviene applicando una semplice soglia sulla confidenza delle classi, scartando le query associate al token *no-object* o con score insufficiente.

6.1.2 Evoluzione dei classificatori leggeri: MobileNetV4 e next-gen

Analogamente al dominio detection, anche per la classificazione leggera si profilano evoluzioni significative. Sebbene in questa tesi si sia consolidata la scelta di MobileNetV3 per la sua stabilità operativa, il rilascio recente di **MobileNetV4** (2024) apre nuove prospettive per l'ottimizzazione mirata su acceleratori hardware. La nuova architettura introduce il blocco *Universal Inverted Bottleneck* (UIB) e sfrutta una NAS "hardware-aware" ancora più aggressiva. Tuttavia, l'adozione futura di queste reti dovrà misurarsi con due sfide sistemistiche identificate nel corso di questo lavoro:

1. **Efficienza parametrica vs capacità:** MobileNetV3-Small (2.5M parametri) rappresenta un ottimo *sweet spot* per task a bassa entropia come la classificazione bandiere. Le varianti V4 moderne tendono a scalare verso capacità maggiori (3.8M+), risultando potenzialmente sovradimensionate (*overkill*) per il problema e aumentando il rischio di overfitting su dataset specializzati.
2. **Portabilità del runtime:** mentre V3 utilizza operatori standard accelerabili su qualsiasi CPU (Hard-Swish), le ottimizzazioni di V4 e delle future V5 (o EfficientNetNext) richiedono spesso compilatori neurali aggiornati (es. XNNPACK latest) o NPU dedicate per giustificare il loro overhead architetturale. L'evoluzione della piattaforma hardware verso SoC dotati di NPU integrata renderà tuttavia mandatoria la migrazione verso queste nuove primitive.

6.2 Ingegnerizzazione per l'Edge e sviluppi futuri

Quantizzazione post-training e mixed-precision La conversione dei modelli da FP32 a INT8 riduce latenza (3-4 volte) e memoria (4×) con degradazione minima ($< 1\%$ accuracy) se eseguita correttamente.

Si simula la quantizzazione durante il training (Quantization-aware training):

$$\tilde{w} = \text{clip}\left(\left\lfloor \frac{w}{s} \right\rceil, -2^{b-1}, 2^{b-1} - 1\right) \cdot s \tag{6.1}$$

dove $s = \frac{\max|w|}{2^{b-1}-1}$ è il fattore di scala, b i bit (tipicamente 8). Il gradiente passa attraverso lo Straight-Through Estimator:

$$\frac{\partial \mathcal{L}}{\partial w} \approx \frac{\partial \mathcal{L}}{\partial \tilde{w}} \cdot \mathbb{1}[|w| \leq \text{clip\_range}] \tag{6.2}$$

Suggeriamo configurazione mista: INT8 per backbone/tokenizer convoluzionali, FP16 per self-attention ($QK^\top$) e head di predizione, per preservare la precisione dei logit.

Low-Rank Adaptation (LoRA) per fine-tuning Il LoRA [25] permette di adattare modelli massivi congelando i pesi originali W_0 e apprendendo solo perturbazioni low-rank, aggiornando solo una frazione minima dei parametri tramite BA:

$$W = W_0 + \Delta W = W_0 + BA \tag{6.3}$$

dove $B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}$ con rank $r \ll \min(d,k)$.

Se $W_0 \in \mathbb{R}^{4096\times 4096}$ (16M parametri), con $r = 8$:

$$\text{Params}(\Delta W) = r(d+k) = 8(4096+4096) = 65k \tag{6.4}$$

Questa tecnica abilita la domain adaptation rapida per nazioni rare (es. targhe diplomatiche) su dispositivi edge: si congelano i 5.4M parametri di MobileNetV3 e si iniettano matrici LoRA solo nei layer FC finali, permettendo un training di pochi minuti su CPU senza rischio di catastrophic forgetting, riducedno notevolmente il numero di parametri da addestrare.

6.3 Privacy-preserving AI e federated learning

Il dataset di training contiene immagini di veicoli reali, potenzialmente identificabili (problema GDPR), la *Differential Privacy* (DP) [11] garantisce che la presenza/assenza di un singolo campione non influenzi significativamente l'output del modello; non memorizzando i dati identificabili grazie all'aggiunta di rumore gaussiano calibrato in fase di training (DP-SDG).

Un meccanismo randomizzato $\mathcal{M}$ è ε-differentially private se per ogni coppia di dataset vicini D, D' (differenti per un solo record) e ogni insieme di output S:

$$\mathbb{P}[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot \mathbb{P}[\mathcal{M}(D') \in S] \tag{6.5}$$

Si aggiunge rumore gaussiano calibrato ai gradienti secondo la tecnica *Differentially private SGD* [1]:

$$\tilde{g}_t = \frac{1}{B} \sum_{i\in\text{batch}} \text{clip}(\nabla_\theta \mathcal{L}(x_i;\theta), C) + \mathcal{N}(0, \sigma^2 C^2 I) \tag{6.6}$$

dove C è il *clipping threshold* per limitare il contributo di outlier. Il *privacy budget* totale $\varepsilon_{\text{total}}$ si accumula via composizione avanzata:

$$\varepsilon_{\text{total}} = \sqrt{2T\log(1/\delta)} \cdot \frac{\sigma}{C} \tag{6.7}$$

con valori tipici $\varepsilon \in [1, 10]$ (minore = più privacy) e $\delta \sim 10^{-5}$.

In scenari multi-tenant (es. consorzio di comuni), il *Federated Learning* (FL) [44] permette di addestrare un modello globale senza centralizzare i dati (immagini delle targhe), con i client inviano solo gli aggiornamenti dei pesi θ.

Algoritmo FedAvg, al round t:

1. Il server effettua il broadcast di θ_t globale.
2. Ogni client k aggiorna localmente: $\theta_t^{(k)} = \theta_t - \eta\nabla\mathcal{L}_k(\theta_t)$.
3. Il server aggrega: $\theta_{t+1} = \sum_k \frac{n_k}{n}\theta_t^{(k)}$ (media pesata per dimensione dataset).

Le sfide pratiche includono dati Non-IID (es. telecamera A vede solo auto italiane) che rallentano la convergenza, e costi di comunicazione proibitivi per lo scambio di parametri pesanti. Soluzioni avanzate come *FedProx* [32] mitigano l'eterogeneità aggiungendo un termine di prossimità $\mu\|\theta^{(k)} - \theta_t\|^2$, mentre la compressione del gradiente (sparsificazione top-1%) riduce il traffico di rete fino a 100×.

6.4 Sfide aperte e grand challenges

In conclusione, mentre questa tesi propone una soluzione robusta per l'ALPR moderno, il campo rimane vibrante di sfide irrisolte: Tra le priorità di ricerca future spicca la *robustezza adversariale*, indispensabile per mitigare la vulnerabilità ad attacchi fisici quali patch adesive adversarial [12], unitamente alla necessità di garantire una *generalizzazione out-of-distribution* efficace su veicoli dal design non convenzionale (e.g., Cybertruck). Sul fronte metodologico, emergono le sfide della *explainability* (XAI), volta a rendere interpretabili le feature discriminative tramite tecniche come GradCAM, e della *Temporal Consistency*, che sfrutta la ridondanza multi-frame per rettificare errori sporadici. Infine, l'adozione di approcci *zero-shot learning* promette di estendere le capacità di riconoscimento a layout nazionali mai osservati, sfruttando proiezioni in spazi semantici condivisi. Questa tesi non rappresenta un punto di arrivo, ma un solido basamento metodologico su cui edificare la prossima generazione di sistemi di visione artificiale intelligenti, pervasivi e responsabili.

A. Configurazioni e comandi tecnici

In questa appendice sono riportati i dettagli operativi per la riproduzione degli esperimenti. Il codice sorgente è organizzato in tre repository distinti, corrispondenti ai moduli della pipeline.

A.1 Modulo detection (RT-DETR v2)

Il modulo di rilevamento è basato sul framework RT-DETR v2, le configurazioni sono definite in file YAML.

A.1.1 Configurazione di training (RT-DETR R18/R50)

Tabella A.1: *Iperparametri di default per il training di RT-DETR v2 su COCO/Dataset proprietario.*

Parametro	**Valore**
Optimizer	AdamW
Learning Rate Base	1×10^{-4}
Weight Decay	1×10^{-4}
LR Scheduler	MultiStepLR (Milestone: 1000, Gamma: 0.1)
Warmup	Lineare (2000 step)
Batch Size	16 (Train) / 32 (Val)
Epochs	120 (R18/R34/R50-vd)
Input Size	640×640
Gradient Clipping	Norm 0.1
EMA (Exponential Moving Average)	$\alpha = 0.9999$

A.1.2 Comandi Operativi

Training da zero:

```
cd C:\Users\Iakta\Desktop\tesi\detection
python tools/train.py -c config/RTDETRV2.yml
```

- **Input:** file di configurazione YAML (definisce dataset paths, architettura, iperparametri).

- **Output:** checkpoint del modello (`.pth`), log di training, eventi TensorBoard nella cartella `output/`.

Abilitazione Augmentations: Per attivare le pipeline di augmentation custom (Albumentations), modificare la chiave `transforms` nel file YAML:

```
train_dataloader:
  dataset:
    transforms:
      ops: [RandomPhotometricDistort, RandomZoomOut,
         RandomIoUCrop]
```

Nota su Test Time Augmentation (TTA): Il framework supporta nativamente TTA (multiscala, flip). Sebbene non utilizzato negli esperimenti finali per preservare la latenza real-time, può essere attivato passando il flag `-use-tta` allo script di validazione.

A.2 Modulo recognition (CCT)

Il riconoscitore ottico (OCR) utilizza un Transformer convoluzionale compatto (CCT).

A.2.1 Configurazione di Training (CCT-XS/S)

Tabella A.2: *Iperparametri addestramento CCT.*

Parametro	Valore
Optimizer	AdamW
Learning Rate (Start)	1×10^{-4}
Learning Rate (End)	Factor 0.1 (Cosine Decay)
Weight Decay	1×10^{-5}
Batch Size	32
Epochs	500
Loss Function	Focal CCE ($\alpha = 0.25, \gamma = 2.0$)
Label Smoothing	0.05
Early Stopping	Patience 20 (su Val Loss)
Data Augmentation	Custom (Albumentations)

A.2.2 Comandi operativi

Training:

```
cd C:\Users\Iakta\Desktop\tesi\recognition
python training.py --config config/default_config.json
```

- **Input:** file di configurazione JSON, Dataset organizzato in cartelle (train/val).
- **Output:** modelli salvati (`.keras`), dump della configurazione utilizzata, log CSV nella cartella `experiment_N/`.

Inference / Eval:

```
python evaluation.py --model_path output/best_model.keras --
    dataset dataset/test
```

- **Input:** File pesi modello (`.keras`), Dataset di test.
- **Output:** Metriche di accuratezza (Character-level, Plate-level) e report degli errori.

Abilitazione Augmentations: Le augmentations sono definite nel file `config.json` sotto la chiave `"augmentation_params"`. Impostare `"use_aug": true` per attivare le distorsioni geometriche e fotometriche durante il training.

Nota su TTA: Lo script di inferenza supporta TTA (media delle probabilità su crop aumentati), attivabile con `-tta`. Tale modalità non è stata impiegata nei benchmark riportati nel Capitolo 5.

A.3 Modulo classification (MobileNetV3)

Il classificatore di nazionalità è un modello leggero basato su MobileNetV3.

A.3.1 Configurazione

Tabella A.3: *Iperparametri MobileNetV3 (Nation/Country Head).*

Parametro	Valore
Input Resolution	320×320
Batch Size	128
Epochs	500
Optimizer	AdamW
LR (Head Phase)	3×10^{-4}
LR (Finetune Phase)	2×10^{-5}
Unfreeze Layers	Ultimi 100 layer
Loss	Categorical CrossEntropy

A.3.2 Comandi operativi

Training:

```
cd C:\Users\Iakta\Desktop\tesi\nations
python training/train.py --config training/default_config/
   default_national_config.json
```

- **Input:** configurazione JSON, Dataset immagini crop (struttura a cartelle per classe).
- **Output:** modello addestrato (`country_classifier.keras`), mappa delle classi (`classes.json`), metriche di validazione.

Abilitazione Augmentations: Per il classificatore, le augmentations (RandomContrast, MotionBlur, GridDistortion) sono iniettate on-the-fly dal 'tf.data.Dataset'. Possono essere configurate o disabilitate modificando il dizionario `aug_config` in `train.py`.

Nota su TTA: Anche per il task di classificazione è possibile abilitare il Test Time Augmentation (es. 5-crop avg) per incrementare l'accuratezza su campioni rumorosi, al costo di un throughput ridotto ($N\times$). Negli esperimenti è stata utilizzata l'inferenza single-pass standard.

B. Specifiche Data Augmentation

In questa sezione riportiamo le configurazioni complete per le pipeline di data augmentation utilizzate nei tre moduli, includendo le probabilità di applicazione (p) per ogni trasformazione. I file originali sono disponibili nel repository del progetto.

B.1 Detection (Albumentations)

Configurazione estratta da `detection/config/RTDETRV2.yml`. Questa pipeline è applicata durante il training del detector RT-DETR.

```yaml
transform:
  __class_fullname__: Compose
  additional_targets: {}
  p: 1.0
  transforms:
    - {
        __class_fullname__: RandomSizedBBoxSafeCrop,
        height: 640,
        p: 0.12,
        width: 640,
      }
    - __class_fullname__: Affine
      p: 0.8
      rotate: [-8.0, 8.0]
      scale: {x: [0.95, 1.05], y: [0.95, 1.05]}
      shear: {x: [-4.0, 4.0], y: [-4.0, 4.0]}
      translate_percent: {x: [-0.05, 0.05], y: [-0.05, 0.05]}
    - __class_fullname__: RandomBrightnessContrast
      brightness_limit: [-0.2, 0.2]
      contrast_limit: [-0.2, 0.2]
      p: 0.6
    - __class_fullname__: OneOf
      p: 0.6
      transforms:
        - {__class_fullname__: GaussianBlur, p: 0.6}
        - {__class_fullname__: MotionBlur, p: 0.3}
        - {__class_fullname__: MedianBlur, p: 0.1}
    - __class_fullname__: OneOf
      p: 0.75
      transforms:
        - {__class_fullname__: GaussNoise, p: 0.6}
        - {__class_fullname__: ISONoise, p: 0.45}
        - {__class_fullname__: PixelDropout, p: 0.2}
    - {__class_fullname__: CLAHE, p: 0.6}
    - __class_fullname__: OneOf
      p: 0.8
      transforms:
        - {__class_fullname__: HueSaturationValue, p: 0.85}
        - {__class_fullname__: RGBShift, p: 0.25}
    - __class_fullname__: OneOf
      p: 0.7
      transforms:
        - {__class_fullname__: RandomGamma, p: 0.75}
        - {__class_fullname__: RandomToneCurve, p: 0.5}
    - __class_fullname__: OneOf
      p: 0.12
      transforms:
        - {__class_fullname__: RandomShadow, p: 0.25}
        - {__class_fullname__: RandomSunFlare, p: 0.12}
    - __class_fullname__: OneOf
      p: 0.15
      transforms:
        - {__class_fullname__: RandomRain, p: 0.35}
        - {__class_fullname__: RandomSnow, p: 0.25}
        - {__class_fullname__: RandomFog, p: 0.15}
```

```
    - __class_fullname__: OneOf
      p: 0.25
      transforms:
        - {__class_fullname__: OpticalDistortion, p: 0.25}
        - {__class_fullname__: GridDistortion, p: 0.12}
        - {__class_fullname__: ElasticTransform, p: 0.08}
    - __class_fullname__: OneOf
      p: 0.85
      transforms:
        - {__class_fullname__: ColorJitter, p: 0.95}
        - {__class_fullname__: Posterize, p: 0.8}
    - __class_fullname__: OneOf
      p: 0.7
      transforms:
        - {__class_fullname__: Sharpen, p: 0.95}
        - {__class_fullname__: UnsharpMask, p: 0.9}
    - {__class_fullname__: Downscale, p: 0.5}
    - {__class_fullname__: ToGray, p: 0.2}
    - {__class_fullname__: CoarseDropout, p: 0.45}
    - {__class_fullname__: core.dataset.WhitePatch, p: 0.4}
```

B.2 Recognition (Custom Augmentations)

Configurazione completa da `recognition/config/custom_augmentation.yaml`. Questa pipeline è ottimizzata per il riconoscimento testo con forte enfasi su blur e distorsioni prospettiche.

```
transform:
  __class_fullname__: Compose
  p: 1.0
  transforms:
  - __class_fullname__: Affine
    p: 0.6
    rotate: [-5.0, 5.0]
    scale: {x: [0.97, 1.03], y: [0.97, 1.03]}
    shear: {x: [-3.0, 3.0], y: [-3.0, 3.0]}
    translate_percent: {x: [-0.03, 0.03], y: [-0.03, 0.03]}
  - __class_fullname__: RandomBrightnessContrast
    brightness_limit: [-0.2, 0.2]
    contrast_limit: [-0.2, 0.2]
    p: 0.55
  - __class_fullname__: OneOf
    p: 0.45
    transforms:
    - {__class_fullname__: GaussianBlur, p: 0.5}
    - {__class_fullname__: MotionBlur, p: 0.4}
    - {__class_fullname__: MedianBlur, p: 0.1}
  - __class_fullname__: OneOf
    p: 0.4
    transforms:
    - {__class_fullname__: GaussNoise, p: 0.4}
    - {__class_fullname__: ISONoise, p: 0.3}
    - {__class_fullname__: PixelDropout, p: 0.3}
  - {__class_fullname__: CLAHE, p: 0.4}
  - __class_fullname__: OneOf
    p: 0.45
    transforms:
    - {__class_fullname__: HueSaturationValue, p: 0.7}
    - {__class_fullname__: RGBShift, p: 0.3}
  - __class_fullname__: OneOf
    p: 0.35
    transforms:
    - {__class_fullname__: RandomGamma, p: 0.7}
    - {__class_fullname__: RandomToneCurve, p: 0.3}
  - __class_fullname__: OneOf
    p: 0.18
    transforms:
    - {__class_fullname__: RandomShadow, p: 0.8}
    - {__class_fullname__: RandomSunFlare, p: 0.2}
  - __class_fullname__: OneOf
    p: 0.15
    transforms:
    - {__class_fullname__: RandomRain, p: 0.5}
    - {__class_fullname__: RandomSnow, p: 0.3}
    - {__class_fullname__: RandomFog, p: 0.2}
  - __class_fullname__: OneOf
    p: 0.25
    transforms:
    - {__class_fullname__: OpticalDistortion, p: 0.5}
    - {__class_fullname__: GridDistortion, p: 0.3}
    - {__class_fullname__: ElasticTransform, p: 0.2}
  - __class_fullname__: OneOf
    p: 0.25
    transforms:
    - {__class_fullname__: ColorJitter, p: 0.6}
    - {__class_fullname__: Posterize, p: 0.4}
```

```
  - __class_fullname__: OneOf
    p: 0.2
    transforms:
    - {__class_fullname__: Sharpen, p: 0.6}
    - {__class_fullname__: UnsharpMask, p: 0.4}
  - {__class_fullname__: Downscale, p: 0.12}
  - {__class_fullname__: ToGray, p: 0.08}
  - {__class_fullname__: CoarseDropout, p: 0.3}
  - {__class_fullname__: InvertImg, p: 0.3}
  - {__class_fullname__: ImageCompression, p: 0.1}
  - {__class_fullname__: RingingOvershoot, p: 0.1}
```

B.3 Classification (TF.Data)

Funzione di augmentation definita in `nations/training/dataset.py`. Queste trasformazioni sono applicate *on-the-fly* durante il caricamento dei dati.

```python
def augment_image(image, label):
    """
    Apply random augmentations.
    """
    # Max delta 0.2 -> range [-0.2, 0.2] additives
    image = tf.image.random_brightness(image, max_delta=0.2)
    # Contrast factor between 0.8 and 1.2
    image = tf.image.random_contrast(image, lower=0.8, upper=1.2)
    # Horizontal flip with 50% probability
    image = tf.image.random_flip_left_right(image)
    return image, label
```

Bibliografia

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] ankandrew. Fast-plate-ocr, 2023.

[3] baudm. Parseq: Scene text recognition with permuted autoregressive sequence models, 2022.

[4] Darwin Bautista and Rowel Atienza. Scene text recognition with permuted autoregressive sequence models. In *European Conference on Computer Vision*, pages 178–196. Springer, 2022.

[5] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[7] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *European Conference on Computer Vision*, pages 17–33. Springer, 2022.

[8] Dinghaoxuan. Samlp: Customized segment anything model for license plate detection, 2024.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[10] Yongkun Du, Zhineng Chen, Caiyan Jia, Xiaoting Yin, Tianlun Zheng, Chenxia Li, Yuning Du, and Yu-Gang Jiang. Svtr: Scene text recognition with a single visual model. In *IJCAI*, 2022.

[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[12] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[13] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva: Exploring the limits of masked visual representation learning at scale, 2023.

[14] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

[15] Guillermo Gallego, Tobi Delbnck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020.

[16] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. Ota: Optimal transport assignment for object detection. In *CVPR*, 2021.

[17] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[18] Global Growth Insights. Automatic license plate recognition: A comprehensive review of 2024 market and technology trends. *Market Report*, 2024.

[19] Haoyan Gong, Yuzheng Feng, Zhenrong Zhang, Xianxu Hou, Jingxin Liu, Siqi Huang, and Hongbin Liu. A dataset and model for realistic license plate deblurring. *arXiv preprint arXiv:2403.00000*, 2024.

[20] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. In *CVPR*, 2021.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in neural information processing systems*, volume 33, pages 6840–6851, 2020.

[23] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[24] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

[25] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[26] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[27] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[28] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8878–8887, 2019.

[29] Rayson Laroca, Luiz A Zanlorensi, Gabriel R Goncalves, Eduardo David, Rodrigo Minetto, and David Menotti. A robust real-time automatic license plate recognition based on the yolo detector. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2018.

[30] Minghao Li et al. Trocr: Transformer-based optical character recognition with pre-trained models, 2021.

[31] Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. Trocr: Transformer-based optical character recognition with pre-trained models. In *AAAI*, 2023.

[32] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[33] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[34] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[35] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[36] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[37] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

[38] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[39] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[40] Wenyu Lv et al. Detrs beat yolos on real-time object detection, 2023.

[41] Wenyu Lv et al. Rt-detr: Real-time detection transformer, 2023.

[42] Wenyu Lv et al. Rt-detr v2: Improved real-time detection transformer, 2024. See also rt_detr_2024.

[43] Wenyu Lv, Shangliang Xu, Yian Zhao, Guanzhong Wang, Jinman Wei, Cheng Cui, Yuning Du, Qingqing Dang, and Yi Liu. Detrs beat yolos on real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16965–16974, 2024.

[44] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[45] Andrew Mvd. Car plate detection, 2020. Accessed: 2025-01-26.

[46] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

[47] PaddlePaddle Team. Paddleocr, 2020.

[48] Roboflow. License plate object detection dataset, 2024. Accessed: 2025-01-26.

[49] Roboflow. Rf-detr: Rt-detr implementation by roboflow, 2024.

[50] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[51] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

[52] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.

[54] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. Yolov9: Learning what you want to learn using programmable gradient information. *arXiv preprint arXiv:2402.13616*, 2024.

[55] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *International Conference on Machine Learning*, pages 5502–5511. PMLR, 2018.

[56] Sergey Zherzdev and Alexey Gruzdev. Lprnet: License plate recognition via deep neural networks. *arXiv preprint arXiv:1806.10447*, 2018.

Ringraziamenti

Il grazie più profondo va alla mia famiglia e in primis ai miei genitori per essere stati il mio porto sicuro in questi anni, sorreggendomi nei momenti bassi e festeggiando con me in quelli alti e mi hanno seguito e dato fiducia lungo tutto il percorso dei 5 anni. Un ringraziamento pieno d'affetto è rivolto a tutti i miei zii, alle mie zie e alle mie cugine per la loro presenza costante e preziosa, con un pensiero speciale e profondo per la zia Lilly che ci ha lasciati lo scorso mese e che so essere in qualche modo qui a festeggiare con noi oggi. Naturalmente un abbraccio dolcissimo va anche al mio piccolo grande cucciolo Pluto, che in questi 13 anni è sempre stato al mio fianco regalandomi il suo affetto incondizionato.

Desidero poi ringraziare il Prof. Alessandro Fiori per la guida e il supporto in questo progetto, insieme a Iakta-Calit per avermi dato l'opportunità di svolgere il tirocinio e sviluppare questa tesi su una tematica così sfidante e attuale, oltre che per i fondamentali aperitivi condivisi. A questo proposito ringrazio in particolar modo Giulio, il mio referente aziendale, per la fiducia e il supporto e gli insegnamenti.

Grazie a chiunque mi abbia accompagnato lungo questi 5 anni e in particolare agli amici di Torino per aver salvato le mie sessioni nei posti studio con le nostre svagate pomeridiane, per gli spritz al Comala e per le infinite serate passate a giocare a poker, Risiko e tutti gli giochi e tipologie di serate, soprattutto alle Panche. Un pensiero va sopratutto agli amici di Parma che rappresentano la mia costante,grazie per ogni singola uscita (soprattutto nei momenti no), ringraziandovi per ogni singola uscita, per la serietà nello studiare insieme all'Università di Parma e non solo, e per esserci sempre stati tra pescate, briscole, tavoli da poker e le recenti serate al Primavera.

Infine vorrei ringraziare Torino e il Polito, un banco di prova sfidante e formativo che mi ha costantemente spinto a dare il massimo. Torino mi ha accolto quando ero solo un giovincello spaesato post maturità e mi ha cresciuto fino a farmi diventare un uomo, facendosi amare in tutti i suoi pregi e difetti. Ora mi sento pronto a lasciarla portando però nel cuore la sua socialità, umanità e la serenità della vita cittadina con tutti quegli eventi che mi hanno arricchito umanamente e culturalmente, trasformando questo momento non in un addio, ma augurandomi che questo sia soltanto un onesto arrivederci.