Importanza della struttura del dataset per la fase di fine-tuning di YOLOv8 per image segmentation

Cavallo	Mazza	Ravagli
Andrea	Pietro	Amedeo
s320122	s314897	s303852

Abstract

Nel progetto portato avanti, ci siamo posti come obiettivo lo studio dell'importanza del dataset scelto per la fase di fine-tuning di una rete neurale. Basandoci su un modello a stato dell'arte, YOLOv8[1], abbiamo ripetutamente eseguito la fase di fine-tuning, cercando di modificare il meno possibile i parametri della rete. Modificando quasi unicamente la struttura del dataset, abbiamo cercato di isolare e analizzare solamente gli effetti che la struttura, la composizione, e la dimensione del dataset hanno sulle performance della rete, dopo il suo addestramento. Partendo da un pool di immagini segmentate manualmente, e di immagini ottenute in modo sintetico, abbiamo generato una serie di dataset con una specifica struttura, e con questi abbiamo addestrato la rete.

1. Introduzione

Negli ultimi anni, le Reti Neurali Convolutive (CNN) sono emerse come uno strumento potente per svariati compiti di visione artificiale, tra cui il rilevamento di oggetti, la classificazione delle immagini e la segmentazione semantica. Tra questi, il modello You Only Look Once (YOLO) ha attirato notevole attenzione, grazie alle sue impressionanti capacità di rilevamento degli oggetti in tempo reale. Tuttavia, le risorse richieste per addestrare da zero questo tipo di modelli non sono ancora alla portata di tutti. Per questo, il fine-tuning rappresenta uno strumento molto importante per adattare i modelli, già addestrati, a utilizzi anche molto specifici e lontani dal contesto originale.

Il processo di fine-tuning di una rete neurale pre-addestrata, come YOLO, comporta la regolazione dei parametri del modello, attraverso l'addestramento eseguito con un nuovo specifico dataset, che porta ad un miglioramento delle prestazioni del modello su compiti, o domini, peculiari. Questo processo di adattamento è influenzato da diversi fattori critici relativi al dataset, tra cui la dimensione (mole di dati), la struttura (organizzazione ed etichettatura dei dati), e la composizione (varietà e bilanciamento delle classi). Ciascuno di questi fattori gioca un ruolo fondamentale nel determinare l'efficacia del processo di fine-tuning e l'accuratezza del modello risultante.

Ispirandoci ad una possibile applicazione reale, abbiamo deciso di utilizzare, come ambito di specializzazione della nostra rete, il rilevamento e la segmentazione di loghi, appartenenti a marchi commerciali, in un'ottica che vede questa rete inserita in un workflow di identificazione, e rimozione, dei loghi presenti nelle immagini, in maniera automatica, tramite Image In painting.

Il nostro lavoro si è incentrato sullo studio dell'impatto del dataset, e per questo abbiamo limitato le modifiche agli iper-parametri della rete (dropout, batch size, ottimizzatore, etc.) in fase di addestramento, così da poter isolare e accentuare al massimo gli effetti delle diverse strutture dei dataset.

Figura 1: immagine generica di loghi

Le caratteristiche su cui ci siamo concentrati sono: la dimensione, la struttura e la composizione del dataset.

- **Dimensione**: si riferisce al numero di immagini presenti per ogni logo;
- **Struttura**: fa riferimento al rapporto tra immagini segmentate manualmente e immagini create sinteticamente;
- **Composizione**: data la presenza di una sola classe, questa caratteristica è legata al numero di loghi diversi, presenti all'interno del dataset.

2. Data

Per addestrare la rete, abbiamo fatto uso di immagini ottenute tramite scraping sul motore di ricerca Google Immagini, e di immagini presenti nel dataset BelgaLogos[2], un dataset pubblico, contenente immagini

[1] Ultralytics: https://www.ultralytics.com/it

[2] BelgaLogos: https://www-sop.inria.fr/members/Alexis.Joly/BelgaLogos/BelgaLogos.html

reali che raffigurano dei loghi in contesti reali. Il dataset contiene 10000 immagini e 26 loghi diversi.

Per la raccolta di immagini, abbiamo collezionato dalle 20 alle 30 immagini per ognuno dei 161 loghi, ottenendo quindi un totale di circa 4300 immagini reali. Il lavoro è stato svolto manualmente, selezionando le immagini trovate online che meglio rispettavano i criteri che abbiamo deciso di adottare:

- Il logo deve essere inserito in un contesto reale e deve essere di dimensioni ragionevoli; quindi, è necessario che sia riconoscibile (è imperativo che sia di dimensioni sufficienti, onde evitare che diventi un "ammasso di pixel"), ma non eccessivamente grande da diventare l'elemento preponderante dell'immagine;
- Le immagini selezionate devono essere di una risoluzione accettabile anche in funzione della dimensione del logo presente;
- Le immagini devono contenere unicamente istanze di un solo marchio, e abbiamo cercato di selezionare immagini che presentassero i loghi in posizioni e rotazioni diverse.

Il logo può, all'interno della stessa immagine, essere presente molteplici volte.

Per quanto riguarda l'annotazione, dopo iniziali tentativi di usare software liberi, abbiamo deciso di usare la versione a pagamento di CVAT[3](Figura 2) anche per la possibilità di usare un tool che, basandosi su modelli neurali, velocizzava la fase di segmentazione. In particolare, il tool in questione usava la rete neurale SAM[4] che, dopo la selezione di alcuni key-point, positivi e/o negativi, sull'immagine, segmentava l'oggetto d'interesse.

Figura 2: utilizzo del programma CVAT per labeling manuale delle immagini

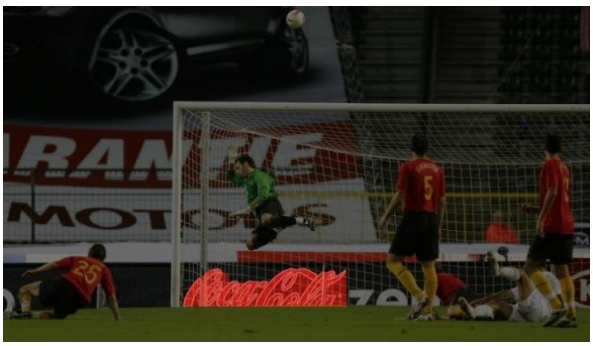
Infine, abbiamo scaricato i dataset segmentati come delle maschere binarie e, successivamente, abbiamo eseguito uno script (tutti gli script sono disponibili su GitHub[5]) che opera una conversione da maschera binaria a file testuale, conforme alla struttura delle annotazioni definita da COCO[6].

Un terzo dei loghi utilizzati sono stati presi da BelgaLogos. Abbiamo preso 30 immagini per ciascuno 16 dei loghi scelti, per un totale di 480 immagini; tra tutte le immagini disponibili sono state scelte quelle in cui il logo fosse abbastanza grande da essere visibile ad occhio nudo. Abbiamo poi creato le maschere per la segmentazione, perché il dataset forniva i bounding-box di ogni logo. Il processo di segmentazione manuale è stato lo stesso usato per le immagini ricavate da Google Immagini.

Oltre alle immagini reali, abbiamo espanso il nostro dataset con più di 87000 immagini generate sinteticamente, incollando l'immagine del logo in primo piano sopra a delle immagini di sfondo. Le 331 immagini di sfondo sono state scaricate da Google Immagini e raffigurano:

Figura 3: esempio di immagine sintetica con la sua maschera binaria

paesaggi naturali, paesaggi urbani e facciate di edifici. I file png che contengono i loghi sono 247 e rappresentano 161 loghi diversi. I .png usati per i loghi sono ad alta risoluzione e presentano uno sfondo trasparente. Le immagini sintetiche sono state create posizionando il logo sull'immagine di sfondo in modo casuale, dopo averlo scalato e ruotato nei tre assi. Lo script implementato controlla anche che, in ogni immagine generata, il logo sia interamente visibile. Per la generazione delle maschere abbiamo creato un'immagine completamente nera, avente la stessa dimensione dell'immagine di sfondo, su cui è stata posizionata la maschera bianca relativa all'immagine del logo, dopo aver applicato le stesse trasformazioni di rotazione e scalamento applicate al logo nell'immagine di partenza.



[3] CVAT: https://www.cvat.ai/
[4] SAM: https://segment-anything.com/
[5] GitHub del progetto: https://github.com/P-ietro/Importanza-della-struttura-del-dataset-per-la-fase-di-fine-tuning-di-YOLOv8-per-image-semgentation.git
[6] COCO: https://cocodataset.org/

Figura 4: maschera binaria generata con GIMP

Le maschere dei singoli loghi sono state generate utilizzando GIMP[7].

Infine, per adattare i dati alla rete YOLO, abbiamo lasciato le immagini inalterate, mentre le maschere sono state trasformate dal formato .png a quello testuale nel quale, in modo conforme al formato richiesto da YOLO, i pixel associati ad un logo sono definiti dall'insieme di punti che descrivono il poligono contenente il logo stesso.

2.1. Come sono stati creati i diversi dataset?

Per la creazione dei singoli database, siamo partiti da un database contenente le informazioni relative a tutte le immagini collezionate. Una volta definita una struttura, abbiamo selezionato, in modo casuale, *L* loghi diversi e, per ciascuno di essi, abbiamo selezionato *R* immagini reali e *S* immagini. Una volta raccolte tutte le immagini, e le relative maschere, abbiamo diviso questo insieme di partenza in due sottogruppi, uno per il training ed uno per la validazione. Abbiamo assegnato, ad ognuno dei due gruppi (sintetiche e reali), il 10% delle immagini al dataset di validazione e il restante 90% a quello di training. La mole di dati con cui abbiamo addestrato la rete dipende, quindi, dalla struttura dei dataset che abbiamo usato.

Per quanto riguarda la realizzazione dei vari test, non è stata fatta alcuna differenziazione tra questi in base ai loghi che contenevano, proprio perché il nostro interesse era quello di trattare tutti i loghi come una classe unica. Il problema che può figurare, nell'utilizzare questo approccio, è che non si sa con certezza se ci siano dei loghi che vengono riconosciuti meglio rispetto ad altri.

3. Metodi

3.1. YOLOv8

La scelta di utilizzare YOLOv8 (Figura 4) deriva dal fatto che è una rete utilizzata per elaborazioni in tempo reale, come pensiamo possa essere utilizzato il nostro modello per la segmentazione dei loghi.

L'implementazione della rete che abbiamo utilizzato è quella fornita da Ultralytics. La configurazione utilizzata è composta da:

- Batch size: 64
- Epoche: 60
- Ottimizzatore: Adamw con momento 0.937
- Dropout: 0.5
- Il learning rate decrementale da 0.01 a 0.00005
- Weight decay: 0.0005
- Loss function della rete[8] è la somma pesata delle tre loss function calcolate dal modello: BCE Loss (Binary Cross Entropy), DFL (Distribution Focused Loss), CIoU (Complete Intersection over Union).

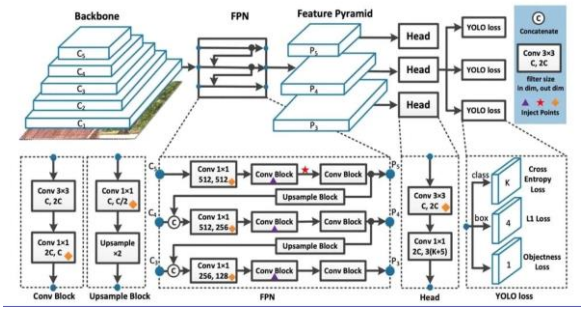


Figura 5: backbone della rete YOLOv8

3.2. Training della rete

Uno dei problemi riscontrati è proprio quello della generazione di immagini sintetiche per la creazione del dataset. Un primo approccio proposto è stato quello di creare un ambiente virtuale tridimensionale, quindi, attraverso un programma di modellazione 3D, posizionare i loghi in un'ambientazione tridimensionale e fare dei render fotorealistici. Questa strada, però, non è stata seguita, a causa dell'estrema complessità realizzativa per la costruzione di un ambiente fotorealistico, e anche per la potenza computazionale necessariamente elevata richiesta per eseguire i render. La seconda idea, quella utilizzata, si basa sui paper [1] e [2], che creano i dataset a partire da immagini dei singoli loghi, a sfondo trasparente, e immagini di background raffiguranti contesti reali. Le immagini dei loghi sono state incollate in primo piano sulle immagini di sfondo, dopo aver applicato una serie di trasformazioni spaziali. Per le maschere siamo partiti da sagome, completamente bianche, dei loghi originali, e dopo aver applicato le stesse trasformazioni spaziali, le abbiamo incollate su immagine su uno sfondo nero in modo da ottenere una maschera binaria.

È stato deciso di seguire questa strada poiché i risultati riportati sui paper di riferimento erano promettenti.

Per valutare l'impatto che la struttura e la dimensione di un dataset hanno sulle performance di una rete, già addestrata, abbiamo deciso di creare diversi dataset, con cui fare fine-tuning della rete, per poi analizzare i risultati ottenuti sul set di validazione.

In particolare, i parametri dei dataset su cui abbiamo lavorato sono: il numero di istanze di loghi diversi

[7] GIMP: https://www.gimp.org/

[8] Formula Loss utilizzata da Ultralytics: https://github.com/ultralytics/ultralytics/issues/3882

presenti *L*, il numero di immagini sintetiche per ogni logo *S* e il numero di immagini reali per ogni logo *R*. Abbiamo deciso di analizzare sei combinazione per ogni valore di *L*: *300S_0R, 280S_20R, 100S_0R, 95S_5R, 90S_10R, 80S_20R*, e abbiamo usato quattro valori di *L*: 10, 50, 100, 155, per un totale, quindi, di 24 esperimenti.

Il numero di immagini per ciascun dataset, quindi, dipende dalla struttura, e varia tra 1000(*L=10 e S+R=100) e 46500(L=155 e S+R=300*).

Data la grande variabilità che i dataset più piccoli possono presentare, sia per via dell'eterogeneità di loghi presenti, che per la diversità di immagini scelte per ogni logo, abbiamo deciso di eseguire più addestramenti, mantenendo fissa la struttura del dataset (ma diversa composizione), e poi di calcolare la media dei risultati, in modo da ottenere dei valori, sperabilmente, più indicativi. Per tutti gli esperimenti con meno di 15000 immagini (tutti quelli con 10 loghi e poi quelli con 50 loghi e 100 immagini per logo) abbiamo eseguito tre addestramenti, mentre per gli esperimenti con 15000 immagini abbiamo eseguito 2 addestramenti (50 loghi e 300 immagini per logo). Infine, per tutti gli altri, abbiamo eseguito un solo esperimento, e questo perché il numero di loghi e di immagini utilizzati ci è sembrato tale da essere considerato indicativo del comportamento generale.

Per quanto riguarda la realizzazione dei vari test, non è stata fatta alcuna differenziazione tra questi in base ai loghi che contenevano, questo perché, innanzitutto, seppur i loghi fossero molteplici, la classe che li andasse a identificare era una, ovvero la classe 'logo', e anche perché ciò che si voleva mettere in evidenza era proprio come i loghi stessi potessero influenzare le prestazioni, e quindi anche i risultati ottenuti, della rete considerata.

4. Risultati

Nella fase di training della rete ogni esperimento è stato eseguito su un dataset di training composto da immagini reali e sintetiche.

Per la fase di test le reti allenate sono state testate usando due diversi dataset. Il primo dataset è composto da, al più, 10 immagini reali (non utilizzate in fase di training) per logo, le quali appartengono a loghi visti in fase di addestramento. Nel secondo dataset le immagini reali sono 20 per logo appartenenti a 50 loghi non utilizzati durante l'addestramento. Con queste due configurazioni possiamo analizzare con quale precisione la rete riesce a riconoscere loghi già visti e loghi mai visti in precedenza.

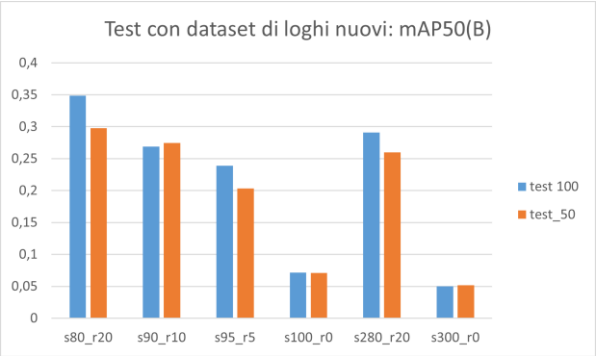


Figura 66: confronto dei risultati di mAP(50) ottenuti in fase di testing eseguiti su loghi mai visti, tra reti addestrate con 50 e 100 loghi

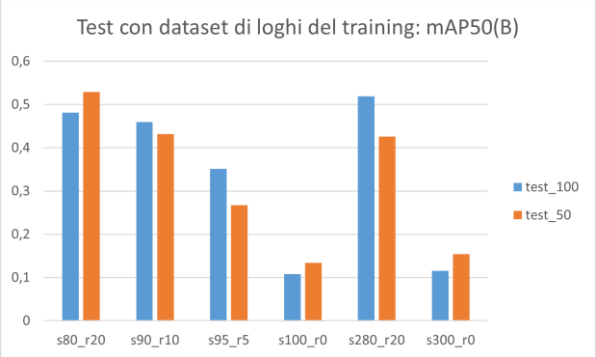


Figura 77: confronto dei risultati di mAP(50) ottenuti in fase di testing eseguito su loghi presenti nel training set

Abbiamo deciso di utilizzare, come metrica per la valutazione degli addestramenti della rete, la mean Average Precision, con una soglia del 50 per cento sul valore di IoU (Intersection over Union). Dai risultati Dei test, fatti con i dataset contenenti loghi mai visti (Figura 6), possiamo dire che:

- l'utilizzo di immagini reali per l'addestramento della rete è determinante per avere delle prestazioni accettabili. Si può infatti notare una differenza di prestazioni, anche notevole, tra i casi in cui sono state usate delle immagini reali nel dataset, e quelli che, al loro interno, presentavano solo immagini sintetiche.
- Questa tendenza è giustificata dalla necessità, che ha la rete, di conoscere il contesto nel quale è inserito un logo, e quindi va anche a mettere in evidenza come il logo, di per sé, non sia sufficiente alla rete per poter imparare meglio a riconoscere nuovi loghi.
- La presenza di più loghi nel dataset di training permette alla rete di generalizzare meglio il concetto di logo.
- i risultati sottolineano un andamento non lineare crescente tra numero di immagini reali e performance della rete: più immagini reali ci sono e più alta è la mAP.

Dai risultati dei test con il dataset contenente gli stessi loghi del training (figura 7), possiamo dire che:

- anche in questo caso la rete migliora le sue prestazioni in modo non lineare, aumentando il numero di immagini reali per ogni logo, e che utilizzando dataset

puramente sintetici si ottengono prestazioni estremamente basse.

- Le immagini sintetiche sono utili per migliorare il riconoscimento dei loghi indipendentemente dal contesto; quindi, avere più immagini dello stesso logo anche sintetiche come si può vedere dal caso S100 R0 e S300 R0 anche se di poco S300 è migliore.

Le figure 8 e 9 ci mostrano dei risultati concreti dei test eseguiti con 100 loghi, 280 immagini sintetiche e 20 reali per logo, con le prestazioni migliori.

In figura 8 viene mostrato il test eseguito su loghi mai visti e i loghi sono stati riconosciuti bene (seppur con confidenze basse), mentre nel caso della figura 9, che mostra il test sui loghi già presenti in fase di training. Tutti i loghi vengono riconosciuti correttamente e con una confidenza leggermente più alta.

Figura 8: esempio di inferenza su immagini con loghi non presenti nel training set. La rete è allenata con 100 loghi, 280 immagini sintetiche e 20 reali

Figura 9: esempio di inferenza su immagini con loghi presenti nel training set. La rete è allenata con 100 loghi, 280 immagini reali e 20 sintetiche

5. Conclusioni

In conclusione, abbiamo verificato le ipotesi che avevamo proposto in partenza, cioè che la presenza di immagini reali sia importante per ottenere delle prestazioni accettabili in termini di mAP, e abbiamo dimostrato che, anche per il compito di segmentazione, è possibile aumentare le dimensioni di un dataset utilizzando delle immagini sintetiche. In particolare, l'utilizzo di immagini sintetiche rinforza l'apprendimento specifico di determinati loghi, indipendentemente dal contesto in cui sono inseriti. Abbiamo verificato, anche, che la rete può generalizzare i loghi visti e riconoscerne di nuovi, e di come la presenza di immagini reali sia importante per questo task, in quanto la rete riconosce un nuovo logo dal suo contesto.

In generale possiamo concludere che la qualità dei dati sia una caratteristica più importante in fase di fine-tuning che non la mole di dati. Nei nostri risultati i miglioramenti che abbiamo ottenuto raddoppiando il numero di loghi sono molto inferiori dei miglioramenti misurati all'aumentare del rapporto tra immagini reali e sintetiche. Quindi utilizzare dei buoni dati reali che rappresentano bene il contesto applicativo finale della rete è molto importante.

Possibili sviluppi futuri sono:

- costruire un dataset in cui sia presente una classe per ogni logo, così da verificare se esistono loghi che la rete riconosce meglio di altri, andando a confrontare i loro corrispettivi valori di mAP.
- Provare ad aumentare ulteriormente il rapporto tra immagini reali e sintetiche sia riducendo il numero di immagini sintetiche, sia aumentando il numero di immagini reali.

Riferimenti

[1] Christian Eggert, Anton Winschel, and Rainer Lienhart. 2015. On the Benefit of Synthetic Data for Company Logo Detection. In Proceedings of the 23rd ACM international conference on Multimedia (MM '15). Association for Computing Machinery, New York, NY, USA, 1283–1286. https://doi.org/10.1145/2733373.2806407.

[2] H. Su, X. Zhu and S. Gong, "Deep Learning Logo Detection with Data Expansion by Synthesising Context," 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 2017, pp. 530-539, doi: 10.1109/WACV.2