

UNIVERSITÀ DEGLI STUDI DI
MILANO-BICOCCA

ADVANCED MACHINE LEARNING
FINAL PROJECT

Classificazione multi-label su VOC2012

Autori:

Alessandro Gilardi - 866035 - a.gilardi7@campus.unimib.it
Alessandro Preziosa - 866142 - a.preziosa1@campus.unimib.it
Mattia Volpato - 866316 - m.volpato4@campus.unimib.it

February 8, 2025



Abstract

In questo lavoro sono stati sviluppati una serie di modelli di deep learning per la classificazione di immagini appartenenti alla **Pascal Visual Object Challenge 2012 (VOC2012)**, contenente immagini rappresentanti uno o più oggetti tra persone, animali, veicoli e oggetti casalinghi. L'approccio adottato consiste nell'addestramento da zero di una CNN, nell'esplorazione del paradigma self-supervised learning e nell'adattamento di modelli neurali profondi pre-addestrati quali ResNet50, VGG16 e MobileNet. Il codice sorgente prodotto è disponibile su GitHub.

1 Introduzione

Il riconoscimento di oggetti in immagini è un problema cruciale nel campo della visione artificiale, con applicazioni che spaziano dalla guida autonoma alla sorveglianza, fino alla robotica. Questo progetto affronta il problema del riconoscimento di oggetti utilizzando il dataset **Pascal Visual Object Challenge 2012 (VOC2012)**, un benchmark ampiamente utilizzato nella comunità scientifica per la valutazione delle performance dei modelli di visione artificiale.

2 Datasets

In questo progetto, è stato utilizzato il dataset **Pascal Visual Object Challenge 2012 (VOC2012)** per l'addestramento e la valutazione dei modelli, composto da 27.675 immagini contenenti oggetti appartenenti alle seguenti venti classi: *aeroplano, bicicletta, uccello, barca, bottiglia, autobus, auto, gatto, sedia, mucca, tavolo da pranzo, cane, cavallo, moto, persona, pianta in vaso, pecora, divano, treno e monitor TV*.

Si tratta di un problema di **classificazione multi-label**, in cui ogni immagine può contenere una o più etichette.

Il dataset VOC 2012 è suddiviso in due sottoinsiemi: training e test. Il dataset di training comprende 11.540 immagini etichettate con gli oggetti presenti. Il dataset di test, invece, contiene 16.135 immagini e non include etichette, rendendolo utile esclusivamente per la valutazione finale sulla piattaforma ufficiale di valutazione. Durante questo progetto, abbiamo utilizzato esclusivamente il dataset di training per addestrare e validare i modelli.

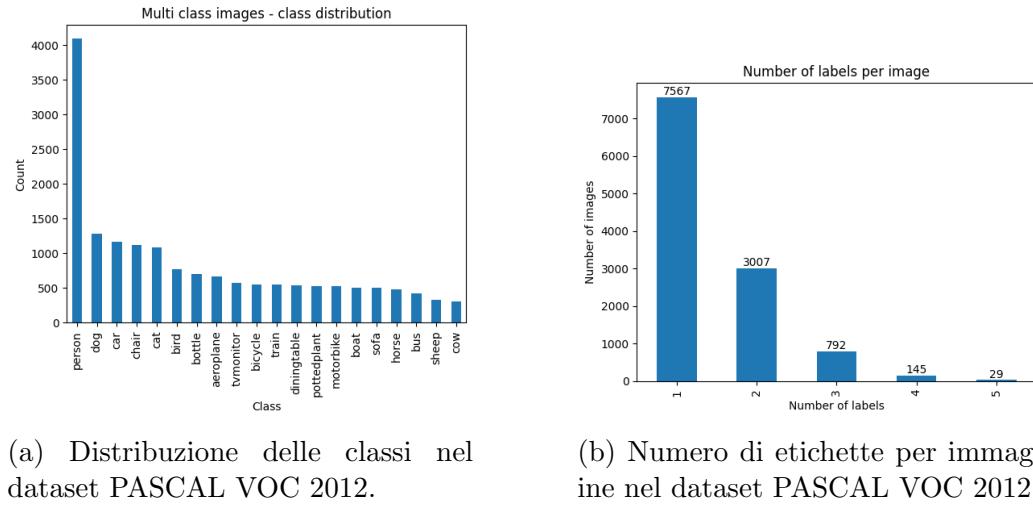


Figure 1: Esempi di immagini presenti nel dataset PASCAL VOC 2012.

2.1 Distribuzione delle classi

I dati di training presentano una distribuzione delle classi sbilanciata (Figura 2a): la categoria *persona* risulta nettamente più frequente rispetto alle altre. A tal riguardo, non sono state effettuate operazioni di ridimensionamento del dataset per il bilanciamento delle classi.

La challenge non specifica il numero massimo di classi che può contenere ogni immagine; tuttavia, l’analisi del training set evidenzia la presenza di un massimo di cinque classi, con più del 90% delle istanze contenenti un massimo di due classi.



(a) Distribuzione delle classi nel dataset PASCAL VOC 2012.

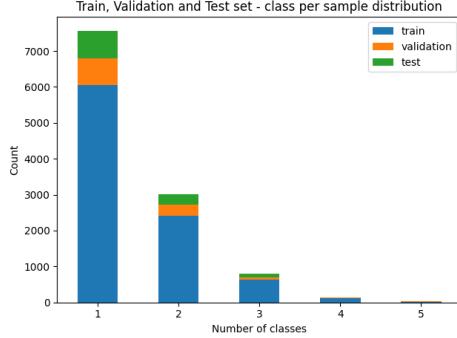
(b) Numero di etichette per immagine nel dataset PASCAL VOC 2012.

2.2 Creazione dei set di Training, Validation e Test

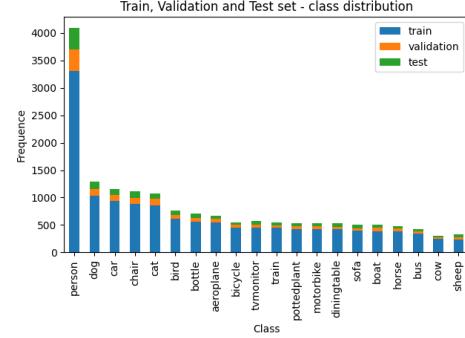
Il dataset è stato suddiviso in tre split: training, validation e test.

La suddivisione è stata effettuata in modo casuale, con il rapporto di 80% di dati per il training, 10% per la validation e 10% per il test. La dis-

distribuzione delle classi, rappresentata nella Figura 2a, risulta essere bilanciata per i diversi split.



(a) Distribuzione del numero di classi per istanza nei diversi split.



(b) Distribuzione delle classi nei diversi split.

2.3 Data Augmentation

In alcuni esperimenti i dati di training sono stati soggetti a *data augmentation* al fine di regolarizzare l’addestramento del modello fornendo versioni modificate dei dati originalmente disponibili. Le trasformazioni applicate sono state:

- Rotazione casuale fino a ± 25 gradi;
- Traslazione orizzontale fino al 20% della larghezza dell’immagine;
- Traslazione verticale fino al 20% dell’altezza dell’immagine;
- Trasformazione prospettica (shear) di $i = \text{rand}(-0.25, 0.25)$ gradi;
- Zoom centrato, di dimensione casuale tra il 70% (zoom-in) e il 120% (zoom-out) delle dimansioni originali;
- Aggiunta, per ogni canale colore, di una quantità $i = \text{rand}(-0.3 * \text{maxImg}, 0.3 * \text{maxImg})$ con maxImg il valore più alto di R, G o B tra tutti i pixel nell’immagine;
- Flip orizzontale con probabilità del 30%.



(a) Immagini originali



(b) Immagini *augmented*.

3 Approccio Metodologico

Come primo approccio si è optato per l’addestramento di una rete neurale convoluzionale da zero (da qui **modello baseline**), impiegata come *baseline* di riferimento per le successive valutazioni.

Seguendo il paradigma *Self-Supervised*, la stessa architettura di rete è stata impiegata nel risolvere diversi *pretext tasks*, come la classificazione del grado di rotazione delle immagini. Dopo di che è stata applicata la tecnica della *network surgery*, seguita dal *fine-tuning* sul task originale utilizzando i dati non modificati di VOC2012.

Infine, sono state selezionate diverse CNN pre-addestrate sul dataset ImageNet, alle quali sono state applicate tecniche di *fine-tuning* al fine di adattarle al problema di interesse.

3.1 Modello baseline

L’architettura del **modello baseline** consiste in una rete convoluzionale composta da sette blocchi principali, ciascuno formato da¹

- uno strato convoluzionale;
- uno strato di normalizzazione;
- una funzione di attivazione *ReLU*;
- uno strato di *max pooling*;

Successivamente, vengono utilizzati:

- uno strato di appiattimento (*flattened*);

¹Il numero di unità dello *strato convoluzionale* e la presenza dello strato di *max pooling* varia da blocco a blocco: consultare la Sezione ‘*Base model*’ del codice sorgente per ulteriori dettagli.

- un singolo strato *fully connected* con attivazione *ReLU*;
- uno strato finale con
 - venti neuroni, pari al numero delle classi;
 - attivazione *sigmoidea*;
 per effettuare la classificazione multi-etichetta.

per un totale di 337.620 parametri. La stessa architettura è stata utilizzata per indurre due modelli diversi:

- *baseline_model*: addestrato sulle immagini originali del dataset senza modifiche;
- *baseline_augmented_model*: addestrato sulle immagini del dataset su cui è stata applicata la **data augmentation** come descritto nella Sezione 2.3.

3.2 Self Supervised Models

Sono stati condotti diversi esperimenti di *Self Supervised Learning* (SSL), utilizzando come pretext task la classificazione del grado di rotazione delle immagini, o una combinazione di questo task con la classificazione del grado di sfocatura. L'uso di questi task nel contesto del SSL consente alla rete di apprendere caratteristiche semantiche rilevanti anche per il downstream task della classificazione delle immagini nelle categorie di VOC2012.

Il successivo *fine-tuning* della rete è stato eseguito con dati reali o *augmented*, a seconda degli esperimenti.

Nel pretext task della classificazione del **grado di rotazione**, ogni immagine di training e validation è stata ruotata in 4 orientamenti: 0° (nessun cambiamento), 90°, 180°, e 270°, generando così 4 immagini a partire dalla stessa.

Quando questo task è stato combinato con la **sfocatura**, il pretext task è diventato quello di classificare sia il grado di rotazione che il grado di sfocatura applicata alle immagini. In questo caso, le immagini sono state prima ruotate in 4 orientamenti (come descritto precedentemente) e poi applicato un filtro di media con dimensioni del kernel $k \times k$, dove k è un valore casuale compreso nell'intervallo [0, 4].

L'idea di combinare pretext task differenti per apprendere caratteristiche più efficaci rispetto a quelle che si apprendono singolarmente da ciascuno di

essi deriva da Yamaguchi et al [1]. Nell’articolo, gli autori esplorano l’idea di un pretext task consistente nel classificare i parametri delle trasformazioni di *Image Enhancement* e rotazione, trovando che blur/sharpening e solarization, insieme alla rotazione, sono le trasformazioni che permettono alla rete di apprendere le caratteristiche più rilevanti.

Gli esperimenti da noi condotti sono stati:

1. Classificazione del grado di rotazione, *fine-tuning* con solo dati reali
2. Classificazione del grado di rotazione, *fine-tuning* con dati aumentati
3. Classificazione del grado di rotazione e sfocatura, *fine-tuning* con solo dati reali

Come verrà mostrato nella Sezione 4, il *fine-tuning* del modello con dati aumentati non ha prodotto miglioramenti rispetto al *fine-tuning* con immagini reali. Alla luce di questi risultati, e considerando che il tempo di addestramento nel caso del *fine-tuning augmented* era molto maggiore, abbiamo deciso di abbandonare questa strada in favore del *fine-tuning* con dati reali. Successivamente è stato cambiato il task e i dati somministrati al modello, che ora sono le immagini di VOC2012 ruotate e sfocate con un kernel di dimensione casuale. Il task diventa quello di classificare il grado di rotazione e sfocatura delle immagini. I risultati ottenuti non sono migliori del caso in cui si classificava la sola rotazione, che quindi risulta l’approccio di SSL migliore di quelli esplorati.

3.3 Transfer Learning

Sono state selezionate tre architetture pre-addestrate ampiamente utilizzate in letteratura: ResNet50[2], VGG16[3] e MobileNet[4]. La scelta di questi modelli è motivata dalle loro caratteristiche:

- **VGG16** è stato scelto per la sua semplicità architetturale e la sua profondità, nonostante l’elevata complessità computazionale dovuta all’elevato numero di parametri.
- **ResNet50** è stato selezionato per il suo elevato grado di profondità, reso possibile dalle *residual connections* che mitigano il problema della scomparsa del gradiente.

- **MobileNet** è stato scelto per la sua efficienza computazionale, che lo rende particolarmente adatto a dispositivi con risorse computazionali limitate.

Modello	Size (MB)	Acc. Top-1	Acc. Top-5	N° Param.	Depth	CPU Time (ms)	GPU Time (ms)
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4

Table 1: Confronto delle performance dei modelli su ImageNet.[5]

I modelli² sono stati raffinati mediante *fine-tuning*, utilizzando la *binary cross-entropy loss*. L’ottimizzazione è stata eseguita con l’algoritmo *RM-Sprop*, adottando un *learning rate* pari a 0.001. Per mitigare il rischio di overfitting, sono state impiegate tecniche di regolarizzazione come *Early Stopping* e *Dropout*.

3.3.1 VGG16

Il modello VGG16 pre-addestrato su ImageNet è stato inizialmente utilizzato congelando tutti gli strati convoluzionali e addestrando solo i nuovi strati aggiunti. Al fine di replicare la struttura originale, sono stati inseriti due strati densi con 128 neuroni ciascuno, entrambi con attivazione ReLU, e uno strato completamente connesso con 20 neuroni, con attivazione sigmoidea per adattare il modello alla classificazione multi-etichetta. Questo modello è stato denominato *base_vgg*.

Come secondo esperimento, si è mantenuta la stessa architettura di *base_vgg*, rendendo addestrabili gli strati appartenenti all’ultimo blocco convoluzionale. Questo modello è stato denominato *ft_vgg*.

3.3.2 ResNet50

Il modello ResNet50 pre-addestrato su ImageNet è stato modificato rimuovendo lo strato finale e congelando i pesi dei layers convoluzionali. Sono stati aggiunti uno strato di *average pooling* e uno strato completamente connesso

²Consultare il codice sorgente per l’architettura completa.

con 20 neuroni, con attivazione sigmoidea per adattare il modello alla classificazione multi-etichetta. Questo modello è stato denominato *base_resnet*.

Come secondo esperimento, si è mantenuta la stessa architettura di *base_resnet*, rendendo addestrabili gli strati appartenenti all’ultimo blocco convoluzionale. Questo modello è stato denominato *ft_resnet*.

3.3.3 MobileNet

Il modello MobileNet pre-addestrato su ImageNet è stato inizialmente modificato rimuovendo gli strati finali e congelando tutti gli strati convoluzionali. Successivamente, è stato definito un nuovo modello sequenziale, composto da un livello di *average pooling* seguito da un livello di *Dropout*. È stato quindi aggiunto uno strato convoluzionale con 20 filtri e kernel di dimensione (1,1). L’output è stato poi rimodellato con un livello *Reshape* per ottenere un vettore di 20 elementi, seguito da una funzione di attivazione sigmoidea. Questo modello è stato denominato *base_mbnet*.

Modello fine-tuned	N° Param.	Modello base	Fine-tuning
base_vgg	138.4M	VGG16	Strati convoluzionali bloccati
ft_vgg	138.4M	VGG16	Ultimo blocco convoluzionale addestrabile
base_resnet	23.6M	ResNet50	Strati convoluzionali bloccati
ft_resnet	23.6M	ResNet50	Ultimo blocco convoluzionale addestrabile
base_mbnet	4.2M	MobileNet	Strati convoluzionali bloccati

Table 2: Sommario modelli fine-tuned.

4 Risultati Ottenuti e Valutazioni

La valutazione dei modelli è stata condotta in due fasi: durante l’addestramento, attraverso l’analisi delle curve relative alle metriche calcolate sul set di validazione e, successivamente, tramite le performance calcolate sul set di test.

4.1 Soglia decisionale

Il problema di **classificazione multi-label** non limita il numero di classi che possono appartenere a una data istanza: si rende quindi necessario stabilire una soglia decisionale per stabilire la presenza di una specifica classe a partire dal valore predetto dal modello, limitato nell'intervallo $[0, 1]$. A tal fine, la soglia di decisione è stata calcolata con l'obiettivo di massimizzare la metrica **F1-score**: come valore, si è scelto quello che andasse a massimizzare tale metrica considerando la curva *Precision-Recall* sul set di validazione. Questa soglia è stata poi utilizzata nella fase di test per la valutazione finale dei diversi modelli.

4.2 Metriche di valutazione

Si assume la seguente notazione:

- N : numero di esempi del dataset;
- K : numero di classi.
- y_{ij} : etichetta reale (0 o 1) per la classe j del campione i ;
- \hat{y}_{ij} : etichetta predetta (0 o 1) per la classe j del campione i ;
- y_i : etichette reali (0 o 1) per tutte le classi del campione i ;
- \hat{y}_i : etichette predetta (0 o 1) per tutte le classi del campione i ;
- \mathbb{I} : funzione indicatrice.

Per valutare le prestazioni dei modelli sul test set, sono state calcolate le seguenti metriche:

- **Binary Accuracy**: calcola con quale frequenza le predizioni delle singole classi corrispondono alle etichette binarie.

$$\text{Binary Accuracy} = \frac{1}{N \cdot K} \sum_{i=1}^N \sum_{j=1}^K \mathbb{I}(\hat{y}_{ij} = y_{ij}) \quad (1)$$

- **Perfect Accuracy**: definita come la percentuale di campioni in cui tutte le classi sono state predette correttamente.

$$\text{Perfect Accuracy} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\hat{y}_i = y_i) \quad (2)$$

- **Mean Average Precision (mAP)**: calcola l'area sotto la curva Precision-Recall per ogni classe e restituisce la media sulle classi.
- **F1 Score**: utile per valutare il modello in caso di dati sbilanciati, come per il nostro problema. E' stata utilizzata la versione pesata, che tiene conto del diverso numero di istanze per classe.
- **Precision e Recall**.

4.3 Modello baseline

Nella Figura 5 vengono riportati i risultati ottenuti con i modelli **baseline** sul test set.

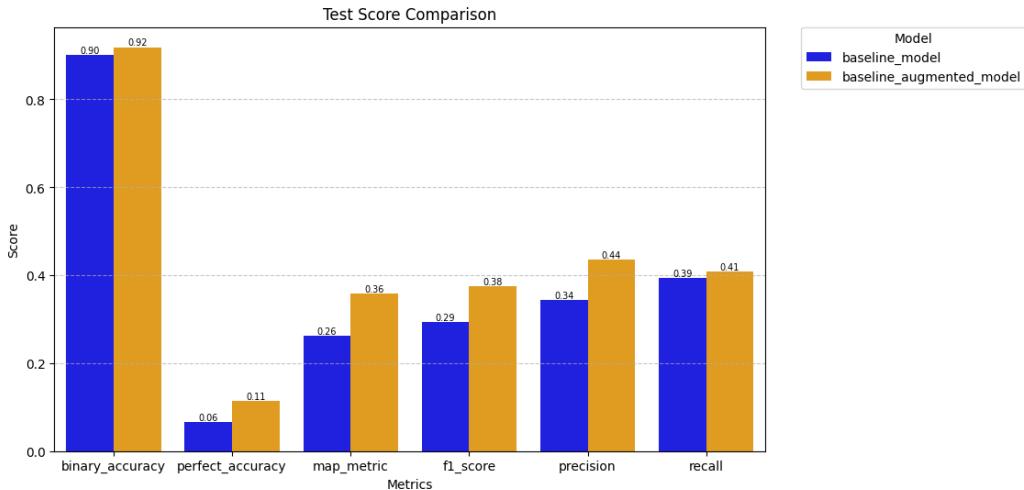


Figure 5: Metriche di valutazione ottenute con i *modelli baseline*.

4.4 Self Supervised Models

Nella Figura 6 vengono riportati i risultati ottenuti con i **modelli SSL** sul test set.

4.5 Transfer Learning

Nella Figura 7 vengono riportati i risultati ottenuti con i **modelli fine-tuned** sul test set.

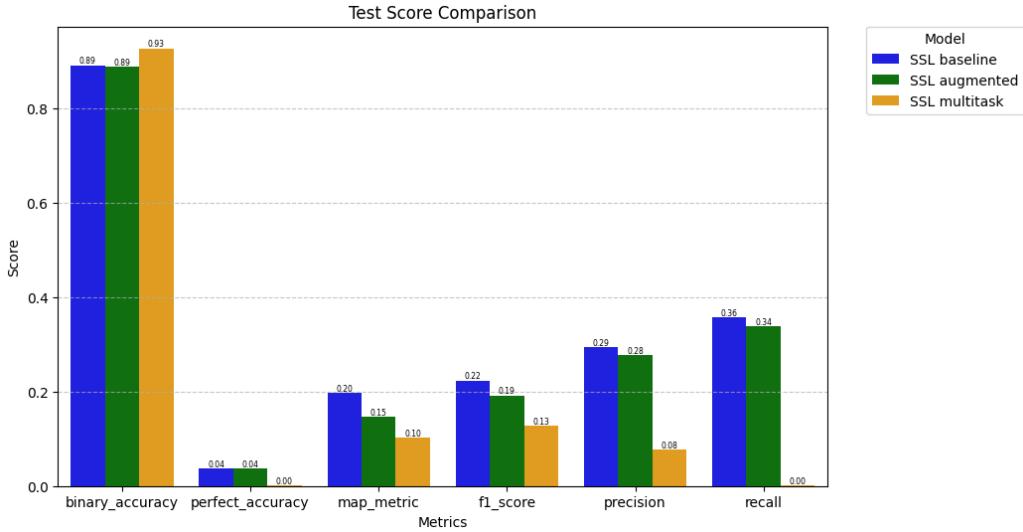


Figure 6: Metriche di valutazione ottenute con i modelli *SSL*.

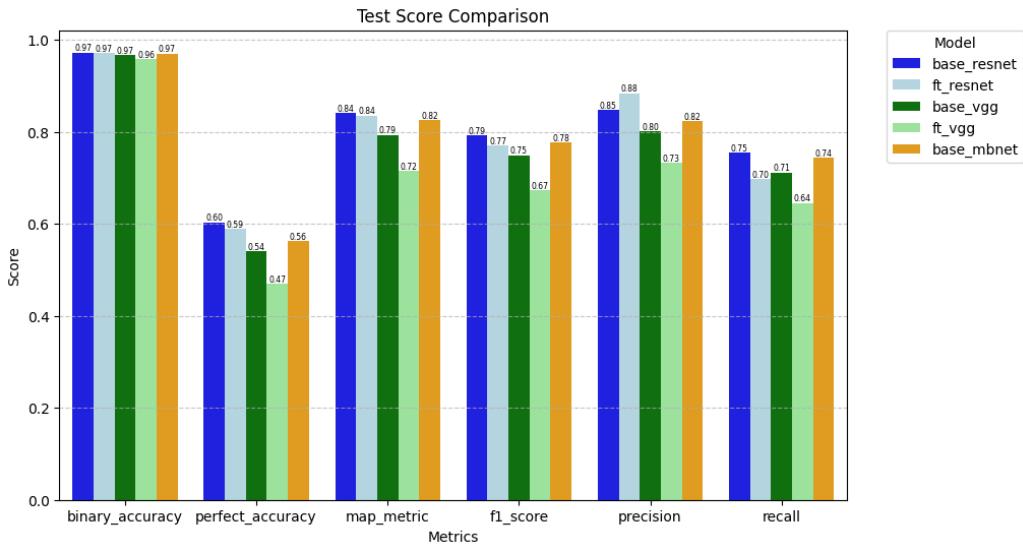


Figure 7: Metriche di valutazione ottenute con i modelli *fine-tuned*.

5 Discussione

5.1 Modello baseline

Pur avendo una profondità limitata rispetto a modelli più conosciuti, il modello di base riesce ad ottenere dei buoni risultati rispetto al problema proposto.

L'uso della *data augmentation* migliora in maniera evidente le prestazioni del modello, producendo in generale un miglioramento in tutte le metriche. Questo risultato è dovuto alla maggiore diversità nei dati di addestramento, che rende il modello più robusto alle variazioni nei dati di test: utilizzando ulteriori dati di addestramento ci si potrebbe aspettare un ulteriore miglioramento, senza necessariamente andare ad aumentare la capacità del modello.

5.2 Modelli Self-Supervised

Osservando le performance del modello di SSL baseline (Figura 6), si nota come le metriche di mAP, F1, precision e recall siano dimezzate rispetto alla stessa architettura addestrata in modo supervisionato. Sempre dalla stessa Figura si evince che le iterazioni successive del modello SSL baseline hanno avuto performance leggermente inferiori (modello augmented) o molto inferiori (modello SSL multitask) rispetto alla baseline SSL.

I risultati del modello augmented confrontati a quelli del modello SSL baseline suggeriscono che l'aumentatation non è stata un introduzione benefica.

In generale, gli esperimenti di SSL hanno prodotto modelli dalle performance molto minori di quelli ottenuti da un allenamento del modello partendo da parametri casuali. L'unica altra differenza tra SSL baseline e CNN baseline, oltre ai parametri del modello prima di addestrarsi sui dati di VOC, sta nel numero di epoch di addestramento, che nel caso degli esperimenti SSL è di molto inferiore al caso dell'addestramento da zero. Il numero inferiore di epoch di addestramento è una possibile spiegazione delle performance generalmente inferiori dei modelli SSL.

5.3 Transfer Learning

I risultati mostrano che il modello *base_resnet* ottiene le migliori prestazioni complessive, con la più alta binary accuracy e dei buoni mAP e F1-score, suggerendo un buon equilibrio tra precisione e recall. Il fine-tuning dell'ultimo

blocco convoluzionale in *ft_resnet*, tuttavia, non ha portato a miglioramenti significativi, bensì a una leggera diminuzione delle metriche. Questo potrebbe essere dovuto a un fenomeno di overfitting in cui il modello, adattandosi troppo ai dati di addestramento, perde la capacità di generalizzare efficacemente su nuovi dati. I modelli *base_vgg* e *ft_vgg* seguono un trend simile, mostrando però prestazioni inferiori rispetto ai modelli basati su ResNet. Questo suggerisce che l’architettura di VGG sia meno adatta per la classificazione multi-label su questo specifico dataset. Infine, MobileNet si dimostra competitivo nonostante la capacità inferiore, mostrando prestazioni simili a quelle di ResNet in termini di binary accuracy e mAP, ma con una perfect accuracy inferiore.

In generale, i modelli basati su ResNet offrono le migliori prestazioni globali, mentre i modelli basati su VGG risultano meno efficaci nel contesto della classificazione multi-label, suggerendo che la complessità di ResNet sia più adatta a questo tipo di compito.

6 Conclusioni

Il modello *baseline_augmented_model* dimostra buone prestazioni rispetto alla sua capacità limitata, confermando l’efficacia della *data augmentation* nel migliorare la generalizzazione.

I modelli addestrati tramite SSL non hanno portato a risultati significativi, riportando risultati inferiori rispetto al caso della stessa architettura addestrata in modo supervisionato. Inoltre, l’uso della *data augmentation* non ha migliorato le prestazioni, così come l’uso di un pretext task combinato.

In conclusione, confrontando il primo modello addestrato da zero con quelli ottenuti tramite SSL è stato possibile evidenziare le limitazioni di tale approccio nel raggiungere risultati significativi. In seguito, l’utilizzo del *transfer learning* con modelli pre-addestrati ha invece consentito di ottenere performance di alto livello sul problema trattato.

References

- [1] S. Yamaguchi, S. Kanai, T. Shioda, and S. Takeda, “Image enhanced rotation prediction for self-supervised learning,” in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 489–493.
- [2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [3] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [4] A. G. Howard, “Mobilennets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [5] Keras, “Keras application,” n.d., accessed: 2025-04-02. [Online]. Available: <https://keras.io/api/applications>