UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

ADVANCED MACHINE LEARNING

FINAL PROJECT

Classificazione automatica di 102 specie di fiori

Teresa Cigna – 813925 - t.cigna@campus.unimib.it

Chiara Di Domenico – 815463 – c.didomenico@campus.unimib.it

Gennaio 2021

Abstract

Questo progetto si pone l'obiettivo di presentare e valutare diversi approcci per il riconoscimento automatico di 102 specie di fiori. L'obiettivo è quello di identificare una combinazione di features che, unita ad un modello efficace ed efficiente, sia in grado di distinguere fiori tra di loro molto simili e difficilmente distinguibili ad un occhio non esperto. A questo scopo si sono utilizzati diversi approcci: utilizzo di *transfer learning* sia attraverso il *fine-tuning* di reti con un basso numero di parametri (*MobileNetV2*, *EfficientNetB0* ed *EfficientNetB2*), sia attraverso l'utilizzo delle suddette come features extractor per l'estrazione dei dati da utilizzare come input di un classificatore *SVM* (Support Vector Machine).

1 Introduzione

Ad oggi la scienza riconosce circa 391'000 specie di piante vascolari, di cui circa 369'000 fiorite. Risulta quindi utile e di particolare interesse il riconoscimento automatico di fiori, per poter riuscire a distinguere tipologie all'apparenza molto simili tra loro. Si è deciso di perseguire questo obiettivo utilizzando features estratte manualmente e facendo uso di *transfer learning* attraverso l'impiego di reti con un basso numero di parametri come *EfficientNet* e *MobileNet* precedentemente allenate su *Imagenet*. In particolare, le suddette reti son state utilizzate come *features extractor* e come reti base da cui partire per effettuare *fine-tuning*.

2 Dataset

I dati presi in considerazione provengono dall' *Oxford's 102 Categories Flowers Dataset.* Si tratta di un dataset contenente 8189 fiori divisi in 102 categorie. La divisione originale prevedeva 6149 dati di test, 1020 dati di train e 1020 di validation; per questo progetto si è deciso di invertire train e test in quanto si è riscontrato che circa 10 immagini per categoria fossero poche per poter addestrare al meglio qualsiasi modello, soprattutto nel caso di specie molto simili tra loro. Insieme al suddetto dataset, ne veniva fornito uno contenente immagini di fiori segmentate tramite lo schema di *Nilsbackand Zisserman*. Per quanto questo metodo funzionasse bene in molti casi, in altri restituiva un'immagine totalmente vuota. Non avendo a disposizione un numero elevato di immagini per ogni classe, si è considerato non ottimale utilizzare questo metodo. Si è quindi deciso di utilizzare un metodo che si basasse sulla rimozione del colore verde, in quanto costituiva la maggior parte degli sfondi. A questo scopo si è approssimato ogni colore dell'immagine al valore *RGB* limite più simile

(i.e. (0,0,255), (0, 255,255), (255,0,0) …), ponendo, per ogni pixel e per ogni canale, una soglia oltre la quale il valore veniva sostituito dal valore massimo (255), e al di sotto della quale veniva sostituito dal valore minimo (0). (*Figura 1*)

Figura 1. Esempio di approssimazione al colore RGB limite più vicino

Questa operazione è stata effettuata per neutralizzare il più possibile sfumature di verde diverse, date anche dalla diversa luminosità tra le immagini. A questo punto si è eliminato il verde (0,255,0). Questo metodo risultava efficace in casi in cui il metodo proposto falliva, ma non dava risultati ottimali in quanto i fiori in primo piano venivano identificati abbastanza bene, ma lo sfondo presentava del rumore. Si è deciso quindi di utilizzare un secondo approccio che si basasse sul metodo *Grab Cut*: un metodo iterativo che, dopo aver calcolato la distribuzione di colore del *foreground* e del *background*, determina se ogni pixel appartiene allo sfondo o al soggetto in primo piano. Questo metodo si è rivelato molto più efficace nonostante si sia notato avere un comportamento singolare nella parte inferiore di alcune immagini; pertanto si è deciso di applicare questo metodo sia all'immagine originale, sia all'immagine stessa ruotata di 180°. Dopo aver riportato quest'ultima all'orientamento iniziale, le due immagini sono state moltiplicate pixel per pixel al fine di ottenere la sola zona di sovrapposizione. *(Figura 2)*

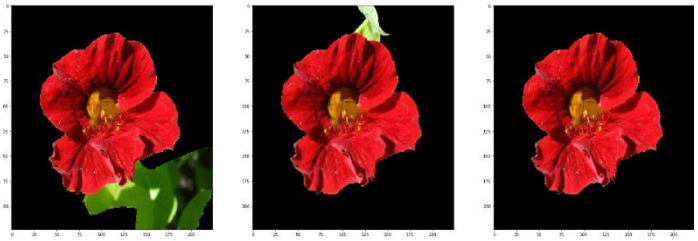

Figura 2. Da sinistra: GrabCut su immagine originale; GrabCut su immagine ruotata di 180° e riportata all'orientamento iniziale; Risultato della sovrapposizione

Questo tipo di approccio è risultato ottimale, anche se per qualche immagine con molti fiori e poco sfondo o con sfondo di un colore simile al colore del fiore, è stata ottenuta un'immagine vuota. In quei casi si è scelto di effettuare una sostituzione utilizzando il metodo della rimozione del colore verde e, nel caso in cui anche quell'immagine fosse risultata non soddisfacente, si è proceduto sostituendola con l'immagine precedente ruotata di 90°. Questa scelta è stata fatta per non diminuire ulteriormente i dati a disposizione, ma, con la rotazione dell'immagine, si è evitato di avere due

immagini perfettamente identiche che avrebbero potuto falsare i risultati in fase di addestramento. Per il resto del progetto è stato quindi utilizzato il dataset formato dalle immagini segmentate.

Di seguito è possibile vedere le differenze fra i tre approcci con le criticità riscontrate e descritte precedentemente. *(Figura 3)*

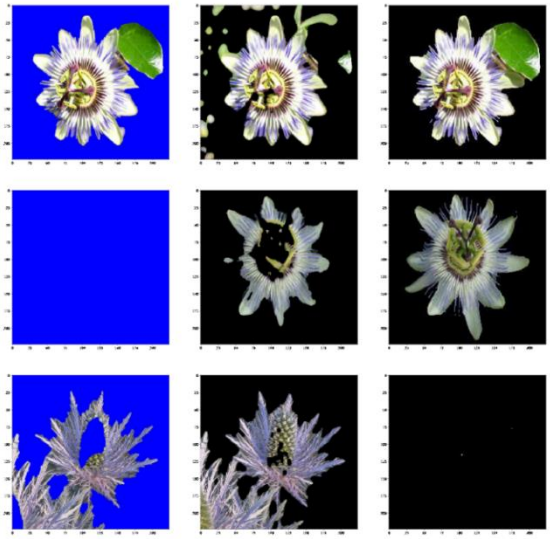

Figura 3. Colonna 1: approccio Nilsbackand Zisserman; Colonna2: approccio rimozione colore verde; Colonna3: approccio GrabCut

3 Approccio metodologico

Per questo progetto si è deciso di utilizzare diversi approcci: utilizzo di specifiche features estratte con funzioni costruite ad hoc da utilizzare come input per una *SVM*, utilizzo di *transfer learning* attraverso *fine-tuning* e utilizzo delle reti pre-addestrate come *features extractor* per estrarre features da dare in input ad una *SVM* sia singolarmente, sia in combinazione con quelle estratte dal primo approccio.

3.1 Hand Crafted Features

Si sono inizialmente definite le caratteristiche che rendono un fiore distinguibile ad occhio umano: il colore, la forma, la trama/texture dei petali, particolari punti distintivi. Si è quindi costruita: una funzione che, data l'immagine segmentata, la trasformasse nel modo indicato in *Figura 1* e successivamente estraesse le percentuali di blu, rosso e verde; una funzione che performasse un *LBP* (Local Binary Pattern) con raggio uguale a 5 e numero di vicini pari a 8 in modo che fosse meno granulare, evitando quindi di identificare trame e texture non d'interesse; una funzione che performasse l'estrazione di keypoints tramite *SURF* (Speeded-Up Robust Features) e *SIFT* (Scale Invariant Features Transform). In quest'ultima ultima funzione, la grossa diseguaglianza di dimensionalità tra i risultati estratti dalle diverse immagini ha reso difficoltoso il loro utilizzo. Si è pensato ad una riduzione di dimensionalità, ad esempio, attraverso la *PCA* (Principal Component

Analysis), ma non è stata scelta questa applicazione perché non compatibile con l'obiettivo di poter classificare una nuova immagine singola, in quanto non ci sarebbero stati abbastanza dati per poter applicare una *PCA*. Si è deciso pertanto di utilizzare una funzione che estraesse gli *HoG* (Histograms of Gradients) con 8 direzioni in quanto simile a *SIFT* come concetto sottostante, ma con il vantaggio di avere una dimensionalità di rappresentazione fissa. Si è scelto di addestrare una *SVM* con diverse combinazioni delle sole features precedentemente descritte.

3.2 Transfer Learning

Avendo una mole di dati di medie dimensioni, si è scelto poi di fare uso di *transfer learning*, sia tramite *fine-tuning*, sia tramite l'utilizzo delle reti come *features extractor*. Le reti scelte per questo task sono state *MobileNetV2, EfficientNetB0, EfficientNetB2* addestrate su *Imagenet*. Questa scelta è stata fatta in quanto queste reti raggiungono buone performance seppur con un basso numero di parametri, rendendole quindi efficienti.

3.2.1. Fine-Tuning

In tutti e tre i casi si è scelto di tagliare la rete all'ultimo layer che precede il *layer fully connected*, in quanto il dataset utilizzato risulta abbastanza simile ad *Imagenet*, essendo immagini a colori, acquisite in un ambiente naturale e con soggetti presenti anche all'interno di *Imagenet* stesso. Inizialmente, si è deciso di aggiungere solo un *layer fully connected* che mappasse i risultati delle reti pre-trainate, a 102 classi. Tuttavia, questo metodo non è risultato adatto in quanto si è presentato un fenomeno di forte overfitting, pertanto si è deciso di aggiungere un layer di dropout con rapporto pari a 0.5.

```
x = base_net.output

x = keras.layers.Dropout(0.5) (x)

pred = keras.layers.Dense(102, activation='softmax')(x)
```

Figura 4. Rete iniziale

Si è poi scelto di procedere aggiungendo a quest'ultima rete un layer con la *ReLu* come funzione di attivazione, e un *layer fully connected* con 128 unità. La scelta di aggiungere questi due layer è stata fatta per dar modo alla nuova rete di addestrarsi meglio prima di arrivare all'output, ma allo stesso tempo si è deciso di non aggiungerne troppi in quanto ciò avrebbe potuto portare ad avere una rete troppo profonda e quindi meno efficiente.

```
x = base_net.output

x = keras.layers.BatchNormalization()(x)
x = keras.layers.ReLU()(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Dense(128, activation = 'relu', kernel_regularizer=regularizers.l2(0.0005))(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Dropout(0.5)(x)

pred = keras.layers.Dense(102, activation='softmax')(x)
```

Figura 5. Rete completa

È stato inoltre scelto di effettuare data augmentation performando rotazione, zoom e shift orizzontale e verticale. Come funzione di perdita è stata scelta la *categorical cross-entropy* e come ottimizzatore è stato utilizzato *Adam* (Adaptive moment estimation) che è un metodo *Stochastic Gradient Descent* che si basa sulla stima dei momenti del primo e del secondo ordine. È computazionalmente efficiente e richiede poca memoria. La rete è stata addestrata per 30 epoche in quanto si è notato che le performance iniziavano ad assestarsi attorno a quella cifra.

3.2.2 CNN come features extractor

Le reti pre-addestrate, tagliate nel medesimo punto scelto per il *fine-tuning*, sono state utilizzate come estrattori di features da utilizzare come input per una *SVM,* sia da sole, sia in combinazione con le *hand crafted features*. È stata scelta la gestione delle classi di tipo bilanciato e un kernel polinomiale di grado 2. La particolarità di questo tipo di kernel è quella di non considerare solo le features di input, ma anche combinazioni delle stesse per calcolare la similarità.

Per scegliere i giusti parametri di *C* e *gamma*, ossia il parametro che determina la forza della regolarizzazione (inversamente proporzionale a C) e il coefficiente del kernel, inizialmente è stato utilizzato un metodo *GridSearchCV* che, data una griglia di valori e un numero di folds per la *cross-validation*, trovasse i valori ottimali, tuttavia questo metodo non è risultato corretto in quanto i parametri scelti portavano ad un forte overfitting (accuracy sul train pari a 1), pertanto, a seconda delle features utilizzate come input, si sono scelti empiricamente i valori di *C* e *gamma*, tenendo conto che più *C* è basso, più la regolarizzazione è forte.

4 Risultati e valutazione

Di seguito son mostrati i modelli con i risultati più rilevanti che evidenziano le differenze tra i vari approcci:

Tabella 1. Risultati ottenuti dai diversi modelli

Approccio	*Features utilizzate*	*Train Accuracy*	*Test Accuracy*
SVM con hand-crafted features	*colori+HoG*	0.30	0.28
	colori+HoG+LBP	0.15	0.12
Fine-tuning	*EfficientNetB2 - Rete completa*	0.87	0.85
	MobileNetV2 - Rete completa	0.85	0.79
	EfficientNetB0 - Rete completa	0.89	0.90
	EfficientNetB0 - Rete iniziale	0.93	0.91
SVM con CNN features + hand-crafted features	*EfficientNetB0*	0.95	0.89
	EfficientNetB0+colori+HoG	0.95	0.84
	EfficientNetB0+colori+HoG+LBP	0.20	0.17

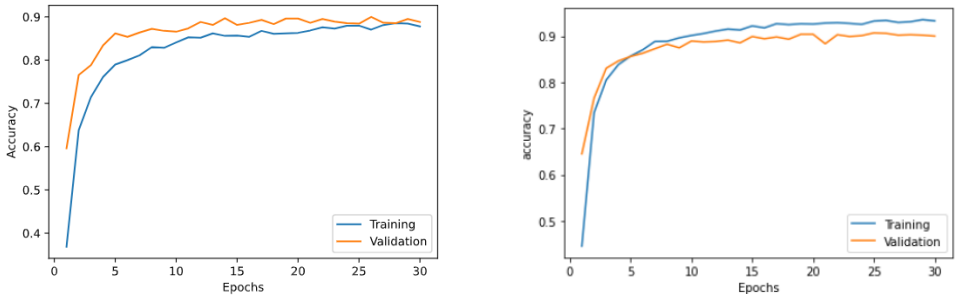

Figura 4. Accuracy di train e validation per ogni epoca dell'addestramento. A sinistra, EfficientNetB0 - Rete completa. A destra, EfficientNetB0 - Rete iniziale.

5 Discussione

Tra le reti pre-addestrate utilizzate, quella che ha portato ai risultati migliori è stata *EfficientNetB0*, pertanto per il terzo approccio son stati riportati solo i risultati ottenuti con quest'ultima. Quello che possiamo notare dai risultati è che l'*SVM* che utilizzava come input le features estratte da *EfficientNetB0*, combinate con colore e *Histogram of Gradients,* ha performance simili a quella che utilizzava solo le features estratte dalle relativa rete. Risulta quindi evidente che le features estratte manualmente, che corrispondono a caratteristiche importanti per il riconoscimento dei fiori ad un occhio umano, non migliorano i risultati ma non li peggiorano nemmeno di molto, perciò si possono considerare informazioni ridondanti già contenute all'interno delle caratteristiche con un alto livello di astrazione estratte dalla rete. Si nota invece che il *Local Binary Pattern*, utilizzato in questo modo,

non è di giovamento a nessun modello, anzi peggiora i risultati. L'*SVM* che utilizzava le features estratte da *EfficientNetB0* produce buoni risultati sul test, ma presenta un fenomeno di overfitting. In *Figura 6* è possibile vedere le performance dei due modelli risultati migliori, con l'approccio *fine-tuning.* È possibile notare che entrambi raggiungono performance simili sul test (0.90, 0.91 rispettivamente), tuttavia il modello *'EfficientNetB0 - Rete iniziale'* mostra un fenomeno di overfitting, seppur non pronunciatissimo. Pertanto si ritiene che, tra i 2, sia più opportuno utilizzare *'EfficientNetB0 - Rete completa'.* I modelli migliori, per i tipi di approcci utilizzati, risultano quindi essere l'*SVM* che utilizza le features estratte da *EfficientNetB0 e 'EfficientNetB0 - Rete completa'*. In *Figura 7* è possibile notare che entrambi riescono ad ottenere ottimi risultati di discriminazione tra fiori molto simili tra loro.

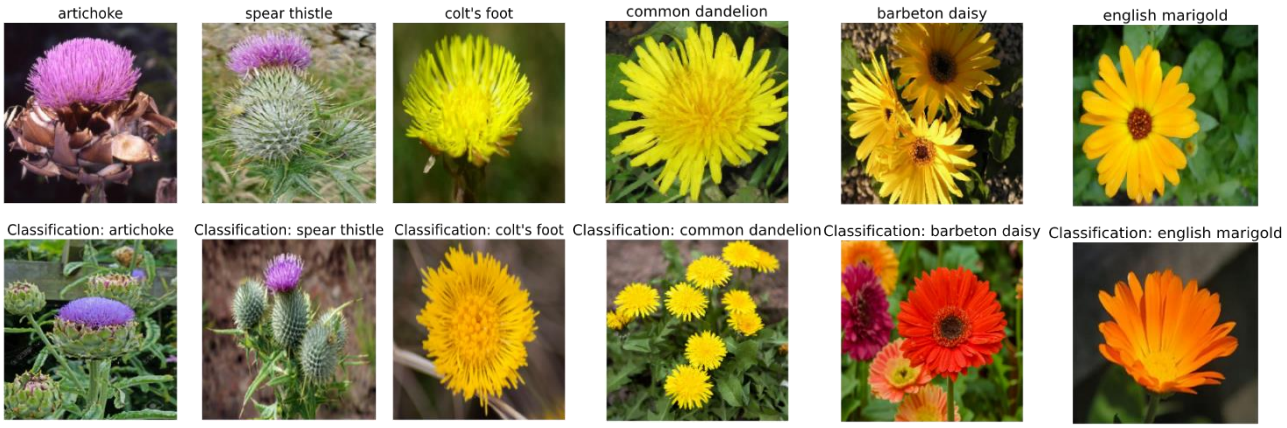

Figura 5. In alto: le immagini originali da classificare. In basso: la classificazione ottenuta da entrambi i modelli migliori. (Le immagini in basso sono immagini esemplificative della categoria predetta)

Visti i deludenti risultati ottenuti dall'utilizzo di *LBP*, che addirittura portava ad un peggioramento dei modelli con configurazioni risultate vincenti, uno sviluppo futuro potrebbe riguardare la ricerca dei parametri ottimali da utilizzare (numero di punti e raggio), in quanto si ritiene che un'informazione sulla trama dei petali possa dare un contributo interessante.

6 Conclusioni

In questo progetto è risultato di fondamentale importanza l'utilizzo di un buon sistema di segmentazione che sia in grado di isolare il soggetto in primo piano e che permetta di utilizzare tutte le immagini a disposizione riducendo la possibilità di ottenere immagini vuote. È emerso poi il fatto che il solo utilizzo delle features relative a colore e *Histogram of Gradients* non è utile alla classificazione. Inoltre, è emerso che l'utilizzo di *transfer learning* risulta efficace sia utilizzando le reti come *features extractor*, sia facendo *fine-tuning*. In particolare *EfficientNetB0* risulta la rete pre-addestrata più performante tra quelle utilizzate.

7 Bibliografia

[1] Maria-Elena Nilsback, Andrew Zisserman. Automated flower classification over a large number of classes, https://www.robots.ox.ac.uk/~vgg/publications/2008/Nilsback08/nilsback08.pdf

[2] M.-E. Nilsback and A. Zisserman. Delving into the whorl of flower segmentation

[3] ArIES, IIT Roorkee. Texture Analysis using LBP, https://medium.com/@ariesiitr/texture-analysis-using-lbp-e61e87a9056d

[4] Zhenhua Guo, Lei Zhang, and David Zhan. A Completed Modeling of Local Binary Pattern Operator for Texture Classification, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5427137

[5] Adrian Rosebrock. OpenCV GrabCut: Foreground Segmentation and Extraction, https://www.pyimagesearch.com/2020/07/27/opencv-grabcut-foreground-segmentation-and-extraction/

[6] Matthijs Hollemans. MobileNet version 2, https://machinethink.net/blog/mobilenet-v2/

[7] Mingxing Tan, Quoc V.Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, https://arxiv.org/pdf/1905.11946.pdf

[8] Diederik P.Kingma, Jimmy Lei Ba. Adam: A Method For Stochastic Optimization, https://arxiv.org/pdf/1412.6980.pdf