Università degli studi di
Milano-Bicocca

ADVANCED MACHINE LEARNING

Progetto Finale

Reti Neurali Convoluzionali: classificazione di frutta e verdura

Autore:
Dario Della Mura - 793751- d.dellamura@campus.unimib.it

15 giugno 2022

Indice

Sommario

La classificazione di oggetti, ad oggi, è uno dei task più significativi il cui sviluppo è in costante crescita per quanto riguarda il campo di ricerca del deep learning. L'obbiettivo di questo studio è lo sviluppo di architetture neurali per la classificazione delle immagini (di frutta e verdura) contenute all'interno del dataset Fruits-360. L'approccio metodologico adottato per affrontare il task in questione è stato lo sviluppo di tre modelli di reti neurali convoluzionali ottenuti applicando tre differenti tecniche tipiche del deep learning: sviluppo di una rete neurale convoluzionale ex novo, utilizzo di una rete neurale convoluzionale pre-addestrata mediante la tecnica del transfer learning e utilizzo di una rete neurale convoluzionale pre-addestrata come estrattore di caratteristiche mediante un approccio di features extraction.

1 Introduzione

Oggi si sente sempre più parlare d'intelligenza artificiale. Ogni giorno le più grandi società High-Tech , Google - Amazon - Microsoft, pubblicano nuovi brevetti/strumenti capaci di alleggerire e facilitare via via in misura crescente la nostra vita privata e lavorativa: gli assistenti vocali ci aiutano a gestire la nostra agenda e i nostri impegni; algoritmi di machine learning ci consentono di prendere decisioni più accurate e nel più breve tempo possibile; modelli di rete neurale ci consentono di scrivere un messaggio semplicemente dettandone il testo.

L'utilizzo di tecniche e modelli d'intelligenza artificiale sta producendo risultati molto positivi anche per quanto riguarda l'applicazione in problemi di Computer Vision come la classificazione di oggetti. Infatti, la classificazione di oggetti, oggi, è uno dei task più significativi il cui sviluppo è in costante crescita per quanto riguarda il campo di ricerca del deep learning e dell'intelligenza artificiale in generale. Lo studio e la progettazione di algoritmi e modelli di object classification forniscono strumenti sempre più performanti che consentono a macchine e/o computer di eguagliare, se non superare, la capacità umana di analizzare, individuare e riconoscere oggetti/persone/animali.

Questo progetto offre una metodologia, per un corretto approccio scientifico alla costruzione di architetture e modelli, utile ad affrontare problemi di object classification.

2 Dataset

Il dataset utilizzato per lo sviluppo dei modelli di Convolutional Neural Network è il Fruits-360 [6]. Il dataset è costituito da 90483 immagini di frutta e verdura di dimensione 100x100 pixeles in formato jpg, suddivise rispettivamente in un training-set composto da 67692 immagini (una frutta o verdura per immagine) e da un test-set composta da 22688 immagini (una frutta o verdura per immagine). Le immagini sono state classificate in 131 classi tra tipologie di frutta e verdura diverse. Differenti varietà dello stesso frutto o verdura (banana per esempio) vengono memorizzate come appartenenti a classi diverse.

2.1 Distribuzione del Training Set

Visto il numero cospicuo di classi da predire, è necessario uno studio della loro distribuzione all'interno del dataset al fine di evitare fenomeni massivi di class imbalance che potrebbero portare a risultati scadenti dei modelli di rete neurale convoluzionale sviluppati. A causa del numero di classi, si è deciso di eseguire una prima analisi preliminare delle 25 classi più rappresentate e delle 25 classi meno rappresentate.

Questa prima analisi mostra come non vi sia una chiara presenza massiva di class imbalance che possa portare a un fallimento nella fase di training dei modelli, infatti per le prime 25 classi maggiormente rappresentate, mediamente, il numero di immagini per classe è nella soglia 600-700; per le 25 classi meno rappresentate, mediamente, il numero di immagini per classe è nella soglia 400-500.

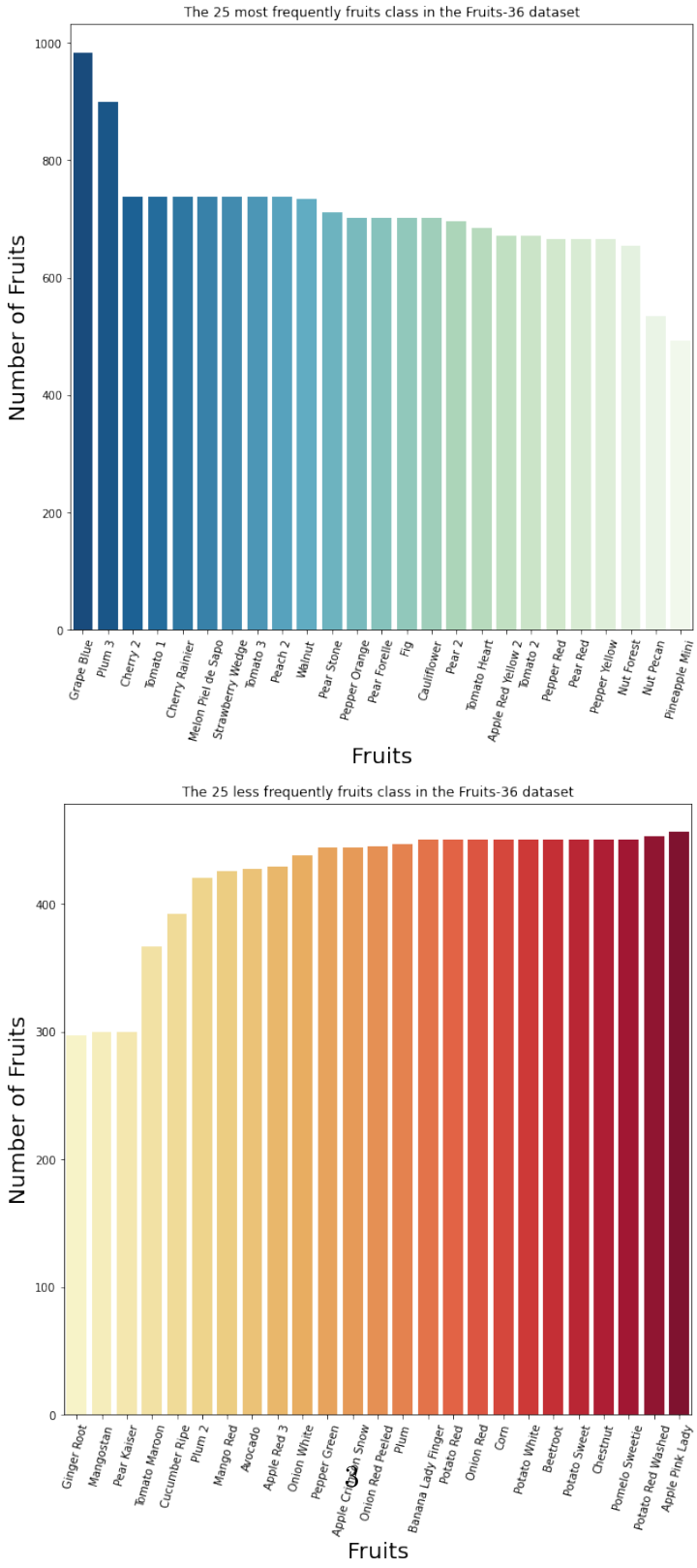

Figura 1: Distribuzione Dataset Fruits-360: in alto le 25 classe più rappresentate; in basso le 25 classi meno rappresentate

Nonostante il rapporto tra la classe più rappresentata "Grape Blue" (984 immagini) e la classe meno rappresentata "Ginger Root" (297 immagini) è di circa 1/3, l'analisi preliminare non mostra la presenza di una class imbalace significativa. Si è deciso comunque di studiare l'andamento generale della distribuzione del dataset, eseguendo un'analisi per threshold. Nello specifico si è suddiviso il dataset in base al numero di classi che avessero una distribuzione delle immagini:

- minore di 401 ;
- maggiore uguale a 401 ma minore di 601;
- maggiore uguale a 601 ma minore di 801;
- maggiore uguale a 801;

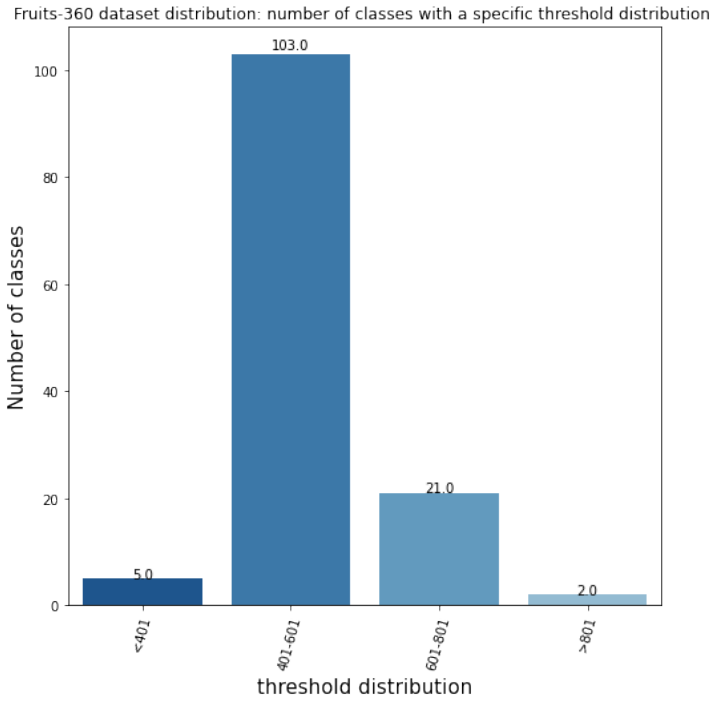

Figura 2: Distribuzione Fruits-360 dataset per threshold

Quest'ultima analisi mostra come all'interno del dataset Fruits 360 la distribuzione delle classi di interesse non è tale da causare problemi di class imbalance significativi.

3 Approccio metodologico

L'approccio metodologico adottato per la risoluzione del problema di object detection sul dataset preso in esame è stato lo sviluppo di 3 differenti modelli di deep learning. In questo modo si è cercato di studiare quale tecnica risultasse essere più performante per problemi di object detection similari. In breve, sono stati sviluppati questi modelli per poter correttamente classificare le 131 classi di frutta o verdura all'interno del dataset Fruits-360:

- Primo Modello: sviluppo di una rete neurale convoluzionale ex novo ;
- Secondo Modello: utilizzo di una rete neurale convoluzionale pre-addestrata mediante la tecnica del transfer learning e data augmentation ;
- Terzo Modello: utilizzo di una rete neurale convoluzionale pre-addestrata come estrattore di caratteristiche mediante un approccio di features extraction.

3.1 Primo Modello

3.1.1 Architettura[1]

Per poter affrontare la classificazione del dataset Fruits-360 con le rispettive 131 classi di frutta e verdura è stata sviluppata ex novo la seguente architettura di rete neurale convoluzionale:

- input layer: strato convoluzionale che riceve in input le immagini del training set, costituito da 32 neuroni con filtro (3x3) e funzione di attivazione "relu";
- pooling layer: Maxpooling[7] con filtro (2x2);
- 3 strati nascosti rispettivamente di 64,128,256 neuroni con filtro (3x3) e funzione di attivazione "relu";

- per ogni strato nascosto è stato aggiunto un pooling layer di tipo "maxpooling" con filtro (2x2);
- strato Flatten[8] per appiattire i neuroni dei precedenti strati;
- strato Dense[9]: composto da 32 neuroni con attivazione "relu";
- strato di regolarizzazione Dropout[10]: pari al 10%;
- output layer: composto da 131 neuroni (per mappare l'output in base al numero di classi) con funzione di attivazione "softmax";

3.1.2 Divisione del set di addestramento

Nella fase di training del modello, dal train set si è estratto un validation set in modo tale da validare, fin dalla fase di addestramento, il modello. Empiricamente la divisione tra train e val test con le quali si sono ottenute le migliori performance segue un rapporto di 90-10 e successivamente 70-30 sul 90%.

3.1.3 Ottimizzazione

Gli iper-parametri scelti per ottimizzare le performance del modello sono stati i seguenti:

- loss function: la funzione di perdita con cui si sono ottenute le migliori performance, tenendo conto delle struttura del problema (classificazione di 131 classi), è la categorical cross entropy[11] che, in combinazione con la funzione di attivazione softmax dell'output layer, offre le migliori prestazioni per la risoluzione di questo task;
- funzione di ottimizzazione: è stata scelta la funzione Adam[12] in quanto è una combinazione delle funzioni di ottimizzazione Rmsprop e Momentum. Inoltre, possiede la capacità di adattare il learning rate tenendo conto dei gradienti precedentemente calcolati (correzione bias dei momenti);
- metrica: le prestazioni del modello sono state valutate in base al punteggio ottenuto nell'accuracy

3.2 Secondo Modello

L'approccio utilizzato per lo sviluppo di questo secondo modello si basa sull'intenzione di applicare due tra le tecniche più utilizzate e importanti quando si parla di algoritmi di machine learning (quindi anche reti neurali), ovvero transfer learning e data augmentation, per verificarne e valutarne gli effetti che producono su problemi di classificazione con un numero cospicuo di classi da predire.

3.2.1 DataAugmentation[2][3]

Il modo migliore per generalizzare meglio un modello di machine learning/deep learning è addestrarlo su più dati. Nella pratica la quantità di dati che abbiamo è limitata. Un modo per aggirare questo problema è creare dati falsi e aggiungerli al training set. Per alcune attività di machine learning, è ragionevolmente semplice creare nuovi dati falsi. Questo approccio è più semplice per per task di classificazione. Un classificatore deve prendere un input ad alta dimensione x e riassumerlo con un'identità di categoria singola y. Ciò significa che il compito principale che deve affrontare un classificatore è di essere invariante rispetto a un'ampia varietà di trasformazioni. Possiamo generare facilmente nuove coppie (x, y) semplicemente trasformando gli input x nel nostro set di allenamento. La data augmentation è una tecnica particolarmente efficace per uno specifico problema di classificazione: il riconoscimento di oggetti. Tuttavia, bisogna fare attenzione a non applicare trasformazioni che possono comportare la distorsione delle immagini e della, per tanto, loro classe di appartenenza.
Mediante l'utilizzo della funzione ImageDataGenerator presente nella libreria Keras di Python, sono state applicate le seguenti trasformazioni:

- zoom range: zoom randomico dell'immagine pari al 20% ;
- horizontal flip: specchia la singola immagine;
- rescale: riscala, normalizzando, tutte le trasformazioni effettuate.

3.2.2 Transfer Learning[4][5]

Il transfer learning è una tecnica di machine learning in cui un modello sviluppato per un'attività viene riutilizzato come punto di partenza per un modello su una seconda attività.
È un approccio popolare nel deep learning in cui i modelli pre-addestrati vengono tipicamente utilizzati come punto di partenza per attività di Computer Vision (come la object detection) e attività di Natural Lenguage Processing date le vaste risorse di calcolo e di tempo necessarie per sviluppare modelli di reti neurali su questi problemi. Ovvero: dati due modelli P1 e P2, P2 sviluppato per svolgere due o più compiti diversi, assumiamo che molti dei fattori che spiegano le variazioni in P1 siano rilevanti per le variazioni che devono essere catturate per l'apprendimento P2. Questo è tipicamente compreso in un contesto di apprendimento supervisionato, in cui l'input è lo stesso ma l'obiettivo può essere di natura diversa. Se sono presenti molti più dati nella prima impostazione (campionata da P1), ciò può aiutare ad apprendere rappresentazioni utili per generalizzare rapidamente solo da pochissimi esempi tratti da P2[Tratto da: "Deep Learning - Goodfellow [2]] .

3.2.3 Architettura

L'architettura pre-addestrata, nonchè il modello di rete neurale pre-addestrato, scelta per affrontare il task di classificazione del dataset Fruits-360 è VGG16[13]. VGG16 è un modello pre-addestrato sul dataset Imagenet pensato per task di image recognition su larga scala. La sua principale caratteristica è l'uso di small receptive fields.

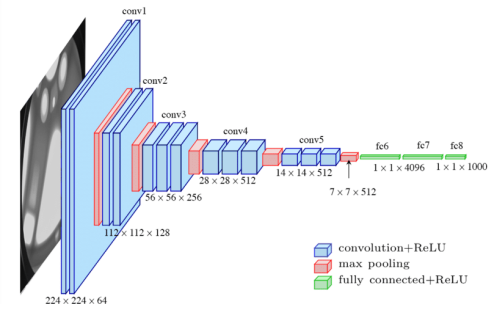

Figura 3: VGG16 Architetture[14]

La funzione di ottimizzazione e la funzione di perdita con le migliori prestazioni, come per il modello precedente, sono state Adam e la categorical cross entropy.

3.3 Terzo Modello

Come terzo approccio, utilizzando modelli pre-addestrati sul dataset ImageNet (come MobileNetV2 e VGG16), è stata utilizzata la tecnica di features extraction con la quale si sono estratte le caratteristiche principali dalle immagini presenti nel set di dati originale in modo tale da addestrare modelli di Machine Learning classici.

3.3.1 Features Extraction

Le tecniche di Features Extraction (FE) sono diventate un'esigenza evidente in molti processi di Computer Vision(object detection, Image processing, Image Retrieval), riconoscimento vocale, data mining, Pattern recognition e machine learning. La features extraction viene utilizzata per estrarre le caratteristiche più distinte presenti in un set di dati (immagine, testo, voce) che vengono utilizzate per rappresentare e descrivere i dati.[15]

Nel Pattern recognition e nell'Image processing, l'estrazione delle caratteristiche è una forma speciale di riduzione della dimensionalità. L'obiettivo principale è per tanto ottenere le informazioni più rilevanti dai dati originali e rappresentare tali informazioni in uno spazio di dimensionalità inferiore. Quando i dati di input di un algoritmo sono troppo grandi per essere elaborati e si sospetta che siano ridondanti (molti dati, ma non molte informazioni), i dati di input verranno trasformati in un insieme di caratteristiche di rappresentazione ridotto (denominato anche vettore di caratteristiche). La trasformazione dei dati di input nell'insieme di funzionalità è denominata estrazione di funzionalità. Se l'algoritmo di features extraction è accurato, questo estrarrà le informazioni pertinenti dai dati di input per eseguire l'attività desiderata utilizzando questa rappresentazione ridotta anziché l'input a dimensione intera.

La tecnica di estrazione delle caratteristiche viene eseguito dopo la fase di pre-processing ed è fondamentale per l'intero processo poiché il classificatore non sarà in grado di riconoscere le funzionalità scarsamente selezionate: in questo processo le caratteristiche rilevanti vengono estratte da oggetti/alfabeti per formare vettori di caratteristiche. Questi vettori vengono quindi utilizzati dai classificatori per riconoscere e associare l'unità di input con l'unità target di output. In sostanza, la FE, consiste nel trovare l'insieme di parametri che definiscono con precisione la forma di un oggetto in modo unico: l'obiettivo principale è estrarre un insieme di funzionalità che massimizza il tasso di riconoscimento con il minor numero di elementi e genera un insieme di caratteristiche simili per varietà di istanze dello stesso oggetto.[16]

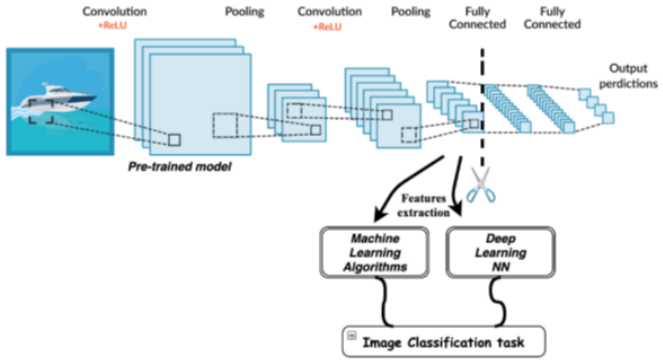

Figura 4: Features extraction for image classification[14]

3.3.2 VGG16

Il modello pre-addestrato utilizzato per estrarre le principali caratteristiche dalle immagini è VGG16. Vista la somiglianza tra la struttura del dataset Fruits-360 e la struttura del dataset utilizzato in fase di addestramento di VGG16 (ImageNet) si è deciso di "tagliare" il modello verso i layers finali in modo tale da estrarre le features più significative per il task di classificazione. Dopo ripetute prove, il taglio con le migliori performance si è rivelato essere il quarto blocco MaxPooling 2D in quanto specializzato nel riconoscimento dei dettagli specifici che consentono una corretta classificazione del soggetto dell'immagine.

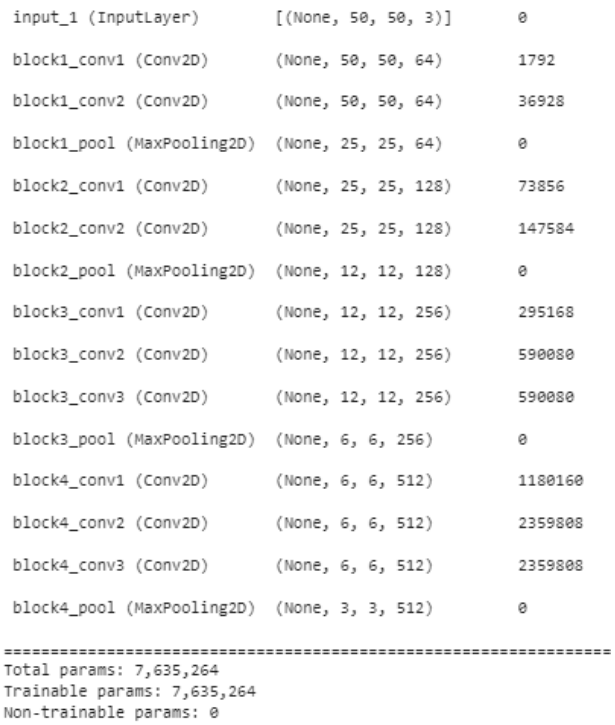

Figura 5: Architettura VGG16 tagliata al quarto blocco MaxPooling per poter estrarre le caratteristiche dal train set[14]

Tramite il taglio della rete VGG16 al quarto blocco di MaxPooling, si è riuscito a estrarre un vettore di caratteristiche principali di dimensione (67692,4608) dall'originale training set.

3.3.3 Modelli Machine Learning

Le features estratte precedentemente sono state utilizzate per addestrare i seguenti modelli tradizionali di machine learning, utilizzando i parametri standard forniti dalla libreria di scikit-learn di Python:

- Support Vector Machine;
- Logistic Regression;
- K-NearestNeighbor.

4 Risultati e Valutazioni

4.1 Primo Modello

La rete neurale sviluppata ex novo, come si può vedere dalla figura 6, riesce a raggiungere risultati eccellenti. Nel complesso, lo score del modello misurato sul validation set è di:

- Test loss: 0.013083158060908318
- Test accuracy: 0.9992614388465881

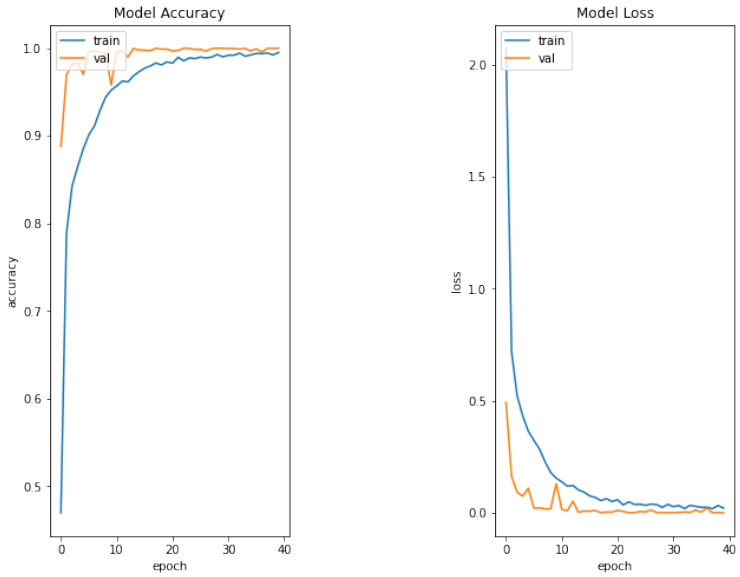

Figura 6: A sinistra il grafico delle curve di Accuracy e validation Accuracy del primo modello. A destra il grafico delle curve di Loss e validation Loss.

I modello riesce a generalizzare molto bene anche suimmagini mai viste prima come dimostrano i risultati ottenuti applicando la cnn al test set:

- Test loss: 0.3135681748390198;
- Test accuracy: 0.9744358062744141

4.2 Secondo Modello

A differenza del primo modello, l'utilizzo della tecnica di transfer learning tramite la rete VGG16 comporta una leggera perdita di performance e produce un piccolo fenomeno di overfitting.

Nonostante ciò, il modello riesce a generalizzare molto bene anche su immagini mai viste prima come dimostrano i risultati ottenuti applicando l modello al test set:

- Test loss: 0.12273755669593811
- Test accuracy0.9689263105392456

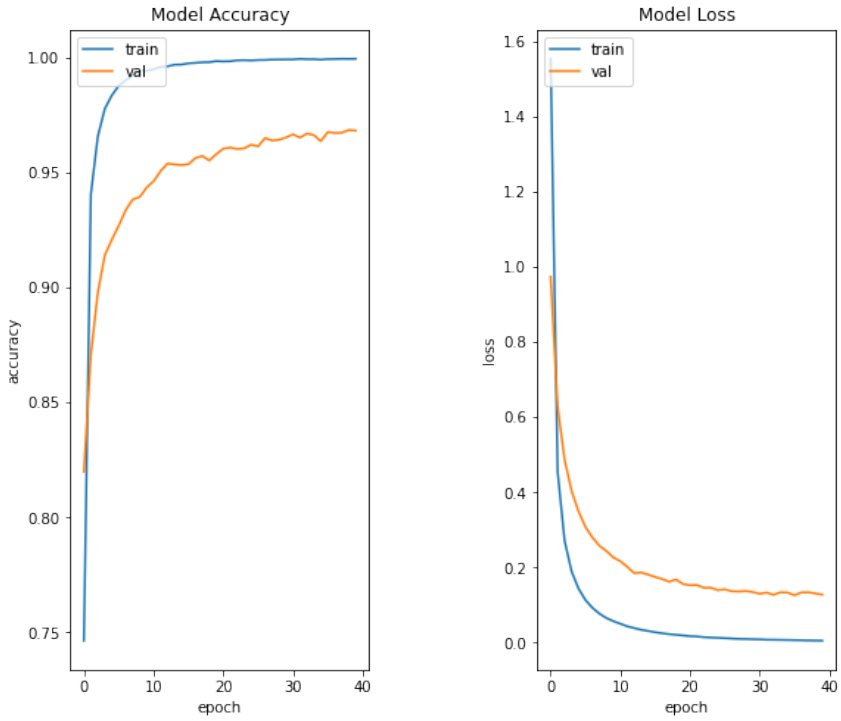

Figura 7: A sinistra il grafico delle curve di Accuracy e validation Accuracy del secondo modello. A destra il grafico delle curve di Loss e validation Loss.

4.3 Terzo Modello

Attraverso il vettore delle caratteristiche principali, estratto grazie alla rete neurale VGG16, utilizzato come vettore di input per i tradizionali modelli di ML, si sono ottenute performance eccellenti sulle immagini presenti nel test-set. Tra gli algoritmi utilizzati, SVM risulta essere il miglior classificatore delle immagini presenti nel dataset Fruits-360.

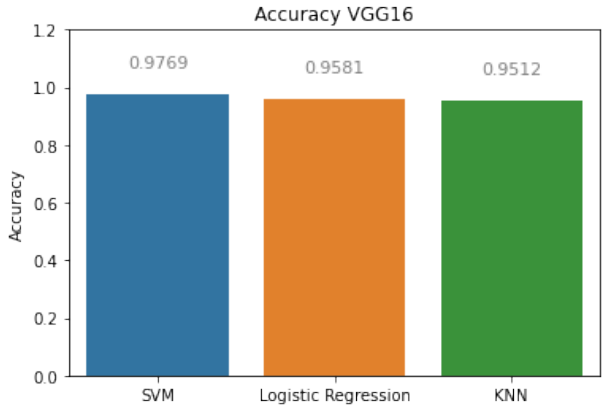

Figura 8: Accuracy dei tre modelli di machine Learning addestrati con le features estratte dal "block_4_pool" di VGG16.

5 Discussione

I risultati più performanti sono stati ottenuti dai modelli di machine Learning (SVM, Logistic Regression e KNN) addestrati utilizzando il modello pre-addestrato VGG16 con un taglio verso i layers finali, nello specifico al layer "block_4_pool", attraverso la tecnica di Features extraction. VGG16 è risultato essere il modello con le migliori performance per la tecnica di features extraction poichè, avendo 14,714,688 di parametri, ha una maggior capacità di riconoscere il soggetto dell'immagine.

Come si può vedere dal grafico sottostante (Figura 9) le performance ottenute grazie alla tecnica di features extraction sono migliori dello 0,36% rispetto alle performance ottenute grazie alla rete neurale convoluzionale sviluppata ex novo. Questo perchè:

- grazie alla features extraction si possono estrarre le caratteristiche più rilevanti di ogni immagine all'interno del dataset, rendendo così la fase di apprendimento molto più efficiente;
- l'utilizzo di un modello pre-addestrato come VGG16 ha un numero significativamente maggiori di parametri addestrabili (14,714,688 vs 466,499) che consento una miglior accuratezza nella classificazione.

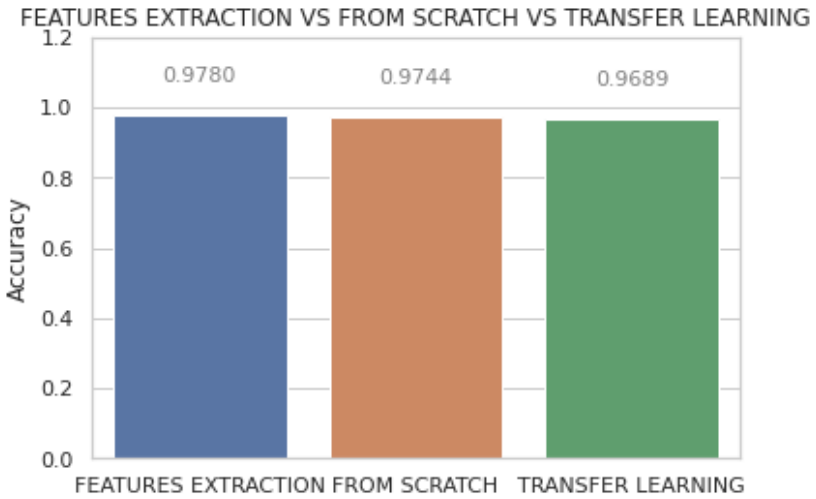

Figura 9: Confronto tra le performance, in termini di Accuracy, ottenute dai tre approcci metodologici applicati per la classificazione delle immagini di frutta e verdura contenute nel dataset Fruits-360.

Tuttavia, data la significativa leggerezza e minor complessità (in termini di architettura) della rete neurale sviluppata ex novo, si consiglia il suo utilizzo (rispetto alla tecnica di features extraction così come presentata) per task semplici di object detection.

5.1 Overfitting Secondo Modello

Come già accennato nella sezione dei risultati del secondo modello, dal grafico delle performance si può notare la presenza di un leggero fenomeno di overfitting. Nonostante la sua entità non preoccupi e, sopratutto, non infici le performance del modello stesso sulle immagini "unseen", è giusto soffermarsi sulle sue possibili cause.
Prendendo come spunto di riflessione i risultati ottenuti dal terzo e sopratutto dal primo modello, questi suggeriscono come per una corretta classificazione delle immagini presenti nel dataset Fruits-360 un'architettura più semplice, o meglio leggera, della rete di classificazione porti a risultati molto positivi senza la presenza di alcun tipo di overfitting. Proprio per questo motivo, l'utilizzo di una rete neurale con un'architettura molto complessa, come quella di VGG16 nella sua completezza, molto utile in problemi di object detection su larga scala comporti un sovra-apprendimento con relativa difficoltà, seppur leggera, nella generalizzazione di immagini mai viste prima.
Per questo motivo, qualora si volesse adottare la tecnica di transfer learning sul dataset Fruits-360, è consigliato l'utilizzo di una rete neurale con un'architettura più leggera (come dimostrano i risultati ottenuti giusto appunto dal primo modello) come può essere MobileNet. Qualora, invece, si preferisse l'utilizzo di una rete neurale più complessa come VGG16, si consiglia di adoperare la tecnica di data augmentation in modo più massivo, cercando di aumentare quanto più possibile il dataset di training (senza tuttavia adoperare filtri che vadano a distorcere troppo il soggetto dell'immagine).

6 Conclusioni

In generale, per tutte le tecniche presentate si possono considerare i risultati ottenuti soddisfacenti nonostante una capacità computazionale limitata.

Per futuri sviluppi, nel caso in cui il task da affrontare avesse una maggior complessità si dovrebbe tenere conto dell'utilizzo di macchine con una capacità computazionale maggiore in modo tale da poter:

- utilizzare modelli pre-addestrati con architetture più complesse e dense;
- utilizzare una dimensione di input delle immagini maggiore per aumentarne la qualità e incrementare l'estrazione di features più significative;
- applicare algoritmi di ricerca che mirano a trovare la miglior combinazione di parametri utili a migliore le performance dei modelli di Machine Learning (come ad esempio GridSearch).

Riferimenti bibliografici

[1] fchollet. The sequential model. Documentation at `https://keras.io/guides/sequential_model/`.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *J. Big Data*, 6:60, 2019.

[4] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, PP:1–34, 07 2020.

[5] Nidhi Agarwal, Akanksha Sondhi, Khyati Chopra, and Ghanapriya Singh. *Transfer Learning: Survey and Classification*, pages 145–155. 01 2021.

[6] Mihai Oltean Horea Muresan. Fruits-360 dataset. Available at `https://github.com/Horea94/Fruit-Images-Dataset`.

[7] Keras Documentation. Maxpooling2d layer keras. Documentation at https://keras.io/api/layers/pooling_layers/max_pooling2d/.

[8] Keras Documentation. Flatten layer keras. Documentation at https://keras.io/api/layers/reshaping_layers/flatten/.

[9] Keras Documentation. Dense layer keras. Documentation at https://keras.io/api/layers/core_layers/dense/.

[10] Keras Documentation. Dropout layer keras. Documentation at https://keras.io/api/layers/regularization_layers/dropout/.

[11] Keras Documentation. Categorical cross entropy keras. Documentation at https://keras.io/api/losses/probabilistic_losses/.

[12] Keras Documentation. Adam optimizer. Documentation at https://keras.io/api/optimizers/adam/.

[13] Keras Documentation. Vgg16 and vgg19. Documentation at https://keras.io/api/applications/vgg/.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[15] Ayodeji Salau and Shruti Jain. Feature extraction: A survey of the types, techniques, applications. pages 158–164, 03 2019.

[16] Gaurav Kumar and Pradeep Bhatia. A detailed review of feature extraction in image processing systems. 02 2014.