

UNIVERSITÀ DEGLI STUDI DI
MILANO-BICOCCA

ADVANCED MACHINE LEARNING
FINAL PROJECT

Riconoscimento di frutta dal Dataset Fruits360

Authors:

Giovanni De Feudis - 820602 - g.defeudis1@campus.unimib.it

Matteo Carcano - 873258 - m.carcano7@campus.unimib.it

Nataliya Zayeva - 867981 - n.zayeva@campus.unimib.it

31 gennaio 2022



Sommario

All'interno di questo report verranno presentati alcuni problemi di classificazione multi-classe di frutta e verdura. Si valuteranno tre tipologie di approcci. Un primo modello CNN hand-made, un modello realizzato tramite Transfert Learning e per ultimo verranno sfruttate le features estratte da CNN per procedere alla classificazione utilizzando tecniche tradizionali. Infine verrà approcciato un problema di riconoscimento di oggetti e classificazione in tempo reale.

1 Introduzione

Il riconoscimento e la classificazione di immagini di oggetti attraverso l'utilizzo di reti neurali è una pratica che può essere sfruttata in molti campi, ad esempio in ambito di raccolta automatizzata, nonché per altre funzionalità legate a gestioni inerenti la grande distribuzione. In particolare, la possibilità di classificare prodotti alimentari come frutta e verdura offre applicazioni utili in tema di organizzazione delle risorse legate all'impresa e mostra le sue potenzialità nel contrasto allo spreco alimentare.

Sebbene negli ultimi anni sono stati registrati molti miglioramenti, purtroppo molti approcci soffrono ancora di tempi di training e numero di falsi positivi molto elevati [1]. Il lavoro sviluppato è stato suddiviso in due sezioni, all'interno della prima sezione verranno confrontati tre diverse tipologie di modelli, il primo riguarderà una rete neurale realizzata e addestrata totalmente sul dataset, il secondo utilizzerà il metodo del *transfert learning* e per ultimo verranno sfruttate le features estratte da *CNN* per procedere alla classificazione utilizzando tecniche tradizionali (*svm-knn-xgboost-Decision Tree*). All'interno della seconda sezione verrà affrontato un problema di object detection e classification in tempo reale. I metodi di Machine Learning sfruttano i computer per simulare l'apprendimento umano e permettere agli stessi di identificare e acquisire conoscenza dal mondo [2], in questo modo i modelli di Machine Learning sono capaci di generalizzare da esempi. Negli ultimi anni sono stati proposti molti approcci che permettessero di applicare metodi di Deep Learning sulla frutta [3], molti dei quali richiedono molto tempo per l'addestramento e risentono di performance non del tutto convincenti.

Le reti neurali sono uno dei metodi più utilizzati all'interno dell'ecosistema del Machine Learning, esse imparano da dati etichettati e sfruttano l'informazione acquisita per permettere di etichettare dati che non lo sono [4]. Questo

ha permesso, grazie alle elevate performance hardware, di poter sfruttare queste metodologie in campo di riconoscimento di pattern, classificazione e previsione. L'introduzione delle reti neurali convoluzionali (CNN), ha portato negli ultimi anni ad un aumento considerevole delle performance risultando però computazionalmente costoso. Per ovviare a queste problematiche sono nati approcci definiti come *Transfert Learning*, in cui per risolvere uno specifico task si può fare affidamento ad un problema affine. Nel caso specifico preso in esame dal nostro studio, verranno utilizzati i pesi appresi durante l'addestramento della rete *MobileNetV2* sul dataset *Imagenet* [5] e verrà applicato del *fine-tuning*.

Le reti neurali possono essere utilizzate anche per task di estrazioni di caratteristiche (*features extraction*), permettendo una riduzione della dimensionalità del input in entrata. Queste caratteristiche verranno poi utilizzate sfruttando metodi di classificazioni classici. Le *Support Vector Machines* o *SVM*, sono un metodo di apprendimento supervisionato utilizzate in task di classificazione e regressione. Il *K-nearest neighbors* è un algoritmo utilizzato nel riconoscimento di pattern per la classificazione di oggetti basato sulle caratteristiche degli esempi più vicini a quello considerato. Il *XGBoost* è un algoritmo potenziato basato sul potenziamento del gradiente dell'algoritmo dell'albero di decisione. Il *decision tree* è un metodo utilizzato nel ML che permette, verificando varie condizioni, di dare delle risposte in output, che nel caso preso in esame permettono di classificare le immagini in classi. Il cuore del lavoro è diviso in due parti:

1. Valutazione dei modelli di classificazione: le performance dei modelli verranno valutate su Train e validation Test;
2. Task di riconoscimento di oggetti e classificazione in real-time: è stata costruita una demo dove viene mostrato alla webcam del computer un frutto che sarà l'input del modello di classificazione.

2 Datasets

Il dataset utilizzato [6] proviene da un archivio presente sul sito di competizioni di Data Science, Kaggle [7]. All'interno del dataset si trovano un totale di 90483 immagini di frutta e verdura, suddivise in tre cartelle per 131 tipologie di frutta: *Train*, *Test* e *Test-multiple-fruits*.

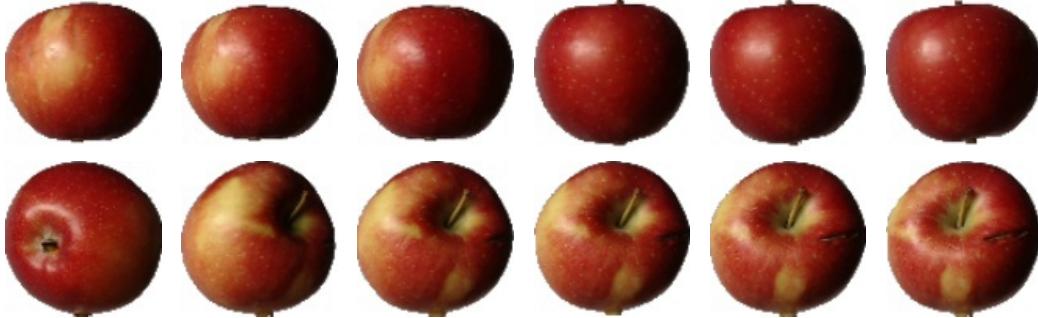


Figura 1: Classe: Apple Braeburn

Il dataset è stato realizzato filmando per mezzo di una webcam (*Logitech C920*), permettendo l'estrapolazione di foto da varie angolature (Figura 1). Per ovviare alle diverse illuminazioni che ogni immagine presentava, i ricercatori, oltre che predisporre uno sfondo bianco, hanno realizzato un algoritmo che permettesse di eliminare il rumore sullo sfondo.

Le immagini presentano tre canali colore e una dimensione di 100x100 pixels. Il formato di compressione è JPEG (*Join Photographic Experts Gropus*).

2.1 Data Augmentation

Data augmentation è una tecnica usata per incrementare il numero dei dati aggiungendo le copie delle immagini esistenti ma con una leggera modifica o dati creati sinteticamente da dati esistenti. Funziona come regolarizzatore e aiuta a prevenire l'overfitting quando il modello viene allenato.

Per questo progetto è stata utilizzata la libreria Keras di TensorFlow che permette di aumentare la diversità del dataset applicando delle trasformazioni. Le immagini del training vengono passate in input per subire le seguenti trasformazioni:

- **Random Flip:** capovolge casualmente le immagini durante l'allenamento, la modalità scelta è *orizzontalmente e verticalmente*;
- **Random Rotation:** l'immagine viene ruotata casualmente durante il train, è stato scelto che venga ruotata di 40° oppure di -40°;
- **Random Contrast:** aggiusta il contrasto dell'immagine scegliendolo in modo casuale da un range di valori [1.0 - lower, 1.0 + upper], *lower* e

upper vengono scelti dall'utente o come float positivo o come una tupla di due dimensioni. La scelta del *lower/upper* è stata uguale a 0.2.

Anche se il dataset del training è abbastanza grande, tuttavia, si è deciso di utilizzare dei metodi di data augmentation per avere una migliore predizione di dati anche in presenza rumori e per ridurre il rischio di overfitting nella fase di training.

2.2 Modifica sfondo

Per la classificazione in tempo reale è stato necessario andare a modificare le immagini aggiungendo un nuovo sfondo. Questa scelta è dovuta al fatto che le immagini presenti nel dataset sono state scattate in condizioni ideali che non sono replicabili tramite webcam del computer (sfondo bianco con rimozione delle diverse illuminazioni). In particolare, come sfondo sono state aggiunte le foto in Figura 2.

3 Aspetti metodologici

3.1 Scelta del modello

Di seguito verranno presentati i modelli selezionati per il task di comparazione delle performance.

3.1.1 Costruzione e addestramento rete neurale

L'idea per l'architettura è la stessa delle reti VGG, ovvero preferire convoluzioni piccole in modo da avere un rete più profonda aumentando così la non linearità. Contemporaneamente si è cercato un buon trade-off tra profondità e numero di parametri basso (questo perché addestrare una CNN può essere computazionalmente dispendioso e molto lungo). Alla fine si è optato per quattro convoluzioni 2×2 , ognuna seguita da un *Max-pooling*, successivamente l'output di queste convoluzioni ($6 \times 6 \times 128$) viene srotolato in un vettore a cui segue un *fully-connected* layer con 150 nodi e infine l'output layer con 131 neuroni, uno per ogni classe. Inoltre, per evitare overfitting è stato utilizzata la tecnica di *Dropout*. In totale ci sono 754,571 parametri addestrabili.

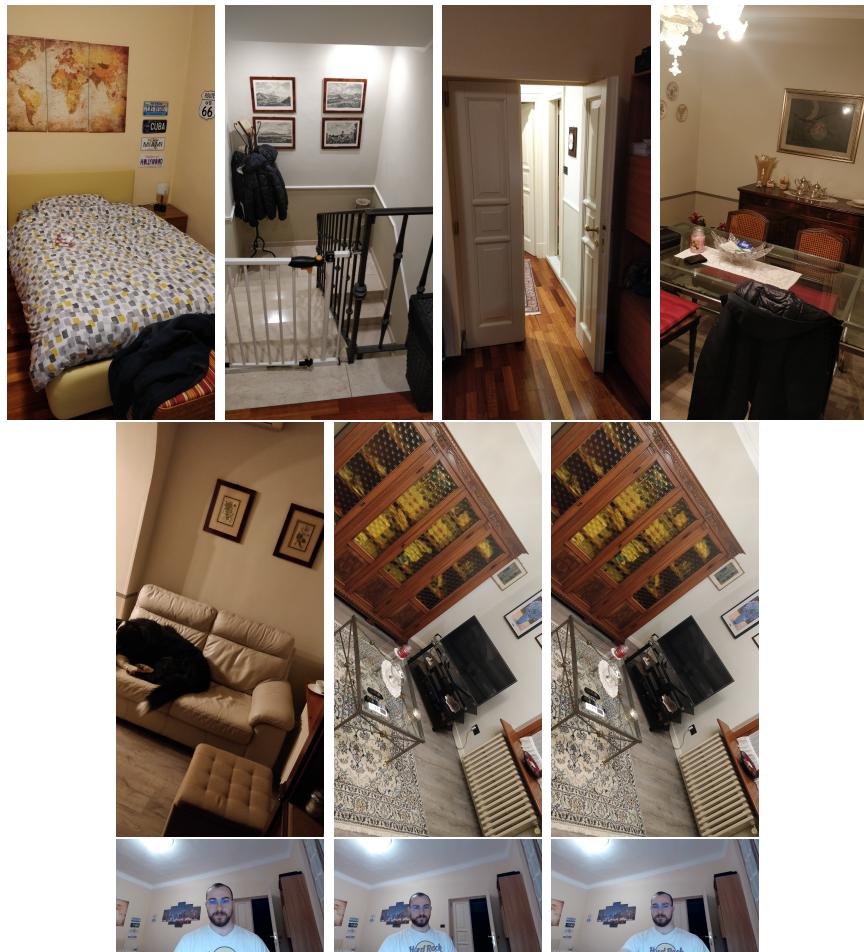


Figura 2: Sfondi utilizzati per modificare le immagini di training.

Come ottimizzatore è stato scelto RMSProp, come funzione di loss si è scelta la categorical-cross entropy e si sono scelte 30 epoche.

3.1.2 Transfert Learning - MobileNetV2

Il modello di transfer learning scelto è il MobileNetV2, che è stato pre-addestrato sull' *ImageNet*, un database di immagini organizzato secondo la gerarchia di WordNet, in cui ogni nodo della gerarchia è rappresentata da centinaia di migliaia di immagini. In presenza, di un dataset non troppo ampio per addestrare il modello è necessario utilizzare *ImageNet*. La rete di MobileNetV2 è stata importata e sono stati tagliati i *fully connected* per permettere di aggiungere un classificatore che si adatti al dataset considerato. I pesi sono stati bloccati per escluderli dal nuovo addestramento.

Nei livelli di pre-elaborazione sono state utilizzate le tecniche di Data Augmentation descritte nel paragrafo 2.1, a tali layer sono stati aggiunti successivamente un *Dense* layer (con 1024 neuroni e funzione di attivazione il *ReLU*), un layer *Dropout* di 0.5 per prevenire l'overfitting, un *Dense* layer (con 512 neuroni, funzione di attivazione il *ReLU*, come kernel inizializzatore è stato utilizzato il *Glorot uniform initializer* e come regolarizzatore del kernel e dei bias il *L₁ Regularizer* con $\lambda = 0.01$). Infine è stato aggiunto un *Dense* layer (con 131 neuroni, uguale al numero delle classi e come attivazione il *Softmax*).

Come ottimizzatore è stato scelto RMSProp, poiché provando con altri ottimizzatori, come Adam o SGD, l'accuracy era minore. La funzione di loss scelta è la Categorical-Crossentropy, in quanto classificazione multi-labels. Sono stati così raggiunti circa 4 milioni di parametri, di cui solo 2 milioni addestrabili. Il modello è stato addestrato per 75 epoche. In Figura 3 si può osservare l'architettura del modello.

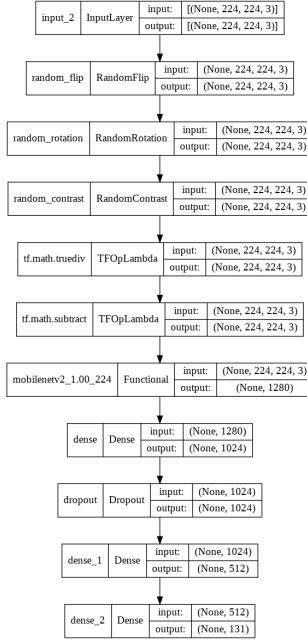


Figura 3: Modello MobileNetV2.

3.1.3 Features extraction e classificatori tradizionali

L’architettura utilizzata per la procedura di transfert learning, come per il paragrafo 3.1.2, è stata una *MobileNetV2*, a cui è stato rimosso il top layer. Successivamente per permettere l’addestramento sono stati aggiunti un layer *Flatten*, un layer *Dense* da 1024 neuroni ed infine per la procedura di classificazione è stato aggiunto un layer *Dense* da 131 neuroni con funzione di attivazione *softmax*. Dopo aver addestrato il modello per 15 epoche, si è passati alla fase di features extraction. In questo modo immettendo all’interno del modello (Figura 4) le immagini presenti nel Dataset si avrà come risultato un vettore di 1024 elementi, che permetteranno ai classificatori presi in esame, di predire le classi del task. Uno dei problemi riscontrati durante la fase di features extraction è stata la difficile gestione della memoria Ram nel momento della creazione della variabile contenente gli array delle immagini. Il dataset è stato quindi suddiviso in parti, in modo tale da immettere queste sezioni all’interno della rete di features extractor e successivamente unire tutti gli outputs (Figura 5).

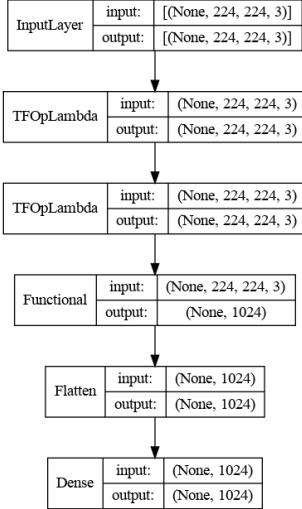


Figura 4: Features Extract MobileNetV2



Figura 5: Il dataset contiene più di 90000 immagini, rendendo proibitiva la gestione delle risorse.

Per la classificazione sono stati utilizzati 4 classificatori differenti:

- *Svm*;
- *Xgboost*;
- *Knn*;
- *Decision Tree*.

I risultati ottenuti saranno poi presentati e discussi all'interno del prossimo paragrafo.

3.2 Riconoscimento di oggetti e classificazione in tempo reale

Per il riconoscimento di oggetti in tempo reale, inizialmente si è deciso di utilizzare il modello migliore che, come si vedrà nella Sezione 4 è transfer learning con SVM. Come detto precedentemente è stato necessario modificare lo sfondo delle immagini, infatti se si applica la SVM addestrata precedentemente non si è in grado di classificare correttamente gli oggetti in quanto le immagini presenti nel dataset sono troppo "ideali". Per risolvere questo problema si è cercato di modificare lo sfondo aggiungendo delle immagini scattate da noi. Per fare ciò è stato necessario costruire una maschera per ogni immagine, sovrapporla all'immagine stessa per "tagliare" solo la parte di interesse, sovrapporre la maschera inversa (invertendo neri e bianchi) alla immagine di sfondo e infine unire le due immagini ottenute, un esempio è mostrato in Figura 6. Queste operazioni impiegano molto tempo per essere eseguite e, dal momento che sono a disposizione molte immagini per ogni classe, si è deciso di applicare uno sfondo a solo una parte delle immagini per ogni classe, in particolare per ogni sfondo e per ogni classe è stato estratto un campione casuale di ampiezza pari alla numerosità della classe diviso il numero di sfondi, uno schema è rappresentato dall'Algoritmo 1.

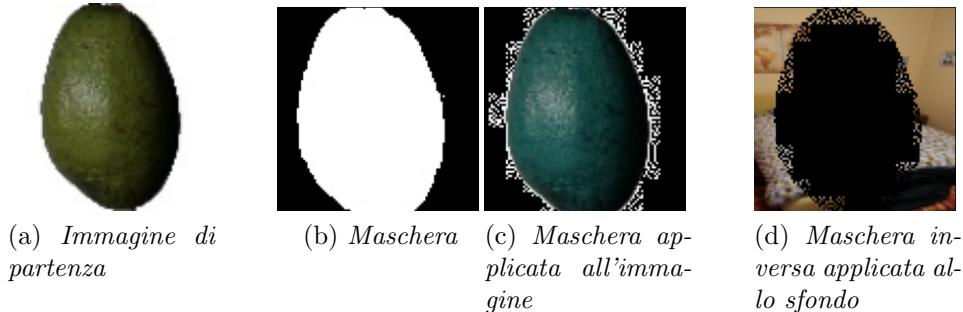


Figura 6: Esempi dei passaggi per l'aggiunta di un nuovo sfondo.

Una volta creato il nuovo dataset è stato quindi riaddestrato il modello, ottenendo però dei risultati non soddisfacenti. Dal momento che i tempi di addestramento per la SVM e la rete neurale sono simili si è preferito modificare la rete neurale piuttosto che la SVM. In particolare alla rete neurale è stata aggiunta la regolarizzazione L_2 con $\lambda = 0.01$. Questa piccola modifica ha fatto sì che il modello funzioni meglio facendo predizioni più veritiere.

Algorithm 1 Modifica sfondo immagini

```
1: Selezionare  $N$  sfondi
2: repeat
3:    $c \leftarrow$  campione casuale di immagini della classe
4:   for  $n = 0$  to  $N$  do
5:     repeat
6:        $Image \leftarrow$  selezionare un immagine da  $c$ 
7:        $Mask \leftarrow$  creazione della maschera
8:        $Image\_masked \leftarrow$  sovrapposizione maschera con l'immagine
di partenza
9:        $Inverse\_mask \leftarrow 255 - Mask$ 
10:       $background\_masked \leftarrow$  sovrapposizione  $Inverse\_mask$  e
sfondo
11:       $new\_image \leftarrow$  sovrapposizione  $Image\_masked$  e
 $background\_masked$ 
12:    until Tutte le immagini nel campione sono state modificate
13: until Tutte le classi sono state modificate
```

4 Risultati e valutazioni

4.1 Risultati modelli

Per la valutazione delle performance dei modelli presentati verrà utilizzata l'accuracy sul training e sul validation test set i cui risultati sono raccolti in Tabella 1.

Tabella 1: Tabella con le accuracy delle architetture considerate.

Modelli	Accuracy training	Accuracy test
Rete CNN	97.88 %	97.69%
MobileNetV2	98.47%	96.88%
SVM	99.99%	98.44%
XGBoost	99.99%	91.81%
KNN	99.99%	98.31%
Decision Tree	99.99%	80.46%



Figura 7: Matrice di confusione per fine tuning con SVM.

Osservando i risultati del test ottenuti sulle specifiche classi e sulla matrice di confusione in Figura 7 si possono notare alcuni casi particolari. La classe 5 (*Apple Granny Smith*) presenta una precision del 73%, alcuni casi sono stati classificati come falsi positivi all'interno della classe 4 (*Apple Golden 3*). La classe 35 (*Corn*) presenta una precision del 75%. Sono stati registrati 13 falsi positivi all'interno della classe 23 (*Cantaloupe*) e 23 falsi positivi all'interno della classe 98 (*Physalis with Husk*). La classe 36(*Corn Husk*) presenta una precision del 73%. Sono stati registrati 29 falsi positivi all'interno della classe 57 (*Kohlrabi*).

4.2 Risultati riconoscimento di oggetti e classificazione in tempo reale

Per quanto riguarda i modello per la classificazione in tempo reale è stata raggiunta un'accuratezza del 95.98% sul training e del 99.43% sul validation e 94.35% sul test set, quindi non sembra esserci overfitting. In questo caso l'accuratezza è più bassa rispetto a prima, questo perché sono state utilizzate anche le immagini modificate e della regolarizzazione, quindi si è andata ad aumentare la capacità di generalizzazione del modello. Infatti, come si può

vedere in Figura 8, ora il modello è in grado di classificare correttamente i frutti fa delle classificazioni con frutti simili. In particolare si può notare come il modello sia stato in grado di riconoscere correttamente la pera, mentre si è considerata anche parzialmente corretta anche l’arancia, che è stata riconosciuta come mandarino, in quanto le immagini di training occupano tutto lo spazio a disposizione perdendo la componente spaziale. Per quanto riguarda la mela è stata erroneamente classificata come pomodoro, riconoscendo almeno il colore.

5 Discussione dei Risultati

Il primo obiettivo all’interno di questo report è stato quello di mettere a confronto alcuni modelli per poterne valutare le performance sull’accuracy. Come mostrato in Tabella 1, la maggior parte dei modelli si sono attestati su livelli di accuracy sul test set decisamente elevate, cioè sopra al 96%. Mostrandosi quindi calzanti per la tipologia di task di classificazione proposto. Il modello che ha raggiunto la miglior accuracy in assoluto è stata l’SVM, cioè il modello che ha sfruttato la features extraction proveniente dalla rete MobileNetV2 presente in Figura 4. Il modello di XGboost e il Decision Tree si sono rivelati i peggior modelli a generalizzare, dove l’accuracy sul training si aggira sul 99% mentre nei test, nel caso del XGboost 91.81% e nel caso del Decision Tree 80.46%.

6 Conclusioni

Il lavoro si è basato sul confronto di alcuni modelli per il riconoscimento della frutta e successivamente si è sviluppato un modello anche per il riconoscimento in tempo reale. In futuro, la realizzazione di un’applicazione per cellulari permetterebbe ad individui con disabilità visive di poter riconoscere all’interno di supermercati non solo frutta e verdura, ma anche il colore e il grado di maturazione, in modo da permettere all’acquirente di scegliere il prodotto migliore per le sue esigenze.

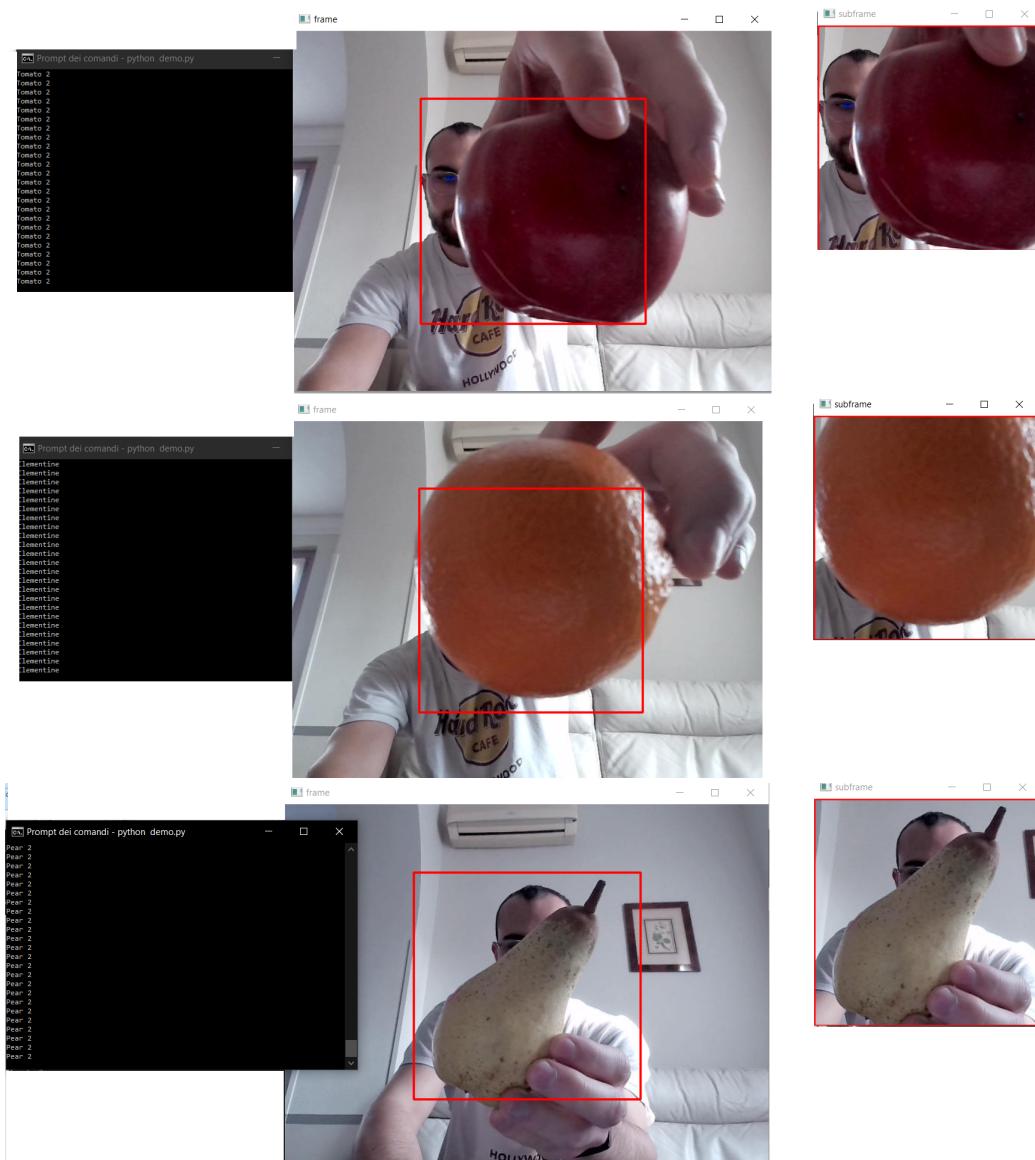


Figura 8: Esempio di previsioni in tempo reale. La parte che viene data come input al modello è rappresentata dal quadrato rosso (di dimensione 100×100 , nel terminale si possono vedere le previsioni).

Riferimenti bibliografici

- [1] L. T. Duong, P. T. Nguyen, C. Di Sipio, and D. Di Ruscio, “Automated fruit recognition using efficientnet and mixnet,” *Computers and Electronics in Agriculture*, vol. 171, p. 105326, 2020.
- [2] I. Portugal, P. Alencar, and D. Cowan, “The use of machine learning algorithms in recommender systems: A systematic review,” *Expert Systems with Applications*, vol. 97, pp. 205–227, 2018.
- [3] J. Muangprathub, N. Boonnam, S. Kajornkasirat, N. Lekbangpong, A. Wanichsombat, and P. Nillaor, “Iot and agriculture data analysis for smart farm,” *Computers and Electronics in Agriculture*, vol. 156, pp. 467–474, 2019.
- [4] D. Svozil, V. Kvasnicka, and J. Pospíchal, “Introduction to multi-layer feed-forward neural networks,” *Chemometrics and Intelligent Laboratory Systems*, vol. 39, pp. 43–62, 1997.
- [5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, 2009.
- [6] H. Mureşan and M. Oltean, “Fruit recognition from images using deep learning,” *Acta Universitatis Sapientiae, Informatica*, vol. 10, pp. 26–42, 2018.
- [7] M.Oltean, “Fruits 360,” 2017. [Online]. Available: <https://www.kaggle.com/moltean/fruits>