

Fine-grained FoodX-251 classification

Kevin Pretell 816725
Simone Benitozzi 889407
Raffaele Cerizza 845512

1 - Analisi del Dataset

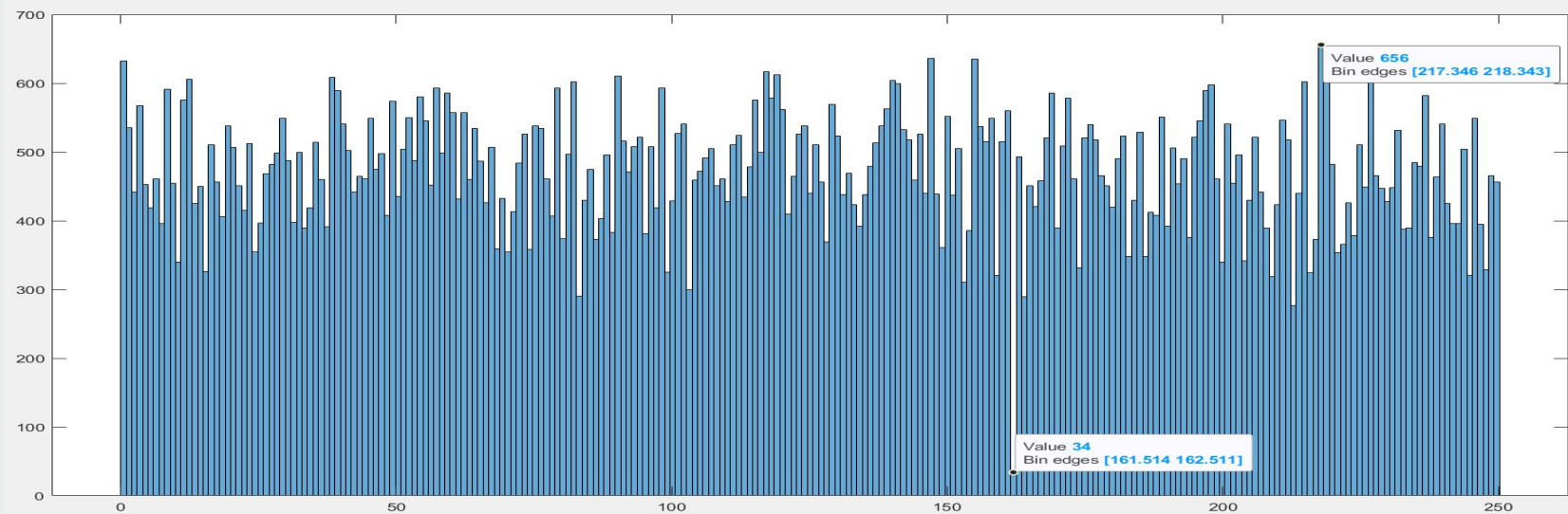


- Il dataset sembra molto ben fornito di immagini per una buona classificazione (più di 100 000 di immagini per l'addestramento)
- Le immagini di Training non sembrano affette da rumore o difetti particolari, tranne la normale compressione JPG, lo stesso per il validation set pulito.
- Risulta interessante invece come il validation set degradato presenti sì dei difetti ma troppo intensi, soprattutto il rumore gaussiano, a prima vista alcune immagini di cibo/piatti sono irriconoscibili.
- Per essere oggettivi si è calcolato la media di BRISQUE (qualità dell'immagine) su entrambi i validation set:
 - Validation set Normale: 28.8350
 - Validation set Degradado: 49.2941

1 - Analisi del Dataset

Di seguito la distribuzione iniziale delle classi del Training set.

Prima di continuare con altre analisi è sorta la necessità di bilanciare e pulire il Training set.



2 - Pulizia del Dataset: analisi preliminare visiva

- A seguito di un'analisi empirica delle immagini del dataset si è osservato come molte immagini non riguardassero il cibo. Essendo un dataset creato da immagini prese dal web c'è stata una fuga di immagini non corrette.
- Essendo il nostro un dataset con molte classi di immagini, avere queste immagini (rumore) presentava un ostacolo per la classificazione.
- Si riportano alcuni esempi di immagini presenti nel dataset e non pertinenti.



2 - Pulizia del Dataset: modalità di pulizia

- Si è quindi resa necessaria una prima fase di processing sul dataset, che ha previsto la pulizia del train set.
- Una prima idea è stata quella di calcolare gli outlier all'interno del dataset, così da effettuare una pulizia automatica. Questo però non era applicabile quantomeno per le seguenti ragioni:
 - anzitutto alcune classi erano composte per la maggior parte da immagini da scartare, e si rischiava quindi di andare a scartare le immagini che effettivamente rappresentano i cibi della classe;
 - inoltre un approccio di questo tipo avrebbe introdotto un bias nel dataset, dal momento che già in principio si sarebbero andato a rimuovere immagini sulla base di feature calcolate analiticamente;
 - infine anche nel caso in cui le immagini di una classe fossero per la maggior parte corrette, gli outlier potrebbero rappresentare una varietà intra-classe che è necessario considerare.
- La scelta finale è ricaduta quindi su una pulizia manuale del dataset, annotando tutte le immagini da scartare, per poi rimuoverle programmaticamente, e ottenere così una versione del dataset controllata e garantita. Per la pulizia del dataset sono stati adottati i seguenti criteri:
 1. sono state tolte le immagini che non contengono alcun cibo;
 2. sono state tolte le immagini che contengono solo strumenti/posate, ma senza cibo;
 3. sono state tenute le immagini dove si vede del cibo, anche se in piccola parte;
 4. sono state tenute le immagini dove il cibo corretto è unito ad altre tipologie di cibo.
- Sono così state rimosse dal training set 4323 immagini.

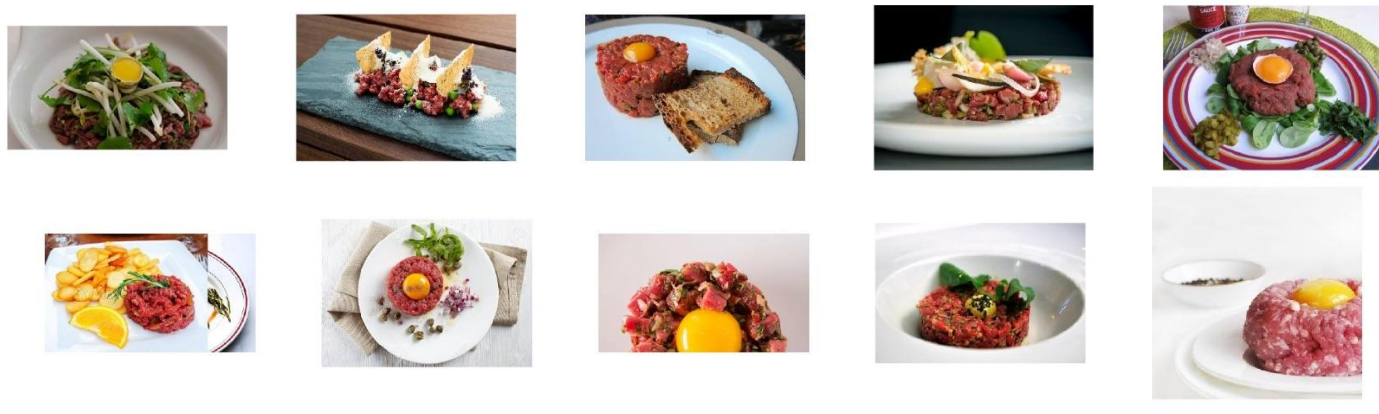
3 - Analisi del Dataset pulito

Similarità tra classi

Ottenuto il dataset pulito un'ulteriore analisi sulle varie classi del dataset ci ha fatto notare la presenza di una forte similarità tra diverse classi, un fenomeno che si riflette nei risultati ottenuti alla fine del nostro esperimento.

Di seguito mostriamo solo alcune paia di classi simili, difficili da distinguere anche per l'occhio umano.

Beef tartare vs Steak tartare





Oyster vs Huitre



Stuffed Tomatoes vs Stuffed Peppers



Buffalo wings vs Chicken Wings

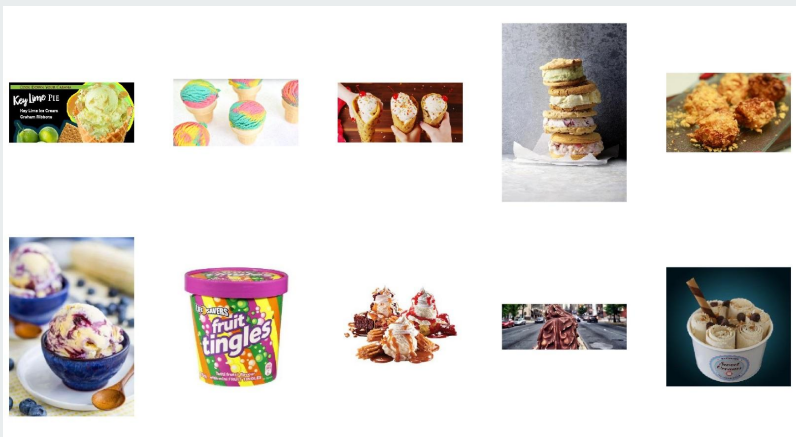


Schnitzel vs Fish&Chips

3 - Analisi del Dataset pulito

Varianza intra-classe

Meno forte invece il fenomeno di varianza intra classe, di seguito mostriamo alcune classi dove l'oggetto da classificare è presente con forme, colore e texture diverse.



Ice Cream



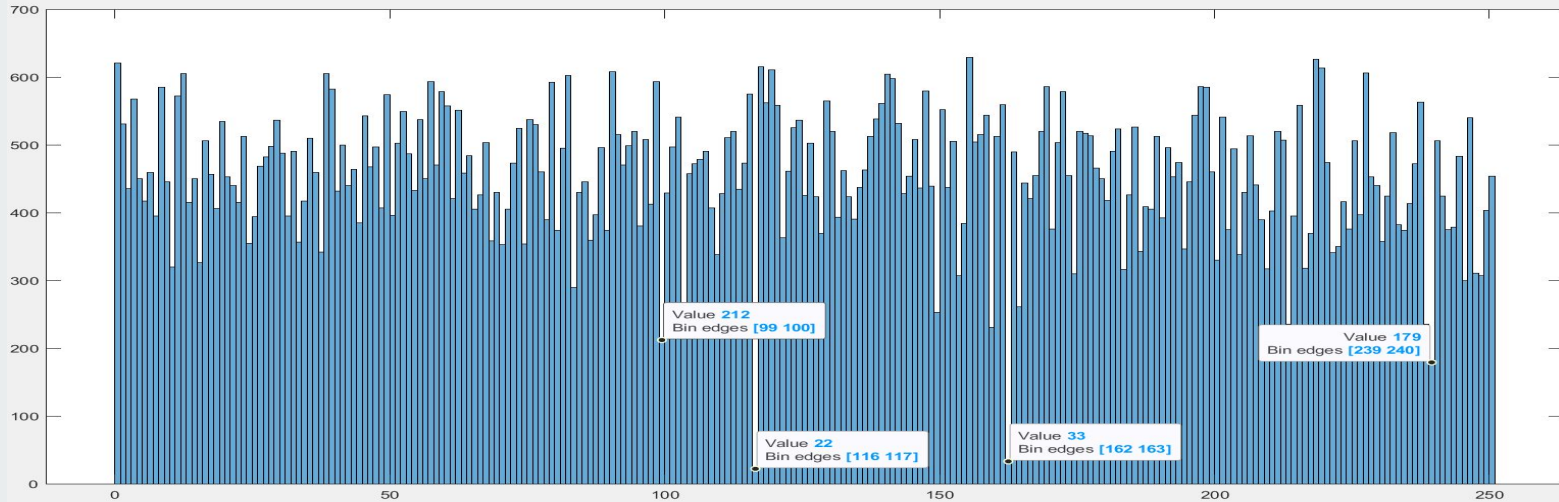
Gingerbread



Torte

4 - Bilanciamento delle classi

- A seguito della pulizia, è sorta la necessità di ribilanciare il dataset, avente classi con numeri molto diversi di immagini, a partire da un minimo di 22 per arrivare ad un massimo di 629.
- È stata scelta una soglia target di 500 immagini per classe, che ha reso il dataset di dimensione totale di 125500 immagini, vicino quindi alle dimensioni originali



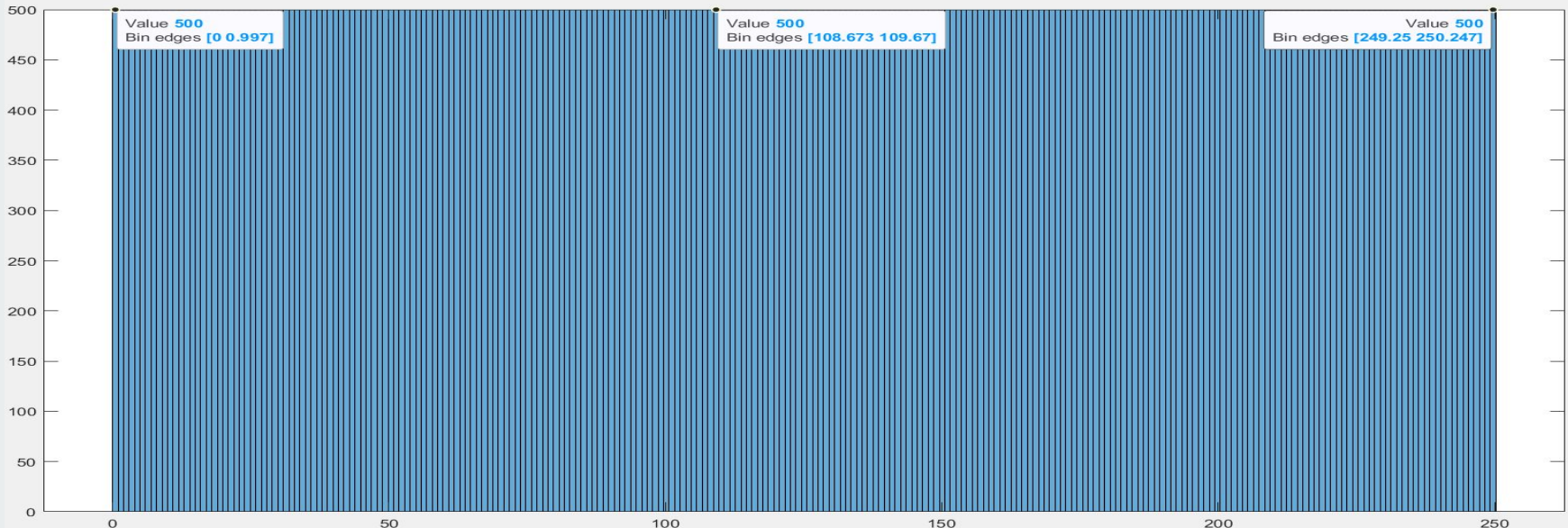
4 - Bilanciamento delle classi: a) Oversampling

- Per le classi minoritarie sono stati provati degli algoritmi statistici per l'augmentation del dataset, quali **SMOTE** e la sua variante **ADASYN**, ma entrambi creavano nuove immagini con forti componenti rumorose, soprattutto nelle variazioni del colore, per tanto i risultati ottenuti sono poi stati scartati
- È stato quindi effettuato un oversampling delle classi minoritarie per mezzo di data augmentation sulle immagini, e nello specifico sulle seguenti componenti:
 - rotation_range
 - shear_range
 - zoom_range
 - brightness_range
 - width_shift_range
 - height_shift_range
 - horizontal_flip



4 - Bilanciamento delle classi: b) Undersampling

- La fase di Undersampling ha invece previsto la rimozione delle immagini in eccesso rispetto alla soglia scelta, mediante un sampling randomico con distribuzione normale
- Ne risulta la seguente distribuzione uniforme delle classi:



5 - Analisi del Validation set Degradato: a) individuazione dei difetti

- L'**obiettivo** del presente lavoro è ottenere le performance di classificazione più elevate su un validation set contenente immagini con condizioni di acquisizione non ottimali.
- Il **problema** posto dal validation set degradato consiste nella maggior difficoltà per un modello di classificazione nel riconoscere le feature caratterizzanti le classi.
- La **soluzione** da noi proposta consiste nell'effettuare un'operazione di data augmentation sul train set. Questa operazione prevede l'individuazione e la replicazione sul train set dei difetti del validation set degradato. In questo modo il train set conterrà sia le immagini originali sia le nuove immagini degradate. Questa operazione comporta quantomeno due effetti:
 - si incrementa il numero di istanze di ogni classe del train set;
 - si introduce maggiore variabilità nel train set.

Questi effetti inducono una migliore generalizzazione dei modelli di classificazione.

Infatti: “the accuracy is inversely proportionate to the number of classes while being directly proportionate to the sample size per class of the dataset” (Zanisham Z., Lee C.P., Lim K.M., *Food Recognition with ResNet-50*, in *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, 2020).

5 - Analisi del Validation set Degradato: a) individuazione dei difetti

- **Blur.**

Il primo artefatto individuato è il blur. Il valore del blur è stato individuato come segue:

- a ogni immagine del validation set non degradato è stato applicato un filtro gaussiano con diversi valori di sigma;
- si è valutata la qualità dell'immagine così degradata con la metrica no reference **Brisque**;
- il valore di sigma accettato è quello che ha portato al valore di Brisque più vicino a quello dell'immagine corrispondente del validation set degradato.

Si è preferita la metrica Brisque alle metriche full reference in quanto la prima ha permesso di ottenere immagini degradate più simili a quelle del validation set degradato secondo un giudizio umano.

Infine si è riscontrato che la presenza del blur era alternativa alla presenza del rumore gaussiano. Pertanto è stato utilizzato un algoritmo di **Noise Level Estimation**[1] per verificare la presenza di rumore gaussiano. In questo modo:

- si è velocizzato il procedimento di analisi delle immagini;
- si sono evitate degradazioni eccessive e indesiderate.



5 - Analisi del Validation set Degradato: a) individuazione dei difetti

- **Problemi di white balancing.**

L'intensità dei colori di molte immagini è alterata. Questo porta un oggetto bianco a non apparire bianco. Si è quindi proceduto a calcolare dei pesi per i canali R, G e B in modo da poter replicare queste alterazioni. In particolare:

- per ogni immagine degradata vengono applicati gli algoritmi di white balancing **Shades of Gray**, **General Gray World**, **Gray Edge (1st order)** e **Gray Edge (2nd order)**;
- questi algoritmi restituiscono dei pesi per i canali R, G e B;
- questi pesi vengono applicati all'immagine ottenuta al passo precedente (in tema di blur);
- si calcola il valore di **Brisque** dell'immagine così ottenuta;
- si accettano i pesi R, G e B che portano al valore di **Brisque** più vicino a quello dell'immagine corrispondente del validation set degradato.

Per l'implementazione degli algoritmi di white balancing è stata utilizzata una funzione esterna[1].

Tra gli algoritmi di white balancing non sono stati utilizzati gli algoritmi **White Point** e **Gray World** in quanto:

- il primo assume che la probabilità di avere una superficie bianca nell'immagine sia alta; questo non è sempre vero; e infatti questo algoritmo ha spesso restituito risultati scadenti;
- il secondo assume che la media della distribuzione dei colori dell'immagine sia un grigio; e anche questo non è sempre vero.



5 - Analisi del Validation set Degradato: a) individuazione dei difetti

● Contrasto.

Le immagini degradate presentano un contrasto alterato. Questo porta le immagini a essere molto sovraesposte o molto sottoesposte. Per trovare il valore di questa alterazione si è proceduto come segue:

- per ogni immagine ottenuta dal passo precedente si applica una **gamma correction globale**: $V_{out} = V_{in}^{\text{gamma}}$ (dove V_{out} è il valore di output e V_{in} è il valore di input);
- vengono applicati diversi valori di gamma;
- viene accettato il valore di gamma che porta al valore più alto della metrica full reference **Structural Similarity Index Measure (SSIM)**. In questo caso la metrica viene calcolata tra l'immagine del validation set degradato e quella corrispondente del validation set pulito che è stata progressivamente degradata. Il valore SSIM è stato calcolato sulle immagini in scala di grigi corrispondenti al canale V dello spazio colore HSV.

In questo caso non è stata utilizzata la metrica Brisque in quanto empiricamente questa metrica ha sempre restituito lo stesso valore nonostante la variazione del livello di gamma.

Si è utilizzata una **gamma correction** globale invece che **locale** in quanto:

- le immagini degradate con gamma globale sono risultate più simili a quelle del validation set degradato sia a un confronto visivo sia per la qualità valutata con le metriche;
- si è preferito in ogni caso avere una maggiore degradazione delle immagini in modo da poter poi estrarre feature maggiormente invarianti.

In ogni caso è stata implementata anche una gamma correction locale adattativa prendendo spunto dalla principale letteratura di riferimento[1].



Immagine originale



Immagine degradata

5 - Analisi del Validation set Degradato: a) individuazione dei difetti

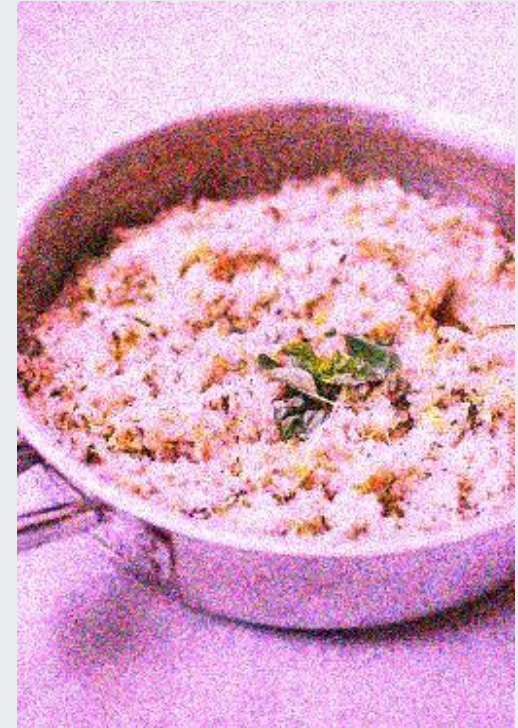
- **Rumore gaussiano.**

Molte immagini sono affette da rumore gaussiano. Si è quindi proceduto a calcolare questo rumore. In particolare:

- per ogni immagine ottenuta dal passo precedente viene applicato l'algoritmo di **Noise Level Estimation** citato in precedenza[1] per determinare se l'immagine è affetta da rumore gaussiano;
- in caso affermativo l'immagine viene convertita in scala di grigi con la stessa modalità esposta per il contrasto;
- poi viene applicato all'immagine un rumore gaussiano ottenuto con diversi valori della potenza di disturbo (varianza del rumore gaussiano);
- viene accettato il valore di rumore gaussiano che porta al valore di **Brisque** più vicino a quello della corrispondente immagine del validation set degradato.

Si è verificato a campione che le immagini erano affette (non da rumore **salt and pepper**, ma) da rumore gaussiano. Infatti:

- il rumore non veniva corretto tramite filtro mediano;
- l'istogramma dei colori dell'immagine degradata era più simile a quello di un'immagine affetta da rumore gaussiano.



5 - Analisi del Validation set Degradato: a) individuazione dei difetti

- Le attività precedenti hanno portato a individuare e a quantificare i difetti delle immagini del validation set degradato. Il tempo computazionale di queste attività è stato di 34231 secondi (circa 9 ore e mezza).
- Si è quindi proceduto ad applicare questi difetti alle corrispondenti immagini del validation set non degradato in modo da verificare la correttezza della procedura anche attraverso un giudizio umano soggettivo. Si riportano qui alcuni esempi dei risultati ottenuti:

Immagine pulita originale vs. Immagine degradata originale vs. Immagine degradata nuova



Immagine pulita originale vs. Immagine degradata originale vs. Immagine degradata nuova



Immagine pulita originale vs. Immagine degradata originale vs. Immagine degradata nuova



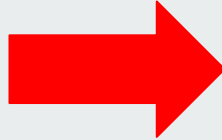
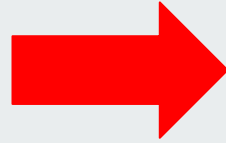
5 - Analisi del Validation set Degradato: b) replicazione dei difetti

- Successivamente all'individuazione dei difetti del validation set degradato, si è passati alla replicazione di questi difetti sul train set. In particolare:
 - sono state prese in considerazione (i) le immagini train set bilanciato come specificato in precedenza e (ii) le immagini del validation set degradato;
 - si è calcolata la distribuzione delle istanze nelle classi del train set e del validation set;
 - si è proceduto ad applicare i difetti del validation set degradato al train set con questo ordine:
 - i difetti della prima immagine degradata alla prima immagine del train set;
 - i difetti della seconda immagine degradata alla seconda immagine del train set; e così via;
 - quando le immagini degradate della classe sono terminate, si ricomincia da capo finché tutte le immagini del train set non sono state degradate.

Le immagini del train set degradato sono state salvate in formato JPG. Il tempo computazionale della replicazione dei difetti è stato di 4449 secondi (circa 1 ora e mezza).

5 - Analisi del Validation set Degradato: b) replicazione dei difetti

- Si riportano alcuni esempi di replicazione dei difetti sul train set:



5 - Analisi del Validation set Degradato: c) Restaurazione Validation set degradato



- Individuati i principali difetti (Blur gaussiano, Rumore gaussiano, Saturazione dei vari canali colore) si è cercato di restaurare il validation set degradato.
- Purtroppo il fatto che tutte le immagini presentino sì questi difetti ma non tutti nello stesso modo e non tutti i difetti allo stesso tempo ma sparsi in modo aleatorio sulle immagini hanno reso inappropriato la creazione di un algoritmo che migliori la qualità delle immagini in automatico.
- Per restaurare il validation set degradato era necessaria una restaurazione di ogni immagine manualmente, ma essendoci quasi 12 000 immagini si è trovata questa soluzione inappropriata e dispendiosa.
- Si è concluso quindi che per affrontare il problema di classificazione del validation set degradato sarebbe stato più opportuno cercare di capirne i difetti e ricrearli come abbiamo visto in precedenza in modo da poter usare un approccio di Data Augmentation.

5 - Analisi del Validation set Degradato:



Per completezza di seguito mostriamo alcune immagini restaurate, il tempo del processing di restaurazione per ogni immagine era variabile e non pensabile su tutto il validation set degradato.

(Inoltre non sempre si riusciva a portare le immagini a un buon stato data la forte intensità dei difetti)

Soluzione a Rumore Gaussiano:

- Si itera l'immagine applicando un filtro gaussiano con valori di sigma compresi tra $[0.01 - 3]$,
- si calcola il valore di BRISQUE
- si prende l'immagine migliore con il BRISQUE più basso.

Soluzione a Blur Gaussiano:

Si applica un filtro di correlazione gaussiano su tutta l'immagine di dimensione $[3 \ 3]$ con deviazione standard pari a 0.5

Soluzione a White Balance:

Si usano gli algoritmi di color constancy (forniti dal professore Bianco) Grey World, Shades of Gray e max-RGB, i quali calcolano i migliori valori da moltiplicare ad ogni canale RGB dell'immagine per renderla "naturale".

5 - Analisi del Validation set Degradato:

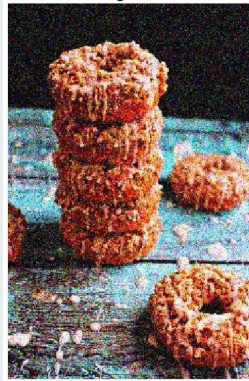
Degradata



Restaurata



Degradata



Restaurata



Restaurata



Degradata



Degradata



Restaurata



6 - Primi Test Computazionali (Processamento dei dati)



Approccio ambiente MatLab - osservazione su soglia da utilizzare

Avendo i dataset di addestramento puliti per proseguire con il processo della classificazione si è voluto testare l'ambiente matlab col fine di capire fin da subito se sarebbe stato opportuno in termini di tempi e calcolo lavorare su questo ambiente.

Sebbene Matlab sia un ambiente molto utile e pratico, al momento di usare le CNN nelle sessioni di laboratorio si era riscontrato una efficienza bassa in assenza di GPU.

Si sono svolte delle sessioni di prova di feature extraction e classificazione KNN andando a prendere un certo numero "soglia" di immagini per classe dal training e test set usando come rete di prova AlexNet.

Dai nostri esperimenti con soglia 100 per il training set e 50 per test set i tempi di calcolo andavano da 1 ora a 3/4 ore sulle diverse macchine.

Si è quindi decise di continuare il processo di classificazione in ambiente Python usando Google Colab come ambiente di lavoro.

6 - Primi Test Computazionali (Processamento dei dati)



Approccio ambiente Python - Google Colab

Usando Google Colab e le varie librerie disponibili non ci sono stati problemi per caricare i nostri dataset.

Inoltre il fatto che l'ambiente Colab ci fornisca potenza di calcolo (GPU) è stato un vantaggio rispetto a Matlab, ripetendo lo stesso esperimento di feature extraction **senza soglia**, ossia usando tutto il dataset di training e di test ci sono voluti pochi minuti per completarlo.

Avendo quindi un ambiente efficiente in termini di tempo su cui lavorare abbiamo iniziato a documentarci su soluzioni in letteratura da cui prendere spunto e confrontarci per il nostro problema di classificazione.


7 - Verso le CNN (Feature Extraction / Transfer Learning / Fine Tuning)

- Per il task di classificazione si è deciso di effettuare delle operazioni di transfer learning (e più precisamente fine-tuning) utilizzando reti pre-addestrate. Questa soluzione è adottata dal paper di riferimento del dataset e da altri paper aventi per oggetto la classificazione di cibi[1]. E questa soluzione sembra ottenere risultati migliori rispetto alla classificazione tramite feature hand-crafted[2].
- A questo proposito si è deciso di utilizzare una CNN baseline di partenza. In particolare come baseline è stata scelta la rete menzionata nel paper del dataset FoodX-251, ossia ResNet101.
- Creata la baseline ogni membro del gruppo si è documentato e ha scelto una rete diversa su cui continuare in parallelo il task di classificazione. Alla fine si sono analizzati i risultati e confrontati i 3 modelli insieme alla baseline.
- Gli ulteriori modelli selezionati sono stati:
 - ResNet50
 - Mobilenet V2
 - EfficientNet B0

[1] Kaur P. et al., *FoodX-251: A Dataset for Fine-grained Food Classification*, arXiv:1907.06167v1 [cs.CV].

[2] Ciocca G., Napoletano P., Schettini R., *CNN-based Features for Retrieval and Classification of Food Images*, in *Computer Vision and Image Understanding*, 2018, secondo cui "features learned by deep Convolutional Neural Networks (CNNs) have been recognized to be more robust and expressive than hand-crafted ones".

a) ResNet101: introduzione



- Il fine-tuning della baseline ResNet101 è stato effettuato utilizzando Python e Google Colab, come già anticipato. In particolare è stato utilizzato il modello di ResNet101 pre-addestrato su ImageNet offerto da Keras[1].
- Il dataset per il fine-tuning è stato caricato tramite la funzione `image_dataset_from_directory` di Keras. Questa funzione riduce l'occupazione della memoria RAM al minimo, mantenendo solo i riferimenti ai file su disco. Quindi è stato possibile caricare e utilizzare tutto il dataset.
- L'intenzione originaria era quella di effettuare il fine-tuning con riaddestramento di tutti i layer quantomeno perché:
 - il paper di riferimento del dataset ha ottenuto risultati migliori con il fine-tuning di tutti i layer invece che del solo ultimo layer;
 - ImageNet contiene poche immagini di cibi e quindi è opportuno riaddestrare la rete sullo specifico oggetto della classificazione[2].
- Tuttavia per problemi di saturazione della memoria RAM è stato possibile riaddestrare solo metà rete.

[1] I modelli pre-addestrati offerti da Keras sono elencati al seguente indirizzo: <https://keras.io/api/applications/>.

[2] Ciocca G., Napoletano P., Schettini R., *CNN-based Features for Retrieval and Classification of Food Images*, in *Computer Vision and Image Understanding*, 2018, secondo cui “the ImageNet database [...] contains the lowest number of food classes” tra un insieme di database analizzati.

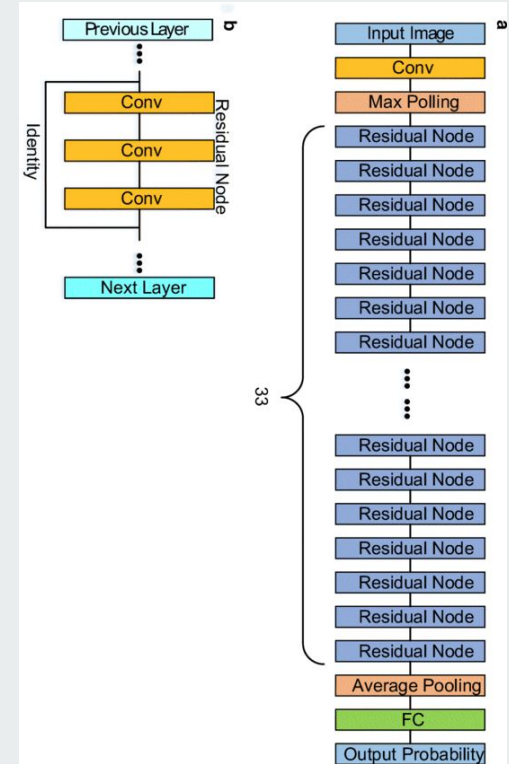
b) ResNet101: implementazione

Transfer Learning:

- freeze della prima metà della rete e riaddestramento della seconda metà;
- sostituzione dell'ultimo layer fully connected con un layer fully connected di 251 neuroni (pari al numero di classi) con funzione di attivazione softmax.

Parametri scelti (configurazione migliore):

- Batch size = 128;
- Loss function = Categorical Cross-Entropy
- Optimizer = Adam
- Epoche = 25 su train set bilanciato e 10 su train set degradato
- Learning Rate = $5e-5$ decrementato di un fattore 10 ogni 10 epoche con il train set bilanciato e ogni 5 con il train set degradato (in modo simile al paper del dataset)
- Metriche = Accuracy Top-1 e Top-3



Architettura di ResNet101[1]

b) ResNet101: risultati

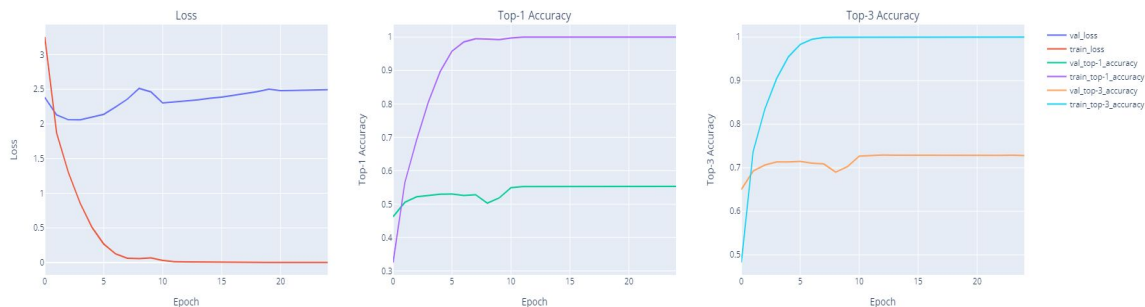
Risultati:

- ResNet101 addestrata sul train set bilanciato:

	Tempo train (s)	Top-1 Accuracy	Top-3 Accuracy
Test set pulito	20254	0.5856	0.7684
Test set degradato	20254	0.2120	0.3402

- ResNet101 addestrata sul train set degradato:

	Tempo train (s)	Top-1 Accuracy	Top-3 Accuracy
Test set pulito	17.040	0.5888	0.7780
Test set degradato	17.040	0.4177	0.6079



Addestramento su train set degradato

Osservazioni:

- L'addestramento sul train set degradato ha portato a un incremento dell'accuracy top-3 sul test set degradato superiore al 25%.
- Si è registrato un forte overfitting sia per la rete addestrata sul train set bilanciato che per la rete addestrata sul train set degradato.
- Le performance sul test set pulito non sono peggiorate a seguito dell'addestramento con il train set degradato. Anzi, sono leggermente migliorate.
- La classe che ha registrato le performance peggiori è la classe 205: lobster_roll_sandwich. In particolare: 0.10 precision, 0.09 recall e 0.10 f-measure sul test set degradato. Il supporto di questa classe è pari a 44 istanze.
- La classe che ha registrato le performance migliori è la classe 124: boiled_egg. In particolare: 0.76 precision, 0.77 recall e 0.77 f-measure sul test set degradato. Il supporto di questa classe è pari a 53 istanze.

b) ResNet50: introduzione



Cercando soluzioni in letteratura ci siamo ritrovati con un task simile già risolto nel 2018 dai professori Ciocca-Schettini-Napoletano nel paper “CNN-based Features for Retrieval and Classification of Food Images”.

Si è quindi cercato di prendere spunto dal loro esperimento anche se non nelle stesse condizioni computazionali per osservare il comportamento su un dataset simile come il nostro.

Rispetto al paper abbiamo preso solo il modello migliore risultato dell'esperimento, ossia ResNet50 addestrato sul dataset Imagenet. (L'architettura delle rete è molto simile a ResNet101 vista prima ma con meno blocchi di vconcoluzione)

Si è prima andato a testare la bontà delle features estratte dai nostri data set usando la rete addestrata su Imagenet. è stato fatto quindi un task di features extraction dall'ultimo layer della rete prima delle predizioni e poi la classificazione con l'algoritmo KNN.

Feature Extraction:

- Estrazione features Training set da layer “avg_pool”
- Estrazione features Test set da layer “avg_pool”
- Classificatore KNN - 5 vicini
- Calcolo predizioni sulle features Test set / Test set degradato

Risultati Ottenuti

Accuracy Test set: 0.005253

Accuracy Test set Degradato: 0.004752

Come si può vedere dai risultati, le features estratte non sono abbastanza discriminanti per ogni classe, si è deciso di proseguire con un approccio **transfer learning**.

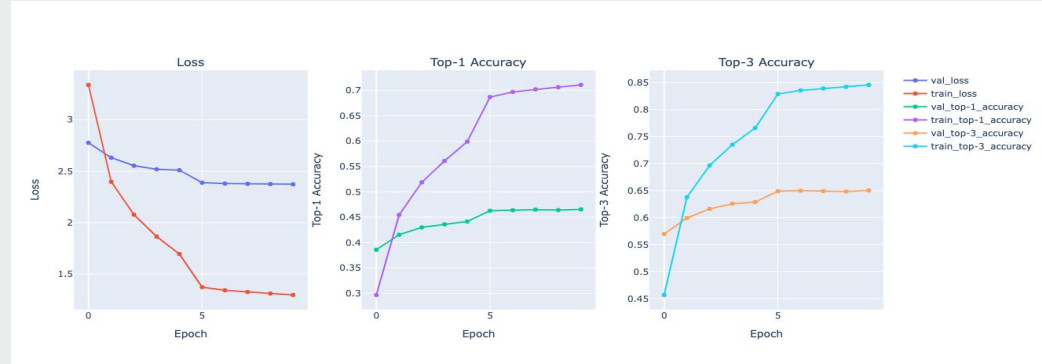
b) ResNet50: implementazione

Transfer Learning:

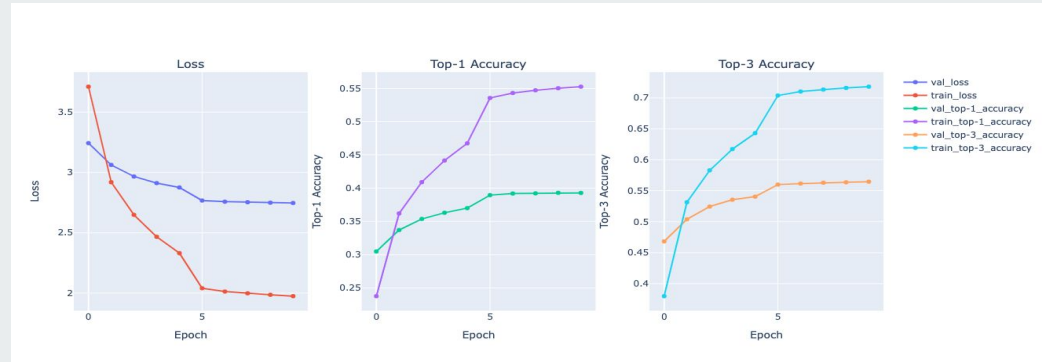
- Freeze dei Layer fino all'ultimo blocco di convoluzione
- Aggiunta nuova parte Fully connected a 251 predizioni con funzione di attivazione "Softmax"

Parametri scelti (Configurazione migliore):

- Batch size = 256
- Loss function = Categorical Cross-Entropy
- Optimizer = Adam (SGD stessi risultati ma più lento)
- Epoche = 10 (con 20 epoche stessi risultati)
- Learning Rate = 0.01, fattore decay 10 ogni 5 epoche
- Metriche = Accuracy Top 1 e Top 3



Addestramento su Training set Bilanciato



Addestramento su Training set Augmented

b) ResNet50: risultati

Risultati:

- I risultati ottenuti sono migliorati di molto rispetto all'approccio Feature Extraction, anche se persiste il problema di overfitting.
- La rete addestrata su entrambi i training set restituisce un'accuracy top 3 uguale su entrambi i validation set.
- C'è un incremento invece dell'accuracy nel validation set degradato con la rete addestrata col training set degradato, anche se comunque bassa.
- Presenza di Overfitting che non si è riusciti a risolvere nonostante diversi addestramenti con parametri diversi

ResNet50 addestrata su Training set bilanciato

	Loss	Accuracy	Top-3 accuracy
Test set	2.0825	0.4998	0.6962
Test set degradato	8.3496	0.0351	0.0737

ResNet50 addestrata su Training set Augmented

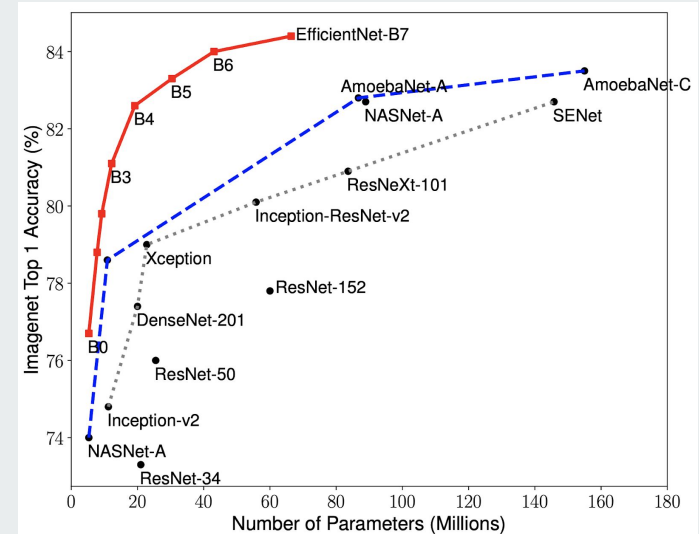
	Loss	Accuracy	Top-3 accuracy
Test set	2.1079	0.4996	0.6858
Test set degradato	5.8540	0.0692	0.1416

c) EfficientNetB0: introduzione

Cercando soluzioni in letteratura su problemi simili, nello specifico su dataset di immagini sul cibo, uno dei risultati più rilevanti ritrovato è stato *Fine-Grained Image Classification on Food-101*[1].

Tra le modalità presentate, tra quelle con le migliori performance c'è la CNN EfficientNet, nello specifico la versione B7, con un'accuracy di ben 93.0%.

Per il nostro problema, EfficientNet si è rivelata un'alternativa valida, in quanto ottimizzata per una computazione veloce ed efficiente. Tuttavia si è scelto di utilizzare la prima versione della rete (B0), che oltre ad essere più light e veloce, offrendo comunque risultati accettabili, ci ha permesso di effettuare un fine-tuning più profondo della parte convoluzionale



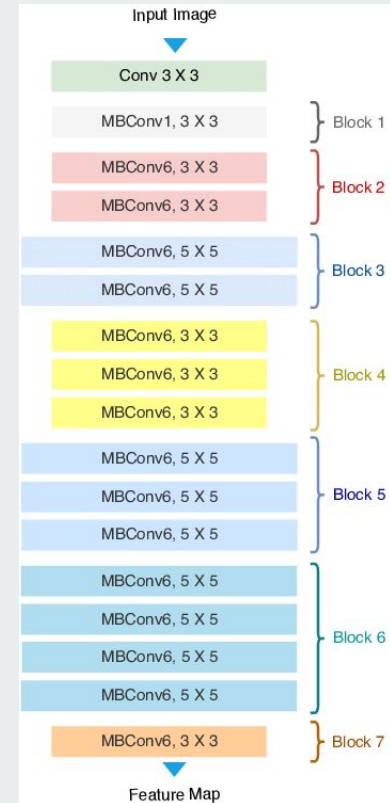
c) EfficientNetB0: implementazione

Transfer Learning:

- freeze dei primi 6 blocchi convoluzionali della rete e riaddestramento a partire da I settimo blocco;
- sostituzione dell'ultimo layer fully connected con un layer fully connected di 251 neuroni (pari al numero di classi) con funzione di attivazione softmax.

Parametri scelti (configurazione migliore):

- Batch size = 512;
- Loss function = Categorical Cross-Entropy
- Optimizer = Adam
- Epoche = 15 su entrambi i train set (bilanciato e degradato)
- Learning Rate = 0.001 (default)
- Metriche = Accuracy Top-1 e Top-3



Architettura di EfficientNetB0

c) EfficientNet B0: risultati



Risultati:

Con la versione B0 è stato possibile effettuare un Fine Tuning a partire dal 7° blocco convoluzionale, il massimo possibile con le risorse disponibili

EfficientNetB0 addestrata su Training set bilanciato

	Loss	Accuracy	Top-3 accuracy
Test set	2.0352	0.5682	0.7518
Test set degradato	5.7149	0.2020	0.3299

EfficientNetB0 addestrata su Training set Augmented

	Loss	Accuracy	Top-3 accuracy
Test set	1.9979	0.5692	0.7598
Test set degradato	2.9612	0.4035	0.5907

d) MobileNetV2: introduzione


- Come secondo modello di comparazione è stato individuato MobileNetV2. Le motivazioni sono quantomeno le seguenti:
 - MobileNetV2 è il modello più piccolo offerto da Keras come dimensione in MB e come numero di parametri. Questo ha permesso di effettuare un fine-tuning di quasi tutti i layer;
 - MobileNetV2 è il modello col minor tempo di inferenza su GPU insieme a MobileNetV1;
 - MobileNetV2 registra comunque un'alta accuracy come top-1 e top-5 su ImageNet rispettivamente pari al 71.3% e al 90.1%.

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7
NASNetMobile	23	74.4%	91.9%	5.3M	389	27.0	6.7
NASNetLarge	343	82.5%	96.0%	88.9M	533	344.5	20.0
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9
EfficientNetB1	31	79.1%	94.4%	7.9M	186	60.2	5.6
EfficientNetB2	36	80.1%	94.9%	9.2M	186	80.8	6.5
EfficientNetB3	48	81.6%	95.7%	12.3M	210	140.0	8.8
EfficientNetB4	75	82.9%	96.4%	19.5M	258	308.3	15.1
EfficientNetB5	118	83.6%	96.7%	30.6M	312	579.2	25.3
EfficientNetB6	166	84.0%	96.8%	43.3M	360	958.1	40.4
EfficientNetB7	256	84.3%	97.0%	66.7M	438	1578.9	61.6

Dati MobileNetV2[1]

[1] Questi dati sono disponibili su: <https://keras.io/api/applications/>.

d) MobileNetV2: implementazione



Transfer Learning:

- riaddestramento di quasi tutta la rete;
- sostituzione dell'ultimo layer fully connected con un layer fully connected di 251 neuroni (pari al numero di classi) con funzione di attivazione softmax;
- aggiunta di un precedente layer fully connected di 512 neuroni con funzione di attivazione ReLU e regolarizzazione L2 (con parametro lambda pari a 0.001) su pesi e bias. In questo modo si è cercato di ridurre l'overfitting;
- aggiunto anche un layer di dropout con probabilità pari a 0.3 per ridurre l'overfitting.

Parametri scelti (configurazione migliore):

- Batch size = 128;
- Loss function = Categorical Cross-Entropy
- Optimizer = Adam
- Epoche = 25 su train set bilanciato e 15 su train set degradato
- Learning Rate = 0.001 decrementato di un fattore 10 ogni 5 epoche
- Metriche = Accuracy Top-1 e Top-3

d) MobileNetV2: risultati

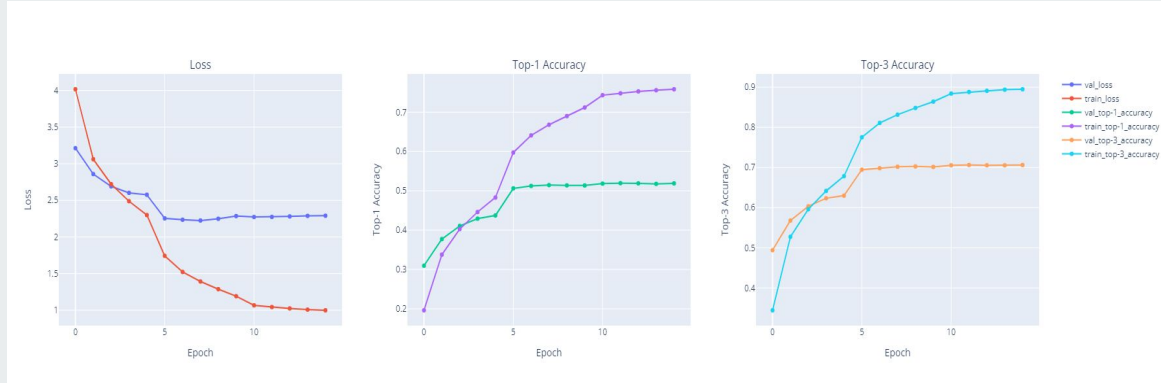
Risultati:

- MobileNetV2 addestrata sul train set bilanciato:

	Tempo train (s)	Top-1 Accuracy	Top-3 Accuracy
Test set pulito	8148	0.5726	0.7626
Test set degradato	8148	0.1375	0.2286

- MobileNetV2 addestrata sul train set degradato:

	Tempo train (s)	Top-1 Accuracy	Top-3 Accuracy
Test set pulito	10114	0.5562	0.7475
Test set degradato	10114	0.4257	0.6115




Addestramento su train set degradato

Osservazioni:

- L'addestramento sul train set degradato ha portato a un incremento dell'accuracy top-3 sul test set degradato quasi pari al 40%.
- Si è registrato un forte overfitting sia per la rete addestrata sul train set bilanciato che per la rete addestrata sul train set degradato.
- Le performance sul test set pulito sono peggiorate dell'1.5% a seguito dell'addestramento con il train set degradato. Pertanto il peggioramento è lieve.
- La classe che ha registrato le performance peggiori è la classe 20: foie_gras. In particolare: 0.00 precision, 0.00 recall e 0.00 f-measure sul test set degradato. Tuttavia il supporto di questa classe è pari a sole 2 istanze.
- La classe che ha registrato le performance migliori è la classe 195: oyster. In particolare: 0.83 precision, 0.74 recall e 0.78 f-measure sul test set degradato. Il supporto di questa classe è pari a 58 istanze.

e) Valutazione approccio features hand-crafted: SIFT



Col training set pulito si è andati a valutare un primo possibile approccio di classificazione con feature hand-crafted.

Estrazione SIFT: si è pensato a questo tipo di descrittore dato che è invariante a scala e rotazione, quindi potenzialmente utile nel nostro task.

Abbiamo fatto delle prove con un numero ridotto di immagini per classe 100 (25000 immagini in totale di training)
Purtroppo i tempi di calcolo e la memoria necessaria richiesta dall'algoritmo era diventato un ostacolo, anche in ambiente Colab.

Provare con un numero inferiore di immagini (es: 10/20) ci è sembrato inutile perché si andava ad sprecare la maggior parte delle immagini di training utili per la classificazione.

Si è deciso di abbandonare questo approccio in fase iniziale.

e) Valutazione approccio features hand-crafted: LBP



- Si è vagliata l'opportunità di estrarre descrittori di tipo Local Binary Pattern (LBP). Si tratta di un descrittore utile per la classificazione di texture.
- Tuttavia non si è proceduto alla classificazione tramite queste feature per diverse ragioni:
 - Anzitutto si è provato a calcolare queste feature su MATLAB. Il tempo computazionale era elevato: circa 5000 immagini in 30 minuti. Allora si è proceduto a calcolare gli LBP solo su 2500 immagini di training e 2500 immagini di test. L'accuracy top-1 ottenuta con un 1-NN è stata molto bassa: 0.0032. Le feature pesavano circa 400 MB per 2500 immagini. Si è provato poi a calcolare le feature con più immagini. Ma il calcolo ha esaurito le risorse di memoria.
 - Successivamente si è provato a calcolare queste feature su Python. Il tempo computazionale per il calcolo è stato basso. In questo caso il problema ha riguardato la RAM. La RAM è stata saturata nel tentativo di calcolare le feature su tutto il dataset. E salvare i risultati (eventualmente anche intermedi) è stato proibitivo dato che per 10000 immagini le feature pesavano circa 4.5 GB.
 - In ogni caso: "features learned by deep Convolutional Neural Networks (CNNs) have been recognized to be more robust and expressive than hand-crafted ones"[1].

f) Esempi di misclassificazione

- Si riporta un esempio di misclassificazione operata dalla rete ResNet101 sul test set pulito:
 - Questa immagine appartiene alla classe 0 macaron. La rete ha classificato questa immagine come appartenente alle classi (top-3): carrot_cake, biryani, croquette. Incidentalmente questa immagine è stata misclassificata anche da MobileNetV2. In questo caso la misclassificazione è evidentemente data dal fatto che il cibo corretto risulta mescolato ad altri cibi.
- Si riporta anche un esempio di misclassificazione operata dalla rete ResNet101 sul test set degradato:
 - Questa immagine appartiene alla classe 0 macaron. La rete ha classificato questa immagine come appartenente alle classi (top-3): ziti, club_sandwich, garlic_bread. In questo caso la degradazione dell'immagine sembra aver inciso in maniera significativa sulla classificazione.




g) Conclusioni: Confronto modelli e Modello migliore

- Si riporta un prospetto di sintesi dei risultati ottenuti come top-1 e top-3 accuracy dai vari modelli addestrati sul train set degradato:

	Top-1 Accuracy Test set pulito	Top-3 Accuracy Test set pulito	Top-1 Accuracy Test set degradato	Top-3 Accuracy Test set degradato
ResNet50	0.4996	0.6858	0.0692	0.1416
ResNet101	0.5888	0.7780	0.4177	0.6079
MobileNetV2	0.5562	0.7475	0.4257	0.6115
EfficientNetB0	0.5692	0.7598	0.4035	0.5907

g) Conclusioni: Confronto modelli e Modello migliore



Sebbene tutti e 3 i modelli testati raggiungono un'accuracy Top1 attorno al 50% e Top3 70% sul validation set pulito, la differenza si è vista nei risultati sul Validation set degradato.

Dai risultati ottenuti sul Validation set degradato dai diversi modelli si può affermare che **il miglior modello** per il nostro dataset risulta essere:

MobileNetV2 con accuracy **Top1: 0.4257** e **Top 3: 0.6115**

Il fatto di essere un rete molto piccola e il vantaggio di poter eseguire fine tuning su quasi tutta la rete in un ambiente limitato come Colab ci ha permesso di aumentare l'accuracy anche sul validation set degradato rispetto agli altri modelli testati.

Possibili cause di un'accuracy modesta nella classificazione:

- La presenza di similarità tra molte classi (e leggera varianza intra classe) si riflette nei risultati ottenuti.
- Nelle immagini di addestramento il piatto/cibo non è sempre in primo piano oppure non è presente in buona parte dell'immagine come nel validation set, molte classi di cibo verranno imparate con caratteristiche diverse a quelle nel validation set;
(**Background clutter**, Il rumore dello sfondo rende difficile cercare il cibo nell'immagine)

Nonostante ciò i risultati ottenuti dalla nostra rete migliore possono considerarsi buoni.

8 - Similarity Retrieval: valutazione oggettiva con metrica $P(n)$

- Il task di similarity retrieval è stato affrontato sotto due profili:
 - si è misurata la qualità della similarity retrieval attraverso una metrica oggettiva;
 - si è valutata la qualità della similarity retrieval attraverso un'osservazione visuale soggettiva.
- La valutazione oggettiva è stata effettuata come segue:
 - Per la valutazione oggettiva è stata utilizzata la metrica $P(n)$. Questa metrica è definita come: $P(n) = Q_n / Q$ dove Q è il numero di immagini di query e Q_n è il numero di query corrette tra le prime n immagini recuperate. In questo caso si è scelto n pari a 3. Questa metrica è utilizzata spesso per il task di similarity retrieval[1,2].
 - In questo caso sono state utilizzate le immagini del validation set come immagini di query e le immagini del train set (originale e depurato delle immagini irrilevanti) come dataset per valutare la qualità della similarity retrieval. In particolare:
 - per tutte le immagini del test set e del train set sono state estratte le rispettive feature attraverso la rete ResNet101;
 - l'estrazione è avvenuta rimuovendo l'ultimo layer (di classificazione) della rete e utilizzando la rete "tagliata" per effettuare le predizioni sul test set e sul train set. L'output di questa operazione sono le feature delle immagini;
 - le prime 3 immagini associate alla query sono state selezionate utilizzando un K-NN con $K = 5$. Come distanza è stata utilizzata la distanza Euclidea;
 - vengono restituite le 3 immagini del train set che presentano le feature più vicine a quelle dell'immagine di query. La query ha una risposta corretta se le immagini restituite del train set appartengono alla stessa classe dell'immagine di query.
- Risultati ottenuti con ResNet101:
 - $P(n) = 1.4651$ per il test set pulito
 - $P(n) = 0.9341$ per il test set degradato

Avendo utilizzato $n = 3$, il valore massimo raggiungibile di $P(n)$ è 3. In questo caso la similarity retrieval risulta corretta (i) per circa la metà dei risultati restituiti utilizzando il test set pulito e (ii) per circa un terzo dei risultati restituiti utilizzando il test set degradato.

[1] Ciocca G., Napoletano P., Schettini R., *Learning CNN-based Features for Retrieval of Food Images*, in *New Trends in Image Analysis and Processing – ICIAP 2017*, 2017.

[2] Farinella G.M. et al., *Retrieval and Classification of Food Images*, in *Computers in Biology and Medicine*, 2016.

8 - Similarity Retrieval: valutazione soggettiva

- La valutazione soggettiva è stata effettuata come segue:
 - è stata presa un'immagine di query da fonti aperte;
 - sono state calcolate le feature di questa immagine come visto per la valutazione oggettiva;
 - sono state riprese le feature del test set calcolate per la valutazione oggettiva;
 - si è calcolata la distanza Euclidea tra le feature dell'immagine di query e le feature delle immagini del test set;
 - sono state recuperate le 3 immagini del test set le cui feature hanno distanza minima dalle feature dell'immagine di query;
 - infine queste 3 immagini vengono mostrate per verificare quanto siano simili all'immagine di query da un punto di vista soggettivo.
- Si riporta un esempio:



Immagine di query:
torta al cioccolato



Primo risultato:
classe 115 torte



Secondo risultato:
classe 68 chocolate_cake



Terzo risultato:
classe 225 boston_cream_pie

9 - Conclusioni

- Il presente lavoro ha avuto per oggetto la classificazione e la similarity retrieval su un dataset di cibi. In particolare si sono volute ottenere le performance migliori su un validation set degradato.
- Per ottenere le performance migliori si è proceduto ad attività di pulizia e bilanciamento del dataset. Inoltre si è proceduto a un'attività di data augmentation consistente nella replicazione dei difetti del validation set degradato sul train set.
- Queste attività hanno permesso di migliorare le performance di classificazione sul validation set degradato in modo significativo. In particolare la top-3 accuracy è stata migliorata di almeno il 25% (con ResNet101) e fino al 40% (con MobileNetV2). In generale la top-3 accuracy si è attestata a circa il 75% sul validation set pulito e a circa il 60% su validation set degradato.
- Infine si è proceduto alla valutazione della similarity retrieval. La similarity retrieval ha portato a individuare le 3 immagini del train set più simili all'immagine di query del validation set. Una valutazione oggettiva di questo task ha portato a verificare che:
 - le immagini del validation set pulito hanno ottenuto in media circa una risposta corretta su 2;
 - le immagini del validation set degradato hanno ottenuto in media circa una risposta corretta su 3.
- In questo quadro le buone performance conseguite sul validation set degradato consentono di trarre le seguenti conclusioni: (i) da un lato le attività di pre-processing si sono rivelate utili per la classificazione e la similarity retrieval; (ii) dall'altro lato risulta confermata la tesi avanzata in letteratura secondo cui "the accuracy is inversely proportionate to the number of classes while being directly proportionate to the sample size per class of the dataset"[1].



**GRAZIE PER
L'ATTENZIONE**

Extra slide: Noise Level Estimation

- Algoritmo per il Noise Level Estimation[1]:
 - L'immagine viene decomposta in patch parzialmente sovrapposte. Ogni patch viene definita in forma vettoriale con un vettore di rumore gaussiano. Si vuole calcolare la deviazione standard data dal rumore gaussiano.
 - Per calcolare la deviazione standard si calcola la direzione della varianza minima attraverso la PCA.
 - Infatti la direzione della varianza minima è l'autovettore associato all'autovalore minimo della matrice di covarianza:

$$\Sigma_y = \frac{1}{N} \sum_{i=1}^N (y_i - \mu)(y_i - \mu)^T,$$

dove N è il numero di dati e μ è la media del dataset $\{y_i\}$.

- La varianza dei dati proiettati sulla direzione della varianza minima è uguale all'autovalore minimo della matrice di covarianza.
- Se si considerano weak textured patches dell'immagine con rumore, allora l'autovalore minimo della matrice di covarianza è circa 0. Questo facilita la stima del rumore. Più precisamente:
 - Viene stimato un livello iniziale di rumore dalla matrice di covarianza. Questo rumore viene stimato usando tutte le patch dell'immagine rumorosa.
 - Poi viene calcolata una soglia per determinare le weak textured patches.
 - Quindi vengono determinate le weak textured patches.
 - Dopodiché viene stimato nuovamente il livello di rumore.
 - Il processo viene iterato finché il livello di rumore rimane stabile (non cambia più).

Extra slide: SMOTE

- Synthetic Minority Oversampling Technique:
 - Tecnica statistica per bilanciare il dataset effettuando un oversampling sulle classi minoritarie, mentre le maggioritarie non vengono toccate.
 - Per ciascuna classe di target, l'algoritmo effettua un sampling del *feature space* dell'istanza da replicare e dei suoi nearest neighbors, generando così nuovi esempi a partire dalla combinazione di queste features.
 - Sebbene sia un algoritmo allo stato dell'arte, non è molto popolare il suo utilizzo su dataset di immagini. Di seguito alcuni esempi di immagini generate, e quindi il motivo per cui si è deciso di scartarlo:

