

Robust Food Classification Project

Visual Information Processing and Management
Università degli Studi di Milano-Bicocca

Matteo Breganni 869549
Lorenzo Monti 869960



Esplorazione del Dataset

- 251 Categorie
- 20 Immagini per classe
- Immagini di **dimensioni** diverse
- Immagini **errate**
- Immagini non esattamente appartenenti alla categoria

Image 1 - Size: 256x309



Image 2 - Size: 256x331



Image 3 - Size: 384x256



Image 2 - Size: 256x768

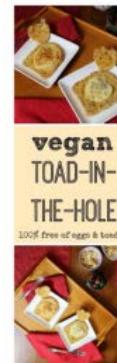


Image 2 - Size: 224x224

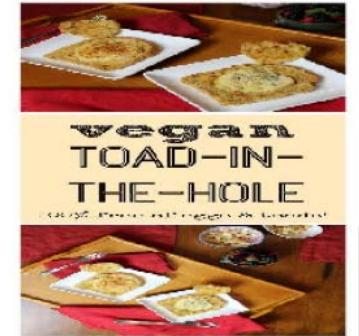


Image 5 - Size: 340x256



Image 6 - Size: 387x256



Image 7 - Size: 256x384



Image 8 - Size: 256x384



Esplorazione del Dataset

- Categorie a volte molto **ampie**

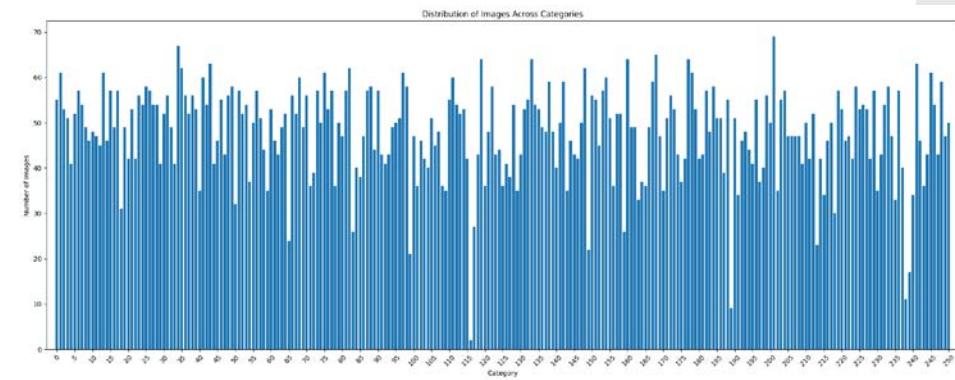
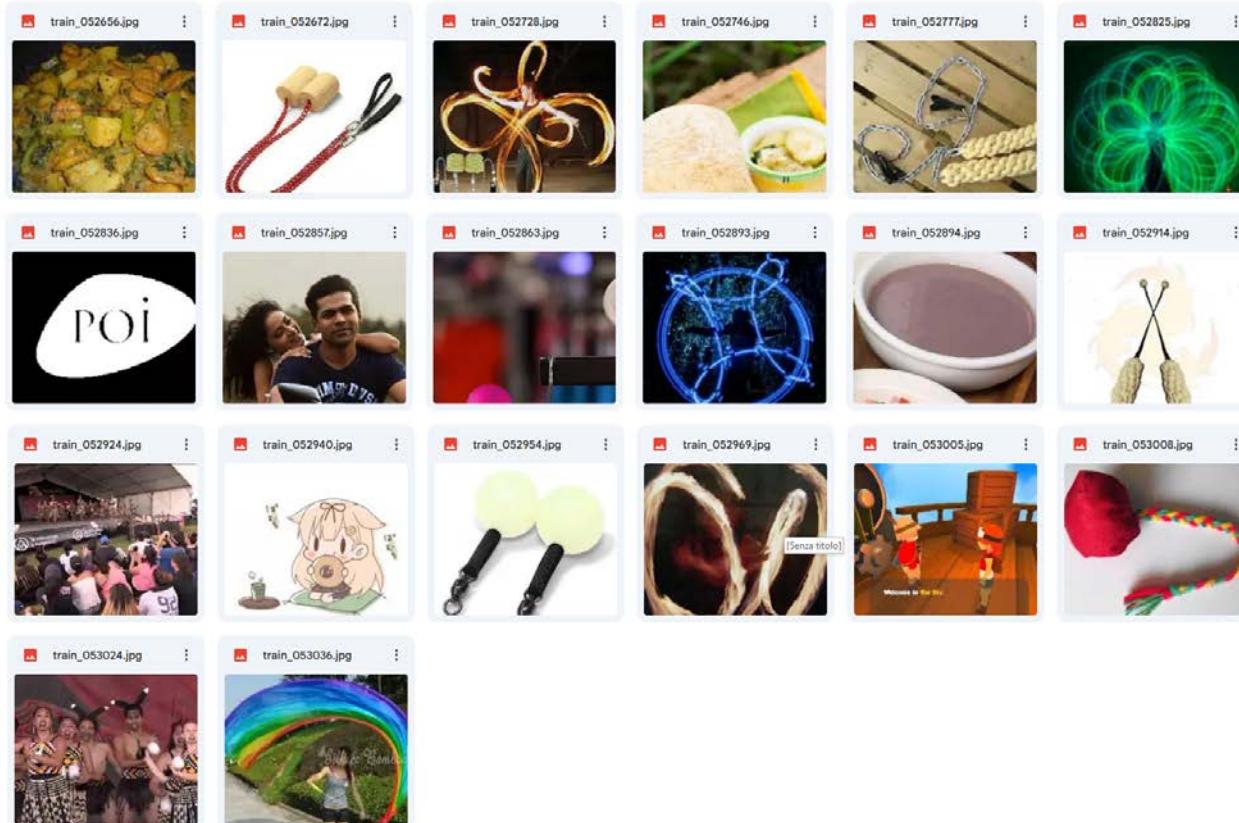
Dal validation set:



Esplorazione del Dataset

- Categorie con principalmente **immagini sporche**
- Relativo validation set **piccolo**
 - Validation set sbilanciato ----->

Training set:



Validation set:



Approcci Iniziali: Feature Extraction

- **Features extraction** con il modello pre-trainato di MobileNetV2 sul dataset Imagenet

Metodi:

- **KNN**
 - 15% di Accuracy
- **SVM**
 - 27% di Accuracy
 - Con ottimizzazione parametri: 28% di Accuracy
- Tentativo di utilizzare il penultimo layer al posto dell'ultimo per feature extraction:
 - ~10GB di array di features vs ~80MB



Approcci Iniziali: CNN Fatta a Mano

Approccio generale:

- Data loaders per ridurre la RAM utilizzata
- Validation set ridotto durante il training per ridurre le computazioni (50 batches)
- Immagini pre-processate e labels con one-hot encoding, softmax output

CNN creata e allenata da zero:

- Normalizzazione tra 0 e 1
- **Performance molto basse** causate dalla bassa grandezza del dataset
 - Test Loss: ~9.70
 - Test Accuracy (Top-1): ~1.60%
 - Top-5 Accuracy: ~5.70%

Layer	Output shape	Param #
Input	(224, 224, 3)	/
Conv, 16, (3, 3), stride 2	(111, 111, 16)	448
Batch normalization	(111, 111, 16)	64
Conv, 32, (3, 3), stride 2	(55, 55, 32)	4'640
Batch normalization	(55, 55, 32)	128
Conv, 64, (3, 3), stride 2	(27, 27, 64)	18'496
Batch normalization	(27, 27, 64)	256
Conv, 128, (3, 3), stride 2	(13, 13, 128)	73'856
Conv, 128, (3, 3), stride 2	(6, 6, 128)	147'584
Flatten	(4608)	0
Dense, 256, 50% dropout	(256)	1'179'904
Output (softmax)	(251)	64'507

Approcci Iniziali: CNN Fine-Tunata

CNN Fine tunata MobileNetV2:

- Pesi congelati sulla prima parte
- Layer fully connected con dropout aggiunto
- Buona performance
 - Test Loss: ~3.49
 - Test Accuracy (Top-1): ~25%
 - Top-5 Accuracy: ~50%
- Totale pesi: 3'042'619
- Totale pesi allenabili: 784'635

Layer	Output shape	Param #
Input	(224, 224, 3)	/
MobileNetV2	(7, 7, 1280)	2'257'984
GlobalAveragePooling	(1280)	0
Dense, 512, 65% dropout	(512)	655'872
Output	(251)	128'763

CNN Fine tunata DenseNet121:

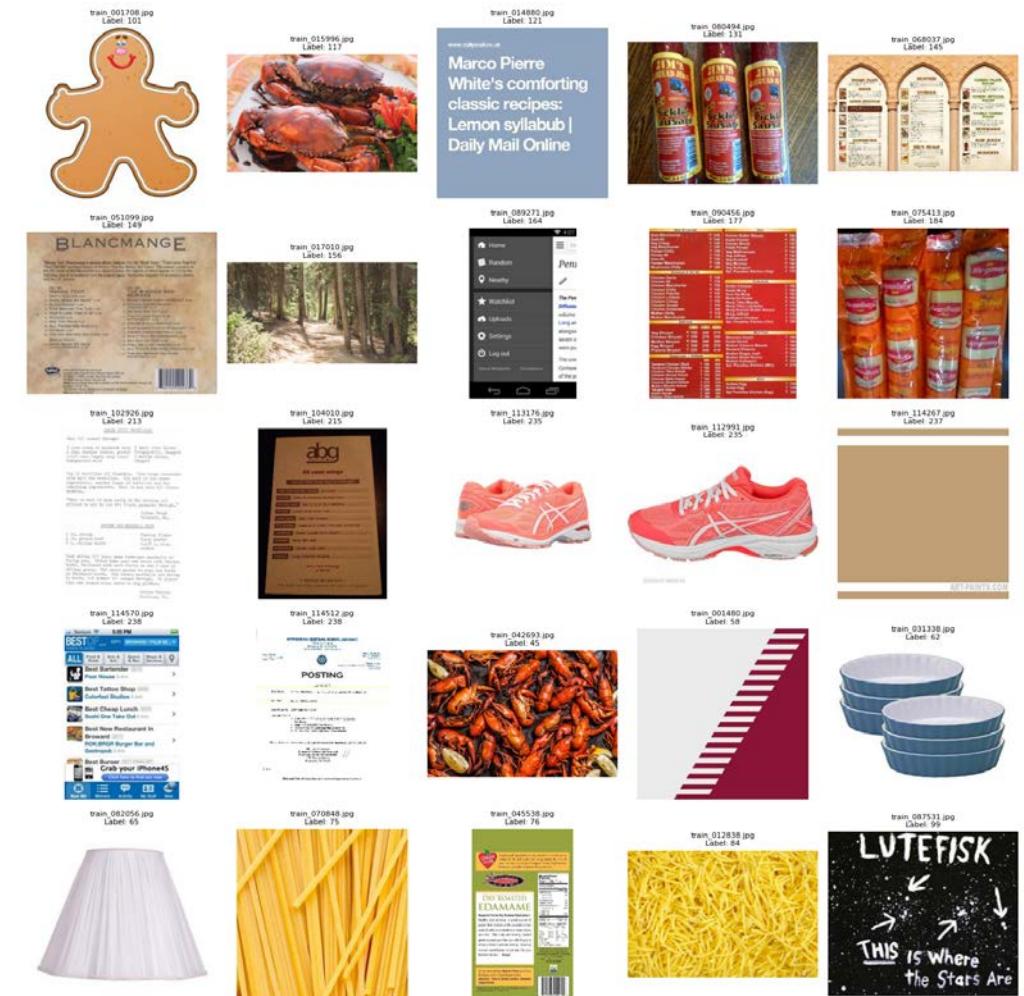
- Stesso funzionamento
- Rete più grande
 - Totale pesi: 7'691'067
 - Totale pesi allenabili: 653'563
- **Performance superiori**
 - Test Loss: ~3.28
 - Test Accuracy: ~26%
 - Top-5 Accuracy: ~53%
- Tempi simili per il training ma più alti per l'inferenza

Layer	Output shape	Param #
Input	(224, 224, 3)	/
DenseNet121	(7, 7, 1024)	7'037'504
GlobalAveragePooling	(1024)	0
Dense, 512, 65% dropout	(512)	524'800
Output	(251)	128'763

Pulizia del Training Set: Metodi

- Metodi per il riconoscimento degli outliers nel train set
 - Obiettivo: **rimuovere le immagini sporche** per migliorare la performance
- Test per valutare l'efficacia di ciascun metodo:
 - Tuning della sensibilità di ciascun metodo per ritornare **25 outliers**
 - Calcolo manuale di 2 score verificando sul val set:
 - Numero di immagini correttamente flaggate
 - Stesso score, sommato agli edge cases
- Metodi:

		score1	score2
• K-Means	->	7/25	9/25
• DBSCAN	->	15/25	17/25
• Isolation Forest	->	15/25	17/25
• NearestNeighbors	->	20/25	22/25
• OneClassSVM	->	6/25	7/25
• KD-Tree	->	8/25	10/25
- Performance diverse e non ottime
- La sensibilità di ciascun metodo può essere adattata per ritornare un diverso numero di immagini.

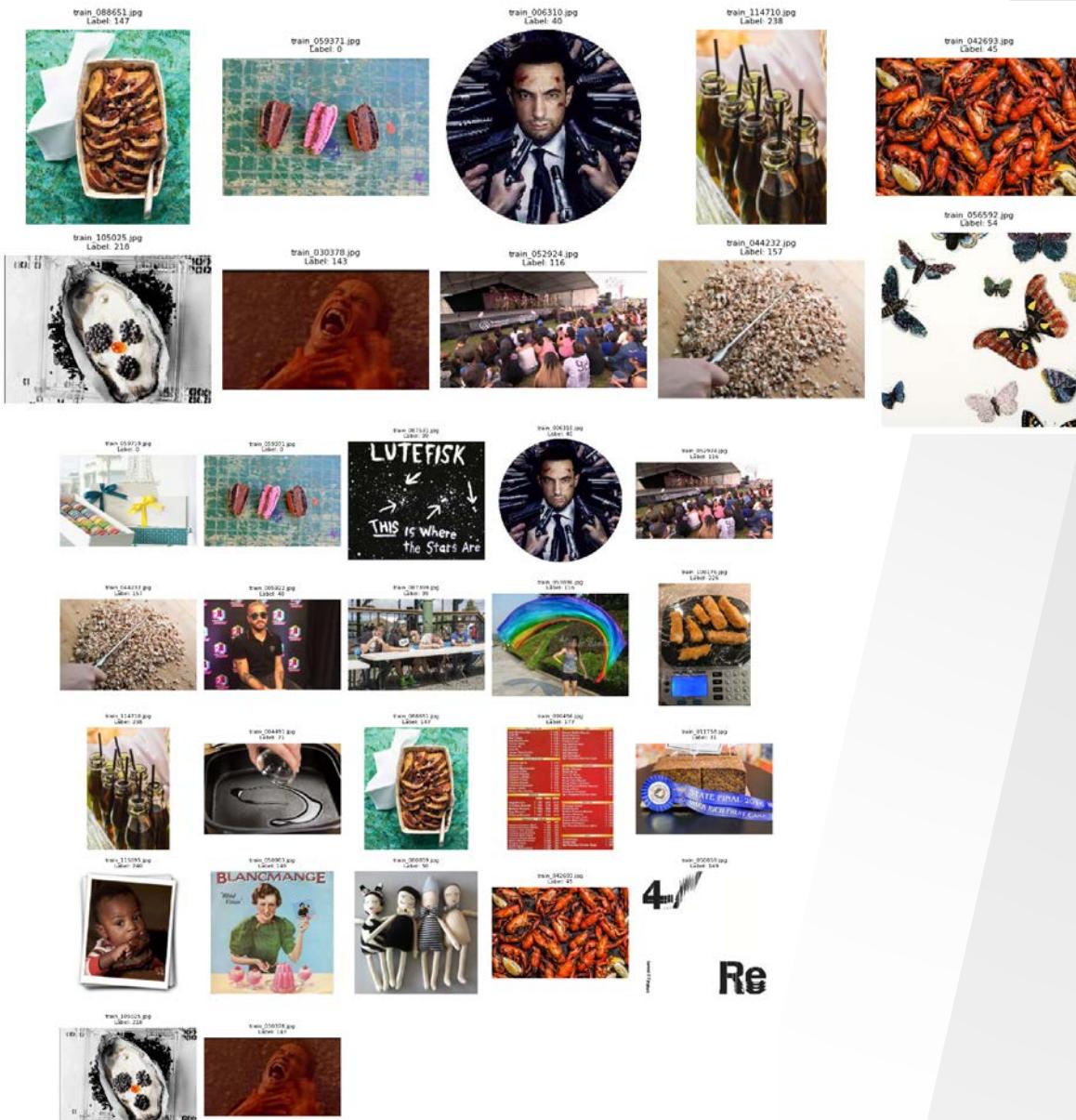


25 outliers ritornati dal metodo NearestNeighbors

Pulizia del Training Set: Consenso

- Performance dei metodi troppo basse per essere usati
- Rischio di rimuovere troppe immagini buone supera i benefici di togliere quelle sporche
- **Metodi di consenso:**
 - Metodi diversi possono commettere errori diversi
 - Outliers trovati da almeno n metodi
 - Risultato con n=3 sui test da 25 immagini --->
 - Metodo non ottimo perché ogni metodo ha la stessa capacità di voto
 - **Consenso pesato** maggiore di n
 - Utilizzando gli score calcolati come stima di efficacia
 - Calcolo pesi (somma a 1)
 - Risultato test con soglia 0.3 ----->

Metodo	Score1	Score2	Peso1	Peso2
K-Means	7/25	9/25	0.1	0.11
DBSCAN	15/25	17/25	0.21	0.21
Isolation Forest	15/25	17/25	0.21	0.21
NearestNeighbors	20/25	22/25	0.28	0.27
SVM	6/25	7/25	0.08	0.09
KD-Tree	8/25	10/25	0.11	0.12



Pulizia del Training Set: Risultati

- **500 immagini per metodo**
 - Cambiando la sensibilità di ciascuno
- Applicando la **funzione di consenso pesata**
- Threshold a 0.5, rimuovendo 422 immagini:
 - Test Loss: ~3.40
 - Test Accuracy: ~25%
 - Top-5 Accuracy: ~51%
 - **Performance peggiorata**, i benefici di rimuovere le immagini sporche non superano i problemi del togliere immagini buone
- Threshold a 0.8, rimuovendo 91 immagini:
 - Test Loss: ~3.32
 - Test Accuracy: ~ 26%
 - Top-5 Accuracy: ~ 53%
 - **Performance simili a prima**
- Come migliorare?
 - Provare un numero diverso di outliers da ritornare per metodo (richiede tempo per fare tuning manual dei parametri)
 - Provare threshold diversi per la funzione di consenso
 - **Sfruttare più dati** per avere outliers migliori

Pulizia del Training Set: Unlabelled

- Idea: sfruttare i dati dell'**unlabelled training set** per migliorare gli outliers trovati da ciascun metodo
 - Più dati aiutano a definire i clusters meglio
 - Il dataset unlabelled sembra avere molte meno immagini sporche
 - **30% del dataset unlabelled usato**, per motivi di computazione (34'036 immagini)
- **Ricalcolo** dei 500 outliers per metodo
 - Funzione di consenso pesata, con gli stessi pesi precedenti
 - Supponendo che l'efficienza relativa di ogni metodo rimanga invariata
- Threshold 0.90 -> 95 immagini ritornate
 - Il threshold necessario per avere un numero simile di immagini è aumentato, indice che **la «confidenza» del metodo è migliorata**
- Performance raggiunte:
 - Test Loss: ~3.22 (**-3%~***)
 - Test Accuracy: ~28% (**+7%~***)
 - Top-5 Accuracy: ~55% (**+4%~***)
- **Possibili miglioramenti:**
 - Tuning numero outliers per metodo, e threshold della funzione di consenso
 - Ricalcolo dei pesi di efficacia relativa

Data Augmentation

- Può essere utile dato il piccolo numero di immagini nel train set
- Augmentation applicata al miglior dataset pulito
- Data augmentation provate: **Traslazioni, Luminosità, Rotazione, Contrasto**



Validation set



Train set

Data Augmentation

- Può essere utile dato il piccolo numero di immagini nel train set
- Augmentation applicata al miglior dataset pulito

Trasformazioni provate singolarmente:

- **Traslazioni**
 - Traslation 0.1 -> Test Loss: 3.4143; Test Accuracy: 0.2496; Top-5 Accuracy: 0.5113
 - **Traslation 0.2** -> Test Loss: 3.4257; Test Accuracy: 0.2505; Top-5 Accuracy: 0.5135
 - Traslation 0.23 -> Test Loss: 3.5944; Test Accuracy: 0.2491; Top-5 Accuracy: 0.5079
- **Luminosità**
 - RandomBrightness 0.1 -> Test Loss: 3.2584; Test Accuracy: 0.2629; Top-5 Accuracy: 0.5381
 - **RandomBrightness 0.12** -> Test Loss: 3.4441; Test Accuracy: 0.2684; Top-5 Accuracy: 0.5307
 - RandomBrightness 0.2 -> Test Loss: 3.4787; Test Accuracy: 0.2480; Top-5 Accuracy: 0.5011
- **Rotazione**
 - RandomRotation 0.15 -> Test Loss: 3.6059; Test Accuracy: 0.2473; Top-5 Accuracy: 0.5088
 - **RandomRotation 0.2** -> Test Loss: 3.4162; Test Accuracy: 0.2540; Top-5 Accuracy: 0.5055
 - RandomRotation 0.6 -> Test Loss: 3.5778; Test Accuracy: 0.2214; Top-5 Accuracy: 0.4676
- **Contrasto**
 - **RandomContrast 0.1** -> Test Loss: 3.4983; Test Accuracy: 0.2405; Top-5 Accuracy: 0.4946
 - RandomContrast 0.15 -> Test Loss: 3.5525; Test Accuracy: 0.2250; Top-5 Accuracy: 0.4748
 - RandomContrast 0.2 -> Test Loss: 3.5525; Test Accuracy: 0.2250; Top-5 Accuracy: 0.4748
- Combinazioni di trasformazioni diverse

Modello Unfrozen e Modello Pesato

Modello iniziale migliorato:

- Dataset migliore dal data cleaning
- Modello **ottimizzato ulteriormente**

Test Loss: ~3.08 (-4%~*)

Test Accuracy: ~30% (+7%~*)

Top-5 Accuracy: ~57% (+4%~*)

Modello Unfrozen:

- Fine tuning ulteriore
 - Modello base **unfrozen** (DenseNet)
 - Learning rate più basso
- **Tempi** di training molto lunghi

Test Loss: ~2.96 (-4%~**)

Test Accuracy: ~32% (+7%~**)

Top-5 Accuracy: ~60% (+5%~**)

Modello pesato:

- Assegnazione di pesi alle classi, relativamente alla **distribuzione**
 - Procedimento di **cleaning** ha **sbilanciato** il dataset
 - Verrà sbilanciato ulteriormente nella sezione successive
- Performance che dovrebbero migliorare ma classi problematiche che hanno perso esempi (come la 116) hanno ancora immagini sporche
- Questo modello diventa la baseline e viene utilizzato nelle successive sezioni

Test Loss: ~3.07 (0%~**)

Test Accuracy: ~30% (0%~**)

Top-5 Accuracy: ~58% (+2%~**)

*Aumento percentuale rispetto al modello migliore del data cleaning con dataset unlabelled

**Aumento percentuale rispetto al modello iniziale migliorato

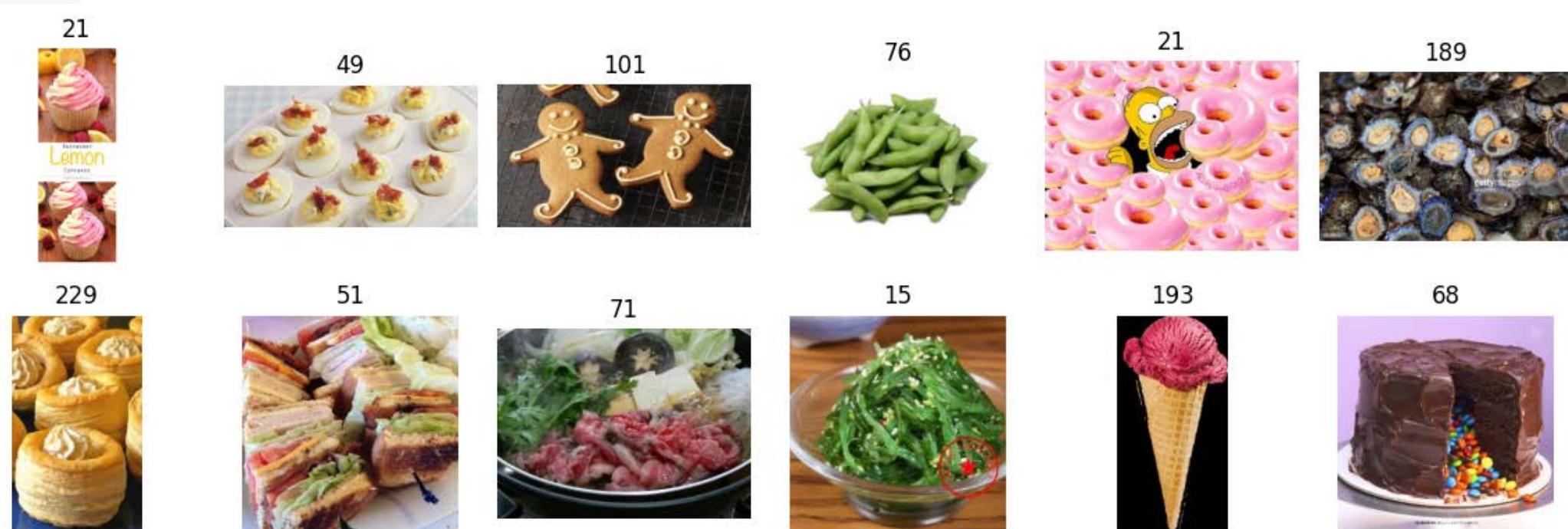
Train Set Unlabelled

- **Aumento dimensione train set aggiungendo dal dataset senza label**
- **Metodi** per selezionare dati da aggiungere, usando il modello pesato per trovare le labels:
 - Confidenza della top class > soglia
 - Confidenza della top class con margine > soglia rispetto alla confidenza della seconda classe
- **Approccio iterativo:**
 - Set della confidenza per ritornare immagini classificate bene (**iniziando con confidenza alta, abbassata dopo ogni iterazione**)
 - Filtrare le immagini ritornate dal metodo, togliendo quelle già aggiunte
 - Limitare al massimo 3 immagini aggiunte per classe ogni iterazione per ridurre lo sbilanciamento
 - **Re-training** del modello con il train set nuovo e ri-calcolando i pesi delle classi
- Dopo aver esaurito le immagini a confidenza molto alta, usare il secondo metodo
- Solo il **10% del dataset senza labels è stato utilizzato**
 - Sempre lo stesso 10%, non random ad ogni iterazione, per esaurire effettivamente le immagini ad alta confidenza

Train Set Unlabelled

Test iniziale del metodo, usando **1% del dataset** senza label, con soglia di confidenza al 90%:

- 19 immagini ritornate
 - Prime 12:



- Controllate manualmente rispetto al validation set, **tutte corrette tranne la quinta**

Train Set Unlabelled

Iterazioni utilizzando sempre lo stesso 10% del dataset senza labels (performance scalerebbero di conseguenza usando l'intero set di dati):

Metodo	Soglia	Immagini trovate	Meno imgs già usate	Massimo 3 per classe	Dimensione dataset	Loss	Top-1	Top-5
/	/	/	/	/	4'925	~3.07	~30%	~58%
Threshold	0.9	219	219	127	5'052	~3.05	~31%	~58%
Threshold	0.8	497	370	205	5'257	~3.04	~31%	~58%
Threshold	0.75	768	438	228	5'485	~3.03	~31%	~59%
Threshold	0.70	982	427	233	5'718	~3.04	~31%	~58%
Threshold	0.65	1500	711	354	6'072	~3.01	~32%	~59%
Margin	0.60	526	517	391	6'463	~3.01	~32%	~59%
Margin	0.60	526	126	126	6'589	~3.00	~33%	~60%

Modello finale: ultimo modello, togliendo i freeze, fine-tunato ulteriormente:

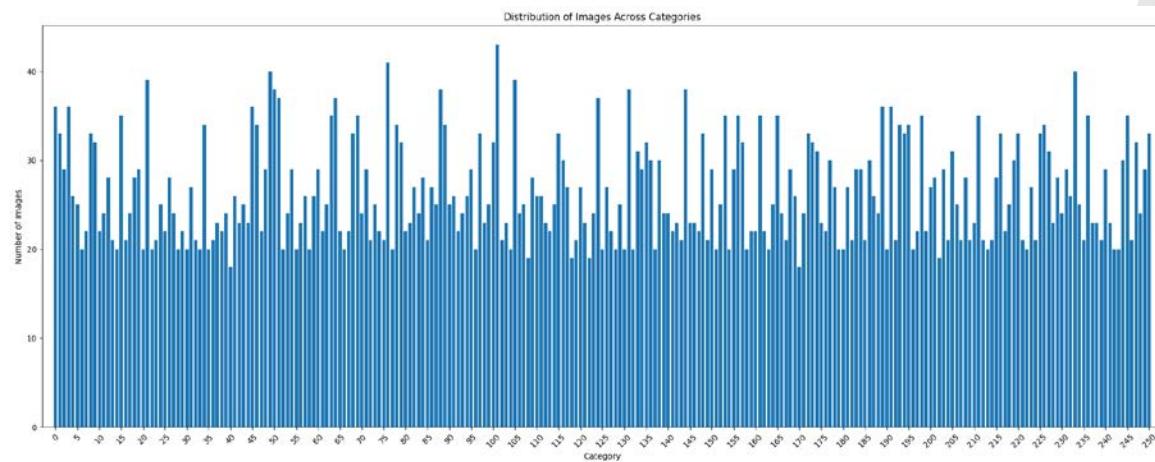
Test Loss: ~2.93 (-1%~*)

Test Accuracy: ~34% (+6%~*)

Top-5 Accuracy: ~61% (+2%~*)

Valutazione Generale

- Valutazioni sul dataset in generale
- Distribuzione delle classi nel dataset finale ----->



- **Confusion matrix e classification report illeggibili**
- **Dataframe** navigabili
- Ordinato per top-1 accuracy:

Class	Top-1 Accuracy	Top-k Accuracy
88	0.775862	0.862069
164	0.756757	0.972973
76	0.754717	0.905660
168	0.723077	0.876923
4	0.707317	0.878049
124	0.704545	0.886364
250	0.700000	0.940000
144	0.695652	0.826087
45	0.695652	0.826087
63	0.693878	0.918367

Ordinato per top-5 accuracy:

Class	Top-1 Accuracy	Top-k Accuracy
164	0.756757	0.972973
60	0.566038	0.943396
250	0.700000	0.940000
63	0.693878	0.918367
134	0.666667	0.907407
76	0.754717	0.905660
204	0.561404	0.894737
124	0.704545	0.886364
174	0.534884	0.883721
191	0.676471	0.882353

- Nessuna classe ha una **top-1** accuracy sopra all'80%
- 6 classi con **top-5** accuracy sopra al 90%

Valutazione Generale

- Ordinamento per top-5 accuracy, **invertito**
- 6 classi con **top-1** accuracy 0
- 2 classi con **top-5** accuracy 0
- Questi dataframe ci danno **indicazioni** molto utili su quali classi andare ad analizzare

Class	Top-1 Accuracy	Top-k Accuracy
213	0.000000	0.000000
239	0.000000	0.000000
85	0.000000	0.078947
238	0.000000	0.090909
99	0.047619	0.095238
84	0.050000	0.100000
65	0.041667	0.125000
40	0.000000	0.142857
132	0.000000	0.181818
58	0.022727	0.181818

Valutazione Classi Migliori

- Migliore classe per top-1: classe 88
- Train set:

train_006245.jpg



train_007324.jpg



train_007435.jpg



train_007513.jpg



train_007437.jpg



train_072618.jpg



train_007625.jpg



train_007315.jpg



train_007383.jpg



train_007231.jpg



train_007212.jpg



train_007300.jpg



- Val set:

val_003888.jpg



val_001333.jpg



val_003385.jpg



val_001683.jpg



val_009512.jpg



val_007114.jpg



val_007297.jpg



val_005160.jpg



val_000811.jpg



val_008394.jpg



val_005415.jpg



Valutazione Classi Migliori

- Migliore classe per top-1: classe 88
- Classi predette per le immagini di test (top-1):

Classe	n
88	30
2	8
142	2
161	2
170	1
194	1
...	...

- Classe confusa più spesso nel top-1:
 - Classe 2

train_062381.jpg



train_062538.jpg



train_062644.jpg



train_062692.jpg



train_062412.jpg



train_016651.jpg



train_062491.jpg



train_062756.jpg



train_062726.jpg



train_062605.jpg



train_062665.jpg



train_062761.jpg



Valutazione Classi Migliori

- Migliore classe per top-1: classe 88
- Quali categorie appaiono più spesso nella top-5 delle immagini di test di questa categoria:
- Classe con più similarità:
 - Classe 133
 - **Sembra identica**



Classe n

88	50
133	21
226	16
155	12
2	8
153	7
...	...

- Man mano diventano meno simili

Classe 226:



Classe 155:



Valutazione Classi Migliori

- **Seconda migliore classe per top-1: classe 164**
- Train set:



- Quali categorie appaiono più spesso nella **top-5** delle immagini di test di questa categoria:

Classe	n
164	36
26	31
100	25
232	11
106	11
201	5
...	...

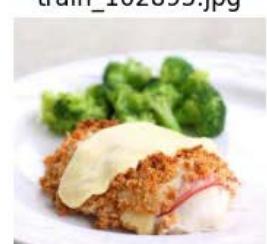
Valutazione Classi Migliori

- Tutte classi di panini, che man mano diventano meno simili



Valutazione Classi Peggiori

- **Peggiora classe per top-5: classe 213**
- Mai presente nella top-5 delle immagini di test della sua classe
- Train set: (il train set prima dell'incremento è molto simile)



- Val set: **immagini molto diverse tra di loro**



Valutazione Classi Peggiori

- **Peggior classe per top-5: classe 213**
- Classi più predette sulle immagini di test della classe 213 molto **sparse** ----->

Classe	n
128	15
76	14
28	13
242	10
53	8
97	7
...	...

- Classi più presenti nella **top-5**:

171: 2
130: 1
196: 1
242: 1
128: 1
15: 1
234: 1
190: 1
108: 1
139: 1
160: 1
28: 1
42: 1
14: 1
120: 1
231: 1
175: 1
49: 1
237: 1
158: 1
185: 1
125: 1

Valutazione Classi Peggiori

- **Peggiora classe per top-5: classe 213**
- Classi più presenti nella top-5:
 - Contengono **elementi sparsi** presenti nel training set della classe 213
 - Classe 128:

train_077952.jpg



train_078064.jpg



train_078021.jpg



train_078006.jpg



train_076159.jpg



train_078039.jpg



- Classe 76:

train_004011.jpg



train_045425.jpg



train_045145.jpg



train_045328.jpg



train_045461.jpg



train_045253.jpg



- Classe 28:

train_056072.jpg



train_056265.jpg



train_056253.jpg



train_056162.jpg



train_055797.jpg



train_056264.jpg



Valutazione Classi Peggiori

- **Seconda peggiore classe per top-5: classe 239**
 - Immagini di categorie diverse nel training set (anche prima dell'aumento)
 - **5 immagini di biscotti e 9 di torte**
 - Train set originale:

train_115176.jpg



train_115334.jpg



train_115312.jpg



train_115290.jpg



train_114938.jpg



train_115323.jpg



train_115185.jpg



train_114992.jpg



train_115018.jpg



train_115166.jpg



train_115074.jpg



train_115096.jpg



- Validation set:

val_010944.jpg



val_011943.jpg



val_010974.jpg



val_012165.jpg



val_011617.jpg



val_010559.jpg



val_011483.jpg



val_011390.jpg



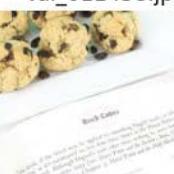
val_011655.jpg



val_012045.jpg



val_011458.jpg



val_011477.jpg



Valutazione Classi Peggiori

- **Seconda peggiore classe per top-5: classe 239**
 - Classi più presenti nella **top-5**:
 - **Mix di biscotti e torte**

Classe	n
101	8
68	8
31	8
115	5
...	...

Classe 101:



Classe 68:

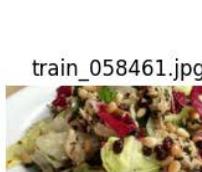


Classe 31:



Valutazione Classi Peggiori

- 2 classi peggiori successive nel top-1, con 0 top-1 accuracy
 - Classe 85: **mix di immagini diverse** (già dal train set originale)



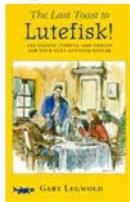
- Classe 238: **mix di immagini diverse** (già dal train set originale)



Valutazione Classi Peggiori

- **Classe 99: maggioranza di immagini sporche** nel dataset iniziale
 - L'aumento del dataset di train aggiunge ulteriori immagini sporche, dato che questa è la classe di maggioranza
 - Train set originale:

train_087382.jpg



train_087523.jpg



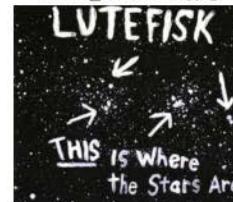
train_087510.jpg



train_087214.jpg



train_087531.jpg



train_087399.jpg



train_087232.jpg



train_087237.jpg



train_087363.jpg



train_087317.jpg



train_087347.jpg



train_087292.jpg



train_087535.jpg



train_087336.jpg



train_087511.jpg



train_087362.jpg



train_087219.jpg



train_087431.jpg



train_087403.jpg



Valutazione Classi Peggiori

- **Classe 99:** maggioranza di immagini sporche nel dataset iniziale
 - Validation set mostra che la classe doveva rappresentare **pesce**
 - Pochissime immagini di pesce nel train set originale



Valutazione su Test Set Reale

- **Immagine 1:** pizza fatta in casa

Original image: dataset/test_set_real/1.jpg



Top-5 predizioni:

Classe	Confidenza
236	52,08%
107	4,74%
183	4,48%
57	3,66%
222	3,47%

Classe 236:

val_010618.jpg



val_011688.jpg



val_011347.jpg



val_011541.jpg



Classe 107:

val_000620.jpg



val_007405.jpg



val_007615.jpg



val_000615.jpg



Classe 183:

val_000316.jpg



val_007573.jpg



val_007930.jpg



val_000126.jpg



Valutazione su Test Set Reale

- **Immagine 2:** basilico per pesto

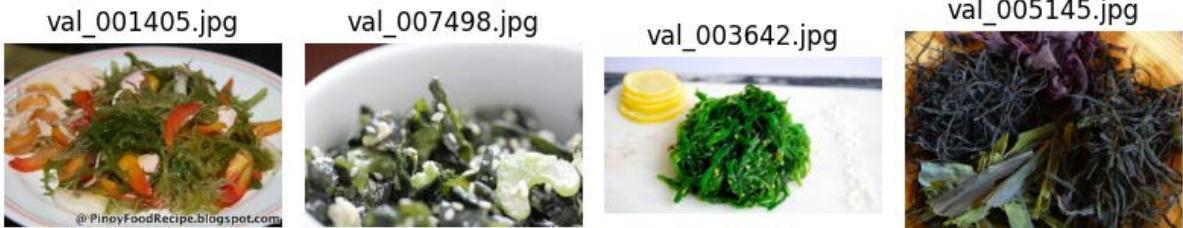
Original image: dataset/test_set_real/2.jpg



Top-5 predizioni:

Classe	Confidenza
15	69,82%
76	8,13%
204	6,70%
114	3,28%
94	2,17%

Classe 15:



Classe 76:



Classe 204:
Classe 114:



Valutazione su Test Set Reale

- **Immagine 3:** strozzapreti fatti in casa

Original image: dataset/test_set_real/3.jpg



Top-5 predizioni:

Classe	Confidenza
88	20,85%
243	7,85%
92	5,37%
117	5,19%
153	4,86%

Classe 88:



Classe 243:



Classe 92:



Valutazione su Test Set Reale

- **Immagine 4:** lasagne fatte in casa

Original image: dataset/test_set_real/4.jpg



Top-5 predizioni:

Classe	Confidenza
157	19,41%
31	11,47%
240	7,28%
119	6,56%
206	6,27%

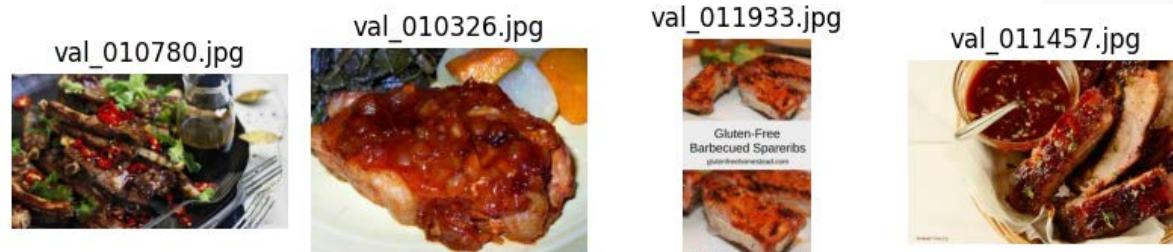
Classe 157:



Classe 31:



Classe 240:



Valutazione su Test Set Reale

- **Immagine 5:** tagliata di manzo

Original image: dataset/test_set_real/5.jpg



Top-5 predizioni:

Classe	Confidenza
217	53,73%
74	8,03%
231	7,24%
158	6,93%
122	2,62%

Classe 217:



Classe 74:



Classe 231:



Degraded Test set

Obbiettivo:

- Migliorare le performance del modello sul validation set degradato.

Modello utilizzato:

- Il modello **migliore precedente freezato** come punto di partenza

Performance sul validation set normale

Test Loss : ~3.00

Test Accuracy: ~33%

Top-5 Accuracy : ~60%



Perdita delle prestazioni

Performance sul validation set degradato

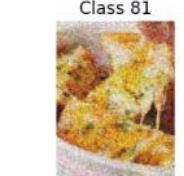
Test Loss : ~3.92

Test Accuracy: ~23%

Top-5 Accuracy : ~45%

Esplorazione delle immagini del degraded-set

- Analisi del contenuto del validation set
 - 90 immagini campionate



Esplorazione delle immagini del degraded-set

Rumore Gaussiano:

Caratterizzato da una distribuzione casuale dei valori dei pixel, con un'ampia gamma di intensità e colori.



Esplorazione delle immagini del degraded-set

Gaussian Noise:

- Verifica che l'immagine degradata corrisponda al tipo di degradazione prevista
- Applicazione del Gaussian Noise all'immagine originale



Esplorazione delle immagini del degraded-set

Compressione JPEG:

Tipo di artefatto visivo che si verifica a causa della compressione dell'immagine nel formato **JPEG**

Original Image



Degraded Image



Esplorazione delle immagini del degraded-set

Compressione JPEG:

- Verifica che l'immagine degradata corrisponda al tipo di degradazione prevista
- Applicazione della compressione JPEG all'immagine originale
 - Trovando i parametri più adatti per raggiungere lo stesso livello di degradazione

Degraded Image (JPEG Compression + Resize)



Degraded Image



Esplorazione delle immagini del degraded-set

Blur:

Il blur è una perdita di nitidezza e dettagli in un'immagine, spesso causata da diversi fattori, come compressione, errori di acquisizione o elaborazione dell'immagine.

Original Image



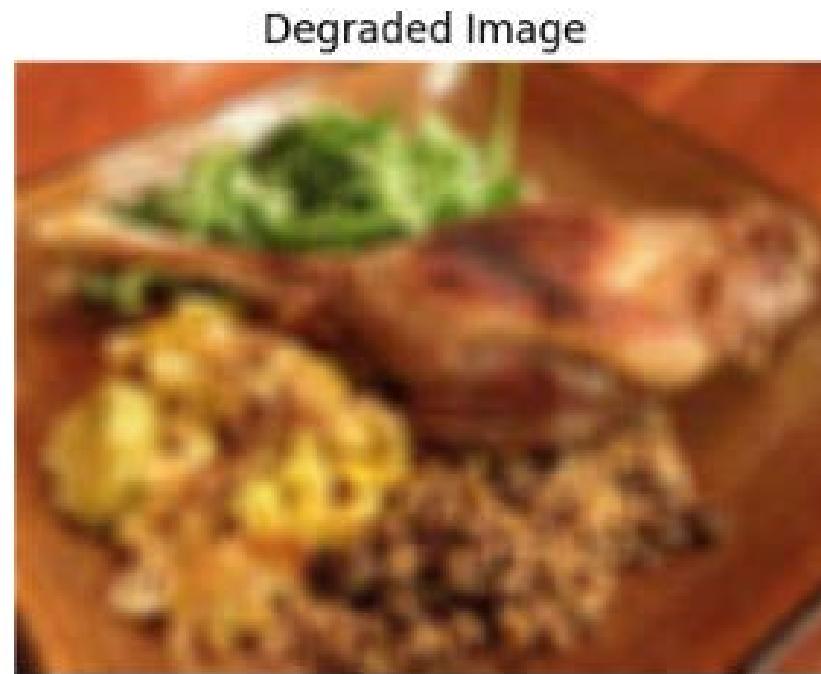
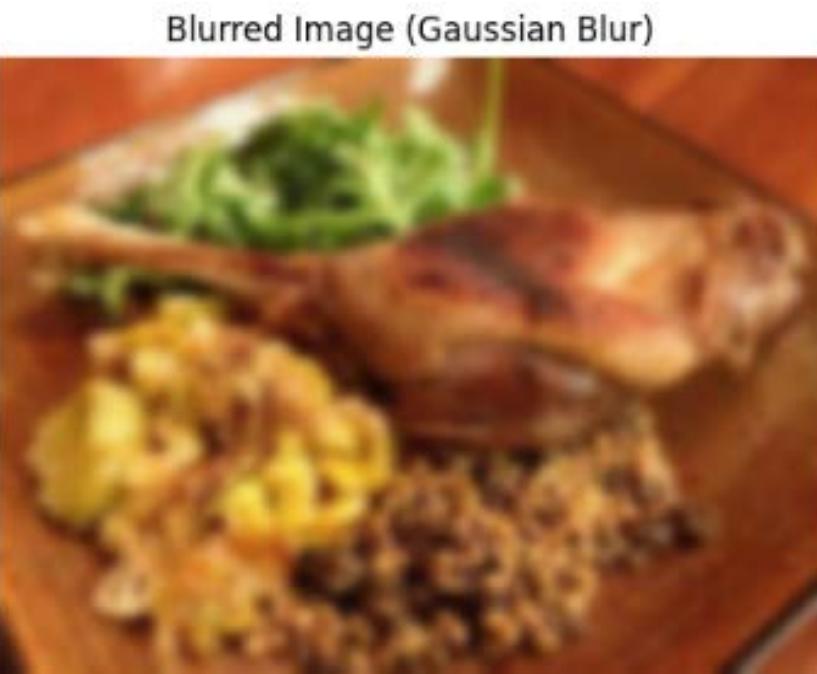
Degraded Image



Esplorazione delle immagini del degraded-set

Blur:

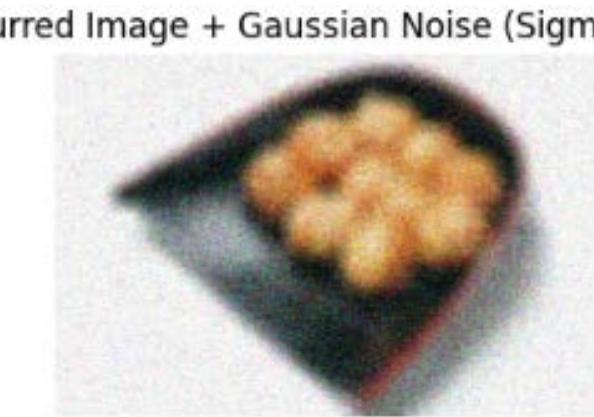
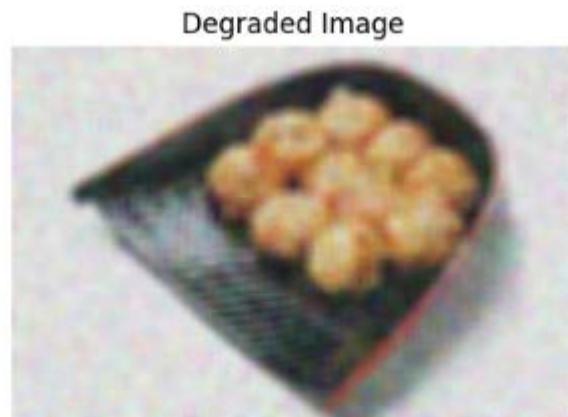
- Verifica che l'immagine degradata corrisponda al tipo di degradazione prevista
- Applicazione blur all'immagine originale



Esplorazione delle immagini del degraded-set

Combinazioni:

- Nel dataset sono presenti diversi tipi di **combinazioni** di degradazione come la gaussian + blur



Trovare i migliori parametri

Obiettivo:

- Analizzare le immagini degradate e rappresentarle mediante una distribuzione di probabilità, considerando le variazioni nei parametri di degradazione anche all'interno della stessa categoria.

Procedimento:

- Cercare un metodo per riconoscere il grado di degradazione e così derivarlo **automaticamente**.
- Oppure **manualmente**.

Procedimento automatico provato:

- **BRISQUE**: Image Quality Assessment, è un algoritmo che prende un'immagine come input e restituisce uno quality score dell'immagine, quindi senza bisogno di un'immagine di riferimento.
 - Scegliere il parametro di degradazione in base al valore Brisque.

Trovare i migliori parametri

Risultati:

- Gaussian Noise, approccio automatico con BRISQUE:

Immagine Originale



Immagine con Rumore (Sigma=21)



Immagine Degradata



Degraded Image
BRISQUE Score: 45.56819161415578



Image with Gaussian Noise
BRISQUE Score: 41.777737308713284



Trovare i migliori parametri

Risultati:

- JPEG compression

Approccio automatico:



Compressed Image (Quality=16)



Degraded Image



Approccio manuale:



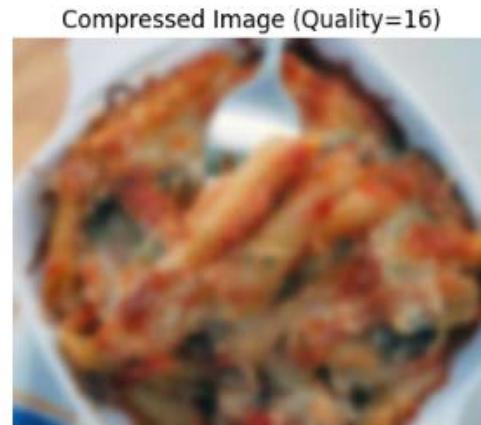
Degraded Image



Trovare i migliori parametri

Risultati:

- Blur



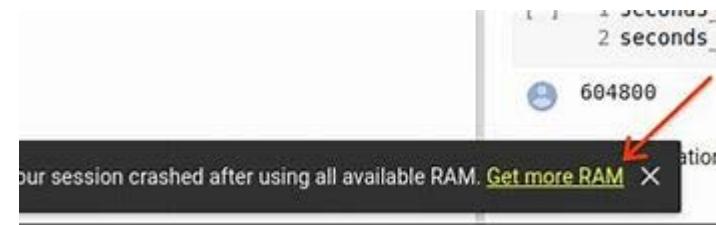
Degradazione del Train Set Aumentato

Procedimento manuale:

- In questa sezione viene mostrato il metodo di degradazione applicato al train set aumentato.
- Proporzioni stimate tra le degradazioni (su un subset):
 - ~17% blur
 - ~46% gaussian noise
 - ~20% jpeg compression
 - ~17% gaussian noise + blur

Primo procedimento:

- Degradazioni casuali applicate ad ogni epoca
 - **Utilizzo di ram troppo alto** ----->



Secondo procedimento:

- Applicazione degradazioni in **proporzione rispetto alle stime**, sull'intero train set aumentato
- **Unione** del train set degradato con quello pulito
 - 50% immagini pulite
 - 50% immagini degradate

Modello Addestrato sul Nuovo Dataset Mixato

- Modello allenato sul dataset aumentato pulito
- Test su validation set degradato
 - Test Loss : ~3.92
 - Test Accuracy: ~23%
 - Top-5 Accuracy : ~45%
- Modello allenato sul dataset aumentato mixato (pulito + degradato)
- Test su validation set degradato
 - Test Loss : ~3.56 (-9%~)
 - Test Accuracy: ~24% (+4%~)
 - Top-5 Accuracy : ~48% (+7%~)

- Modello allenato sul dataset aumentato pulito
- Test su validation set pulito
 - Test Loss : ~3.00
 - Test Accuracy: ~33%
 - Top-5 Accuracy : ~60%
- Modello allenato sul dataset aumentato mixato (pulito + degradato)
- Test su validation set pulito
 - Test Loss : ~3.10 (+3%~)
 - Test Accuracy: ~30% (-9%~)
 - Top-5 Accuracy : ~57% (-5%~)

Test sui singoli tipi di degradazione

Predizioni su un sottoinsieme di immagini degradate divise per tipo di degradazione

- Blur (50)
- Rumore gaussiano (50)
- JPEG compressione (50)

Modello addestrato con il train set mixato (pulito + degradato)

Accuracy for Blur: 18.00% | F1-score: 0.18 | Top-5 Accuracy: 52.00%

Accuracy for Gauss: 12.00% | F1-score: 0.14 | Top-5 Accuracy: 26.00%

Accuracy for Jpeg: 4.00% | F1-score: 0.03 | Top-5 Accuracy: 14.00%

Modello addestrato con il train set aumentato pulito

Accuracy for Blur: 14.00% | F1-score: 0.14 | Top-5 Accuracy: 38.00%

Accuracy for Gauss: 2.00% | F1-score: 0.03 | Top-5 Accuracy: 8.00%

Accuracy for Jpeg: 2.00% | F1-score: 0.02 | Top-5 Accuracy: 10.00%

Tentativo di miglioramento delle immagini degradate

Obiettivo:

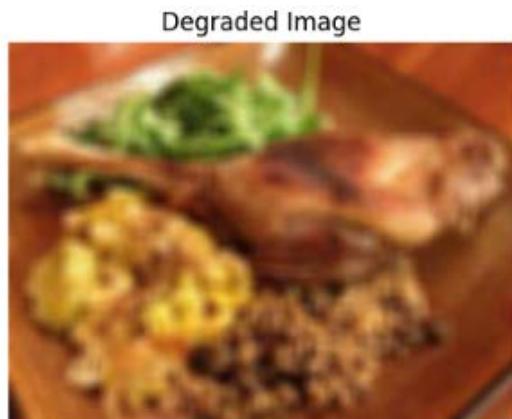
- Trovare un metodo automatico per **identificare le degradazioni**
- Trovare un metodo automatico per **correggere le degradazioni**

Identificazione degli errori:

- **Metodo con varianza:**
 - Varianza Originale: 4291.40
 - Varianza Degradata: 4419.12
- **Metodo con filtro Laplaciano:**
 - Varianza laplaciana originale: 431.66
 - Varianza laplaciana degradata: 37886.42



Tentativo di miglioramento delle immagini degradate: metodo del filtro laplaciano



Laplacian Variance Originale: 2144.82
Laplacian Variance Degradata: 13.15



Laplacian Variance Originale: 716.10
Laplacian Variance Degradata: 642.11

Conclusioni e Possibili Ulteriori Sviluppi

- **Train set di partenza** causa problemi enormi a tutti gli approcci programmativi utilizzati
 - Classi non ben definite
 - Classi mischiate
 - Classi molto simili
 - Immagini sporche che a volte superano il numero delle immagini pulite nella classe
- **Pulizia manuale** del training set
 - Aumento e bilanciamento del training set iniziale **manualmente**
 - Porterebbe a risultati certamente migliori ma sarebbe stata poco interessante per questo progetto
- **Prestazioni top-1** che difficilmente può diventare molto alta (~34%)
 - Grande varietà di immagini in ciascuna classe
 - Alta similarità tra classi diverse
- **Prestazioni top-5** raggiunte buone (~61%)
- **Possibili ulteriori sviluppi:**
 - **Più fine-tuning** su modelli non freezati
 - Utilizzo dell'**intero train set unlabelled** (10% esplorato)
 - Esplorazione di ulteriori **metodi di semi-supervised learning**
 - Esplorazione di metodi per **migliorare le immagini del validation set degradato**

Grazie per l'attenzione

Robust Food Classification Project

Visual Information Processing and Management
Università degli Studi di Milano-Bicocca

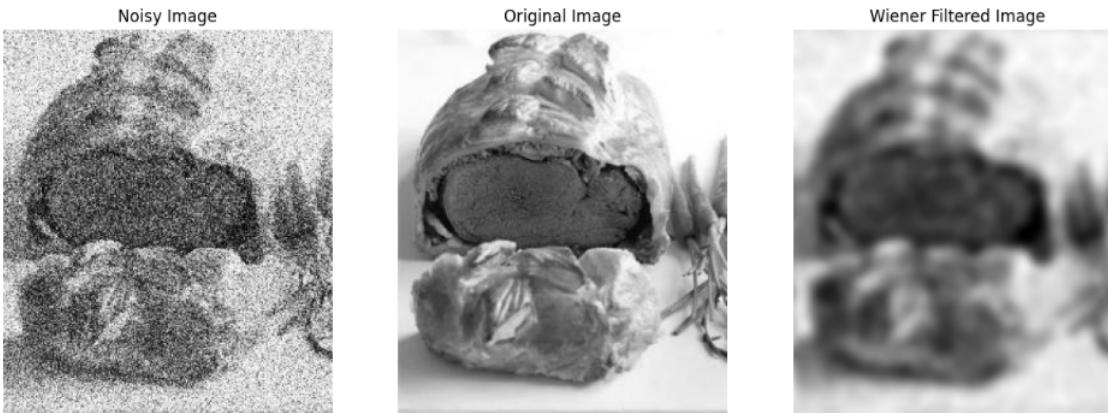
Matteo Breganni 869549
Lorenzo Monti 869960



Tentativo di correzione della degradazione gaussiana

Correzione della degradazione gaussiana

- Wiener

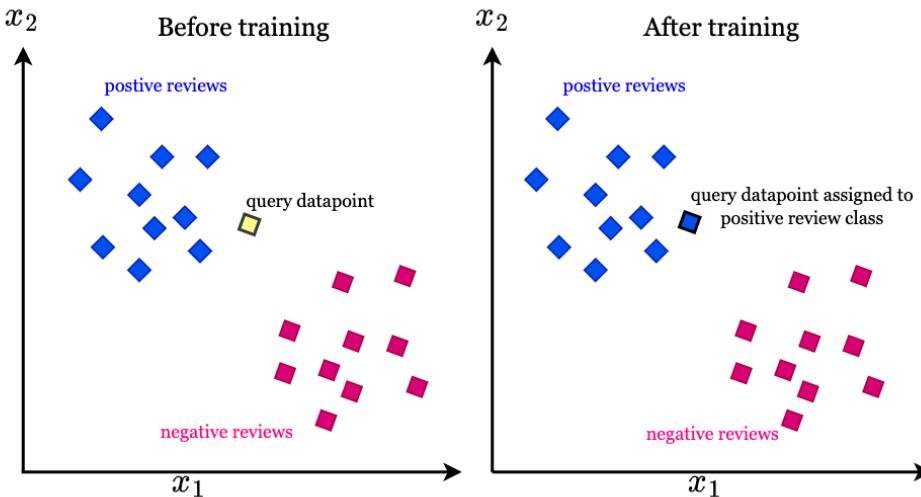


- Non-local Means Denoising



KNN

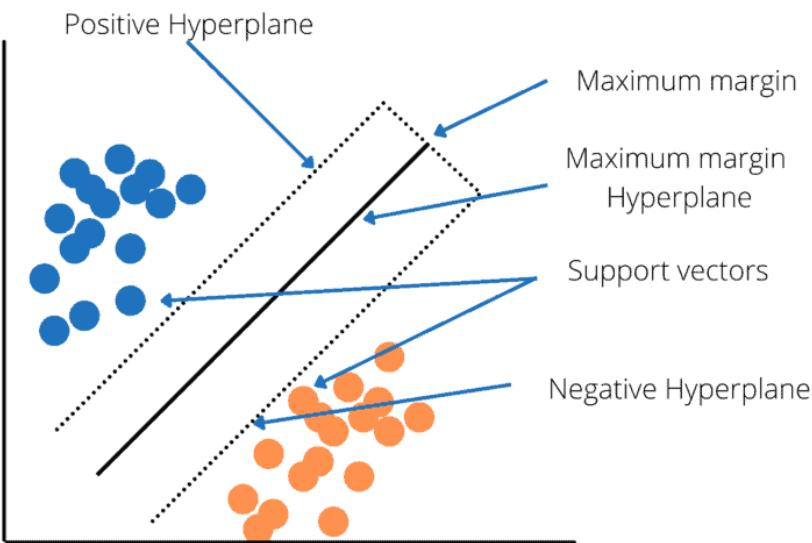
- K-Nearest Neighbors (KNN) è un algoritmo di classificazione e regressione basato sulla vicinanza tra i dati.
- Per classificare un nuovo punto, calcola la distanza (es. Euclidea) dai punti di addestramento e considera i K più vicini. La classe più frequente tra questi vicini viene assegnata al nuovo punto



- Nella versione Nearest Neighbors lo abbiamo utilizzato per trovare gli outliers:
 - Si calcola la distanza di ogni punto dai suoi K vicini più prossimi.
 - Se un punto ha una distanza molto più alta rispetto alla media delle altre distanze, è considerato un outlier.

SVM

- Trova un iperpiano ottimale che separa le classi, massimizzando il margine tra i punti più vicini di ciascuna classe (support vectors).
- Se i dati non sono linearmente separabili, usa **kernel trick** per trasformarli in uno spazio a dimensione superiore dove diventano separabili.



- Lo abbiamo anche usato per la ricerca degli outliers nella forma **OneClassSVM** dove il modello trova una regione che circoscrive i dati normali. A ciascun dato viene assegnato un punteggio di anomalia, in base alla distanza dal confine.

MobileNetV2

- **MobileNetV2** è una rete neurale leggera progettata per dispositivi mobili
- Ha una convoluzione standard 2D con kernel 3x3 e stride 2 per estrarre le caratteristiche di base.
- Segue una serie di blocchi «**bottleneck**», ogni blocco ha tre fasi:
 - 1x1 Expansion Layer → Espande i canali con un fattore di espansione.
 - 3x3 Depthwise Convolution → Applica convoluzioni leggere per filtrare i dati.
 - 1x1 Projection Layer → Comprime i canali per ridurre la dimensionalità.
 - Skip connections usate quando input e output hanno lo stesso numero di canali.
- Convolution 2D 1x1 finale per combinare le feature estratte
- Average Pooling + Fully Connected Layer seguito da una convoluzione 1x1 e lo strato di classificazione softmax.

DenseNet121

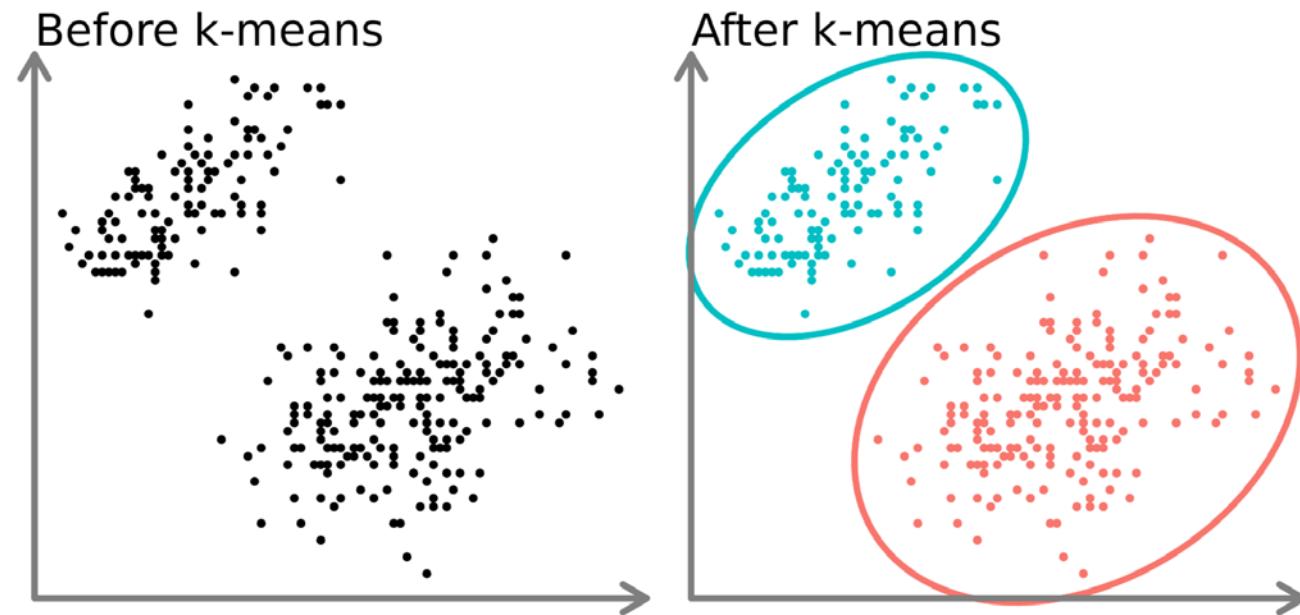
- In **DenseNet**, ogni layer riceve input non solo dal layer precedente, ma anche da tutti i layer precedenti. Questo tipo di architettura consente al modello di apprendere in modo più efficiente e ridurre il problema del vanishing gradient, che è comune nelle reti profonde.

Caratteristiche principali:

- **Connessioni Dense (Dense Connections):** Ogni layer è connesso a tutti i precedenti.
- **Numero di layers:** DenseNet121 contiene 121 layers, che è una delle versioni più leggere della famiglia DenseNet
- **Growth Rate:** Ogni layer ha un "growth rate" che determina quanti nuovi feature maps vengono generati per ogni layer. Ad esempio, in DenseNet121, il growth rate è generalmente impostato su 32, il che significa che ogni layer aggiunge 32 nuove feature maps.
- **Bottleneck Layers e Compressione:** DenseNet121 utilizza anche il concetto di "bottleneck layers" per ridurre la dimensionalità prima di passare ai layers successivi.
- **Architettura a blocchi:** DenseNet è composta da blocchi di layer chiamati **Dense Blocks**. Ogni blocco è seguito da un **Transition Layer**, che riduce la dimensione spaziale e il numero di feature maps.

K-Means

- Algoritmo di clustering non supervisionato.
- Raggruppa i dati in K cluster, minimizzando la distanza tra punti e centroidi.
- I punti che non appartengono a nessun cluster chiaro (cioè lontani dai centroidi) possono essere considerati **outliers**, in quanto non si adattano bene al modello di clustering.
- Tramite una soglia possiamo tunare la sensibilità degli outliers ritornati.



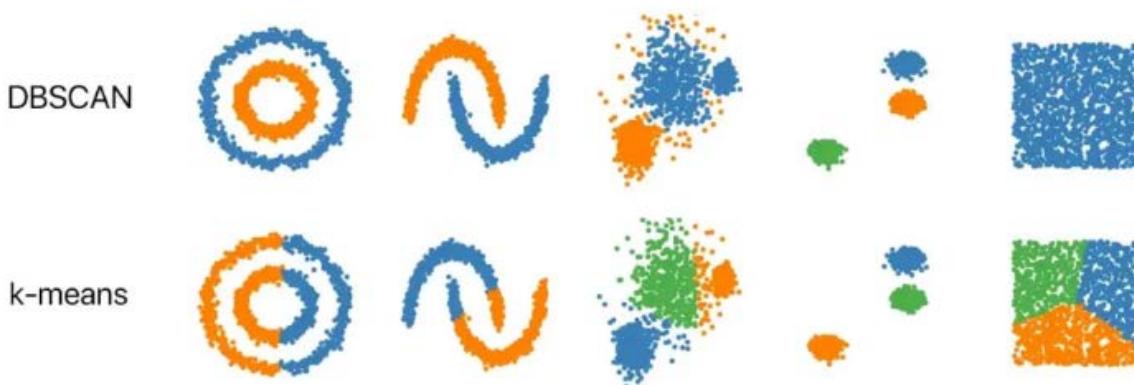
DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

- Algoritmo di clustering basato sulla densità.
- Raggruppa punti vicini con alta densità in cluster, separando le aree meno dense come rumore (outliers).
- Passi:
 - Seleziona un punto casuale e trova i punti vicini.
 - Se ci sono abbastanza punti vicini, forma un cluster.
 - Espandi il cluster includendo i punti vicini ad altri punti già nel cluster.
 - Ripeti fino a visitare tutti i punti.

Per trovare outliers:

- I punti che non appartengono a nessun cluster (bassi nella densità) sono considerati **outliers** o **rumore**.



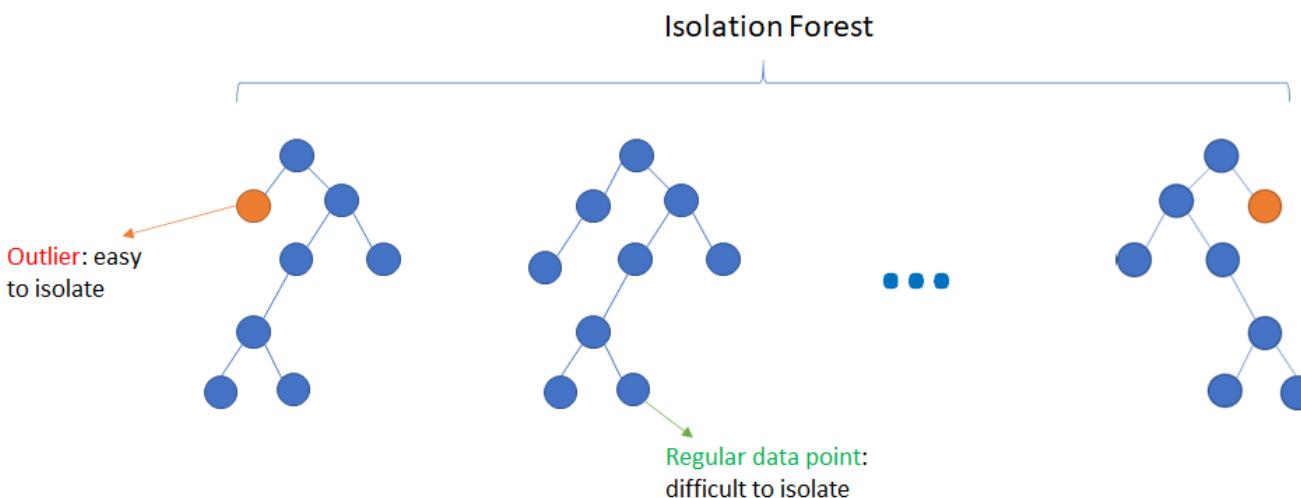
Isolation Forest

Isolation Forest:

- Algoritmo basato sull'isolamento dei punti anomali (outliers).
- Utilizza alberi decisionali per "isolare" i punti, cercando quelli che sono più facilmente separabili dal resto dei dati.
- Passi:
 - Costruisci alberi decisionali con sottogruppi casuali dei dati.
 - Misura la profondità a cui un punto viene isolato (più è isolato, più è anomalo).
 - Ripeti per molti alberi e calcola la media delle profondità.

Per trovare outliers:

- I punti che sono facilmente separabili sono considerati **outliers**.



Rumore Gaussiano

Il **rumore gaussiano** è un tipo di disturbo che segue una distribuzione normale (gaussiana), caratterizzata da una media (tipicamente zero) e una deviazione standard che controlla l'intensità del rumore.

È un rumore casuale che si aggiunge ai segnali (come le immagini), con valori che variano in modo imprevedibile.

Per applicare questo disturbo ad un'immagine abbiamo:

- generato un array di valori casuali con distribuzione normale (gaussiana)
- lo abbiamo sommato all'immagine originale
- abbiamo limitato i valori dell'immagine risultante nell'intervallo [0, 255]
- e restituito l'immagine con il rumore aggiunto.

Immagine Originale



Immagine con Rumore (Sigma=71)



Compressione JPEG

La **compressione JPEG** riduce la dimensione di un'immagine mantenendo una qualità visiva accettabile.

- **Trasformazione in YCbCr:** L'immagine RGB viene convertita nel formato YCbCr, separando la luminosità (Y) dai colori (Cb e Cr).
- **Sottocampionamento cromatico:** I canali di colore (Cb e Cr) sono ridotti in risoluzione, poiché l'occhio umano è meno sensibile ai dettagli cromatici rispetto alla luminosità.
- **Blocchi 8x8:** L'immagine viene divisa in blocchi di 8x8 pixel.
- **Trasformata discreta del coseno (DCT):** Ogni blocco di 8x8 pixel viene trasformato in frequenze (DCT), separando i dettagli ad alta frequenza (meno visibili) da quelli a bassa frequenza (più visibili).
- **Quantizzazione:** Le frequenze alte vengono approssimate (quantizzate) per ridurre la quantità di dati. I dettagli meno rilevanti vengono eliminati, riducendo ulteriormente la dimensione.

Gaussian Blur

Il **Gaussian Blur** sfoca un'immagine applicando un filtro gaussiano, che pesa di più i pixel centrali e meno quelli periferici. Si crea una matrice (kernel) gaussiana e si esegue una **convoluzione**, ossia si calcola una media pesata dei pixel all'interno di una finestra del kernel per ogni pixel dell'immagine. Questo riduce i dettagli e il rumore, creando un effetto di sfocatura.

Original Image



Compressed Image (Quality=16)



BRISQUE

Workflow

Fig. 5 shows the steps involved in calculating BRISQUE.

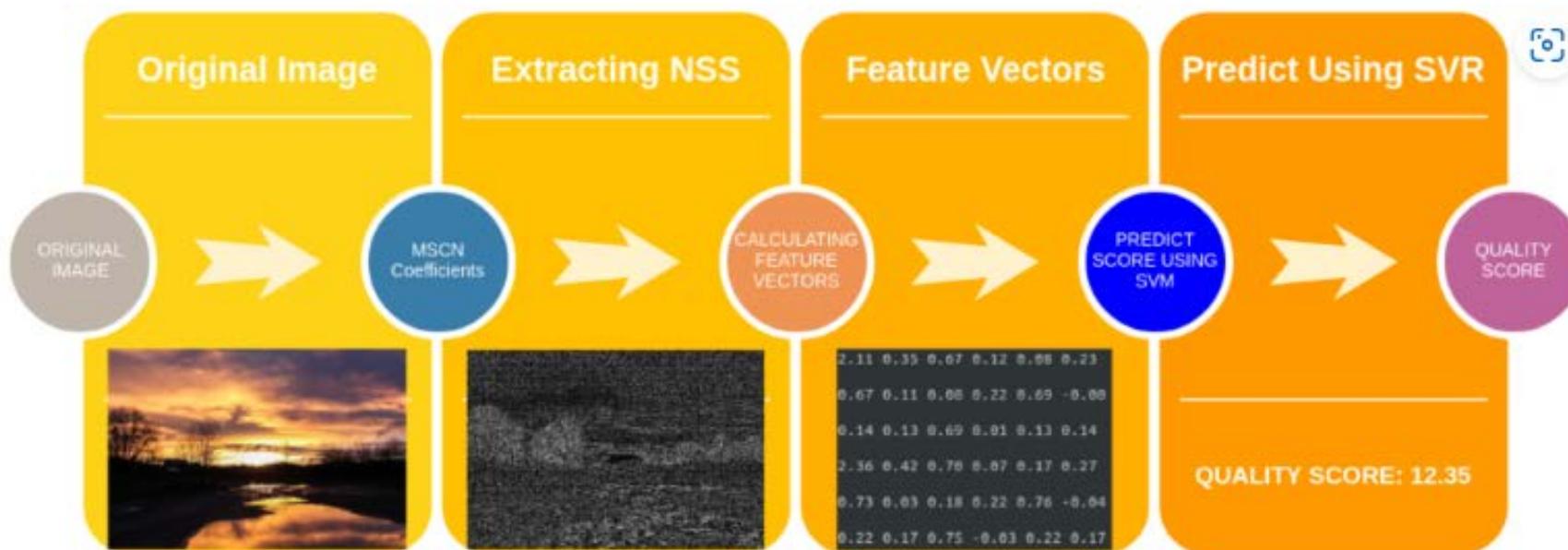


Fig. 5 Steps to Calculate Image Quality Score using BRISQUE Model

BRISQUE

1. Estrazione delle caratteristiche NSS con i coefficienti MSCN :

- Nel **primo passo**, si estraggono le **Statistiche delle Scene Naturali (NSS)** per analizzare la distribuzione delle intensità dei pixel. Le immagini naturali hanno una distribuzione delle intensità diversa rispetto a quelle distorte, e questa differenza diventa più evidente dopo la normalizzazione.
- Dopo la normalizzazione, le immagini naturali seguono una distribuzione gaussiana (a campana), mentre quelle distorte no. La deviazione dalla distribuzione ideale è quindi un indicatore del livello di distorsione presente nell'immagine.
- **MSCN** (Mean Subtracted Contrast Normalized) normalizza l'immagine sottraendo il valore medio locale e ridimensionando il contrasto, migliorando la separazione delle caratteristiche.
- Si calcolano i **coefficienti MSCN** per ogni pixel, che descrivono l'intensità relativa in un'area locale rispetto ai pixel circostanti, fornendo informazioni sulla struttura spaziale.

2. Calcolare vettore delle features

- Le caratteristiche estratte tramite MSCN vengono usate per costruire **feature vectors**. Questi vettori descrivono le proprietà spaziali, come la regolarità o la distorsione dell'immagine, raccogliendo statistiche come media, deviazione standard e altre misure sui coefficienti MSCN.

Predizione del punteggio usando SVM:

- I **feature vectors** vengono passati a un modello **SVM** (Support Vector Machine) addestrato per classificare la qualità dell'immagine.
- L'SVM predice un punteggio di qualità basato sullo spazio delle caratteristiche, dove il punteggio finale rappresenta la qualità percepita dell'immagine (più alto è il punteggio, migliore è la qualità).

Filtro laplaciano

Codice usato: `laplacian_original = cv2.Laplacian(image_array, cv2.CV_64F).var()`

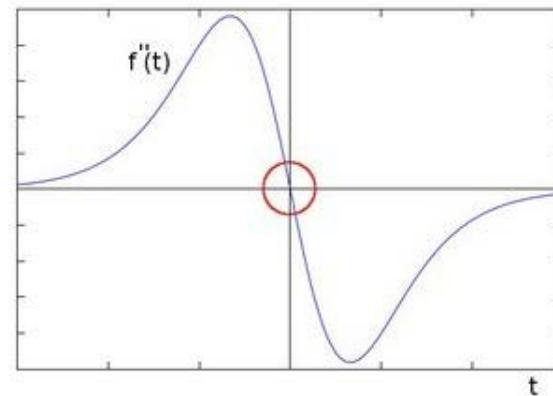
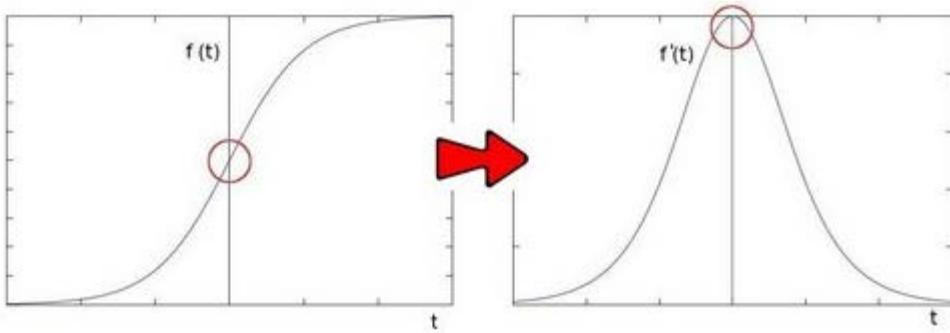
Link alla libreria: [OpenCV: Laplace Operator](#)

Laplacian operator

```
# Apply Laplace function
dst = cv.Laplacian(src_gray, ddepth, ksize=kernel_size)
```

- The arguments are:
 - *src_gray*: The input image.
 - *dst*: Destination (output) image
 - *ddepth*: Depth of the destination image. Since our input is *CV_8U* we define *ddepth* = *CV_16S* to avoid overflow
 - *kernel_size*: The kernel size of the Sobel operator to be applied internally. We use 3 in this example.
 - *scale*, *delta* and *BORDER_DEFAULT*: We leave them as default values.

Teoria: Nell'area del bordo, l'intensità dei pixel mostra un salto o un elevata variazione di intensità. Dalla derivata prima possiamo osservare un maximum, mentre dalla derivata seconda 0 :



Filtro laplaciano

Visto che usa il gradiente dell'immagine chiama internamente l'operatore sobel per performare la computazione, visto che l'operatore sobel utilizza due maschere di convoluzione, una per il gradiente prima direzione e un'altra per gradiente seconda direzione.

$$\text{Laplace}(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- **Derivata rispetto a x ($\frac{\partial f}{\partial x}$):**

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- **Derivata rispetto a y ($\frac{\partial f}{\partial y}$):**

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Questa misura viene spesso utilizzata per **valutare la nitidezza** di un'immagine. Se la varianza del Laplaciano è **bassa**, significa che non ci sono molte variazioni di intensità e l'immagine è probabilmente **sfocata**. Se invece è alta, indica la presenza di molti dettagli e bordi ben definiti.