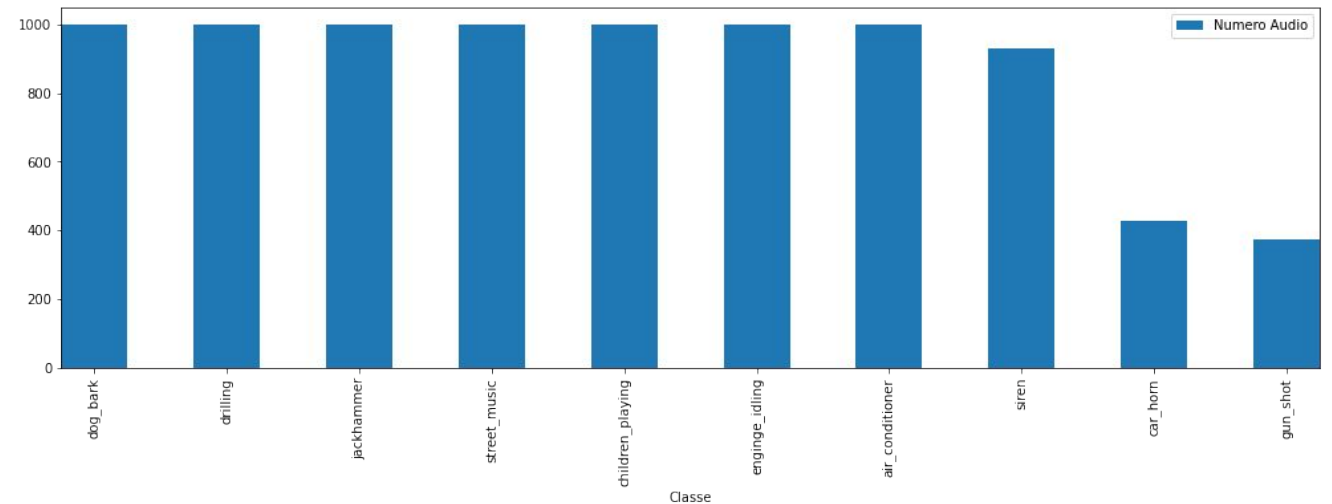


Digital Signal and Image Management

Aurora Cerabolini – 839327
Veronica Morelli – 839257

Classificazione Mono-dimensionale – UrbanSound8K

- 8000 tracce audio rilevate in ambiente urbano
- 10 classi:
 - air_conditioner: aria condizionata
 - car_horn: auto
 - children_playing: bambini che giocano
 - dog_bark: abbaio del cane
 - drilling: foratura/trivellazione
 - engine_idling: motore acceso al minimo
 - gun_shot: sparo di pistola
 - jackhammer: martello pneumatico
 - siren: sirene
 - street_music: musica di strada



Distribuzione audio per classe

Diment, A., et al. "Tut rare sound events, development dataset." *DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events*. 2017.

Steps

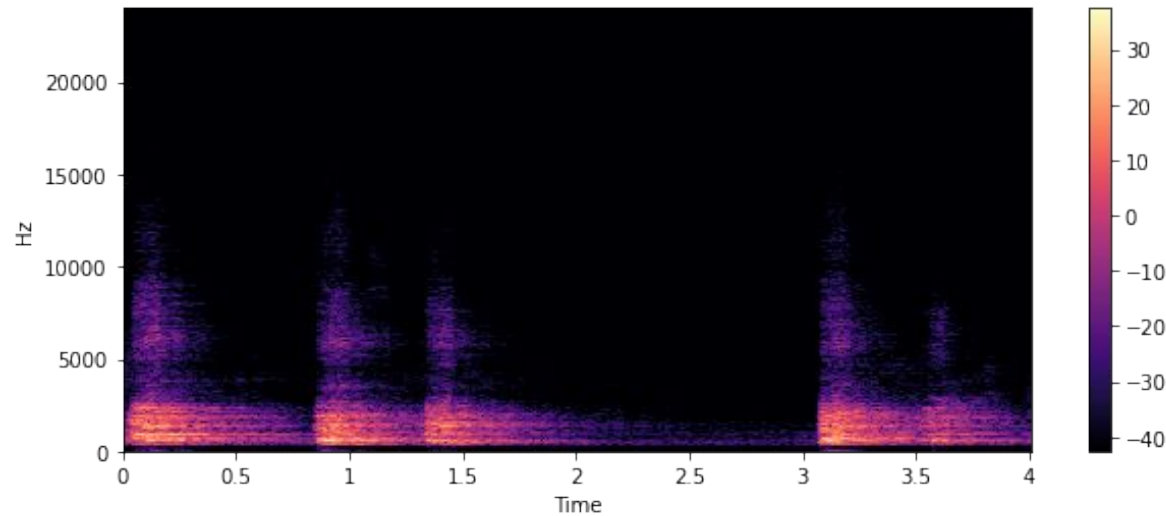
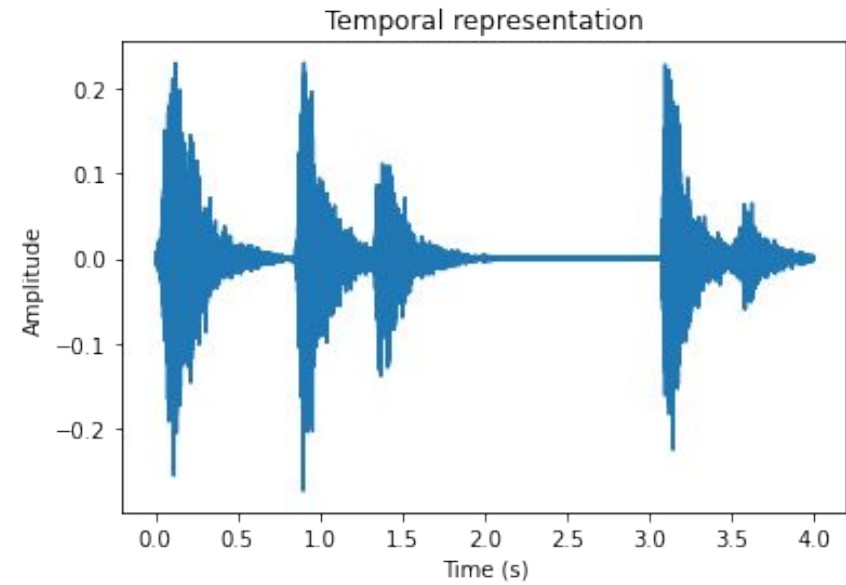
1. Caricamento e Organizzazione Dataset
2. Analisi Esplorativa
3. Data Augmentation: velocizzazione + rallentamento
4. Dichiarazione Features
5. Data Loading + Divisione dati in training e validation
6. Classificazione SVM
7. Classificazione CNN con Spettrogramma di Mel

Analisi Esplorativa

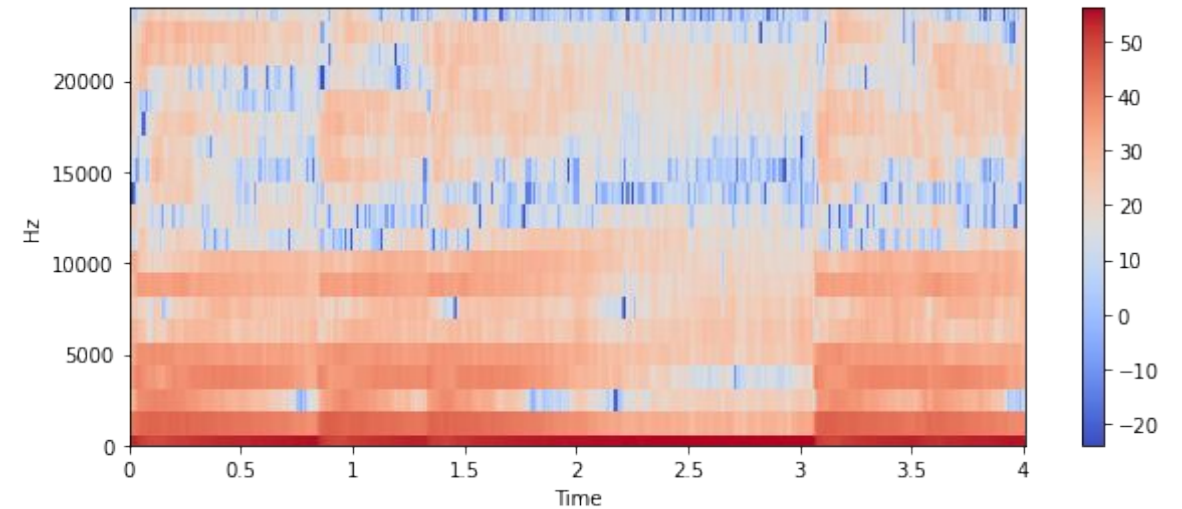
Visualizzazioni audio classe: *children playing*

Caratteristiche:

- Dimensione sound data: 192000
- Sound rate: 48000



Spettrogramma



MFCC: Mel Spectrogram Ceptral Coefficient

Features hand-crafted

- Energia
- Durata
- Zero Crossing Rate
- Spettrogramma
- Spettrogramma di Mel
- Mel Frequency Cepstral Coefficient (MFCC)
- Combo 1: Energia + Durata
- Combo 2: Energia + Durata + Zero Crossing Rate

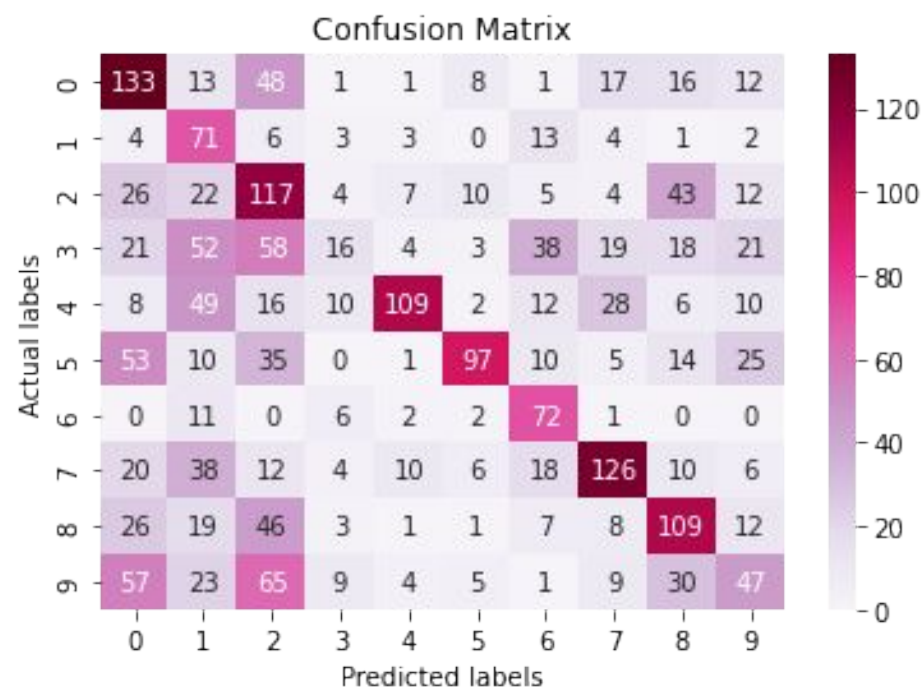
Classificazione SVM

Features	Accuracy
Energia	0.20
Durata	0.18
Zero Crossing Rate	0.20
Spettrogramma	0.11
Spettrogramma di Mel	0.26
MFCC – Mel Frequency Cepstral Coefficient	0.17
Combo 1	0.20
Combo 2	0.42

Classificazione con dataset con data augmentation non porta miglioramenti.

Risultato migliore
features combo 2:

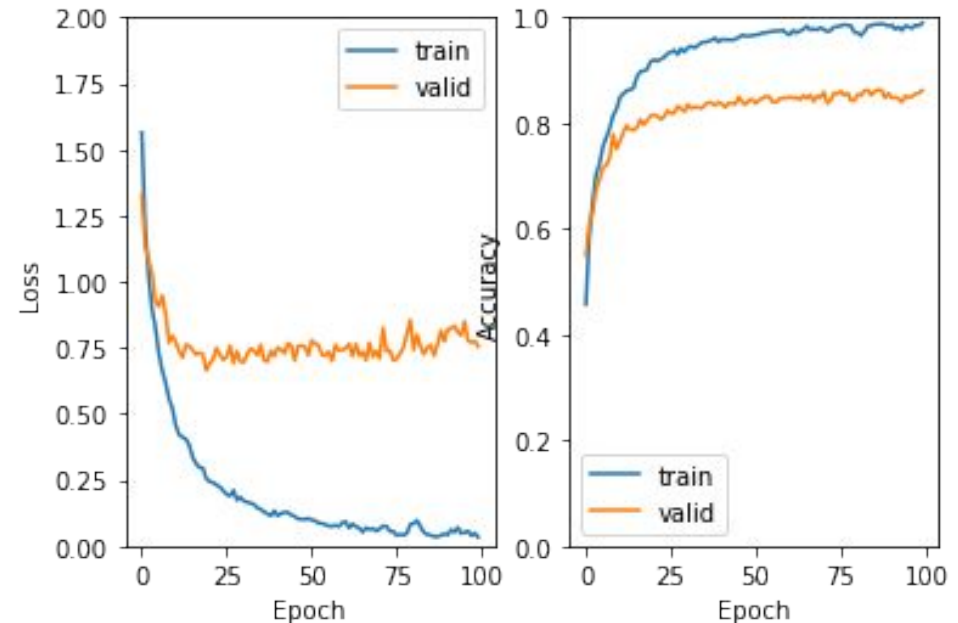
- energia
- durata
- zero crossing rate

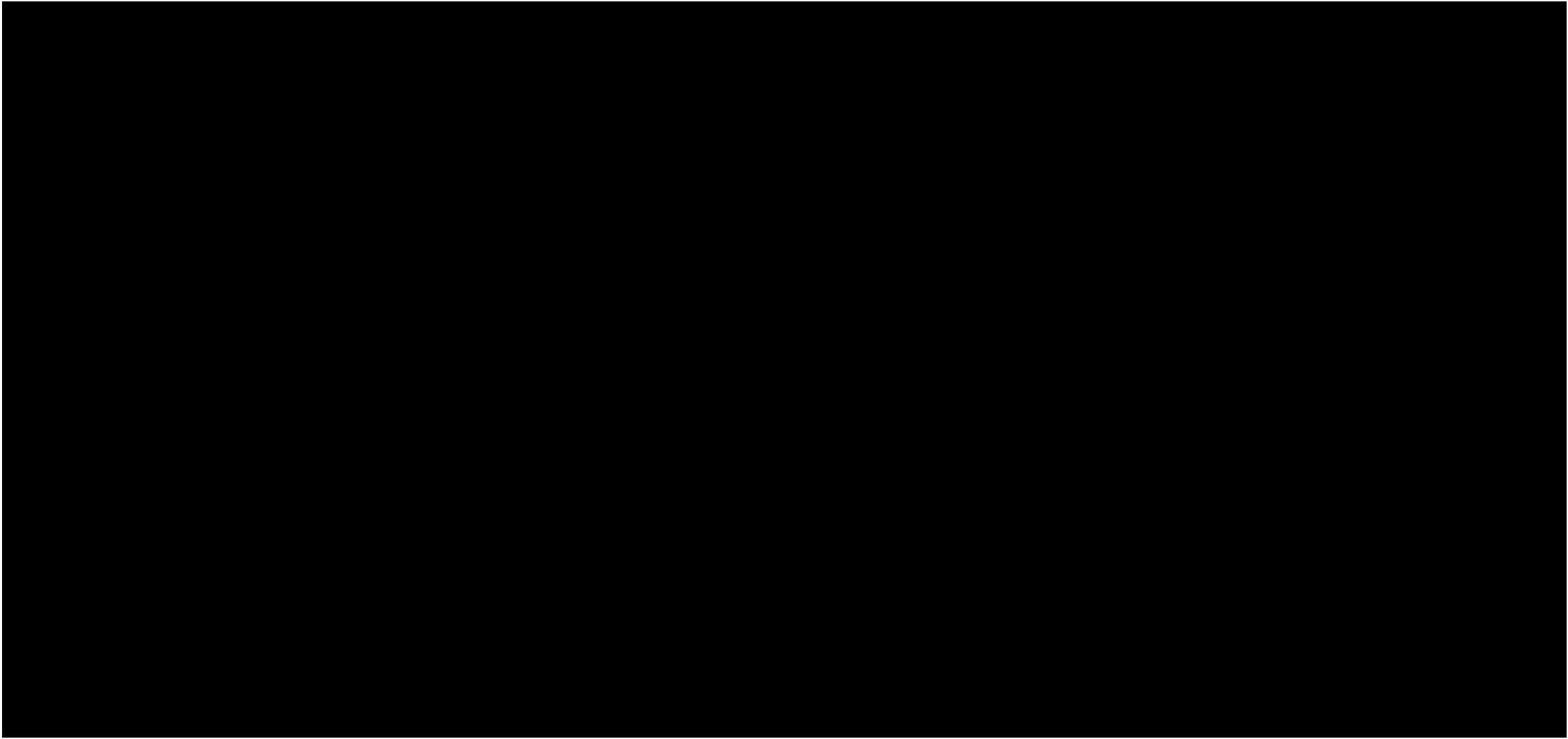


Classificazione Mel Spectrogram con CNNs

Architettura Rete	Accuracy
<ul style="list-style-type: none">• 3-layer convoluzionali• ReLU• Max pooling• Dropout• 1-layer Dense	Accuracy training: 0.98 Accuracy validation: 0.89
<ul style="list-style-type: none">• 3-layer convoluzionali• ReLU• Max pooling• Batch Normalization• Dropout• 2-layer dense	Accuracy training: 0.96 Accuracy validation: 0.86

Risultato migliore: CNN1

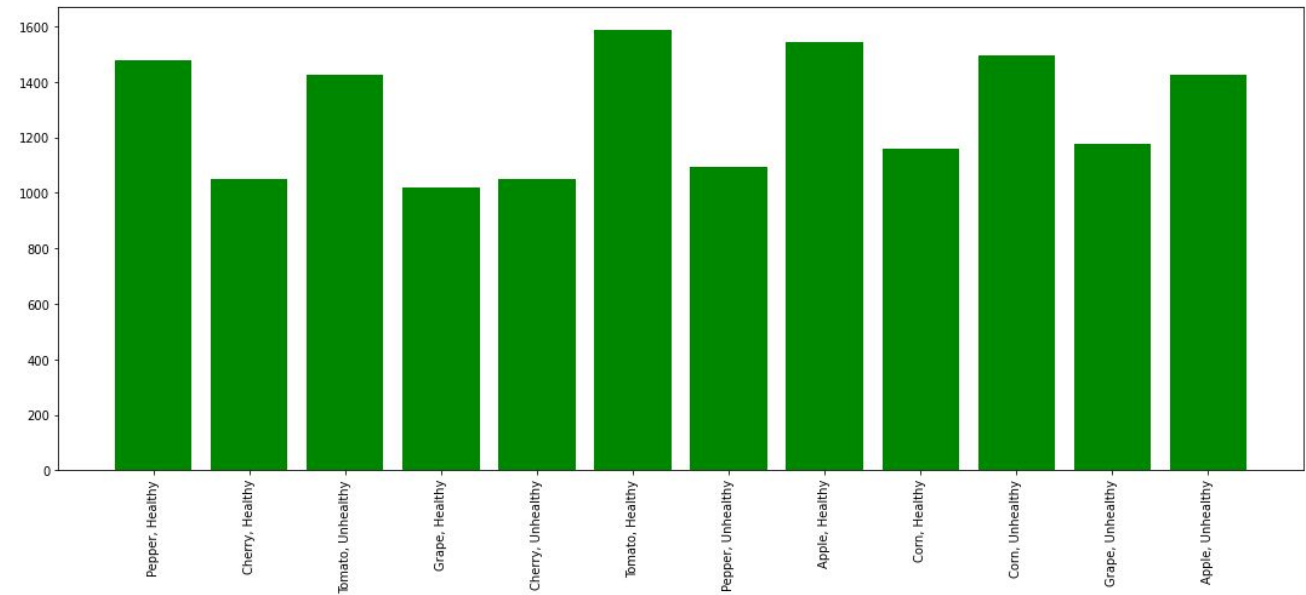




Classificazione Bi-dimensionale – PlantVillage

15500 immagini, ognuna delle quali appartiene a due tra le seguenti 8 classi:

- Healthy
- Unhealthy
- Pepper
- Cherry
- Tomato
- Grape
- Apple
- Corn



Distribuzione delle immagini nelle classi

Obiettivo: classificare lo stato di salute e il diverso tipo di foglia

Steps

1. Caricamento e organizzazione Dataset
2. Divisione dati in training, validation e test
3. Classificazione CNN
4. Valutazione performance

['Apple', 'Healthy']



['Tomato', 'Unhealthy']



['Apple', 'Unhealthy']



['Tomato', 'Healthy']



['Pepper', 'Healthy']



['Apple', 'Healthy']



['Tomato', 'Healthy']



['Corn', 'Healthy']



['Corn', 'Healthy']



['Tomato', 'Unhealthy']



['Apple', 'Unhealthy']



['Apple', 'Healthy']



Caricamento e organizzazione Dataset

1. A partire dalla cartella contenente le immagini viene creato un dizionario avente come chiavi il percorso di ogni immagine e come valori le label
2. Il dizionario viene convertito in dataframe
3. Le label vengono rappresentate in modo binario

	image	label	Healthy	Unhealthy	Apple	Cherry	Corn	Grape	Pepper	Tomato
0	Pepper, Healthy/7b3b16d2-e0d2-4c91-acc1-3778b3...	[Pepper, Healthy]	1	0	0	0	0	0	1	0
1	Pepper, Healthy/c0183c29-b09d-42a6-a376-90806c...	[Pepper, Healthy]	1	0	0	0	0	0	1	0
2	Pepper, Healthy/9b3b71d1-ad33-4512-8dd2-a7277e...	[Pepper, Healthy]	1	0	0	0	0	0	1	0
3	Pepper, Healthy/71e8540f-309c-4022-97c7-7ce276...	[Pepper, Healthy]	1	0	0	0	0	0	1	0
4	Pepper, Healthy/5b6539ac-0d6d-401b-a862-fd4730...	[Pepper, Healthy]	1	0	0	0	0	0	1	0
...
15504	Apple, Unhealthy/3908d563-a8f8-4206-b09e-1ae34...	[Apple, Unhealthy]	0	1	1	0	0	0	0	0
15505	Apple, Unhealthy/52d8da3b-5d74-43ba-b527-bc8fc...	[Apple, Unhealthy]	0	1	1	0	0	0	0	0
15506	Apple, Unhealthy/6344cfa8-2c26-4498-b366-817d5...	[Apple, Unhealthy]	0	1	1	0	0	0	0	0
15507	Apple, Unhealthy/fb911d9c-0445-4d0f-886e-50458...	[Apple, Unhealthy]	0	1	1	0	0	0	0	0
15508	Apple, Unhealthy/80ddd81b-9c75-4711-8012-7ee68...	[Apple, Unhealthy]	0	1	1	0	0	0	0	0

15509 rows x 10 columns

Divisione dati in training, validation e test

- Utilizzando la tecnica di campionamento stratificato, il 70% dei dati è stato utilizzato come training, il 20% come validation e il 10% come test.
- Tutte le immagini hanno dimensione 224x224
- I pixel assumono valori nel range 0-1

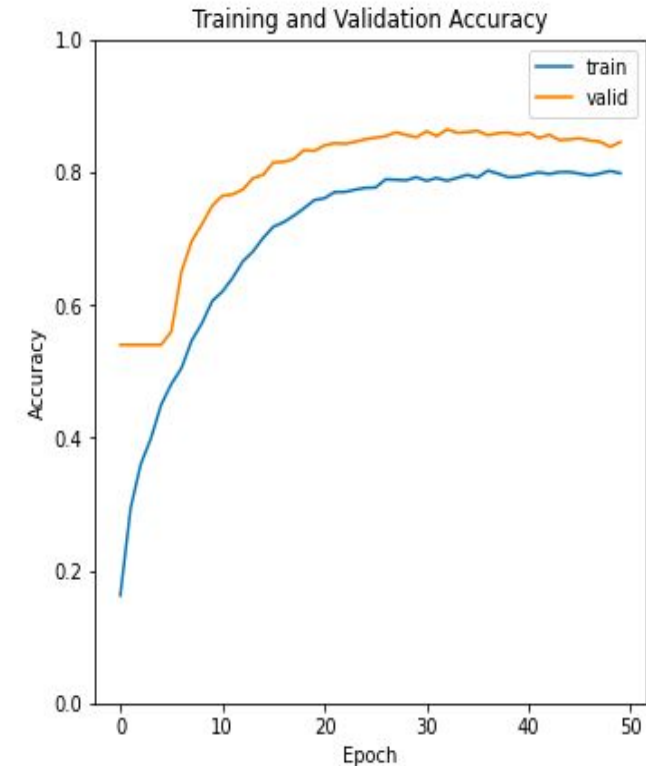
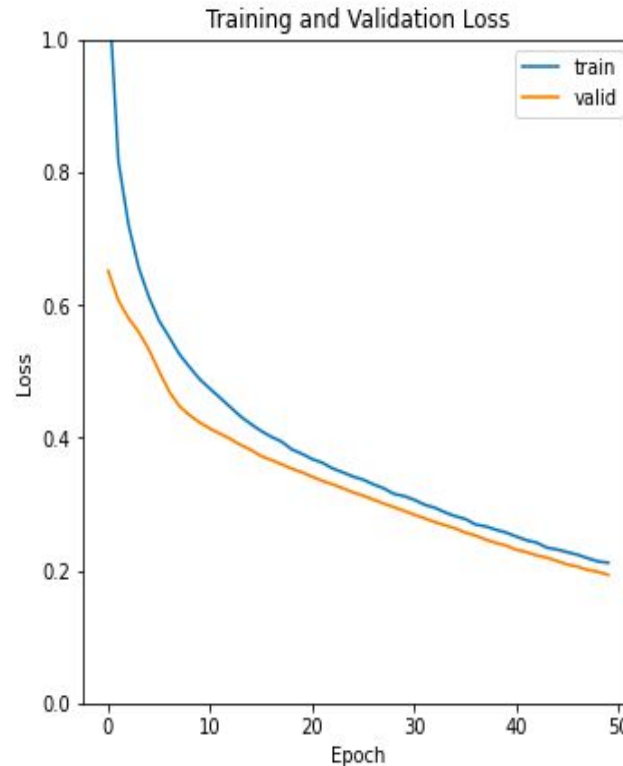
Classificazione CNN

Model name	Network Architecture	Categorical accuracy and Loss values
Model 1	CNN (32, 64, 128) + batch normalization + 1 layer Dense (128)	Acc. 0.77, Loss 0.27 Val_Acc. 0.77, Loss 0.27
Model 2	Data Augmentation CNN (32, 64, 128) + batch normalization + 1 layer Dense (128) + Dropout (0.5)	Acc. 0.79, Loss 0.38 Val_Acc. 0.86, Loss 0.33
Model 3	Data Augmentation CNN (32, 64, 128) + batch normalization + Dropout (0.25) + 2 layer Dense (256, 512) + Dropout (0.5)	Acc. 0.75, Loss 0.23 Val_Acc. 0.79, Loss 0.3
Model 4	Data Augmentation CNN (32, 64, 128) + batch normalization + Dropout (0.1) + 2 layer Dense (256, 512) + Dropout (0.5)	Acc. 0.79, Loss 0.21 Val_Acc. 0.85, Loss 0.19
Model 5	Data Augmentation Pre-trained MobileNetV2 + 3 layer Dense (128, 256, 512) + Dropout (0.5)	Acc. 0.53, Loss 0.50 Val_Acc. 0.57, Loss 0.49
Model 6	Data Augmentation Pre-trained ResNet50 + 3 layer Dense (128, 256, 512) + Dropout (0.5)	Acc. 0.56, Loss 0.49 Val_Acc. 0.55, Loss 0.47

Modello Migliore

Architettura

- Input Layer
- 2D convolutional layer: 3x3 filter, 32 neurons
- Batch Normalization layer
- ReLu activation function
- Dropout layer (0.1)
- Max Pooling layer (3x3)
- 2D convolutional layer: 3x3 filter, 64 neurons
- Batch Normalization layer
- ReLu activation function
- Dropout layer (0.1)
- Max Pooling layer (3x3)
- 2D convolutional layer: 3x3 filter, 128 neurons
- Batch Normalization layer
- ReLu activation function
- Dropout layer (0.1)
- Global Max Pooling layer (3x3)
- Dense layer: 256 neurons, ReLu activation function
- Dropout layer (0.5)
- Dense layer: 512 neurons, ReLu activation function
- Dropout layer (0.5)
- Output layer: Sigmoid activation function

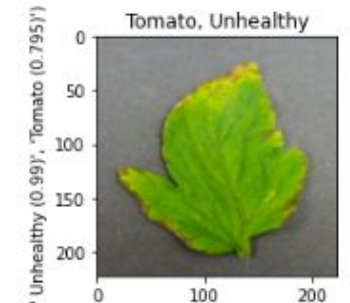
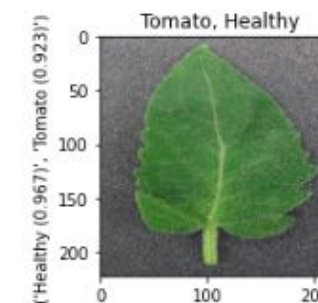
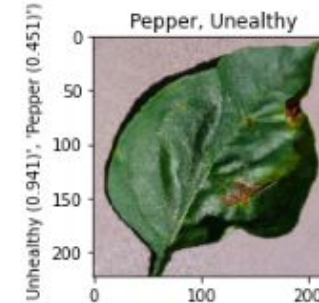
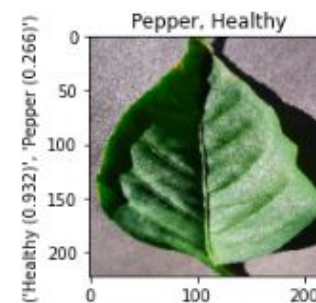
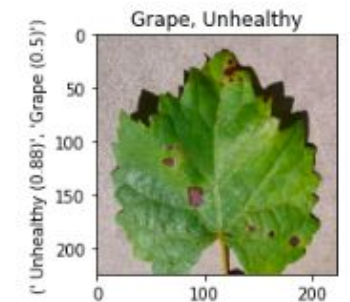
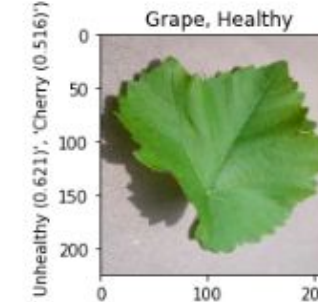
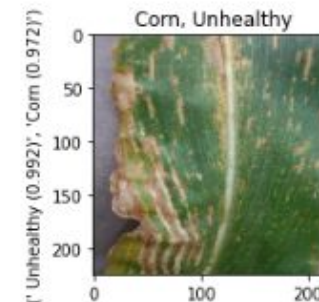
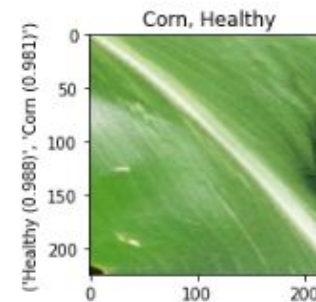
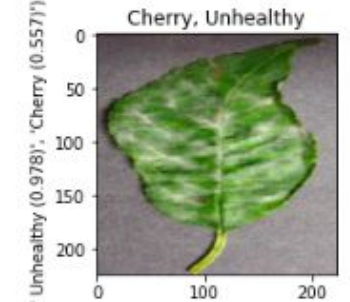
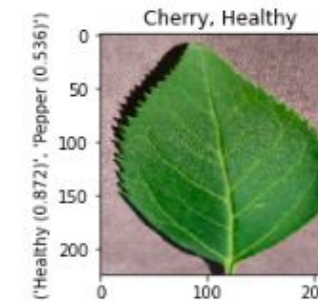
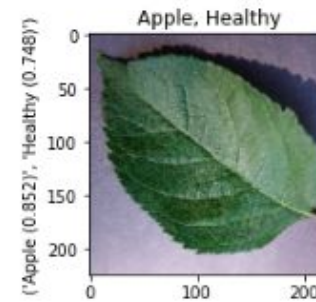


Valutazione Modello Migliore

Sul dataset di Test ottiene un valore di Accuracy di 0.82 e un valore di Loss di 0.18.

Nell'esempio classifica correttamente 10 immagini su 12:

- non riesce a individuare correttamente la foglia di ciliegio e la foglia d'uva
- riesce a determinare correttamente lo stato di salute per tutte le foglie



File

{x}

gdrive

sample_data

Apple,Unhealthy.JPG

Cherry,Healthy.JPG

Tomato,Healthy.JPG

+ Codice + Testo

Evaluation - Demo

Model 4

✓ [4] model4=None
model4=keras.models.load_model('gdrive/MyDrive/DSIM Project/BI-DIMENS

2 s


✓

▶

 classes = np.array(data_label)
img = image.load_img('gdrive/MyDrive/DSIM Project/BI-DIMENSIONAL/Apple,Unhealthy.JPG')
img = image.img_to_array(img)
img = img/255
proba = model4.predict(img.reshape(1,224,224,3))
top_2 = np.argsort(proba[0])[:-3:-1]
for i in range(2):
 print("{} ".format(classes[top_2[i]])+ " {:.3f}".format(proba[0][top_2[i]]))
plt.imshow(img)

RAM
Disco

Apple,Unhealthy.JPG



<>

Disco 53.80 GB disponibili

Content-Based Image Retrieval – Food101

- 101000 immagini di diversi tipi di cibo
- 101 classi, ognuna con 1000 immagini

Obiettivo: restituire le 10 immagini più simili all'immagine di input



Steps

1. Caricamento e Organizzazione Dataset
2. Creazione delle Features
 - MobileNetV2
 - Autoencoder
3. Estrazione delle immagini più simili
 - KDtree

Caricamento e organizzazione del dataset

È stata creata una funzione per selezionare:

- quante immagini estrarre in totale
- quante sottocartelle considerare
- quante immagini di cibo considerare per ogni sottocartella

Specificando il feature extractor, vengono estratte le features dalle immagini prese in considerazione

Creazione delle Features

- Rete pre-addestrata MobileNetV2
- Autoencoder:
 - Encoder: riduce la dimensionalità del dato in input
 - Decoder: ricostruisce il dato originale
 - Attraverso il training l'encoder impara a generare una rappresentazione compressa limitando al minimo la perdita di dati.

Siradjuddin, Indah Agustien, Wrida Adi Wardana, and Mochammad Kautsar Sophan. "Feature extraction using self-supervised convolutional autoencoder for content based image retrieval." *2019 3rd International Conference on Informatics and Computational Sciences (ICICoS)*. IEEE, 2019.

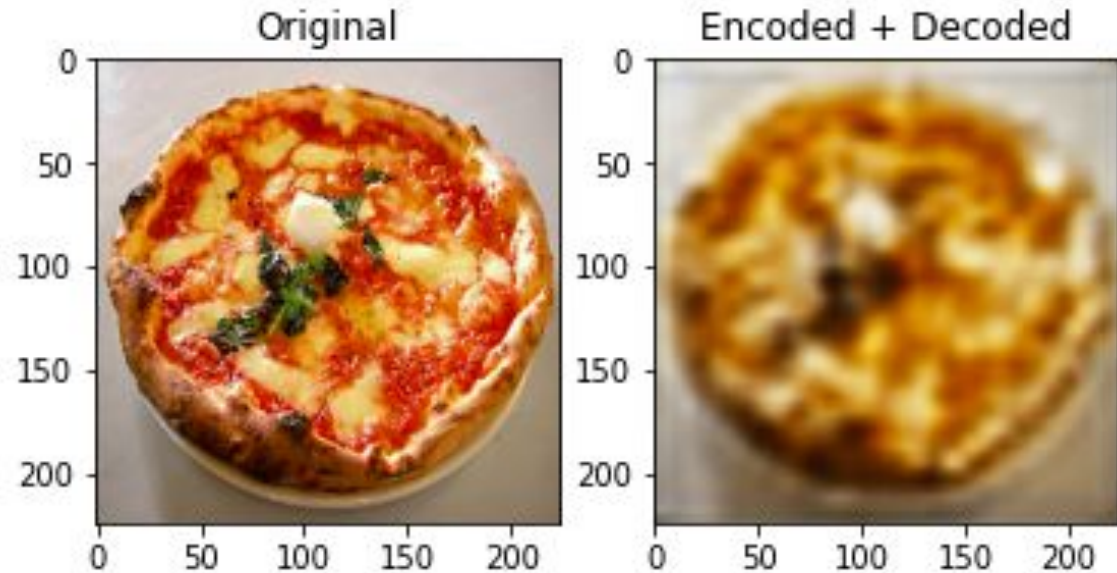
Architettura Autoencoder

Encoder:

- Conv2D layer: 64 filtri 3x3, ReLU
- MaxPooling: 2x2
- Conv2D layer: 32 filtri 3x3, ReLU
- MaxPooling: 2x2
- Conv2D layer: 16 filtri 3x3, ReLU
- MaxPooling: 2x2

Decoder:

- Conv2DTranspose: 16 filtri 3x3, ReLU
- UpSampling2D 2x2
- Conv2DTranspose: 32 filtri 3x3, ReLU
- UpSampling2D 2x2
- Conv2DTranspose: 64 filtri 3x3, ReLU
- UpSampling2D 2x2
- Conv2D: 3 filtri 3x3, ReLU



Estrazione delle immagini più simili

Per cercare le immagini più simili presenti nel training set, è stato utilizzato il modello KDtree.



Query Image

Immagini più simili - MobileNetV2 + KDtree

1. risotto
dist: 81.294



2. risotto
dist: 81.956



3. risotto
dist: 83.299



4. risotto
dist: 83.45



5. spaghetti_carbonara
dist: 84.122



6. macaroni_and_cheese
dist: 84.669



7. risotto
dist: 85.454



8. clam_chowder
dist: 85.651



9. shrimp_and_grits
dist: 85.716



10. macaroni_and_cheese
dist: 85.788



Immagini più simili - Encoder + KDtree

1. apple_pie
dist: 11.633



2. shrimp_and_grits
dist: 15.749



3. cup_cakes
dist: 16.249



4. clam_chowder
dist: 16.393



5. apple_pie
dist: 16.545



6. caesar_salad
dist: 16.616



7. risotto
dist: 16.691



8. club_sandwich
dist: 16.699



9. waffles
dist: 16.718



10. creme_brulee
dist: 16.94



Sommario

Content-Based Image Retrieval

Import packages

Import Data

Data Creation

MobileNetV2

Feature extraction with MobileNetV2

Retrieving similar image

Features MobileNetV2 + KDTree

Autoencoder

Test with image from training set

Test with image not used for training

Feature extraction encoder layer in convolutional autoencoder

Retrieving similar image

Features encoder + KDTree

+ Sezione


+ Codice + Testo

RAM
Disco

Features MobileNetV2 + KDTree

✓ [13] %%time
#calcolo l'albero basato sulle features di mobilenet
tree_mn = KDTree(X_train_mn)

CPU times: user 2 s, sys: 126 ms, total: 2.13 s
Wall time: 2.13 s

#query image
img_query = cv.imread('gdrive/MyDrive/DSIM Project/CBIR/club_sandwich.jpg')
img_query = cv.resize(img_query, (224, 224))
query_features = mobilenetv2_features(img_query)
query_features = np.expand_dims(query_features, axis = 0)
distance, idx = tree_mn.query(query_features, k=10)

↑ ↓ ↶ ↷ ⚙ 📄 🗑 ⋮

✓ [23] plt.imshow(cv.cvtColor(img_query, cv.COLOR_BGR2RGB))
plt.axis("off")
plt.title("Query image")
plt.show()

✓ [24] plt.figure(figsize=(20, 20))
for i in range(10):
img = cv.imread(paths_mn[idx[0][i]])
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)

Grazie per la vostra attenzione!