

PLANT classifier

Mattia Spazzoli

Addestramento del classificatore - Settings

```
jupyter plant_classifier_training (autosaved) Python 3 (ipykernel) Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)
In [1]: #LIBRARY IMPORT
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
print(tf.__version__)

2.5.0

In [2]: #CONSTANT DEFINE
IMAGE_SIZE = 256
BATCH_SIZE = 32
CHANNELS = 3
EPOCHS = 200

In [3]: #IMPORT DATASET WITH LABELS FROM DIRECTORY
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "dataset_pv",
    shuffle=True,
    image_size = (IMAGE_SIZE, IMAGE_SIZE),
    batch_size = BATCH_SIZE,
)

Found 9140 files belonging to 10 classes.

In [4]: #PRINT LABELS NAME
class_names = dataset.class_names
class_names

Out[4]: ['Apple',
'Blueberry',
'Corn',
'Grape',
'Peach',
'Pepper',
'Potato',
'Raspberry',
'Strawberry',
'Tomato']

In [5]: #PRINT BATCHS NUMBER
len(dataset)

Out[5]: 286
```

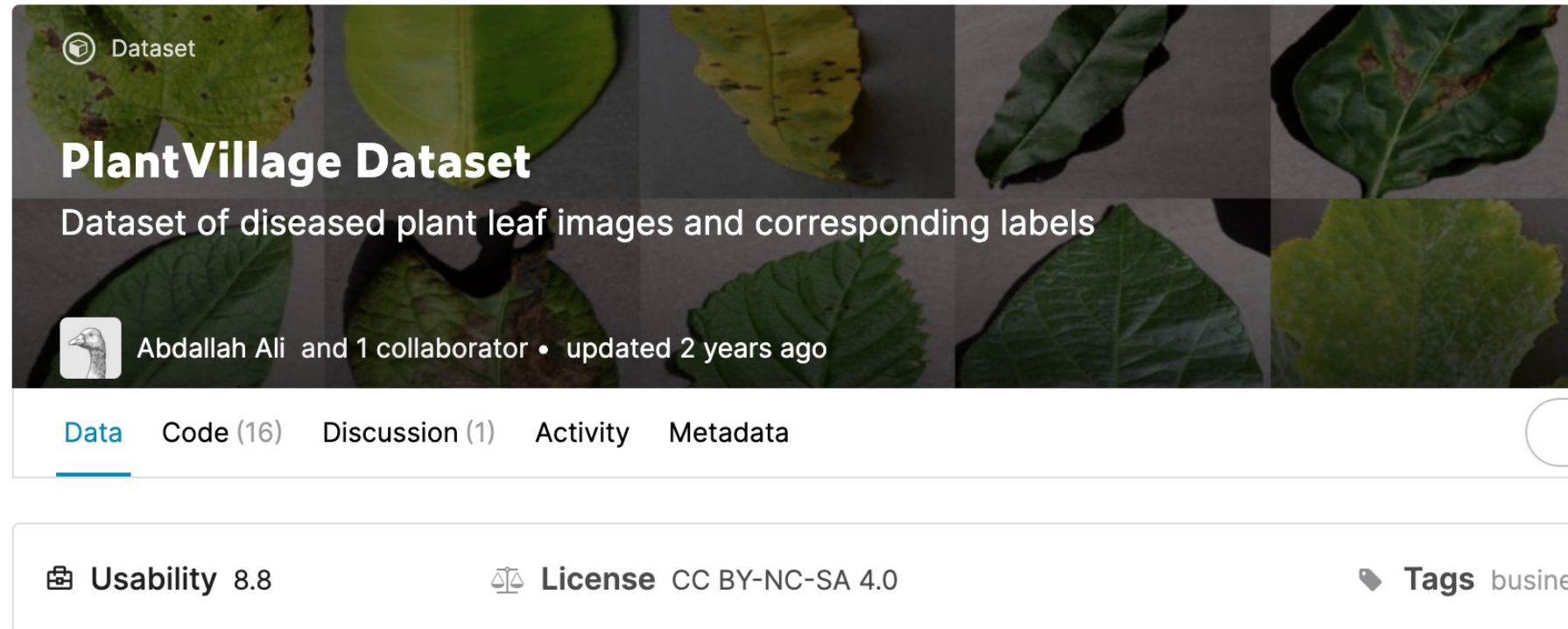


Addestramento del classificatore - DataSet

→ kaggle

→ 50000+ immagini


→ 38 classi






Dataset

PlantVillage Dataset

Dataset of diseased plant leaf images and corresponding labels

 Abdallah Ali and 1 collaborator • updated 2 years ago

[Data](#) [Code \(16\)](#) [Discussion \(1\)](#) [Activity](#) [Metadata](#)

 **Usability** 8.8  **License** CC BY-NC-SA 4.0  **Tags** business

<https://www.kaggle.com/abdallahalidev/plantvillage-dataset>

Addestramento del classificatore - Preprocessing

```
#IMPORT DATASET WITH LABELS FROM DIRECTORY
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "dataset_pv",
    shuffle=True,
    image_size = (IMAGE_SIZE, IMAGE_SIZE),
    batch_size = BATCH_SIZE,
)
```

Caricamento del dataset

```
#FETCH SOME FUNCTIONS TO USE THEM IN FOLLOW EPOCHS
train_ds=train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds=val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds=test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

Prefetch

```
#SCALE RGB TENSOR TO 0-1 VALUES
resize_and_rescale= tf.keras.Sequential([layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
                                          layers.experimental.preprocessing.Rescaling(1.0/255)])
```

Resize and rescale

```
#FLIP AND ROTATE RANDOMLY IMAGES
data_augmentation=tf.keras.Sequential([layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
                                       layers.experimental.preprocessing.RandomRotation(0.2)])
```

Flip and rotation

Addestramento del classificatore - Architettura

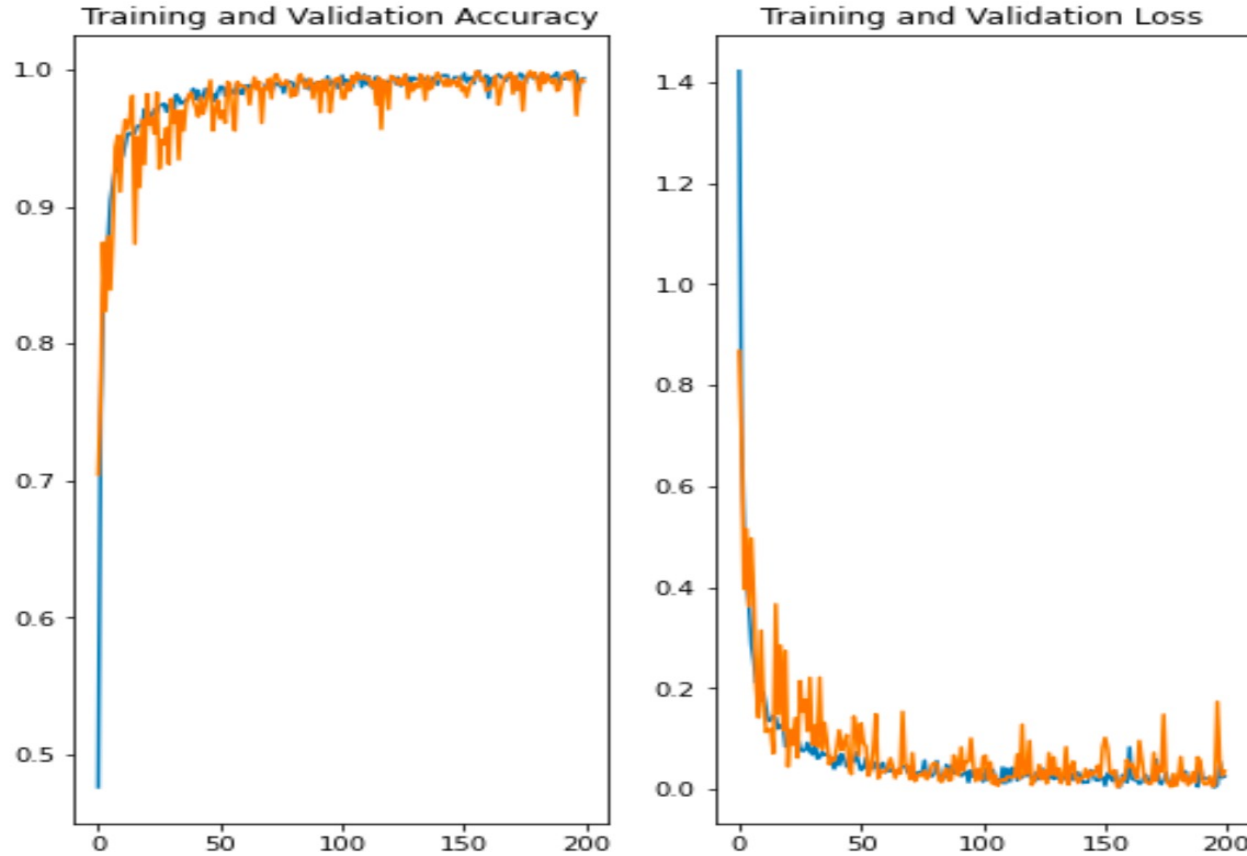
- 6 layer convoluzionali e 6 di pooling
- Funzione di loss: SparseCategoricalCrossentropy
- Batch da 32 immagini
- 10 classi di output

```
#CNN ARCHITECTURE CREATE
input_shape=(BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = len(class_names)
model = models.Sequential([
    #Applies preprocessing layers
    resize_and_rescale,

    #Applies CNN layers
    layers.Conv2D(32,(3,3), activation='relu', input_shape=input_shape), #Convolutional layer
    layers.MaxPooling2D((2,2)), #Pooling layer
    layers.Conv2D(64, kernel_size=(3,3), activation='relu'), #Convolutional layer
    layers.MaxPooling2D((2,2)), #Pooling layer
    layers.Conv2D(64, kernel_size=(3,3), activation='relu'), #Convolutional layer
    layers.MaxPooling2D((2,2)), #Pooling layer
    layers.Conv2D(64,(3,3), activation='relu'), #Convolutional layer
    layers.MaxPooling2D((2,2)), #Pooling layer
    layers.Conv2D(64,(3,3), activation='relu'), #Convolutional layer
    layers.MaxPooling2D((2,2)), #Pooling layer
    layers.Conv2D(64,(3,3), activation='relu'), #Convolutional layer
    layers.MaxPooling2D((2,2)), #Pooling layer

    #Flat neurons
    layers.Flatten(),
    #Applies the rectified linear unit activation function
    layers.Dense(64, activation="relu"),
    #Converts a vector of values to a probability distribution
    layers.Dense(n_classes, activation='softmax')
])
```

Addestramento del classificatore - Risultati



Configurazioni di training

200 epoche

30 ore di addestramento

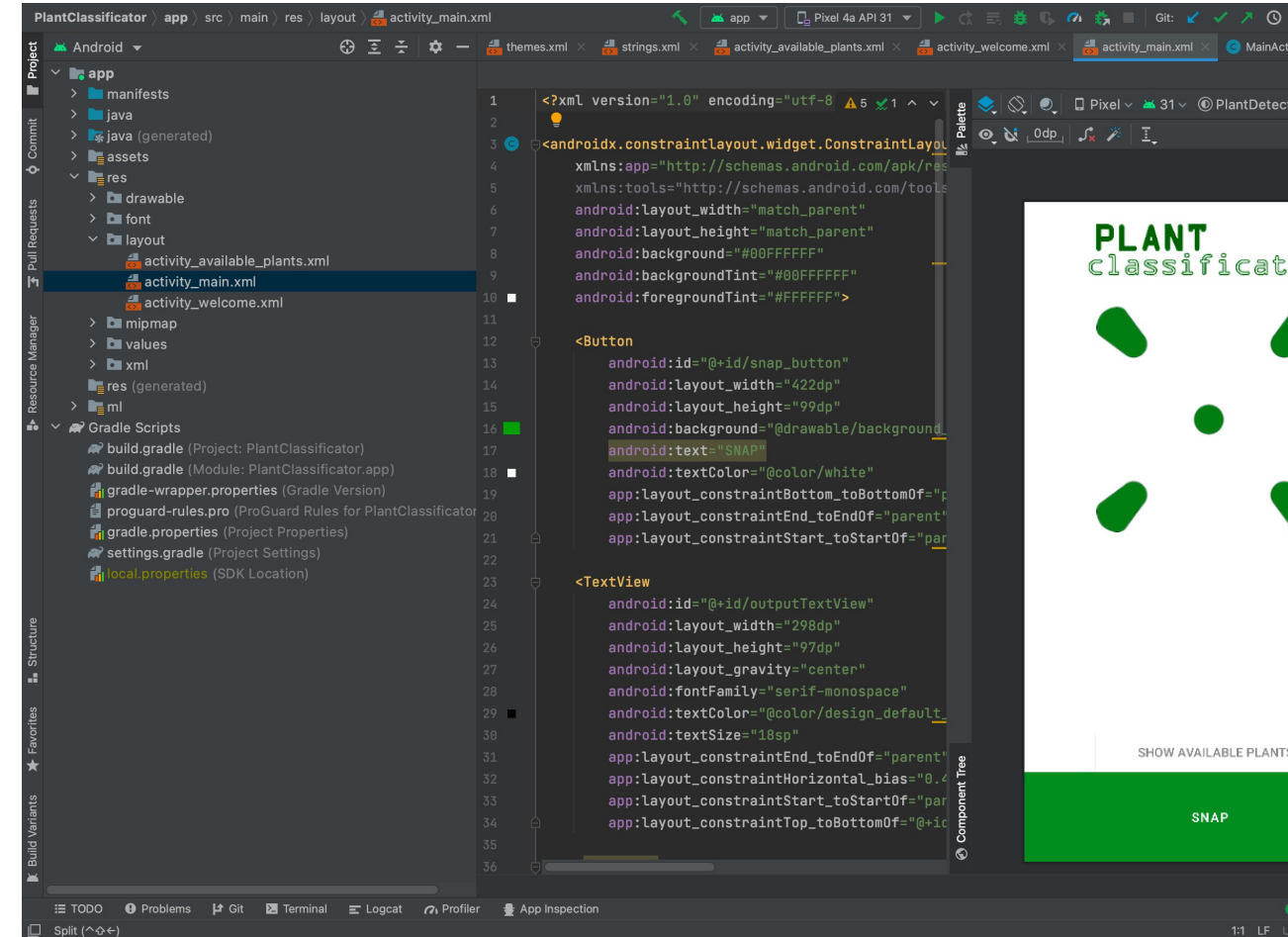
Prestazioni del modello sul DataSet di Test

Loss: 0.008

Accuracy: 99.7 %

Applicazione Android - Introduzione

- Android Studio Artic Fox 2020.3.1
- Activity class .java
- Activity layout .xml
- AndroidManifest.xml e build.gradle



Applicazione Android – Apertura



Looper da 1 secondo

Permission per fotocamera

SNAP

SHOW AVAILABLE PLANTS

Applicazione Android - SNAP

Apertura fotocamera

Acquisizione dell'immagine

Inferenza sul modello TensorflowLite

Match tra probabilità e labels

Setting dell'ImageView e della
TextView con i risultati ottenuti

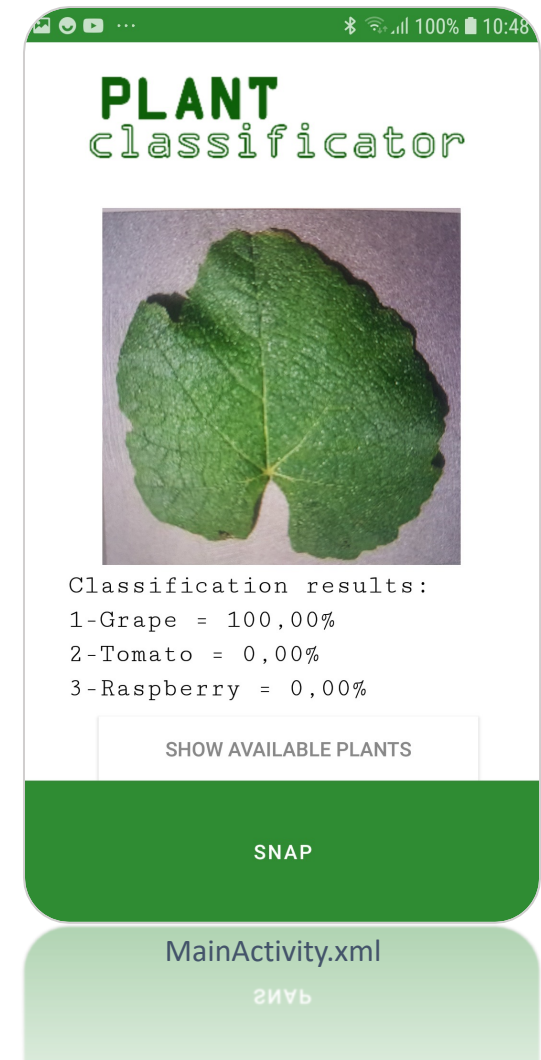
```
// Create Buffer di input
TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 256, 256, 3}, DataType.FLOAT32);
inputFeature0.loadBuffer(tensorImage.getBuffer());

// Create Buffer di output
float[][] result = new float[1][10];

MappedByteBuffer tfliteModel
    = FileUtil.loadMappedFile(context, this,
        filePath: "tf_lite_model.tflite");
InterpreterApi tflite = new InterpreterFactory().create(
    tfliteModel, new InterpreterApi.Options());

// Run inference
if (null != tflite) {
    tflite.run(inputFeature0.getBuffer(), result);
}
```

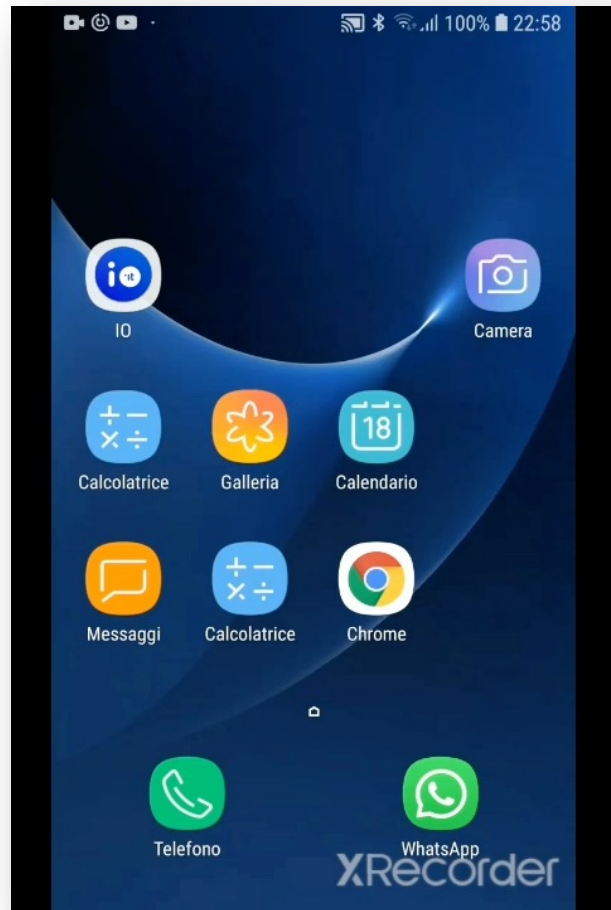
Tempo dell'inferenza: 147 ms



MainActivity.xml

SNAP

Applicazione Android – Show available plants



AvailablePlants.xml