RRC (Red Rose Condition – Condición de Rosa Roja) (February de 2022)

Johann S. Kwan A., student of la Universidad Surcolombiana

Abstract

Context: It is said that a person can identify if a red rose is withered or not, but there have been cases that, due to mistrust of the customer to the seller, they come to ask "Isn't it damaged?", with what refers to if the red rose is withered or no longer in good condition.

Methods: There have been works that it is possible to measure the quality of roses and other projects have managed to design efficient artificial vision models. However, the works are oriented to the agricultural environment, given this, a model is needed that clarifies the characteristics to a client..

Results: This project categorizes whether a red rose is in “aceptable” condition, “marchita desde hace 1 a 4 días” or “marchita desde hace mucho”, being “mucho” more than 4 days, with an average accuracy percentage of 81.23%..

Conclusion: The results show a good precision, according to the rubric with which this project is evaluated, with this managing to satisfy the initial problem.

Keywords: red rose condition categorization, red rose condition classification model, detect withered red rose.

Resumen.

Contexto: Se dice que una persona puede identificar si una rosa roja esta marchita o no, pero se han dado casos que, por desconfianza del cliente al vendedor, se llega a preguntar “¿No esta dañada?”, con lo que se refiere a que si la rosa roja se encuentra marchita o ya no se ve en buenas condiciones.

Método: Han existido trabajos que es posible medir la calidad de las rosas y otros proyectos han logrado diseñar modelos visión artificial eficientes. No obstante, los trabajos son orientados al entorno agrícola, dado esto se necesita de un modelo que aclare las características a un cliente.

Resultados: Este proyecto categoriza si una rosa roja está en condición “aceptable”, “marchita desde hace 1 a 4 días” o “marchita desde hace mucho”, siendo “mucho” más de 4 días, con un porcentaje de precisión promedio del 81.23%.

Conclusión: Los resultados muestran una precisión buena, según la rúbrica con la cual se evalúa este proyecto, con esto logrando satisfacer la problemática inicial.

Palabras clave: categorización de la condición de una rosa roja, modelo de clasificación del estado de una rosa roja, detectar rosa roja marchita.

Estudiante de Ingeniería de Software de la Universidad Surcolombiana. Neiva, Colombia.
Contacto: u20191179607@usco.edu.co

I. Introduction

This document shows how the "RRC" project was carried out for the subject of "Data Science", what was needed, what was used, how it works, what its purpose is and how it fulfilled this.

II. Problem Statement

When buying roses, it is common for a customer to ask: "Are they in good condition?" And depending on the situation of the seller it can be difficult to answer. To answer this question correctly, a program or application can be developed that allows us to know approximately how many days the red rose has left, without water or with water, before it stops looking presentable.

There are between 5 to 500 florists for each municipality in Colombia approximately and red roses are sold in all of them, and they are even sold in more types of stores, it is normal for customers to want to know if the red roses are in good condition or if they are already withered . The problem with this is that it causes confusion and distrust for the seller, in case the roses last less than expected.

How can an artificial vision model be implemented to this problem? To solve the problem, it is possible to create an application that can tell if a red rose is in good condition or withered, according to the image that is captured of the red rose. Since my mother is the owner of a flower shop, I know the reality of the problem, I will be able to take many guide images, some free of charge and others with a minimum cost, which will serve to feed the intelligence of the application and also by having the results you can decide if it is useful or not.

Specific objectives

- Make the application able to identify the presentable lifetime of a red rose.
- Be easy to use.
- be visibly nice.

III. State of the art

- Design of an Artificial Vision System for the Analysis of Quality and Production of Roses: This project seeks to evaluate the quality of roses, it helped me for my project since within it certain

aspects similar to mine are dealt with, these aspects are: the use of artificial vision, the analysis of a rose, the identification of unpresentable parts of a rose and the methodology used by this project is helpful to me. [3]

Methodology: The methodology used for this project consisted of five parts, these were:

- Documentation and research on artificial vision techniques applied to the agricultural sector.
- Evaluation of the feasibility of the implementation in Matlab.
- System design.
- Development of the different algorithms.
- Testing and bug fixing.

Average Accuracy Percentage: 67.1% [3]

- Design of a flower identification and classification system: This project seeks to identify different parameters of roses, in order to classify the degree of quality of roses. This project deals with elements similar to mine, things such as the identification of aspects of a rose with artificial vision, quality rating of a rose, this being similar since it is often said that a withered rose or a rose with signs of being close to wilt is a low quality product.[4][17][18]
  Methodology: In this project he proposes the following step by step to identify and classify roses:
  - Acquisition and digitization of the image.
  - Processing.
  - Description.
  - Classification.
  - Decision making. [4]

  Maximum accuracy percentage achieved: 93%.
- Rose classification system of the explorer variety, using computer vision: In the classification system project roses are classified according to their characteristics, one of the characteristics that is classified is the color, which is carried out correctly, so it was useful for my project, since by changing color the roses can be identified as wilted or with signs of being close to wilting. [5]
- Classification of images (dogs and cats): This code represents the website, once the artificial intelligence model is created and trained with Python, Tensorflow and other utilities, given this and that a mathematical model similar to the one I use is used, it was useful to me, to base myself on this code.
  Average Accuracy Percentage: 67.1%. [2] [6]
- API_DeepLearning: In this code a neural network is implemented using keras-tensorflow and executed in a flask web service. [1]

.

IV. Theoretical Framework/Conceptual

¿What is RRC?, An application that allows to identify approximately how much presentable life a rose has left, defining that a rose is presentable when it does not look completely or partially withered, also specifying that a red rose begins to look withered when certain parts of the petals turn brown.

¿How works RRC?, The application needs to import the image of a red rose whose presentable lifetime is desired, the image must have the same specifications that were used to take the photos of the dataset, after the image is imported, it will be analyzed and will tell, via text, how many days of presentable time the red rose has left.

¿Where is it applied RRC?, it serves in the event that a person or persons wish to know the condition of a rose.

¿ What does it mean when a red rose is withered?, It is known that a red rose or a common plant is withered when its leaves or petals are brittle, if the stem changes color from green to black, if the stem loses its flexibility and that the head of the rose, also called flower, is listless.

What does it mean for a rose to be in good or fair condition?, a red rose is acceptable or in good condition when its petals, leaves and stem are flexible, another aspect is that the stem is green and the flower or head of the rose is raised. [9] [10] [11]

V. Mathematic Model

The problem was analyzed if a classification system was chosen, which would divide the information into three categories, "acceptable", "Withered for 1 to 4 days" and "Withered for a long time", the first category indicating that the red rose is is in good condition, the second indicates that the rose has been wilting for one to three days and the last indicates that the rose has been wilting for more than 4 days and that it has not been provided with water for a period of time considerable.

The mathematical model chosen was the convolutional neural network, with three output neurons, these being the three categories in which the classification was made.

Since the images are resolution adjusted to 224x224pixels and with rgb color channels, it resulted in 150528 input neurons. A pre-trained model called "MobileNet" was used in its version "v2", so when representing the processes carried out by the model they are called "Hidden Layers". [7] [12] [13] [19] [20] [21] [23]

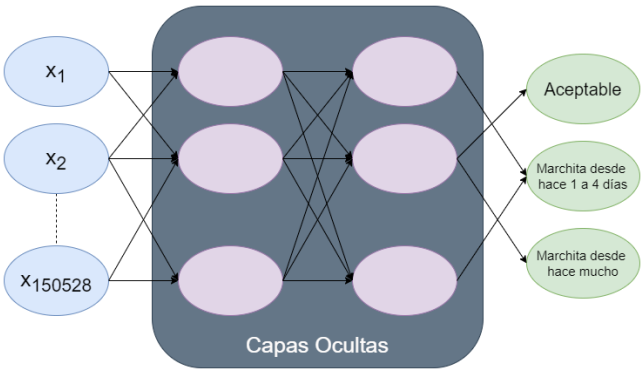


Fig. 1. Mathematical model, graphically represented.

VI. Dataset Construction

The "Camara FV-5" application for android was used, ensuring that the camera was in the same position for all the photos, the photos were taken between 10:30 p.m. and 11:10 p.m., with the illumination given by a ring of light.
3465 photos were taken of 10 roses from different angles, during 6 days to 5 roses that started in good conditions and on the last day to 5 roses that had been withered and dehydrated for a long time. When taking each photo, the red rose was rotated, zoomed in and out, with a white background and two sheets of paper that they made as a guide for the location of the roses.

Fig. 2. Example of dataset construction.

TABLA I
DATASET DISTRIBUTION

DATASET DISTRIBUTION			
Training	**80%**	**Validation**	**20%**
2772		**693**	
3465			

VII. Creation of the Prediction Model

To implement the knowledge of the problem statement, theoretical framework, mathematical model and to be able to develop a code capable of classifying the constructed dataset, the "Google Collaboratory" tool was used, also called "Colab", which runs in the cloud, outside the local system the personal computer that was used, it runs with the Python programming language.
The step by step for the implementation was:

1. The dataset is saved and organized within the “Google Drive” service.

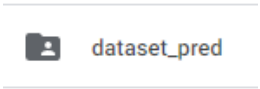


Fig. 3. Step 1, colab code.

A new code is created in “Colab”, called“RRC”.

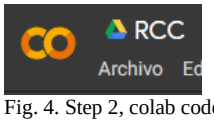


Fig. 4. Step 2, colab code.

2. Coding inside “Colab”:
   2.1. A command is executed so that “Colab” can connect to “Google Drive”.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Fig. 5. Step 3.1, colab code.

   2.2. The “dataset” folder is created inside the main folder.

```
!mkdir dataset
```

Fig. 6. Step 3.2, colab code.

   2.3. The folders "acceptable", "withered_day-1-4" and "withered_long_ago" are created.

```
!mkdir dataset/aceptable
!mkdir dataset/marchita_dia_1-4
!mkdir dataset/marchita_hace_mucho
```

Fig. 7. Step 3.3.1, colab code.

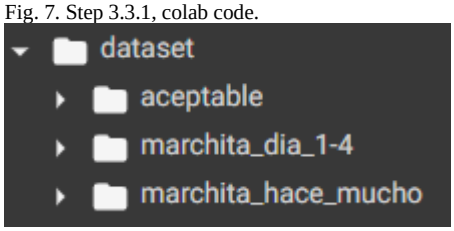


Fig. 8. Step 3.3.2, colab code.

   2.4. The “os” and “shutil” libraries are imported.

```
import os
import shutil
```

Fig. 9. Step 3.4, colab code.

   2.5. Now using certain code, the information from the “Google Drive” folders is passed to the folders created within “Colab”.

```
#Copiar imagenes que subimos a carpetas del dataset
carpeta_fuente = '/content/drive/MyDrive/datasetmater/dataset_pred/falta_1_dia'
carpeta_destino = '/content/dataset/aceptable'

imagenes = os.listdir(carpeta_fuente)
for i, nombreimg in enumerate(imagenes):
  #Copia de la carpeta fuente a la destino
  shutil.copy(carpeta_fuente + '/' + nombreimg, carpeta_destino + '/' + nombreimg)
```

Fig. 10. Step 3.5, colab code.

   2.6. Just to confirm the passage of the images, some code is executed, which counts the number of files inside the folders created previously in “Colab”.

```
!ls /content/dataset/aceptable | wc -l
1485
!ls /content/dataset/marchita_dia_1-4 | wc -l
1485
!ls /content/dataset/marchita_hace_mucho | wc -l
495
```

Fig. 11. Step 3.6, colab code.

2.7. "tensorflow.keras", "numpy" and "matplotlib" are imported. [8]

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt
```

Fig. 12. Step 3.7, colab code.

2.8. The dataset generator is created, to allow you to use the color images and assign "20%" of the dataset for validation.

```
datagen = ImageDataGenerator(
    rescale=1. / 255,
    validation_split=0.2 #20% para pruebas
)
```

Fig. 13. Step 3.8.1, colab code.

The training and validation sets are generated, also called tests, and the images are rescaled to a resolution of 224x224 pixels.. [16]

```
data_gen_entrenamiento = datagen.flow_from_directory('/content/dataset', target_size=(224,224),batch_size=32, shuffle=True, subset='training')
data_gen_pruebas = datagen.flow_from_directory('/content/dataset', target_size=(224,224),batch_size=32, shuffle=True, subset='validation')
```

Fig. 14. Step 3.8.2, colab code.

To prove that the images were loaded, rescaled and that only 3 categories remained, 10 random images are shown from those selected for training.

```
for imagen, etiqueta in data_gen_entrenamiento:
  for i in range(10):
    plt.subplot(2,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(imagen[i])
  break
plt.show()
```

```
Found 2772 images belonging to 3 classes.
Found 693 images belonging to 3 classes.
```

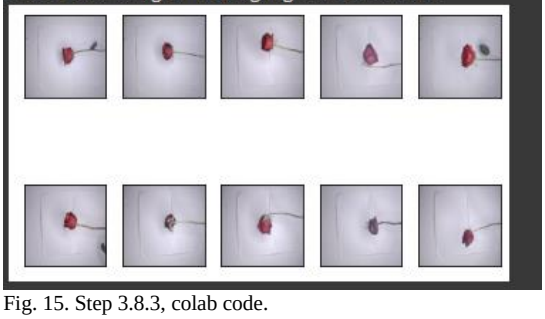


Fig. 15. Step 3.8.3, colab code.

2.9. "tensorflow", "tensorflow_hub" is imported and the pre-trained model "mobilenetv2" is imported.[7][12]

```
import tensorflow as tf
import tensorflow_hub as hub

url = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"
mobilenetv2 = hub.KerasLayer(url, input_shape=(224,224,3))
```

Fig. 16. Step 3.9, colab code.

2.10. The pretrained model is frozen so that the new trainings are added to the new model used.

```
mobilenetv2.trainable = False
```

Fig. 17. Step 3.10, colab code.

2.11. "Model" is defined, it is assigned to be for 3 categories or classes. [8]

```
modelo = tf.keras.Sequential([
    mobilenetv2,
    tf.keras.layers.Dense(3, activation='softmax')
])
```

Fig. 18. Step 3.11, colab code.

2.12. "model.sumary()" is executed, to know how many parameters will be used for training.

```
modelo.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 1280)              2257984

 dense (Dense)               (None, 3)                 3843

=================================================================
Total params: 2,261,827
Trainable params: 3,843
Non-trainable params: 2,257,984
_________________________________________________________________
```

Fig. 19. Step 3.12,colab code.

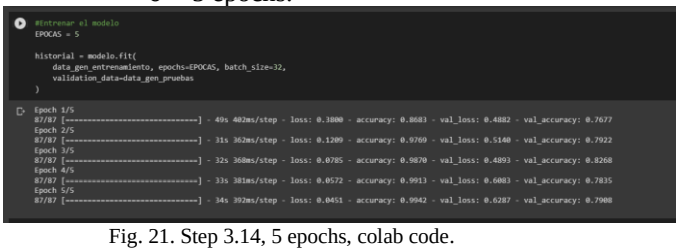
2.13. Compile so that optimizer uses "Adam" and adjust data loss to use "categorical crossentropy". [14]

```
modelo.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Fig. 20. Step 3.13, colab code.

2.14. When training the model, 5 sequences of epochs or epochs were made, in the following order:

- 5 epochs.



```
#Entrenar el modelo
EPOCAS = 5

historial = modelo.fit(
    data_gen_entrenamiento, epochs=EPOCAS, batch_size=32,
    validation_data=data_gen_pruebas
)
```

```
Epoch 1/5
87/87 [==============================] - 49s 402ms/step - loss: 0.3800 - accuracy: 0.8683 - val_loss: 0.4882 - val_accuracy: 0.7677
Epoch 2/5
87/87 [==============================] - 31s 362ms/step - loss: 0.1209 - accuracy: 0.9769 - val_loss: 0.5140 - val_accuracy: 0.7922
Epoch 3/5
87/87 [==============================] - 32s 368ms/step - loss: 0.0785 - accuracy: 0.9870 - val_loss: 0.4893 - val_accuracy: 0.8268
Epoch 4/5
87/87 [==============================] - 33s 381ms/step - loss: 0.0572 - accuracy: 0.9913 - val_loss: 0.6083 - val_accuracy: 0.7835
Epoch 5/5
87/87 [==============================] - 34s 392ms/step - loss: 0.0451 - accuracy: 0.9942 - val_loss: 0.6287 - val_accuracy: 0.7908
```

Fig. 21. Step 3.14, 5 epochs, colab code.

- 10 epochs.

```
#Entrenar el modelo
EPOCAS = 10

historial = modelo.fit(
    data_gen_entrenamiento, epochs=EPOCAS, batch_size=32,
    validation_data=data_gen_pruebas
)

Epoch 1/10
87/87 [==============================] - 34s 390ms/step - loss: 0.0369 - accuracy: 0.9975 - val_loss: 0.6958 - val_accuracy: 0.7763
Epoch 2/10
87/87 [==============================] - 33s 385ms/step - loss: 0.0310 - accuracy: 0.9975 - val_loss: 0.7119 - val_accuracy: 0.7792
Epoch 3/10
87/87 [==============================] - 34s 388ms/step - loss: 0.0263 - accuracy: 0.9978 - val_loss: 0.6043 - val_accuracy: 0.8167
Epoch 4/10
87/87 [==============================] - 34s 388ms/step - loss: 0.0227 - accuracy: 0.9989 - val_loss: 0.6355 - val_accuracy: 0.8153
Epoch 5/10
87/87 [==============================] - 34s 389ms/step - loss: 0.0200 - accuracy: 0.9993 - val_loss: 0.6849 - val_accuracy: 0.8081
Epoch 6/10
87/87 [==============================] - 33s 385ms/step - loss: 0.0181 - accuracy: 0.9993 - val_loss: 0.7620 - val_accuracy: 0.7908
Epoch 7/10
87/87 [==============================] - 33s 385ms/step - loss: 0.0157 - accuracy: 0.9993 - val_loss: 0.6944 - val_accuracy: 0.8139
Epoch 8/10
87/87 [==============================] - 33s 385ms/step - loss: 0.0146 - accuracy: 0.9993 - val_loss: 0.7646 - val_accuracy: 0.7980
Epoch 9/10
87/87 [==============================] - 34s 386ms/step - loss: 0.0126 - accuracy: 0.9996 - val_loss: 0.7115 - val_accuracy: 0.8153
Epoch 10/10
87/87 [==============================] - 33s 384ms/step - loss: 0.0118 - accuracy: 1.0000 - val_loss: 0.7150 - val_accuracy: 0.8167
```

Fig. 22. Step 3.14, 10 epochs, colab code.

- 20 epochs.

```
Epoch 11/20
87/87 [==============================] - 33s 384ms/step - loss: 0.0049 - accuracy: 1.0000 - val_loss: 0.8550 - val_accuracy: 0.8124
Epoch 12/20
87/87 [==============================] - 33s 383ms/step - loss: 0.0045 - accuracy: 1.0000 - val_loss: 0.8158 - val_accuracy: 0.8182
Epoch 13/20
87/87 [==============================] - 33s 384ms/step - loss: 0.0042 - accuracy: 1.0000 - val_loss: 0.9467 - val_accuracy: 0.7980
Epoch 14/20
87/87 [==============================] - 33s 382ms/step - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.8373 - val_accuracy: 0.8182
Epoch 15/20
87/87 [==============================] - 33s 383ms/step - loss: 0.0039 - accuracy: 1.0000 - val_loss: 0.9224 - val_accuracy: 0.8110
Epoch 16/20
87/87 [==============================] - 33s 382ms/step - loss: 0.0036 - accuracy: 1.0000 - val_loss: 0.8349 - val_accuracy: 0.8182
Epoch 17/20
87/87 [==============================] - 33s 381ms/step - loss: 0.0033 - accuracy: 1.0000 - val_loss: 0.8708 - val_accuracy: 0.8211
Epoch 18/20
87/87 [==============================] - 33s 381ms/step - loss: 0.0031 - accuracy: 1.0000 - val_loss: 0.8632 - val_accuracy: 0.8211
Epoch 19/20
87/87 [==============================] - 33s 381ms/step - loss: 0.0030 - accuracy: 1.0000 - val_loss: 0.9177 - val_accuracy: 0.8110
Epoch 20/20
87/87 [==============================] - 33s 378ms/step - loss: 0.0029 - accuracy: 1.0000 - val_loss: 0.8967 - val_accuracy: 0.8167
```

Fig. 23. Step 3.14, 20 epochs, colab code.

- 40 epochs.

```
Epoch 31/40
87/87 [==============================] - 33s 381ms/step - loss: 6.1941e-04 - accuracy: 1.0000 - val_loss: 1.1367 - val_accuracy: 0.8153
Epoch 32/40
87/87 [==============================] - 33s 381ms/step - loss: 6.0168e-04 - accuracy: 1.0000 - val_loss: 1.1344 - val_accuracy: 0.8139
Epoch 33/40
87/87 [==============================] - 33s 381ms/step - loss: 5.6329e-04 - accuracy: 1.0000 - val_loss: 1.1438 - val_accuracy: 0.8139
Epoch 34/40
87/87 [==============================] - 33s 380ms/step - loss: 5.4546e-04 - accuracy: 1.0000 - val_loss: 1.1512 - val_accuracy: 0.8139
Epoch 35/40
87/87 [==============================] - 33s 381ms/step - loss: 5.1893e-04 - accuracy: 1.0000 - val_loss: 1.2026 - val_accuracy: 0.8095
Epoch 36/40
87/87 [==============================] - 33s 382ms/step - loss: 4.8415e-04 - accuracy: 1.0000 - val_loss: 1.2286 - val_accuracy: 0.8081
Epoch 37/40
87/87 [==============================] - 33s 380ms/step - loss: 4.8417e-04 - accuracy: 1.0000 - val_loss: 1.1934 - val_accuracy: 0.8139
Epoch 38/40
87/87 [==============================] - 33s 380ms/step - loss: 4.5145e-04 - accuracy: 1.0000 - val_loss: 1.1995 - val_accuracy: 0.8124
Epoch 39/40
87/87 [==============================] - 33s 379ms/step - loss: 4.3777e-04 - accuracy: 1.0000 - val_loss: 1.2106 - val_accuracy: 0.8124
Epoch 40/40
87/87 [==============================] - 33s 380ms/step - loss: 4.2166e-04 - accuracy: 1.0000 - val_loss: 1.1810 - val_accuracy: 0.8153
```

Fig. 24. Step 3.14, 40 epochs, colab code.

- 60 epochs.

```
Epoch 51/60
87/87 [==============================] - 34s 385ms/step - loss: 4.9022e-05 - accuracy: 1.0000 - val_loss: 1.6226 - val_accuracy: 0.8038
Epoch 52/60
87/87 [==============================] - 33s 384ms/step - loss: 4.8658e-05 - accuracy: 1.0000 - val_loss: 1.5395 - val_accuracy: 0.8139
Epoch 53/60
87/87 [==============================] - 33s 385ms/step - loss: 4.3517e-05 - accuracy: 1.0000 - val_loss: 1.5302 - val_accuracy: 0.8139
Epoch 54/60
87/87 [==============================] - 33s 385ms/step - loss: 4.3076e-05 - accuracy: 1.0000 - val_loss: 1.6190 - val_accuracy: 0.8066
Epoch 55/60
87/87 [==============================] - 33s 384ms/step - loss: 4.1047e-05 - accuracy: 1.0000 - val_loss: 1.5456 - val_accuracy: 0.8139
Epoch 56/60
87/87 [==============================] - 33s 382ms/step - loss: 3.9660e-05 - accuracy: 1.0000 - val_loss: 1.5668 - val_accuracy: 0.8139
Epoch 57/60
87/87 [==============================] - 34s 385ms/step - loss: 4.1041e-05 - accuracy: 1.0000 - val_loss: 1.5650 - val_accuracy: 0.8139
Epoch 58/60
87/87 [==============================] - 34s 386ms/step - loss: 3.6791e-05 - accuracy: 1.0000 - val_loss: 1.6128 - val_accuracy: 0.8153
Epoch 59/60
87/87 [==============================] - 34s 385ms/step - loss: 3.6003e-05 - accuracy: 1.0000 - val_loss: 1.5874 - val_accuracy: 0.8139
Epoch 60/60
87/87 [==============================] - 33s 386ms/step - loss: 3.3788e-05 - accuracy: 1.0000 - val_loss: 1.6176 - val_accuracy: 0.8139
```

Fig. 25. Step 3.14, 60 epochs, colab code.

Resulting in training accuracy being **100%** and validation or testing accuracy being **81.23%** on average.

2.15. To observe the accuracy of the model, a graph was made for each sequence of epochs.

- 5 epochs.

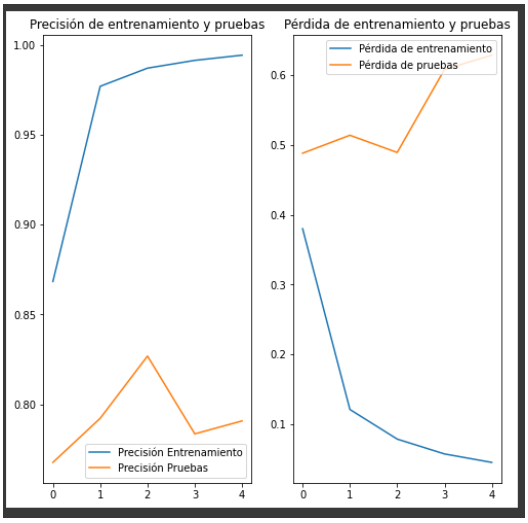


Fig. 26. Step 3.15, plot 5 epochs, colab code.

- 10 epochs.

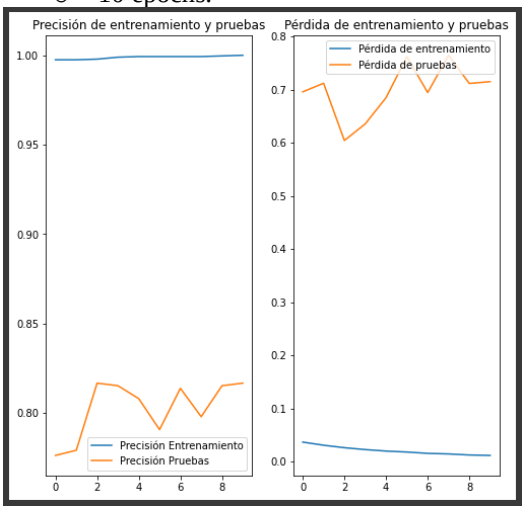


Fig. 27. Step 3.15, plot 10 epochs, colab code.

- 20 epochs.

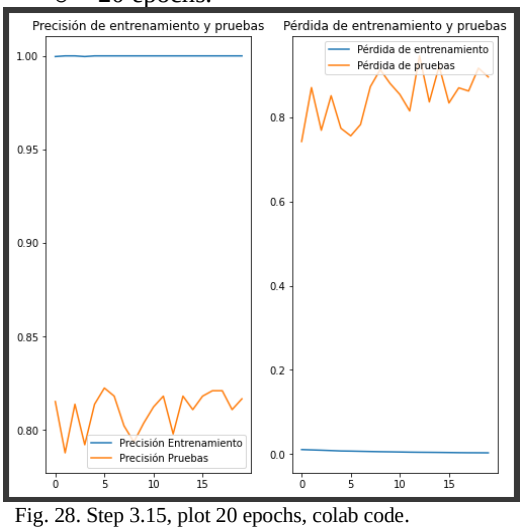


Fig. 28. Step 3.15, plot 20 epochs, colab code.

- 40 epochs.

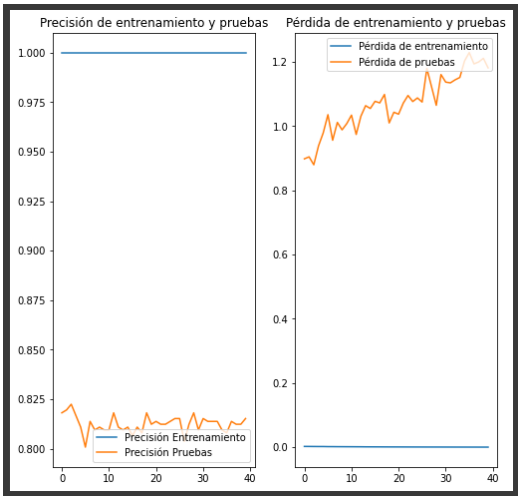


Fig. 29. Step 3.15, plot 40 epochs, colab code.

- 60 epochs.

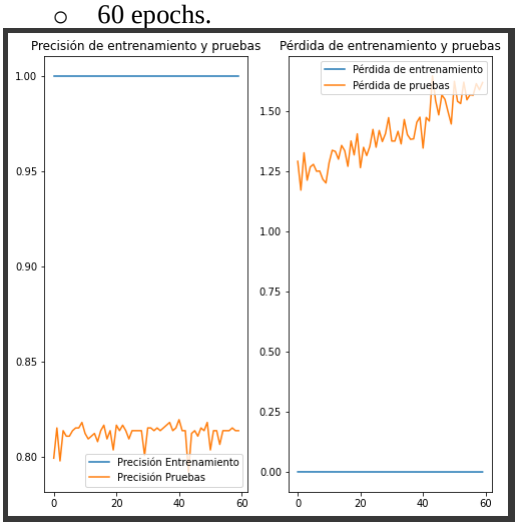


Fig. 30. Step 3.15, plot 60 epochs, colab code.

2.16. Se guarda el modelo con formato ".h5". [15] [24]

```
from keras.models import load_model
modelo.save('model_Rosas.h5')
```

Fig. 31. Step 3.16, colab code.

2.17. In order to quickly test the model, you can load an image from a url.

```
#Categorizar una imagen de internet
from PIL import Image
import requests
from io import BytesIO
import cv2

def categorizar(url):
  respuesta = requests.get(url)
  img = Image.open(BytesIO(respuesta.content))
  img = np.array(img).astype(float)/255

  img = cv2.resize(img, (224,224))
  prediccion = modelo.predict(img.reshape(-1, 224, 224, 3))
  return np.argmax(prediccion[0], axis=-1)
```

```
#0 = aceptable, 1 = marchita_dia_1-4, 2 = marchita_hace_mucho
url = ''

prediccion = categorizar (url)
print(prediccion)

if prediccion == 0:
  print("aceptable")
elif prediccion == 1:
  print("marchita entre 1 a 4 dias")
else:
  print("marchita hace mucho")
```

Fig. 32. Step 3.17, colab code.

VIII. SIMULATION OF PREDICTION RESULTS

To test the operation of the model, 6 tests were carried out with different images, these were the results:

1.

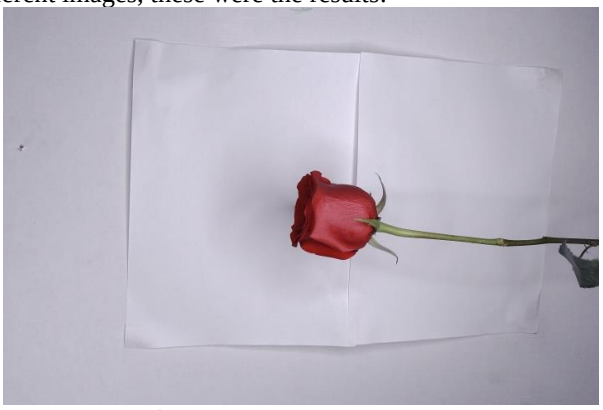


Fig. 33. Simulación desde resultados, ImagenPrueba_1.jpg.

```
#Simulacion 1
#0 = aceptable, 1 = marchita_dia_1-4, 2 = marchita_hace_mucho

url = 'https://i.ibb.co/b1tZjtN/Imagen-Prueba-1.jpg'
prediccion = categorizar (url)
print(prediccion)

if prediccion == 0:
  print("aceptable")
elif prediccion == 1:
  print("marchita entre 1 a 4 dias")
else:
  print("marchita hace mucho")
```

```
0
aceptable
```

Fig. 34. Simulation from results, simulation 1. [19]

2.

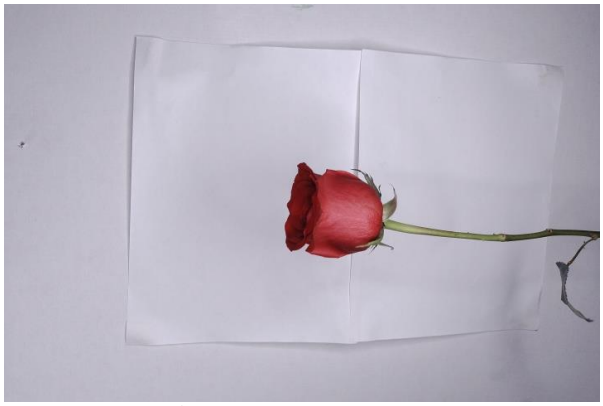


Fig. 35. Simulation from results, ImagenPrueba_2.jpg.

```
#Simulacion 2
#0 = aceptable, 1 = marchita_dia_1-4, 2 = marchita_hace_mucho

url = 'https://i.ibb.co/YXMjBVB/Imagen-Prueba-2.jpg'
prediccion = categorizar (url)
print(prediccion)

if prediccion == 0:
  print("aceptable")
elif prediccion == 1:
  print("marchita entre 1 a 4 dias")
else:
  print("marchita hace mucho")
```

```
0
aceptable
```

Fig. 36. Simulation from results, simulation 2. [19]

3.

Fig. 37. Simulation from results, ImagenPrueba_3.jpg.

```
#Simulacion 3
#0 = aceptable, 1 = marchita_dia_1-4, 2 = marchita_hace_mucho

url = 'https://i.ibb.co/tLQCJyQ/Imagen-Prueba-3.jpg'
prediccion = categorizar (url)
print(prediccion)

if prediccion == 0:
  print("aceptable")
elif prediccion == 1:
  print("marchita entre 1 a 4 dias")
else:
  print("marchita hace mucho")
```

```
1
marchita entre 1 a 4 dias
```

Fig. 38. Simulation from results, simulation 3. [19]

4.

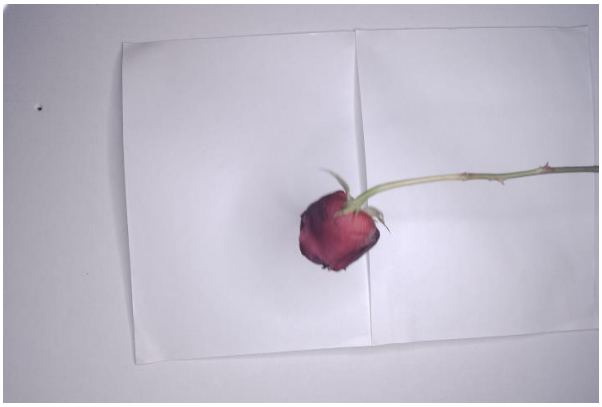


Fig. 39. Simulation from results, ImagenPrueba_4.jpg.

```
#Simulacion 4
#0 = aceptable, 1 = marchita_dia_1-4, 2 = marchita_hace_mucho

url = 'https://i.ibb.co/kXz7gYW/Imagen-Prueba-4.jpg'
prediccion = categorizar (url)
print(prediccion)

if prediccion == 0:
  print("aceptable")
elif prediccion == 1:
  print("marchita entre 1 a 4 dias")
else:
  print("marchita hace mucho")
```

```
1
marchita entre 1 a 4 dias
```

Fig. 40. Simulation from results, simulation 4. [19]

5.

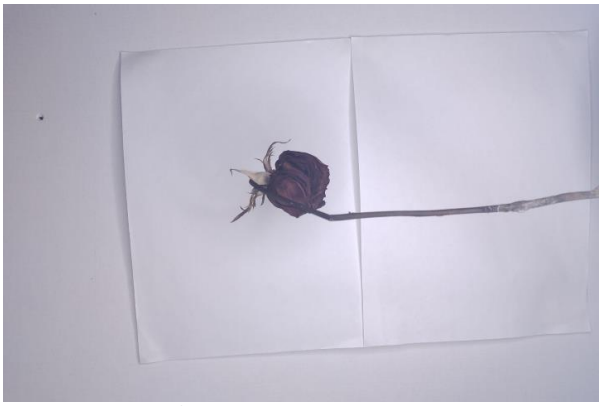


Fig. 41. Simulation from results, ImagenPrueba_5.jpg.

```
#Simulacion 5
#0 = aceptable, 1 = marchita_dia_1-4, 2 = marchita_hace_mucho

url = 'https://i.ibb.co/njyJd9J/Imagen-Prueba-5.jpg'
prediccion = categorizar (url)
print(prediccion)

if prediccion == 0:
  print("aceptable")
elif prediccion == 1:
  print("marchita entre 1 a 4 dias")
else:
  print("marchita hace mucho")
```

```
1
marchita entre 1 a 4 dias
```

Fig. 42. Simulation from results, simulation 5. [19]

6.

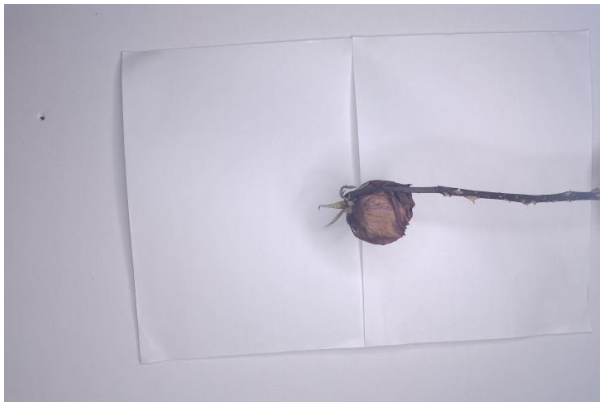


Fig. 43. Simulation from results, ImagenPrueba_6.jpg.

```
#Simulacion 6
#0 = aceptable, 1 = marchita_dia_1-4, 2 = marchita_hace_mucho

url = 'https://i.ibb.co/jJqGhtN/Imagen-Prueba-6.jpg'
prediccion = categorizar (url)
print(prediccion)

if prediccion == 0:
  print("aceptable")
elif prediccion == 1:
  print("marchita entre 1 a 4 dias")
else:
  print("marchita hace mucho")
```

```
2
marchita hace mucho
```

Fig. 44. Simulation from results, simulation 6. [19]

IX. Model test in Colab(without web service and without database)

A test was carried out to load the model and that it continued to work correctly.

```
from keras.applications.imagenet_utils import preprocess_input
from keras.models import load_model

import tensorflow as tf
import tensorflow_hub as hub
import numpy as np
import cv2
import matplotlib.pyplot as plt
from PIL import Image
import requests
from io import BytesIO

width_shape = 224
height_shape = 224

names = ['aceptable','marchita_dia_1-4','marchita_hace_mucho']

path = 'models/model_Rosas.h5'
modelo = tf.keras.models.load_model((path),custom_objects={'KerasLayer':hub.KerasLayer})
print("Modelo cargado exitosamente")

imaget_path = "ImagenPrueba_1.jpg"
imaget=cv2.resize(cv2.imread(imaget_path), (width_shape, height_shape), interpolation = cv2.INTER_AREA)

xt = np.asarray(imaget)
xt=preprocess_input(xt)
xt = np.expand_dims(xt,axis=0)

print("Predicción")
preds = modelo.predict(xt)

print("Predicción:", names[np.argmax(preds)])
plt.imshow(cv2.cvtColor(np.asarray(imaget),cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

Fig. 45.

```
Modelo cargado exitosamente
Predicción
Predicción: marchita_hace_mucho
```

Fig. 46.

X. Prediction test with Web service and database

To implement the web service, the following steps were carried out:

1. Create necessary folders and upload the files.

```
!mkdir models
!mkdir static
!mkdir templates

!mkdir static/css
!mkdir static/js
```

Fig. 47.

2. Install the "flask-ngrok" and "pygrok" libraries, which are used to create a connection to a service that makes a tunnel from the local host of "Colab" to a temporary host generated on the web.

```
!pip install flask-ngrok
!pip install pyngrok --quiet
from pyngrok import ngrok
from flask_ngrok import run_with_ngrok
```

Fig. 48.

3. ngrok authentication token is loaded.

```
!ngrok authtoken 25PC5a0BW7mU4bCLAFB3snSxKE7_4xG5EnEhmNoVHhMQ9K5WW
NGROK_AUTH_TOKEN = "25PC5a0BW7mU4bCLAFB3snSxKE7_4xG5EnEhmNoVHhMQ9K5WW"
```

Fig. 49.

4. The libraries are loaded, the size of the images is adjusted, it is decided that the application will use ngrok, the model is loaded, the prediction model is defined, it is defined that the “GET” and “GET” methods can be used from the main page. POST”. It is defined that in the “/predict” subpage “GET” and “POST” can be used, the images that are loaded on the web page are also categorized and finally the prediction is sent.

```
from werkzeug.utils import secure_filename
import flask
import flask
from flask_ngrok import run_with_ngrok
from flask import Flask, request, render_template
import os
import numpy as np
import cv2
import tensorflow as tf
import tensorflow_hub as hub
import matplotlib.pyplot as plt
import psycopg2

width_shape = 224
height_shape = 224

#names = ['aceptable','marchita_dia_1-4','marchita_hace_mucho']
names = ['0','1','2']

app = Flask(__name__)
run_with_ngrok(app)

path = 'models/model_Rosas.h5'
modelo = tf.keras.models.load_model((path),custom_objects={'KerasLayer':hub.KerasLayer})
print("Modelo cargado exitosamente")

def model_predict(img_path, model):
    img=cv2.resize(cv2.imread(img_path),
                   (width_shape, height_shape),
                   interpolation = cv2.INTER_AREA)
    x=np.asarray(img)
    x=preprocess_input(x)
    x = np.expand_dims(x,axis=0)

    preds = model.predict(x)
    return preds

@app.route('/', methods=['GET', 'POST'])
def index():
    return render_template('index.php')

@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        f = request.files['file']

        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)

        preds = model_predict(file_path, model)

        print('PREDICCIÓN', names[np.argmax(preds)])

        result = str(names[np.argmax(preds)])
        return result
    return None

app.run()
```

Fig. 50.

5. When running the application, it will show the link that ngrok created, through which you will enter the main page.

```
Modelo cargado exitosamente
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
INFO:werkzeug: * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Running on http://427a-34-75-157-152.ngrok.io
 * Traffic stats available on http://127.0.0.1:4040
```

Fig. 51.

6. When you enter the link you see the main page.

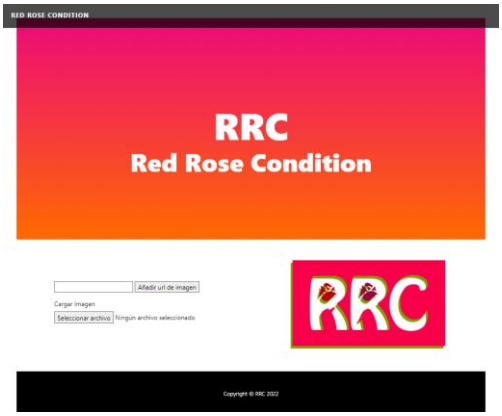


Fig. 52.

XI. PREDICTION TEST WITH DATABASE

A system with prediction without connection to the database was carried out, because the previous test did not bear fruit, due to technical problems, not linked to the prediction model. For the elaboration of these steps, we start with importing the libraries, adjusting the size of the images, assigning names to the categories, loading the model and adjusting the values of the images.

```
from werkzeug.utils import secure_filename
import os
import numpy as np
import cv2
import tensorflow as tf
import tensorflow_hub as hub
import matplotlib.pyplot as plt
import psycopg2

width_shape = 224
height_shape = 224

#names = ['aceptable','marchita_dia_1-4','marchita_hace_mucho']
names = ['0','1','2']

path = 'models/model_Rosas.h5'
modelo = tf.keras.models.load_model((path),custom_objects={'KerasLayer':hub.KerasLayer})
print("Modelo cargado exitosamente")

def model_predict(img_path, model):
    img=cv2.resize(cv2.imread(img_path),(width_shape, height_shape),interpolation = cv2.INTER_AREA)
    x=np.asarray(img)
    x=preprocess_input(x)
    x = np.expand_dims(x,axis=0)

    preds = model.predict(x)
    return preds
```

Fig. 53.

After the connection to the database is made, which was created through "ElephantSQL", a database hosting service, managed by "PgAdmin", followed by the connection, a SELECT is made from the images table, which only has a column that contains links to the images with which the model will be tested.

```
try:
  connection=psycopg2.connect(
      host='rosie.db.elephantsql.com',
      user='tnvfomkj',
      password='y22QIO3M0vRcv0ydEvak3JTIY-MhrmyL',
      database='tnvfomkj'
  )
  print("Conexión exitosa")
  cursor=connection.cursor()
  cursor.execute("SELECT * FROM imagen")
  rows=cursor.fetchone()
  for row in rows:
    print(row)
except Exception as ex:
    print(ex)
```

Fig. 54.

Next, it will be defined how the images are categorized and from where the prediction will be made.

```
from PIL import Image
import requests
from io import BytesIO
import cv2

def categorizar(url):
  respuesta = requests.get(url)
  img = Image.open(BytesIO(respuesta.content))
  img = np.array(img).astype(float)/255

  img = cv2.resize(img, (224,224))
  prediccion = modelo.predict(img.reshape(-1, 224, 224, 3))
  return np.argmax(prediccion[0], axis=-1)
```

Fig. 55.

And finally the result of the prediction is shown.

```
#0 = aceptable, 1 = marchita_dia_1-4, 2 = marchita_hace_mucho

url = row
prediccion = categorizar (url)
print(prediccion)

if prediccion == 0:
  print("aceptable")
elif prediccion == 1:
  print("marchita entre 1 a 4 dias")
else:
  print("marchita hace mucho")
```

Fig. 56.

XII. Results and Discussion

To analyze the results given by the simulation, they were reviewed one by one, with the following points corresponding to the order in which they occurred in the simulation of results:

1. An image with a rose in “aceptable” state was uploaded and the model gave a **correct answer**.
2. An image with a rose in “aceptable” state was uploaded and the model gave a **correct answer**.
3. An image was uploaded with a rose in a “marchita entre 1 a 4 días” state and the model gave a **correct answer**.
4. An image was uploaded with a rose in a “withered between 1 to 4 days” state and the model gave a **correct answer**.
5. An image was uploaded with a rose in a “long withered” state and the model gave an **incorrect answer**.
6. An image was uploaded with a rose in a “long withered” state and the model gave a **correct answer**.

It was tested with two images of each category and it was observed that of the 6 simulations only one printed the incorrect prediction.

Test-by-test analysis:

- No web service and no database: it can be seen quite simple, but without drawbacks, easy to do and the results were correct.
- With web service and database: in this case it could not be developed correctly, as much as possible was tried, but the desired objective was not achieved, since everything was in the cloud, new knowledge was obtained, it was possible to connect Google Colab with a temporary web page hosting service (ngrok), designed for developers, but there were problems when trying to load information from the web page to Google Colab and to the ElephantSQL database, these errors can be seen in the console, the errors are not solved.

```
INFO:werkzeug:127.0.0.1 - - [24/Feb/2022 06:52:15] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [24/Feb/2022 06:52:15] "GET /static/css/main.css HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [24/Feb/2022 06:52:16] "GET /js/scripts.js HTTP/1.1" 404 -
INFO:werkzeug:127.0.0.1 - - [24/Feb/2022 06:52:16] "GET /static/js/main.js HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [24/Feb/2022 06:52:16] "GET /js/scripts.js HTTP/1.1" 404 -
INFO:werkzeug:127.0.0.1 - - [24/Feb/2022 06:52:16] "GET /favicon.ico HTTP/1.1" 404 -
```

Fig. 57.

- Without a web service and with a database, it does not have the best graphic sections, the usability is not the best, but it achieves the objective, it predicts mostly with correct results with the three different categories.

XIII. Conclusions

This document showed how the "RRC" project was carried out for the subject of "Data Science", what was needed, what was used, how it works, what its purpose is and how it fulfilled this. This project categorizes whether a red rose is in “fair” condition, “withered for 1 to 4 days” or “withered for a long time”, being “much” more than 4 days, with an average accuracy percentage of **81.23%.**

The results show a good precision, according to the rubric with which this project is evaluated, with this managing to satisfy the initial problem. maybe in "Results and discussion" there has been an incorrect answer but it does not detract from the value of the project and is in accordance with the precision achieved.

XIV. Future Works

The project can be complemented in various ways such as: increasing the dataset, creating new categories and developing a web service or an API that works correctly, so that the project has better usability in the use of the model.

This project will be used to complement the project of the same name that will be delivered in the subject " Visión Artificial" and perhaps in some other subject not yet foreseen.

XV. References

[1] [1] D. R. Luna, «API_DeepLearning,» 15 Enero 2021. [En línea]. Available: https://github.com/DavidReveloLuna/API_DeepLearning. [Último acceso: 21 Febrero 2022].

[2] [2] R. T. «Clasificación de imágenes (perros y gatos),» 13 Septiembre 2021. [En línea]. Available: https://github.com/ringa-tech/clasificador-perros-gatos. [Último acceso: 21 Febrero 2022].

[3] [3] O. A. M. Amaya, «Diseño De Un Sistema De Visión Artificial Para El Análisis De Calidad Y Producción De Rosas,» 2019. [En línea]. Available: http://repository.pedagogica.edu.co/bitstream/handle/20.500.12209/12074/TE-24132.pdf?sequence=1. [Último acceso: 21 Febrero 2022].

[4] [4] P. A. C. Aguilar, «Diseño de un sistema de identificación y clasificación de flores,» 2017. [En línea]. Available: https://ciencia.lasalle.edu.co/cgi/viewcontent.cgi?article=1000&context=ing_automatizacion. [Último acceso: 21 Febrero 2022].

[5] [5] B. Y. Alexis, C. G. Jonathan, A. B. Ángela, Y. V. Marco y L. R. Samuel, «Sistema de clasificación de rosas de la variedad explorer usando visión por computadora,» 2019. [En línea]. Available: https://investigacion.utmachala.edu.ec/proceedings/index.php/utmach/article/download/442/382/799. [Último acceso: 21 Febrero 2022].

[6] [6] R. T. «Clasificación de imágenes (perros y gatos),» 13 Septiembre 2021. [En línea]. Available: https://colab.research.google.com/drive/1DQc8a-WOTctenvoy5T0lWUXn47EuteCy?usp=sharing. [Último acceso: 21 Febrero 2022].

[7] [7] M. Sandler, «tf2-preview/mobilenet_v2/feature_vector,» TensorFlowHub, 13 Enenro 2019. [En línea]. Available: https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4. [Último acceso: 21 Febrero 2022].

[8] [8] «Keras,» Keras, [En línea]. Available: https://keras.io/api/models/model_saving_apis/#save_model-function. [Último acceso: 21 Febrero 2022].

[9] [9] «Red Leaves On Roses: What To Do For Red Leaves On A Rose Bush,» Gardening know how, 26 Julio 2021. [En línea]. Available: https://www.gardeningknowhow.com/ornamental/flowers/roses/red-leaves-on-roses.htm. [Último acceso: 21 Febrero 2022].

[10] [10] C. Boeckmann, «Roses,» Almanac, 2019. [En línea]. Available: https://www.almanac.com/plant/roses. [Último acceso: 21 Febrero 2022].

[11] [11] Jackson & Perkins, «Rose Anatomy 101,» Jackson & Perkins, 11 Septiembre 2021. [En línea]. Available: https://www.jacksonandperkins.com/blog/rose-blogs/rose-anatomy/b/rose-anatomy/. [Último acceso: 21 Febrero 2022].

[12] [12] A. Howard, «MobileNetV2: Inverted Residuals and Linear Bottleneck,» Arvix, 13 Enero 2018. [En línea]. Available: https://arxiv.org/abs/1801.04381. [Último acceso: 21 Febrero 2022].

[13] [13] M. Sanjeevi, «Chapter 8 .0: Convolutional neural networks for deep learning,» Medium, 15 Octubre 2018. [En línea]. Available: https://medium.com/deep-math-machine-learning-ai/chapter-8-0-convolutional-neural-networks-for-deep-learning-364971e34ab2. [Último acceso: 21 Febrero 2022].

[14] [14] A. Kumar, «Keras – Categorical Cross Entropy Loss Function,» Vitalflux, 28 Octubre 2020. [En línea]. Available: https://vitalflux.com/keras-categorical-cross-entropy-loss-function/. [Último acceso: 21 Febrero 2022].

[15] [15] A. Khaled, «Unknown layer: KerasLayer when i try to load_model,» Assem Khaled, 16 Mayo 2020. [En línea]. Available: https://stackoverflow.com/questions/61814614/unknown-layer-keraslayer-when-i-try-to-load-model. [Último acceso: 21 Febrero 2022].

[16] [16] itdxer, «What is batch size in neural network?,» itdxer, 5 Abril 2019. [En línea]. Available: https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network. [Último acceso: 21 Febrero 2022].

[17] [17] M. Koziarkiewicz, «Rose Rosette Disease diagnosis through Deep Learning,» softwaremill, 3 Junio 2020. [En línea]. Available: https://blog.softwaremill.com/rose-rosette-disease-diagnosis-through-deep-learning-155ec3b45a6b. [Último acceso: 21 Febrero 2022].

[18] [18] Ó. M. d. l. F. Sanz, «Google Colab: Python y Machine Learning en la nube,» adictosaltrabajo, 4 Junio 2019. [En línea]. Available: https://www.adictosaltrabajo.com/2019/06/04/google-colab-python-y-machine-learning-en-la-nube/. [Último acceso: 21 Febrero 2022].

[19] [19] imgbb, «imgbb,» imgbb, 22 Enero 2022. [En línea]. Available: https://es.imgbb.com/. [Último acceso: 21 Febrero 2022].

[20] [20] J. Barrios, «Redes Neuronales Convolucionales,» Juan Barrios, 2022. [En línea]. Available: https://www.juanbarrios.com/redes-neurales-convolucionales/. [Último acceso: 21 Febrero 2022].

[21] [21] M. Padriñan, «Matemáticas de las redes neuronales convolucionales,» ichi.pro, 2021. [En línea]. Available: https://ichi.pro/es/matematicas-de-las-redes-neuronales-convolucionales-30697178023872. [Último acceso: 21 Febrero 2022].

[22] [22] J. Cuevas, «Diseño de una red neuronal convolucional para la clasificación de señales,» 2019. [En línea]. Available: https://www.ecorfan.org/proceedings/Proceedings_Ciencias_de_la_Tierra_Fisica_y_Matematicas_TI/Proceedings_Ciencias_de_la_Tierra_Fisica_y_Matematicas_TI_4.pdf. [Último acceso: 21 Febrero 2022].

[23] [23] L. D. Solutions, «Deep Learning, redes neuronales y visión artificial,» LIS Data Solutions, 2021. [En línea]. Available: https://www.lisdatasolutions.com/blog/deep-learning-redes-neuronales-y-vision-artificial/. [Último acceso: 21 Febrero 2022].

[24] [24] TensorFlow, «Guardando y Serializando Modelos con TensorFlow Keras,» TensorFlow Core, 05 11 2021. [En línea]. Available: https://www.tensorflow.org/guide/keras/save_and_serialize#registering_the_custom_object. [Último acceso: 21 Febrero 2022].

[25] [25] D. Magic, «github,» Data Magic, 24 Febrero 2021. [En línea]. Available: https://github.com/datamagic2020/deploy-ML-model-with-webapp. [Último acceso: 21 Febrero 2022].

[26] [26] M. Rivas, «GitHub,» Marcos Rivas, 10 Septiembre 2017. [En línea]. Available: https://github.com/marcosrivasr/Sitio-Web-CSS3. [Último acceso: 21 Febrero 2022].

[27] [27] O. Garcia, «GitHub,» Oscar Garcia, 21 Enero 2022. [En línea]. Available: https://github.com/UskoKruM. [Último acceso: 21 Febrero 2022].

[28] [28] S. Muria, «Youtube,» Shak Muria, 7 Julio 2021. [En línea]. Available: https://www.youtube.com/watch?v=62n_MhpHfoU. [Último acceso: 21 Febrero 2022].

[29] [29] Vida MRR, «YouTube,» Vida MRR, 12 Abril 2018. [En línea]. Available: https://www.youtube.com/watch?v=GNfTYvCVjSc. [Último acceso: 21 Febrero 2022].

[30] [30] Hot Examples, «Hot Examples,» Hot Examples, [En línea]. Available: https://python.hotexamples.com/es/examples/OpenLegLib.utils/-/fetch/python-fetch-function-examples.html. [Último acceso: 21 Febrero 2022].

[31] [31] w3resource, «w3resource,» w3resource, 19 Octubre 2021. [En línea]. Available: https://www.w3resource.com/PostgreSQL/PostgreSQL-insert.php. [Último acceso: 21 Febrero 2022].

[32] [32] My PHP.net, «php,» My PHP.net, [En línea]. Available: https://www.php.net/manual/es/function.pg-insert.php. [Último acceso: 21 Febrero 2022].

[33] [33] informaticapc.com, «informaticapc.com,» informaticapc.com, [En línea]. Available: https://informaticapc.com/tutorial-php/bases-de-datos-postgresql.php. [Último acceso: 21 Febrero 2022].

[34] [34] Vishal, «PYnative,» PYnative, 9 Marzo 2021. [En línea]. Available: https://pynative.com/python-cursor-fetchall-fetchmany-fetchone-to-read-rows-from-table/. [Último acceso: 21 Febrero 2022].

[35] startbootstrap, «startbootstrap,» startbootstrap, 21 Septiembre 2013. [En línea]. Available: https://startbootstrap.com/theme/one-page-wonder. [Último acceso: 21 Febrero 2022].