Alumnos:

Yépez Callo, Juan Carlos

Pérez Pacheco, Elizabeth Victoria

12MBID_Proyecto_Programacion_Entrega1

May 22, 2022

0.1 Proyecto de programación ”Deep Vision in classification tasks”

[ ]:
```
%%capture
!pip freeze
```

Hola voy a ejecutar el comando `!pip freeze` para cotillear a Google Colab

[ ]:
```
#Importemos TensorFlow 2.X y Numpy
import numpy as np
import tensorflow as tf
tf.__version__
```

[ ]:
```
'2.8.0'
```

- Cargando el conjunto de datos

[ ]:
```
# Nos aseguramos que tenemos instalada la última versión de la API de Kaggle en␣
 ↪Colab
!pip install --upgrade --force-reinstall --no-deps kaggle
```

```
Collecting kaggle
  Downloading kaggle-1.5.12.tar.gz (58 kB)
     |                  | 58 kB 5.6 MB/s
Building wheels for collected packages: kaggle
  Building wheel for kaggle (setup.py) … done
  Created wheel for kaggle: filename=kaggle-1.5.12-py3-none-any.whl size=73051
sha256=748fcc8ac52e0a022e8ef1470c4f55ef0ec672d7f8f243be3ac9109397e79cbb
  Stored in directory: /root/.cache/pip/wheels/62/d6/58/5853130f941e75b2177d281e
b7e44b4a98ed46dd155f556dc5
Successfully built kaggle
Installing collected packages: kaggle
  Attempting uninstall: kaggle
    Found existing installation: kaggle 1.5.12
    Uninstalling kaggle-1.5.12:
      Successfully uninstalled kaggle-1.5.12
Successfully installed kaggle-1.5.12
```

[ ]:

```
# Seleccionar el API Token personal previamente descargado (fichero kaggle.json)
from google.colab import files
files.upload()
```

```
<IPython.core.display.HTML object>


Saving kaggle.json to kaggle.json
```

[ ]:

```
{'kaggle.json':
 b'{"username":"juankyep","key":"4b1baf860edb1438375ef56e737ec3bd"}'}
```

[ ]:

```
!ls
```

```
kaggle.json  sample_data
```

[ ]:

```
# Creamos un directorio en el que copiamos el fichero kaggle.json
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

[ ]:

```
# Ya podemos listar los datasets disponibles en kaggle para su descarga
!kaggle datasets list
```

```
ref                                                                   title
size  lastUpdated          downloadCount  voteCount  usabilityRating
--------------------------------------------------------------------
--------------------------------------------------  -----  -------------------
-------------  ----------  ---------------
muratkokludataset/date-fruit-datasets                                 Date Fruit
Datasets                                 408KB  2022-04-03 09:25:39
8615       1171  0.9375
victorsoeiro/netflix-tv-shows-and-movies                              Netflix TV
Shows and Movies                           2MB  2022-05-15 00:01:23
1041         42  0.9411765
mdmahmudulhasansuzan/students-adaptability-level-in-online-education  Students
Adaptability Level in Online Education       6KB  2022-04-16 04:46:28
5762        145  1.0
muratkokludataset/rice-image-dataset                                  Rice Image
Dataset                                  219MB  2022-04-03 02:12:00
1685        972  0.875
paradisejoy/top-hits-spotify-from-20002019                            Top Hits
Spotify from 2000-2019                      94KB  2022-04-26 17:30:03
1770         44  1.0
victorsoeiro/disney-tv-shows-and-movies                               Disney+ TV
Shows and Movies                         718KB  2022-05-13 23:24:57
548         28  1.0
muratkokludataset/raisin-dataset                                      Raisin
```

```
Dataset                                      112KB  2022-04-03 00:23:16
697        863  0.9375
muratkokludataset/pistachio-dataset                                   Pistachio
Dataset                                     2MB  2022-04-03 08:38:21
635        882  0.9375
muratkokludataset/rice-msc-dataset                                    Rice MSC
Dataset                                    102MB  2022-04-03 01:33:52
270        850  0.9375
muratkokludataset/pistachio-image-dataset                             Pistachio
Image Dataset                              27MB  2022-03-28 18:01:27
585        928  0.9375
muratkokludataset/grapevine-leaves-image-dataset                      Grapevine
Leaves Image Dataset                      109MB  2022-04-03 09:00:54
206        889  0.875
muratkokludataset/durum-wheat-dataset                                 Durum
Wheat Dataset                                 983MB  2022-04-03 00:02:29
116        861  0.875
muratkokludataset/pumpkin-seeds-dataset                               Pumpkin
Seeds Dataset                               393KB  2022-03-28 18:28:16
685        857  0.9375
muratkokludataset/dry-bean-dataset                                    Dry Bean
Dataset                                      5MB  2022-04-02 23:19:30
584        860  0.9375
psycon/daily-coffee-price                                             Daily
Coffee Price                                   71KB  2022-05-20 17:46:09
271         11  1.0
rinichristy/covid19-coronavirus-pandemic                              COVID-19
Coronavirus Pandemic                         9KB  2022-04-05 08:43:16
4513        103  1.0
muhmores/spotify-top-100-songs-of-20152019                            Spotify
Top 100 Songs of 2010-2019                  139KB  2022-04-09 06:35:36
5764        118  0.88235295
surajjha101/stores-area-and-sales-data
Supermarket store branches sales analysis            10KB  2022-04-29 11:10:16
1618         67  1.0
aslanahmedov/walmart-sales-forecast                                   Walmart
Sales Forecast                                3MB  2022-04-21 05:28:20
2809         70  1.0
satoshidatamoto/colleges-and-universities-a-comprehensive-datasee     Colleges
and Universities: A Comprehensive Dataset  923KB  2022-05-14 11:15:33
255         41  1.0
```

[ ]:

```
!kaggle competitions download -c the-nature-conservancy-fisheries-monitoring
```

```
Downloading the-nature-conservancy-fisheries-monitoring.zip to /content
100% 2.11G/2.11G [00:57<00:00, 58.7MB/s]
100% 2.11G/2.11G [00:57<00:00, 39.1MB/s]
```

```
[ ]: # Creemos un directorio para descomprimir los datos
     !mkdir my_dataset
```

```
[ ]: # Descomprimimos los datos y los dejamos listos para trabajar
     !unzip the-nature-conservancy-fisheries-monitoring.zip -d my_dataset
```

```
Archive:  the-nature-conservancy-fisheries-monitoring.zip
  inflating: my_dataset/sample_submission_stg1.csv.zip
  inflating: my_dataset/sample_submission_stg2.csv.zip
  inflating: my_dataset/test_stg1.zip
  inflating: my_dataset/test_stg2.7z
  inflating: my_dataset/train.zip
```

```
[ ]: # %%capture
     !unzip my_dataset/train.zip
```

```
Archive:  my_dataset/train.zip
   creating: train/
  inflating: train/.DS_Store
   creating: __MACOSX/
   creating: __MACOSX/train/
  inflating: __MACOSX/train/._.DS_Store
   creating: train/ALB/
  inflating: train/ALB/img_00003.jpg
  inflating: train/ALB/img_00010.jpg
  inflating: train/ALB/img_00012.jpg
  inflating: train/ALB/img_00015.jpg
  inflating: train/ALB/img_00019.jpg
  inflating: train/ALB/img_00020.jpg
  inflating: train/ALB/img_00029.jpg
  inflating: train/ALB/img_00032.jpg
  inflating: train/ALB/img_00037.jpg
  inflating: train/ALB/img_00038.jpg
  inflating: train/ALB/img_00039.jpg
  inflating: train/ALB/img_00041.jpg
  inflating: train/ALB/img_00043.jpg
  inflating: train/ALB/img_00045.jpg
  inflating: train/ALB/img_00055.jpg
  inflating: train/ALB/img_00057.jpg
  inflating: train/ALB/img_00074.jpg
  inflating: train/ALB/img_00085.jpg
  inflating: train/ALB/img_00090.jpg
  inflating: train/ALB/img_00097.jpg
  inflating: train/ALB/img_00110.jpg
  inflating: train/ALB/img_00121.jpg
  inflating: train/ALB/img_00130.jpg
  inflating: train/ALB/img_00134.jpg
  inflating: train/ALB/img_00136.jpg
```

```
 inflating: train/YFT/img_07750.jpg
 inflating: train/YFT/img_07752.jpg
 inflating: train/YFT/img_07759.jpg
 inflating: train/YFT/img_07761.jpg
 inflating: train/YFT/img_07765.jpg
 inflating: train/YFT/img_07775.jpg
 inflating: train/YFT/img_07782.jpg
 inflating: train/YFT/img_07828.jpg
 inflating: train/YFT/img_07849.jpg
 inflating: train/YFT/img_07852.jpg
 inflating: train/YFT/img_07853.jpg
 inflating: train/YFT/img_07854.jpg
 inflating: train/YFT/img_07891.jpg
 inflating: train/YFT/img_07901.jpg
 inflating: train/YFT/img_07911.jpg
```

```
[ ]: # %%capture
     !ls train/
```

```
ALB  BET  DOL  LAG  NoF  OTHER  SHARK  YFT
```

```
[ ]: # Visualizacion de subcarpetas de Train que vienen a ser las etiquetas
     !ls train/ | wc -l
```

```
8
```

1 Escritura de datos tomando como referencia un BASE_FOLDER

```
[5]: # Conectamos con nuestro Google Drive
     from google.colab import drive
     drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

```
[ ]: # Creamos las carpetas de Train y Test dentro del directorio de nuestro␣
      ↪proyecto creado previamente en Google Drive
     !mkdir /content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/
      ↪Train
     !mkdir /content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Test
```

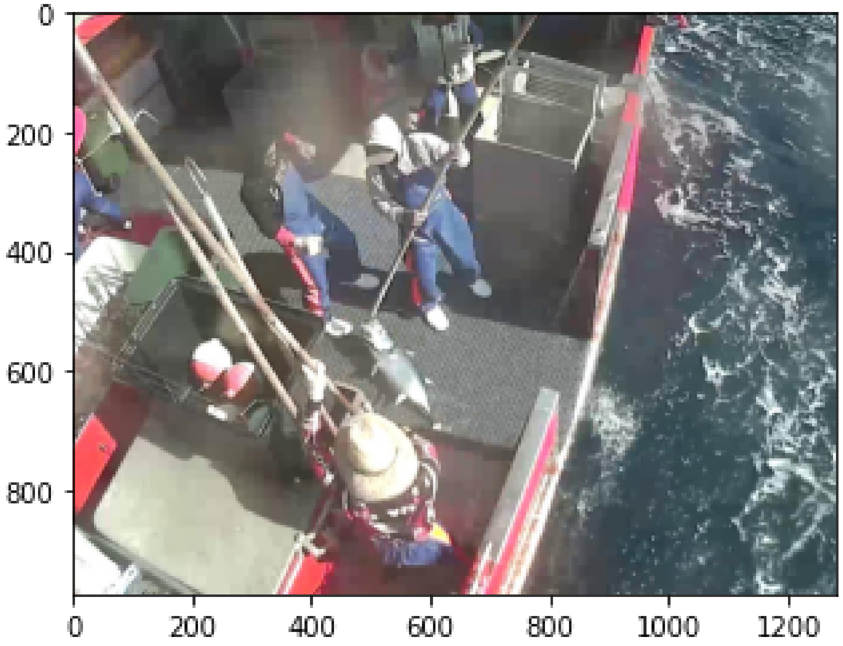
```
[6]: # Establezco una ruta absoluta a un directorio existente de mi Google Drive
     BASE_FOLDER = "/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/
      ↪my_dataset/"
```

```
[ ]: !ls train/ALB | wc -l
```

```
1719
```

```
[ ]: import matplotlib.pyplot as plt
     import numpy as np
     import cv2

     # Escogiendo y mostrando una imagen al azar del conjunto de test

     indx = 12
     img = cv2.imread('train/ALB/img_00012.jpg', cv2.COLOR_BGR2RGB)
     img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
     plt.imshow(img)
```

```
[ ]: <matplotlib.image.AxesImage at 0x7f27c531db90>
```



```
[ ]: import os
     categorias = []
     categorias = os.listdir('train/')
     categorias.remove('.DS_Store')
     print(categorias)
```

```
['DOL', 'NoF', 'BET', 'SHARK', 'LAG', 'YFT', 'OTHER', 'ALB']
```

```
[ ]: # Creacion de subdirectorios por cada categoria con sus respectivas imagenes
     import shutil
     from PIL import Image

     imagenes = []
     labels = []

     # Generando subcarpetas con imagenes  por cada categoria
     idx = 0
     for cat in categorias:
         parent_dir = BASE_FOLDER + "Train/"
         path = os.path.join(parent_dir, cat)
         print(path)
         os.mkdir(path)
         # creamos las imagenes para el entranamiento redimensionandolos
         for imagen in os.listdir('train/' + cat):
           img = cv2.imread(os.path.join('train/' + cat, imagen))
           img = cv2.resize(img, (200,200))
           cv2.imwrite(path + '/' + imagen, img)
           img = np.asarray(img)
           imagenes.append(img)
           labels.append(idx)
         idx += 1
```

```
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/DOL
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/NoF
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/BET
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/SHARK
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/LAG
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/YFT
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/OTHER
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/ALB
```

```
[ ]: # Generando datos con etiquetas a partir de imagenes  por cada categoria para␣
      ↪el test
     x_test =[]
     y_test =[]

     idx = 0
     for cat in categorias:
         parent_dir = BASE_FOLDER + "Test/"
         path = os.path.join(parent_dir, cat)
         print(path)
         os.mkdir(path)
         shuf = np.random.permutation(os.listdir('train/' + cat))
         # creamos imagenes de test en sus subdirectorios por categorias
         for i in range(int(len(shuf) / 10)):
```

```
      img = cv2.imread(os.path.join('train/' + cat, shuf[i]))
      img = cv2.resize(img, (200,200))
      img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
      cv2.imwrite(path + '/' + shuf[i], img)
      os.remove(os.path.join(BASE_FOLDER + 'Train/' + cat , shuf[i]))
      img = np.asarray(img)
      x_test.append(img)
      y_test.append(idx)
    idx += 1
```

```
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Test/DOL
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Test/NoF
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Test/BET
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Test/SHARK
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Test/LAG
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Test/YFT
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Test/OTHER
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Test/ALB
```

```
[ ]: # # Generando datos de entrenamiento extrayendo del directory Train/
x_train = []
y_train = []

# # Generando dataset de entrenamiento con etiqutas (x_test, y_test)
idx = 0
for cat in categorias:
    parent_dir = BASE_FOLDER + "Train/"
    path = os.path.join(parent_dir, cat)
    print(path)
    for imagen in os.listdir(path):
      img = cv2.imread(os.path.join(path, imagen))
      img = cv2.resize(img, (200,200))
      img = np.asarray(img)
      x_train.append(img)
      y_train.append(idx)
    idx += 1
```

```
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/DOL
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/NoF
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/BET
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/SHARK
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/LAG
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/YFT
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/OTHER
/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/my_dataset/Train/ALB
```

[ ]:

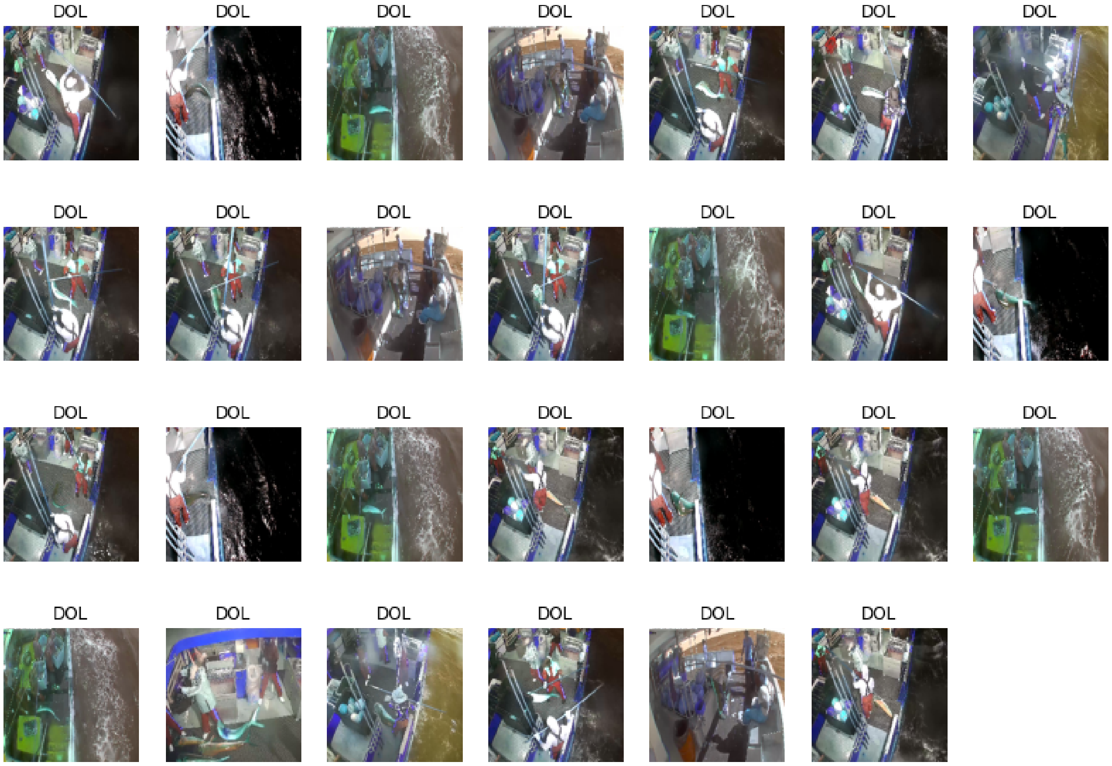
```
x_train = np.asarray(x_train)
y_train = np.asarray(y_train)
x_test = np.asarray(x_test)
y_test = np.asarray(y_test)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(3404, 200, 200, 3)
(3404,)
(373, 200, 200, 3)
(373,)
```

2 Verificando las imagenes

[ ]:

```
fig = plt.figure(figsize=(14,10))
for i in range(1, 28):
  fig.add_subplot(4,7,i)
  plt.xticks([])
  plt.yticks([])
  plt.grid(False)
  plt.imshow(imagenes[i])
  plt.title(categorias[labels[i]])
  plt.axis('off')
```



```
[ ]: # Pre-procesado obligatorio cuando trabajo con redes neuronales
     from tensorflow.keras.utils import to_categorical
     from sklearn.model_selection import train_test_split
     from tensorflow.keras.backend import expand_dims

     x_train, x_te = x_train / 255.0, x_test / 255.0 #Cambio al rango 0-1 ->␣
      ↪Disminuyo CC
     #¿Que pasa si empleo labels con etiquetas número entero?
     print(y_train[0])
     #y_train = to_categorical(y_train, num_classes=10) #One-hot encoding para␣
      ↪minimizar error
     #y_te = to_categorical(y_test, num_classes=10)
     x_tr, x_val, y_tr, y_val = train_test_split(x_train, y_train, test_size=0.1,␣
      ↪random_state=42) # 3 subconjuntos es de vital importancia
     #Expandir dimensiones porque en CNN tengo que especificar el número de canales
     print(x_tr.shape)
     print(x_val.shape)
     print(x_te.shape)
```

```
0
(3063, 200, 200, 3)
(341, 200, 200, 3)
(373, 200, 200, 3)
```

```
[ ]: print(y_val.shape)
```

```
(341,)
```

3 CREANDO TOPOLOGIA DE RED NEURONAL (CNN) Y ENTRENANDOLA

```
[ ]: # Construccion de una red CNN
     from tensorflow.keras.models import Sequential
     from tensorflow.keras import layers
     # Red feedforward API secuencial
     convnet = Sequential()

     # BASE MODEL
     convnet.add(layers.Conv2D(32,(3,3),input_shape=(200,200,3),activation='relu'))
     convnet.add(layers.MaxPooling2D((2,2)))

     convnet.add(layers.Conv2D(64,(3,3),activation='relu'))
     convnet.add(layers.MaxPooling2D((2,2)))

     convnet.add(layers.Conv2D(64,(3,3),activation='relu'))

     #TOP MODEL
     convnet.add(layers.Flatten())
     convnet.add(layers.Dense(64,activation='relu'))
     convnet.add(layers.Dense(8,activation='softmax'))
```

```
[ ]: convnet.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 198, 198, 32)      896

 max_pooling2d (MaxPooling2D  (None, 99, 99, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 97, 97, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 48, 48, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 46, 46, 64)        36928

 flatten (Flatten)           (None, 135424)            0
```

```
 dense (Dense)                (None, 64)               8667200

 dense_1 (Dense)              (None, 8)                520

=================================================================
Total params: 8,724,040
Trainable params: 8,724,040
Non-trainable params: 0
_________________________________________________________________
```

```
[ ]: convnet.compile(optimizer='adam',
                   loss='sparse_categorical_crossentropy',
                   #loss='categorical_crossentropy', #If labels are one-hot encoded
                   metrics=['accuracy'])
```

```
[ ]: H = convnet.fit(x_tr, y_tr, epochs=20, batch_size=64, validation_data=(x_val,␣
      ↪y_val))
```

```
Epoch 1/5
48/48 [==============================] - 19s 132ms/step - loss: 1.6318 -
accuracy: 0.5021 - val_loss: 1.1280 - val_accuracy: 0.5689
Epoch 2/5
48/48 [==============================] - 4s 82ms/step - loss: 0.7243 - accuracy:
0.7578 - val_loss: 0.6046 - val_accuracy: 0.8358
Epoch 3/5
48/48 [==============================] - 4s 82ms/step - loss: 0.3082 - accuracy:
0.9053 - val_loss: 0.3920 - val_accuracy: 0.9003
Epoch 4/5
48/48 [==============================] - 4s 82ms/step - loss: 0.1196 - accuracy:
0.9660 - val_loss: 0.3851 - val_accuracy: 0.9267
Epoch 5/5
48/48 [==============================] - 4s 82ms/step - loss: 0.0429 - accuracy:
0.9876 - val_loss: 0.4177 - val_accuracy: 0.9208
```

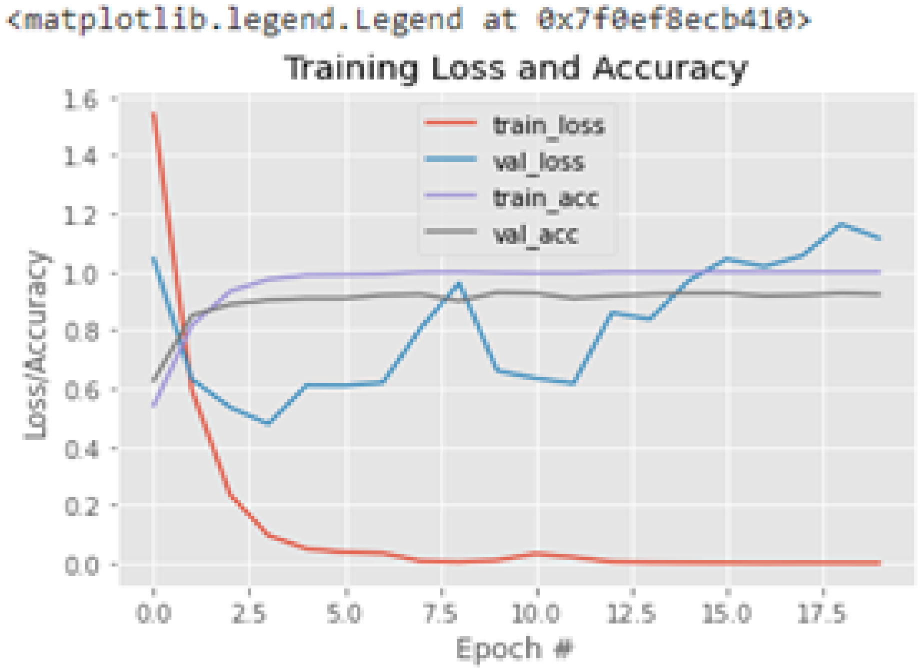
```
[ ]:
```

4 Observando el proceso de entrenamiento para tomar decisiones

```
[ ]: import matplotlib.pyplot as plt
     import numpy as np
     # Muestro gráfica de accuracy y losses
     plt.style.use("ggplot")
     plt.figure()
     plt.plot(np.arange(0, 20), H.history["loss"], label="train_loss")
```

```
plt.plot(np.arange(0, 20), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, 20), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, 20), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()
```

```
[ ]: <matplotlib.legend.Legend at 0x7f27b00e0e90>
```



```
<matplotlib.legend.Legend at 0x7f0ef8ecb410>
```

- Probando el conjunto de datos en el subset de test y evaluando el performance del modelo

```
[ ]: from sklearn.metrics import classification_report
     # Evaluando el modelo de predicción con las imágenes de test
     print("[INFO]: Evaluando red neuronal...")
     predictions = convnet.predict(x_te, batch_size=128)
     #print(y_te[0])
     #print(predictions[0])
     print(classification_report(y_test, predictions.argmax(axis=1)))
```

```
[INFO]: Evaluando red neuronal…
              precision    recall  f1-score   support

           0       1.00      0.82      0.90        11
```

```
           1       0.87      0.87      0.87        46
           2       1.00      0.95      0.97        20
           3       1.00      0.88      0.94        17
           4       1.00      1.00      1.00         6
           5       0.91      0.97      0.94        73
           6       1.00      0.90      0.95        29
           7       0.95      0.96      0.96       171

    accuracy                           0.94       373
   macro avg       0.97      0.92      0.94       373
weighted avg       0.94      0.94      0.94       373
```

[ ]:

```python
print(convnet.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 198, 198, 32)      896

 max_pooling2d (MaxPooling2D  (None, 99, 99, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 97, 97, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 48, 48, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 46, 46, 64)        36928

 flatten (Flatten)           (None, 135424)            0

 dense (Dense)               (None, 64)                8667200

 dense_1 (Dense)             (None, 8)                 520

=================================================================
Total params: 8,724,040
Trainable params: 8,724,040
Non-trainable params: 0
_________________________________________________________________
None
```

[ ]:

```python
path = os.path.join(BASE_FOLDER, 'MODELO')
os.mkdir(path)
convnet.save(path + '/modelo.h5')
```

```
convnet.save_weights(path + '/pesos.h5')
```

5 PREDICCION

```
[ ]: # Conectamos con nuestro Google Drive
     from google.colab import drive
     drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

```
[ ]: # Establezco una ruta absoluta a un directorio existente de mi Google Drive
     BASE_FOLDER = "/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/
      ↪my_dataset/"
     import os
     categorias = []
     categorias = os.listdir(BASE_FOLDER + 'Train')
     print(categorias)
```

```
['DOL', 'NoF', 'BET', 'SHARK', 'LAG', 'YFT', 'OTHER', 'ALB']
```

```
[ ]: from matplotlib.font_manager import list_fonts
     import numpy as np
     import cv2
     from keras.models import load_model

     modelo = BASE_FOLDER + 'MODELO/modelo.h5'
     pesos = BASE_FOLDER + 'MODELO/pesos.h5'
     cnn = load_model(modelo)
     cnn.load_weights(pesos)
     list_img =[]
     test_stg = []
     shuf = np.random.permutation(os.listdir(BASE_FOLDER + 'test_stg1/'))
     for i in range(10):
           list_img.append(shuf[i])
           img = cv2.imread(os.path.join(BASE_FOLDER + 'test_stg1/', shuf[i]))
           img = cv2.resize(img, (200,200))
           img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
           img = np.asarray(img)
           test_stg.append(img)
     test_stg = np.asarray(test_stg)
     arreglo = cnn.predict(test_stg)
     print( test_stg.shape)
```

```
(10, 200, 200, 3)
```

[ ]:

```
print( test_stg.shape[0])
```

```
10
```

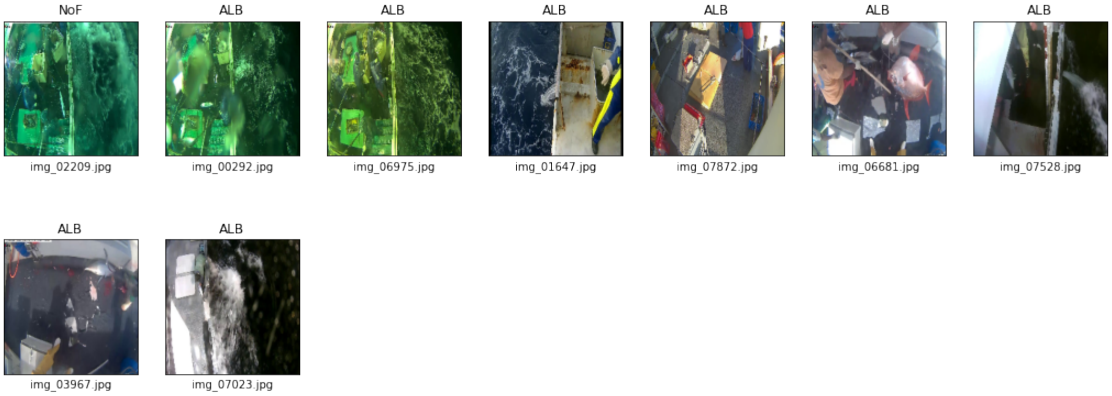
[ ]:

```
list_respuesta = []
list_resultado = []
for i in range(len(arreglo)):
    resultado = arreglo[i]
    respuesta = categorias[np.argmax(resultado)]
    list_img.append(resultado)
    list_respuesta.append(respuesta)
    list_resultado.append(np.argmax(resultado))
print(list_respuesta, list_resultado )
```

```
['YFT', 'NoF', 'ALB', 'ALB', 'ALB', 'ALB', 'ALB', 'ALB', 'ALB', 'ALB'] [5, 1, 7,
7, 7, 7, 7, 7, 7, 7]
```

[ ]:

```
# Visualizacion de imagenes predecidas
import matplotlib.pyplot as plt
fig_stg = plt.figure(figsize=(18,14))

for i in range(1, 10):
  fig_stg.add_subplot(4,7,i)
  plt.xticks([])
  plt.yticks([])
  plt.grid(False)
  plt.imshow(test_stg[i])
  plt.title(list_respuesta[i])
  plt.xlabel(list_img[i])
  # plt.axis('off')
plt.show()
# print(resultado)
```



12MBID_Proyecto_Programacion_Entrega2

May 22, 2022

0.1 Proyecto de programación ”Deep Vision in classification tasks Pre-entrenado”

[ ]:
```
%%capture
!pip freeze
```

Hola voy a ejecutar el comando `!pip freeze` para cotillear a Google Colab

[ ]:
```
#Importemos TensorFlow 2.X y Numpy
import numpy as np
import tensorflow as tf
tf.__version__
```

[ ]:
```
'2.8.0'
```

0.2 - Cargando el conjunto de datos

1 Escritura de datos tomando como referencia un BASE_FOLDER

[ ]:
```
# Conectamos con nuestro Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

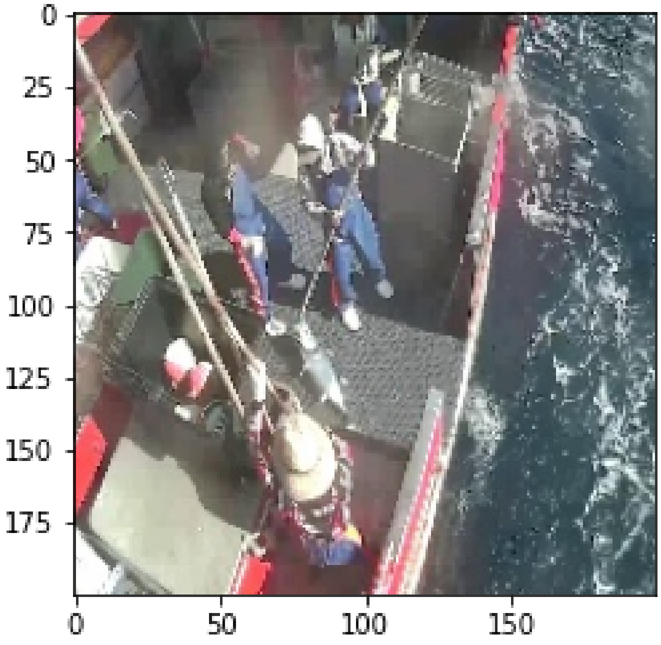
```
Mounted at /content/drive
```

[ ]:
```
# Establezco una ruta absoluta a un directorio existente de mi Google Drive

BASE_FOLDER = "/content/drive/MyDrive/12MBID_Proyecto_Programacion_Colab/
 ↪my_dataset/"
```

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import cv2

# Escogiendo y mostrando una imagen al azar del conjunto de test
```

```
indx = 12
img = cv2.imread(BASE_FOLDER + 'Train/ALB/img_00012.jpg', cv2.COLOR_BGR2RGB)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img)
```

```
[ ]: <matplotlib.image.AxesImage at 0x7f2cf6240110>
```



```
[ ]: import os
categorias = []
categorias = os.listdir(BASE_FOLDER + 'Train/')
print(categorias)
```

```
['YFT', 'SHARK', 'OTHER', 'NoF', 'LAG', 'DOL', 'BET', 'ALB']
```

```
[ ]: # Generando datos con etiquetas a partir de imagenes  por cada categoria para␣
 ↪el test
x_train =[]
y_train =[]

# Generando subcarpetas con imagenes  por cada categoria
idx = 0
for cat in categorias:
    parent_dir = BASE_FOLDER + "Train/"
    path = os.path.join(parent_dir, cat)
```

```
    # creamos las imagenes para el entranamiento redimensionandolos
    for imagen in os.listdir(path):
      img = cv2.imread(os.path.join(path, imagen))
      img = cv2.resize(img, (200,200))
      img = np.asarray(img)
      x_train.append(img)
      y_train.append(idx)
    idx += 1
x_train = np.asarray(x_train)
y_train = np.asarray(y_train)
print(x_train.shape)
print(y_train.shape)
```

```
(3404, 200, 200, 3)
(3404,)
```

[ ]:

```
# Generando datos de entrenamiento extrayendo del directory Test/
x_test = []
y_test = []

idx = 0
for cat in categorias:
    parent_dir = BASE_FOLDER + "Test/"
    path = os.path.join(parent_dir, cat)
    # creamos las imagenes para el entranamiento redimensionandolos
    for imagen in os.listdir(path):
      img = cv2.imread(os.path.join(path, imagen))
      img = cv2.resize(img, (200,200))
      img = np.asarray(img)
      x_test.append(img)
      y_test.append(idx)
    idx += 1
x_test = np.asarray(x_test)
y_test = np.asarray(y_test)
print(x_test.shape)
print(y_test.shape)
```

```
(373, 200, 200, 3)
(373,)
```

[ ]:

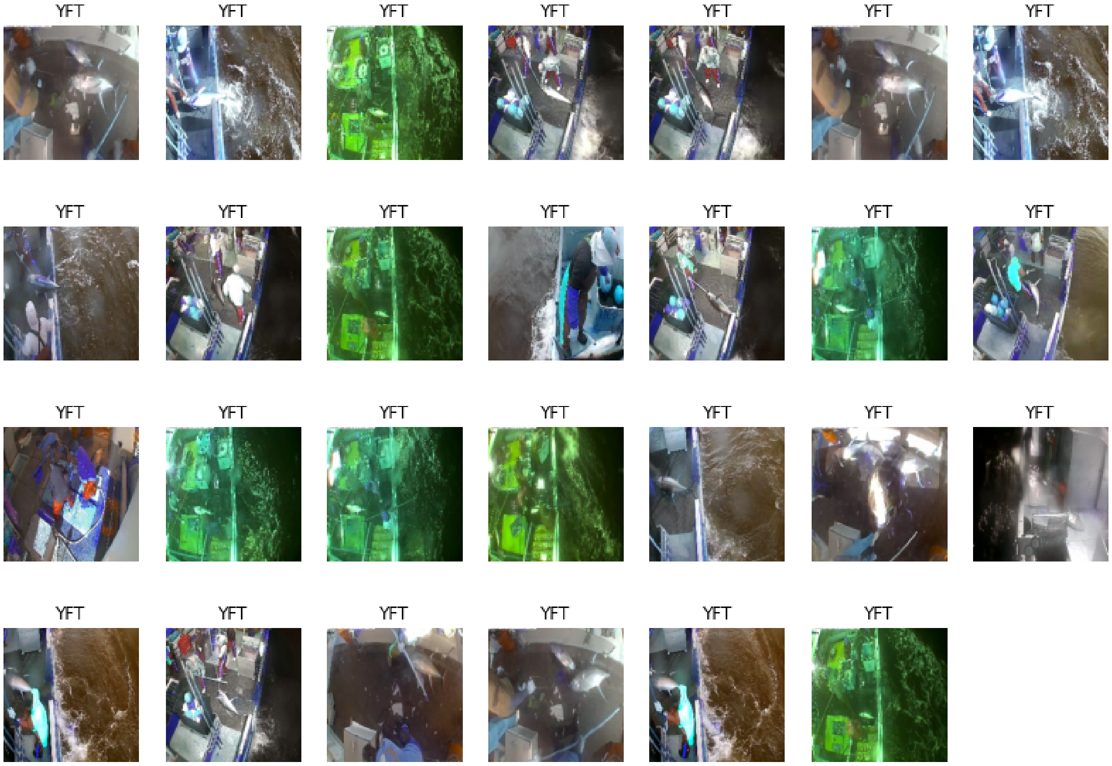
```
import pandas as pd
data_df = pd.DataFrame(y_test)
y_test = np.asarray(y_test)
print(y_test.shape)
```

```
(373,)
```

2 Verificando las imagenes

[ ]:

```
fig = plt.figure(figsize=(14,10))
for i in range(1, 28):
  fig.add_subplot(4,7,i)
  plt.xticks([])
  plt.yticks([])
  plt.grid(False)
  plt.imshow(x_train[i])
  plt.title(categorias[y_train[i]])
  plt.axis('off')
```



3 TRABAJANDO CON REDES PRE-ENTRENADAS

3.1 Acondicionamiento del conjunto de datos como en la VGG

[ ]:

```
from tensorflow.keras.applications import imagenet_utils
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.backend import expand_dims

```

```
#One-hot encoding
lb = LabelBinarizer()
trainY = lb.fit_transform(y_train)
testY = lb.transform(y_test)

# IMPORTANTE: Se normalizan los datos como se normalizaron en el entrenamiento␣
 ↪con ImageNet!!
trainX = imagenet_utils.preprocess_input(x_train)
testX = imagenet_utils.preprocess_input(x_test)

labelNames = categorias

print(trainX.shape)
print(trainY.shape)
print(testX.shape)
print(testY.shape)
```

```
(3404, 200, 200, 3)
(3404, 8)
(373, 200, 200, 3)
(373, 8)
```

3.2 Creando un contenedor DataGenerator para el aumento automatico de muestras

[ ]:

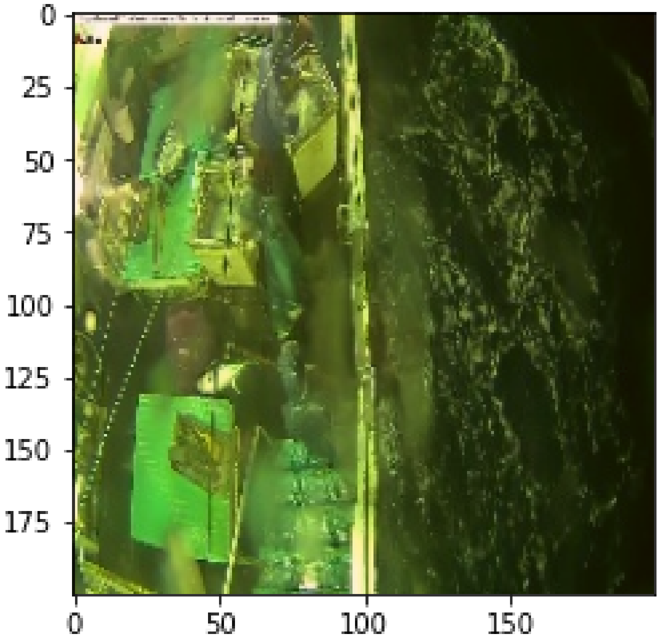
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=15, # grados de rotacion aleatoria
    width_shift_range=0.2, # fraccion del total (1) para mover la imagen
    height_shift_range=0.2, # fraccion del total (1) para mover la imagen
    horizontal_flip=True, # girar las imagenes horizontalmente (eje vertical)
    # shear_range=0, # deslizamiento
    zoom_range=0.2, # rango de zoom
    # fill_mode='nearest', # como rellenar posibles nuevos pixeles
    # channel_shift_range=0.2 # cambios aleatorios en los canales de la imagen
)
```

3.3 Inspeccionando las muestras generadas sinteticamente

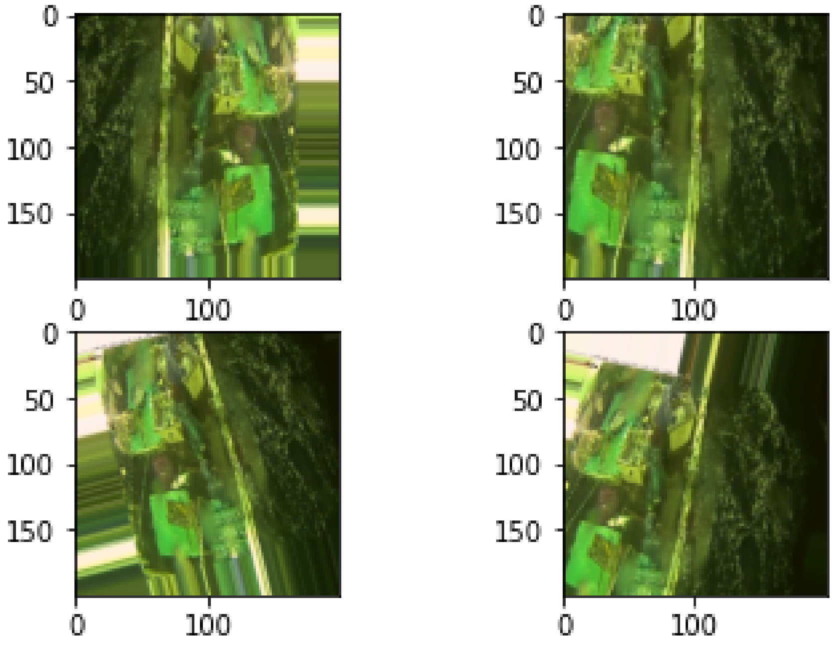
[ ]:

```
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
%matplotlib inline

sample = 45
```

```
plt.imshow(image.array_to_img(trainX[sample]))
plt.show()
print('Label = {}'.format(labelNames[trainY[sample].argmax(axis=0)]))

fig, axes = plt.subplots(2,2)
i = 0
for batch in datagen.flow(trainX[sample].reshape((1,200,200,3)),batch_size=1):
    #plt.figure(i)
    axes[i//2,i%2].imshow(image.array_to_img(batch[0]))
    i += 1
    if i == 4:
        break
plt.show()
```



```
Label = YFT
```



4 Cargando la topologia de CNN (base model)

[ ]:

```
#keras incluye varias arquitecturas
# VGG16, VGG19, ResNet50, Xception, InceptionV3, InceptionResNetV2,␣
 ↪MobileNetV2, DenseNet, RasNet
# documentacion https://keras.io/applications/
# Visual Geometry Group 16 / 19 (numero de layers)
# 1 y 2 en la competicion ImageNet 2014
# Kernels pequeños de 3x3

from tensorflow.keras.applications import VGG16

base_model = VGG16(weights='imagenet',
                  include_top=False, # No incluir el top model, i.e. la parte␣
 ↪densa destinada a la clasificación (fully connected layers)
                  input_shape=(200,200,3))
base_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
58900480/58889256 [==============================] - 0s 0us/step
Model: "vgg16"
```

```
-----------------------------------------------------------------
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 200, 200, 3)]     0

 block1_conv1 (Conv2D)       (None, 200, 200, 64)      1792

 block1_conv2 (Conv2D)       (None, 200, 200, 64)      36928

 block1_pool (MaxPooling2D)  (None, 100, 100, 64)      0

 block2_conv1 (Conv2D)       (None, 100, 100, 128)     73856

 block2_conv2 (Conv2D)       (None, 100, 100, 128)     147584

 block2_pool (MaxPooling2D)  (None, 50, 50, 128)       0

 block3_conv1 (Conv2D)       (None, 50, 50, 256)       295168

 block3_conv2 (Conv2D)       (None, 50, 50, 256)       590080

 block3_conv3 (Conv2D)       (None, 50, 50, 256)       590080

 block3_pool (MaxPooling2D)  (None, 25, 25, 256)       0

 block4_conv1 (Conv2D)       (None, 25, 25, 512)       1180160

 block4_conv2 (Conv2D)       (None, 25, 25, 512)       2359808

 block4_conv3 (Conv2D)       (None, 25, 25, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 12, 12, 512)       0

 block5_conv1 (Conv2D)       (None, 12, 12, 512)       2359808

 block5_conv2 (Conv2D)       (None, 12, 12, 512)       2359808

 block5_conv3 (Conv2D)       (None, 12, 12, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 6, 6, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
-----------------------------------------------------------------
```

4.1 Creando el top model y congelando TODAS las capas convolucionales (TRANSFER LEARNING)

```
[ ]: # conectarlo a nueva parte densa
     from tensorflow.keras.models import Sequential
     from tensorflow.keras import layers

     base_model.trainable = False # Evitar que los pesos se modifiquen en la parte␣
      ↪convolucional -> TRANSFER LEARNING
     pre_trained_model = Sequential()
     pre_trained_model.add(base_model)
     pre_trained_model.add(layers.Flatten())
     pre_trained_model.add(layers.Dense(256, activation='relu'))
     pre_trained_model.add(layers.Dense(8, activation='softmax'))

     pre_trained_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 6, 6, 512)         14714688

 flatten (Flatten)           (None, 18432)             0

 dense (Dense)               (None, 256)               4718848

 dense_1 (Dense)             (None, 8)                 2056

=================================================================
Total params: 19,435,592
Trainable params: 4,720,904
Non-trainable params: 14,714,688
_________________________________________________________________
```

4.2 Entrenando la Solucion

```
[ ]: # Import the necessary packages
     import numpy as np
     from tensorflow.keras import backend as K
     from tensorflow.keras.layers import Input, Conv2D, Activation, Flatten, Dense,␣
      ↪Dropout, BatchNormalization, MaxPooling2D
     from tensorflow.keras.models import Model
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.optimizers import SGD, Adam
     from sklearn.metrics import classification_report
```

```python
import matplotlib.pyplot as plt
from google.colab import drive

# Compilar el modelo
print("[INFO]: Compilando el modelo...")
pre_trained_model.compile(loss="categorical_crossentropy", optimizer=Adam(lr=0.0005,decay=0, beta_1=0.9, beta_2=0.999, epsilon=1e-08), metrics=["accuracy"])

# Entrenamiento de la red
print("[INFO]: Entrenando la red...")
H_pre = pre_trained_model.fit(trainX, trainY, batch_size=128, epochs=20, validation_split=0.2)
# Almaceno el modelo en Drive
# Montamos la unidad de Drive
# drive.mount('/content/drive')
# Almacenamos el modelo empleando la función mdoel.save de Keras
pre_trained_model.save(BASE_FOLDER+"deepCNN_FISH_pretrained.h5") #(X)

# Evaluación del modelo
print("[INFO]: Evaluando el modelo...")
# Efectuamos la predicción (empleamos el mismo valor de batch_size que en training)
predictions = pre_trained_model.predict(testX, batch_size=128)
# Sacamos el report para test
print(classification_report(testY.argmax(axis=1), predictions.argmax(axis=1), target_names=labelNames))

# Gráficas
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, 20), H_pre.history["loss"], label="train_loss")
plt.plot(np.arange(0, 20), H_pre.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, 20), H_pre.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, 20), H_pre.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.show()
```

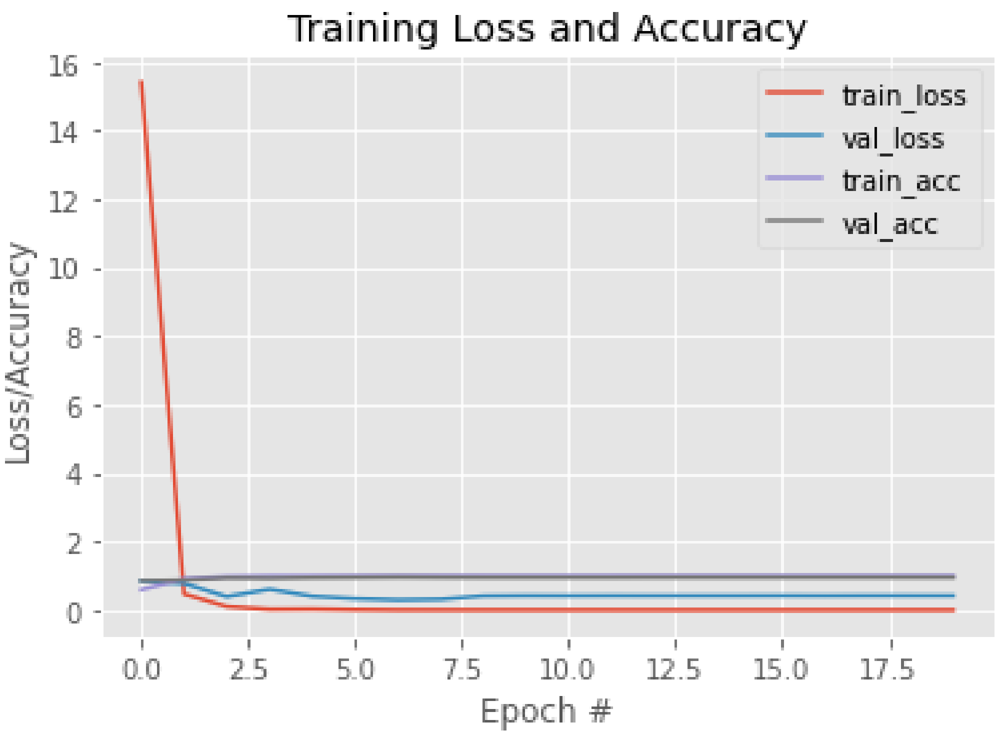
```
[INFO]: Compilando el modelo…
[INFO]: Entrenando la red…
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105:
UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
```

```
Epoch 1/20
```

```
22/22 [==============================] - 38s 930ms/step - loss: 15.4111 -
accuracy: 0.5986 - val_loss: 0.8578 - val_accuracy: 0.8605
Epoch 2/20
22/22 [==============================] - 14s 636ms/step - loss: 0.4734 -
accuracy: 0.9141 - val_loss: 0.7814 - val_accuracy: 0.8811
Epoch 3/20
22/22 [==============================] - 14s 650ms/step - loss: 0.1049 -
accuracy: 0.9791 - val_loss: 0.3969 - val_accuracy: 0.9413
Epoch 4/20
22/22 [==============================] - 15s 667ms/step - loss: 0.0249 -
accuracy: 0.9952 - val_loss: 0.6112 - val_accuracy: 0.9325
Epoch 5/20
22/22 [==============================] - 15s 676ms/step - loss: 0.0298 -
accuracy: 0.9967 - val_loss: 0.4064 - val_accuracy: 0.9501
Epoch 6/20
22/22 [==============================] - 15s 698ms/step - loss: 0.0128 -
accuracy: 0.9985 - val_loss: 0.3362 - val_accuracy: 0.9559
Epoch 7/20
22/22 [==============================] - 16s 721ms/step - loss: 0.0023 -
accuracy: 0.9996 - val_loss: 0.3007 - val_accuracy: 0.9648
Epoch 8/20
22/22 [==============================] - 16s 729ms/step - loss: 0.0014 -
accuracy: 1.0000 - val_loss: 0.3172 - val_accuracy: 0.9589
Epoch 9/20
22/22 [==============================] - 15s 709ms/step - loss: 8.9068e-04 -
accuracy: 1.0000 - val_loss: 0.4165 - val_accuracy: 0.9559
Epoch 10/20
22/22 [==============================] - 15s 698ms/step - loss: 7.5947e-04 -
accuracy: 1.0000 - val_loss: 0.4231 - val_accuracy: 0.9559
Epoch 11/20
22/22 [==============================] - 15s 699ms/step - loss: 6.4752e-04 -
accuracy: 1.0000 - val_loss: 0.4208 - val_accuracy: 0.9559
Epoch 12/20
22/22 [==============================] - 15s 711ms/step - loss: 5.6603e-04 -
accuracy: 1.0000 - val_loss: 0.4227 - val_accuracy: 0.9559
Epoch 13/20
22/22 [==============================] - 16s 717ms/step - loss: 5.0197e-04 -
accuracy: 1.0000 - val_loss: 0.4216 - val_accuracy: 0.9589
Epoch 14/20
22/22 [==============================] - 16s 715ms/step - loss: 4.5130e-04 -
accuracy: 1.0000 - val_loss: 0.4219 - val_accuracy: 0.9589
Epoch 15/20
22/22 [==============================] - 16s 712ms/step - loss: 4.1108e-04 -
accuracy: 1.0000 - val_loss: 0.4221 - val_accuracy: 0.9589
Epoch 16/20
22/22 [==============================] - 15s 711ms/step - loss: 3.7272e-04 -
accuracy: 1.0000 - val_loss: 0.4235 - val_accuracy: 0.9589
Epoch 17/20
```

```
22/22 [==============================] - 15s 710ms/step - loss: 3.4572e-04 -
accuracy: 1.0000 - val_loss: 0.4214 - val_accuracy: 0.9589
Epoch 18/20
22/22 [==============================] - 16s 712ms/step - loss: 3.1707e-04 -
accuracy: 1.0000 - val_loss: 0.4240 - val_accuracy: 0.9589
Epoch 19/20
22/22 [==============================] - 16s 711ms/step - loss: 2.9062e-04 -
accuracy: 1.0000 - val_loss: 0.4235 - val_accuracy: 0.9589
Epoch 20/20
22/22 [==============================] - 15s 711ms/step - loss: 2.6918e-04 -
accuracy: 1.0000 - val_loss: 0.4229 - val_accuracy: 0.9589
[INFO]: Evaluando el modelo…
              precision    recall  f1-score   support

         YFT       0.82      0.85      0.83        73
       SHARK       1.00      0.88      0.94        17
       OTHER       0.76      0.55      0.64        29
         NoF       0.86      0.91      0.88        46
         LAG       1.00      0.33      0.50         6
         DOL       1.00      0.64      0.78        11
         BET       0.85      0.85      0.85        20
         ALB       0.87      0.94      0.90       171

    accuracy                           0.86       373
   macro avg       0.89      0.74      0.79       373
weighted avg       0.86      0.86      0.86       373
```



```
[ ]: inputs = Input(shape=(trainX.shape[1], trainX.shape[2], trainX.shape[3]))
     print(inputs.shape)
```

```
(None, 200, 200, 3)
```

5 REDUCIENDO OVERFITTING MEDIANTE DATA AUGMENTATION

5.1 Creando el top model y descongelando bloques convolucionales (FINE TUNING)

```
[ ]: # Imports que vamos a necesitar
     %tensorflow_version 1.x
     # from tensorflow.keras.datasets import cifar10
     from tensorflow.keras.layers import Input, Conv2D, Activation, Flatten, Dense,␣
      ↪Dropout, BatchNormalization, MaxPooling2D
     from tensorflow.keras.applications import VGG16, imagenet_utils
     from tensorflow.keras.utils import to_categorical
     from tensorflow.keras import optimizers
     from tensorflow.keras.layers import Dropout, Flatten, Dense
     from tensorflow.keras import Model
```

```python
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
import numpy as np

#Cargamos el dataset CIFAR10
# (trainX, trainY), (testX, testY) = cifar10.load_data()
inputs = Input(shape=(trainX.shape[1], trainX.shape[2], trainX.shape[3]))
# Normalizamos las entradas de idéntica forma a como lo hicieron para entrenar␣
 ↪la VGG16 en imageNet
trainX = imagenet_utils.preprocess_input(x_train)
testX = imagenet_utils.preprocess_input(x_test)

# Definimos dimensiones de nuestros datos de entrada y lista con las categorias␣
 ↪de las clases
input_shape = (200, 200, 3)
# labelNames = ["Avión", "Automóvil", "Pájaro", "Gato", "Ciervo", "Perro",␣
 ↪"Rana", "Caballo", "Barco", "Camión"]
labelNames = categorias
# En caso de inestabilidades numéricas pasar datos a one-hot encoding
trainY = to_categorical(y_train)
testY = to_categorical(y_test)

# Importamos VGG16 con pesos de imagenet y sin top_model especificando tamaño␣
 ↪de entrada de datos
base_model = VGG16(weights='imagenet', include_top=False,␣
 ↪input_shape=input_shape)
# Mostramos la arquitectura
base_model.summary()

# Congelamos las capas de los 4 primeros bloques convolucionales, el quinto se␣
 ↪re-entrena
# En base_model.layers.name tenemos la información del nombre de la capa
for layer in base_model.layers:
  if layer.name == 'block3_conv1':
    break
  layer.trainable = False
  print('Capa ' + layer.name + ' congelada...')

# Cogemos la última capa del model y le añadimos nuestro clasificador␣
 ↪(top_model)
last = base_model.layers[-1].output
x = Flatten()(last)
x = Dense(1024, activation='relu', name='fc1')(x)
x = Dropout(0.3)(x)
x = Dense(256, activation='relu', name='fc2')(x)
predictions = Dense(8, activation='softmax', name='predictions')(x)
```

```
model_aug = Model(base_model.input, outputs=predictions)


# Compilamos el modelo
# model_aug.compile(optimizer='sgd', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
model_aug.compile(loss="categorical_crossentropy", optimizer=Adam(lr=0.
 ↪001,decay=0, beta_1=0.9, beta_2=0.999, epsilon=1e-08), metrics=["accuracy"])

# Vamos a visualizar el modelo prestando especial atención en el número de␣
 ↪pesos total y el número de pesos entrenables.
# ¿tiene sentido en comparación al ejemplo de transfer learning?
model_aug.summary()

# Unimos las entradas y el modelo mediante la función Model con parámetros␣
 ↪inputs y ouputs (Consultar la documentación)
# model_aug = Model(inputs=inputs, outputs=predictions)

# Entrenamos el modelo
# H = model_aug.fit(trainX, trainY, validation_split=0.2, batch_size=256,␣
 ↪epochs=20, verbose=1)
H = model_aug.fit(datagen.flow(trainX, trainY, batch_size=128),  epochs=20,␣
 ↪validation_data=(testX, testY))

# Evaluación del modelo
print("[INFO]: Evaluando el modelo...")
predictions = model_aug.predict(testX, batch_size=64)
# Obtener el report de clasificación
print(classification_report(testY.argmax(axis=1), predictions.argmax(axis=1),␣
 ↪target_names=labelNames))

# Gráficas
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, 20), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, 20), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, 20), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, 20), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.show()
```

```
TensorFlow is already loaded. Please restart the runtime to change versions.
Model: "vgg16"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 200, 200, 3)]     0

 block1_conv1 (Conv2D)       (None, 200, 200, 64)      1792

 block1_conv2 (Conv2D)       (None, 200, 200, 64)      36928

 block1_pool (MaxPooling2D)  (None, 100, 100, 64)      0

 block2_conv1 (Conv2D)       (None, 100, 100, 128)     73856

 block2_conv2 (Conv2D)       (None, 100, 100, 128)     147584

 block2_pool (MaxPooling2D)  (None, 50, 50, 128)       0

 block3_conv1 (Conv2D)       (None, 50, 50, 256)       295168

 block3_conv2 (Conv2D)       (None, 50, 50, 256)       590080

 block3_conv3 (Conv2D)       (None, 50, 50, 256)       590080

 block3_pool (MaxPooling2D)  (None, 25, 25, 256)       0

 block4_conv1 (Conv2D)       (None, 25, 25, 512)       1180160

 block4_conv2 (Conv2D)       (None, 25, 25, 512)       2359808

 block4_conv3 (Conv2D)       (None, 25, 25, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 12, 12, 512)       0

 block5_conv1 (Conv2D)       (None, 12, 12, 512)       2359808

 block5_conv2 (Conv2D)       (None, 12, 12, 512)       2359808

 block5_conv3 (Conv2D)       (None, 12, 12, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 6, 6, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
Capa input_5 congelada…
Capa block1_conv1 congelada…
```

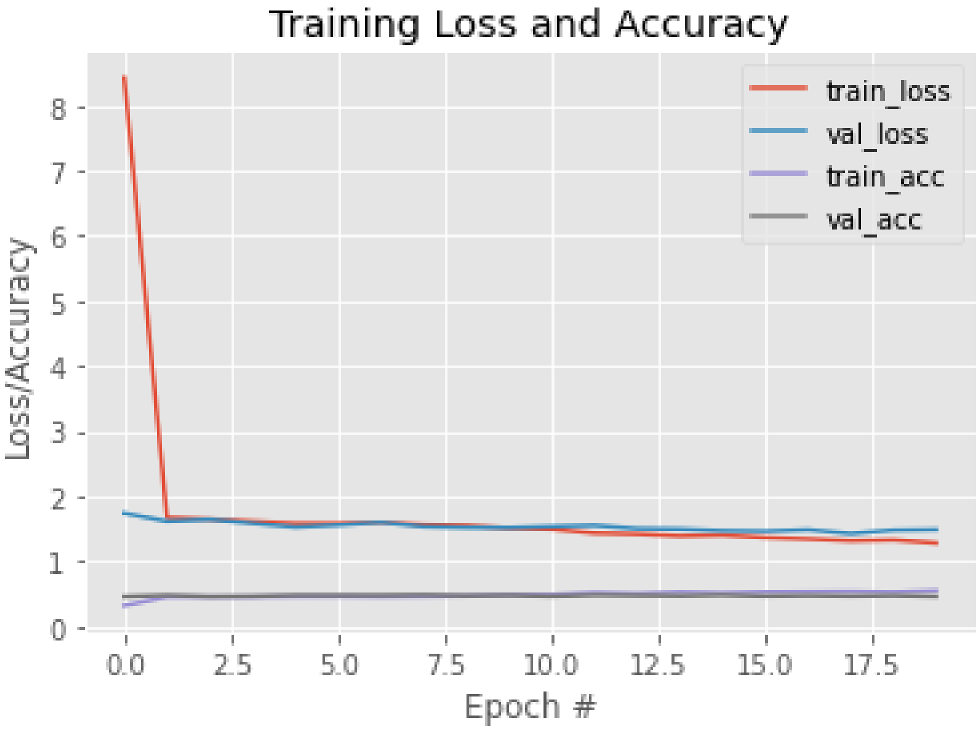
```
Capa block1_conv2 congelada…
Capa block1_pool congelada…
Capa block2_conv1 congelada…
Capa block2_conv2 congelada…
Capa block2_pool congelada…
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 200, 200, 3)]     0

 block1_conv1 (Conv2D)       (None, 200, 200, 64)      1792

 block1_conv2 (Conv2D)       (None, 200, 200, 64)      36928

 block1_pool (MaxPooling2D)  (None, 100, 100, 64)      0

 block2_conv1 (Conv2D)       (None, 100, 100, 128)     73856

 block2_conv2 (Conv2D)       (None, 100, 100, 128)     147584

 block2_pool (MaxPooling2D)  (None, 50, 50, 128)       0

 block3_conv1 (Conv2D)       (None, 50, 50, 256)       295168

 block3_conv2 (Conv2D)       (None, 50, 50, 256)       590080

 block3_conv3 (Conv2D)       (None, 50, 50, 256)       590080

 block3_pool (MaxPooling2D)  (None, 25, 25, 256)       0

 block4_conv1 (Conv2D)       (None, 25, 25, 512)       1180160

 block4_conv2 (Conv2D)       (None, 25, 25, 512)       2359808

 block4_conv3 (Conv2D)       (None, 25, 25, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 12, 12, 512)       0

 block5_conv1 (Conv2D)       (None, 12, 12, 512)       2359808

 block5_conv2 (Conv2D)       (None, 12, 12, 512)       2359808

 block5_conv3 (Conv2D)       (None, 12, 12, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 6, 6, 512)         0

 flatten_2 (Flatten)         (None, 18432)             0
```

```
 fc1 (Dense)                 (None, 1024)              18875392

 dropout_1 (Dropout)         (None, 1024)              0

 fc2 (Dense)                 (None, 256)               262400

 predictions (Dense)         (None, 8)                 2056

=================================================================
Total params: 33,854,536
Trainable params: 33,594,376
Non-trainable params: 260,160
-----------------------------------------------------------------
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105:
UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
```

```
Epoch 1/20
27/27 [==============================] - 40s 1s/step - loss: 8.4299 - accuracy:
0.3193 - val_loss: 1.7380 - val_accuracy: 0.4584
Epoch 2/20
27/27 [==============================] - 39s 1s/step - loss: 1.6706 - accuracy:
0.4539 - val_loss: 1.6145 - val_accuracy: 0.4799
Epoch 3/20
27/27 [==============================] - 39s 1s/step - loss: 1.6459 - accuracy:
0.4548 - val_loss: 1.6424 - val_accuracy: 0.4584
Epoch 4/20
27/27 [==============================] - 39s 1s/step - loss: 1.6128 - accuracy:
0.4548 - val_loss: 1.5821 - val_accuracy: 0.4638
Epoch 5/20
27/27 [==============================] - 38s 1s/step - loss: 1.5825 - accuracy:
0.4595 - val_loss: 1.5259 - val_accuracy: 0.4799
Epoch 6/20
27/27 [==============================] - 38s 1s/step - loss: 1.5821 - accuracy:
0.4615 - val_loss: 1.5560 - val_accuracy: 0.4799
Epoch 7/20
27/27 [==============================] - 39s 1s/step - loss: 1.5983 - accuracy:
0.4568 - val_loss: 1.5974 - val_accuracy: 0.4799
Epoch 8/20
27/27 [==============================] - 39s 1s/step - loss: 1.5615 - accuracy:
0.4618 - val_loss: 1.5332 - val_accuracy: 0.4853
Epoch 9/20
27/27 [==============================] - 39s 1s/step - loss: 1.5477 - accuracy:
0.4680 - val_loss: 1.5192 - val_accuracy: 0.4745
Epoch 10/20
27/27 [==============================] - 39s 1s/step - loss: 1.5102 - accuracy:
0.4880 - val_loss: 1.5152 - val_accuracy: 0.4799
```

```
Epoch 11/20
27/27 [==============================] - 39s 1s/step - loss: 1.4925 - accuracy:
0.4888 - val_loss: 1.5334 - val_accuracy: 0.4638
Epoch 12/20
27/27 [==============================] - 39s 1s/step - loss: 1.4325 - accuracy:
0.5182 - val_loss: 1.5499 - val_accuracy: 0.4826
Epoch 13/20
27/27 [==============================] - 39s 1s/step - loss: 1.4182 - accuracy:
0.5032 - val_loss: 1.5009 - val_accuracy: 0.4772
Epoch 14/20
27/27 [==============================] - 39s 1s/step - loss: 1.3894 - accuracy:
0.5273 - val_loss: 1.4938 - val_accuracy: 0.4718
Epoch 15/20
27/27 [==============================] - 39s 1s/step - loss: 1.4023 - accuracy:
0.5138 - val_loss: 1.4696 - val_accuracy: 0.4826
Epoch 16/20
27/27 [==============================] - 39s 1s/step - loss: 1.3642 - accuracy:
0.5300 - val_loss: 1.4614 - val_accuracy: 0.4665
Epoch 17/20
27/27 [==============================] - 39s 1s/step - loss: 1.3425 - accuracy:
0.5335 - val_loss: 1.4847 - val_accuracy: 0.4745
Epoch 18/20
27/27 [==============================] - 38s 1s/step - loss: 1.3146 - accuracy:
0.5376 - val_loss: 1.4247 - val_accuracy: 0.4665
Epoch 19/20
27/27 [==============================] - 39s 1s/step - loss: 1.3268 - accuracy:
0.5288 - val_loss: 1.4787 - val_accuracy: 0.4745
Epoch 20/20
27/27 [==============================] - 39s 1s/step - loss: 1.2797 - accuracy:
0.5494 - val_loss: 1.4865 - val_accuracy: 0.4611
[INFO]: Evaluando el modelo…
              precision    recall  f1-score   support

         YFT       0.08      0.01      0.02        73
       SHARK       0.31      0.24      0.27        17
       OTHER       0.00      0.00      0.00        29
         NoF       0.48      0.22      0.30        46
         LAG       0.00      0.00      0.00         6
         DOL       0.00      0.00      0.00        11
         BET       0.00      0.00      0.00        20
         ALB       0.48      0.92      0.63       171

    accuracy                           0.46       373
   macro avg       0.17      0.17      0.15       373
weighted avg       0.31      0.46      0.34       373

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318:
```

```
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



[4]:
```
name_IPYNB_file = '12MBID_Proyecto_Programacion_Entrega1.ipynb'
get_ipython().system(
        "apt update >> /dev/null && apt install texlive-xetex␣
 ↪texlive-fonts-recommended texlive-generic-recommended >> /dev/null"
    )
get_ipython().system(
            "jupyter nbconvert --output-dir='$BASE_FOLDER'␣
 ↪'$BASE_FOLDER''$name_IPYNB_file' --to pdf"
```

```
        )
```

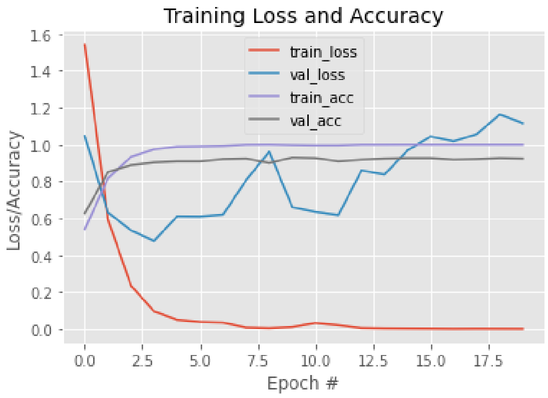
```
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
[NbConvertApp] Converting notebook /content/drive/MyDrive/12MBID_Proyecto_Progra
macion_Colab/my_dataset/12MBID_Proyecto_Programacion_Entrega1.ipynb to pdf
[NbConvertApp] Support files will be in
12MBID_Proyecto_Programacion_Entrega1_files/
[NbConvertApp] Making directory ./12MBID_Proyecto_Programacion_Entrega1_files
[NbConvertApp] Making directory ./12MBID_Proyecto_Programacion_Entrega1_files
[NbConvertApp] Making directory ./12MBID_Proyecto_Programacion_Entrega1_files
[NbConvertApp] Making directory ./12MBID_Proyecto_Programacion_Entrega1_files
[NbConvertApp] Writing 219562 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', './notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', './notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 1019498 bytes to /content/drive/MyDrive/12MBID_Proyecto_P
rogramacion_Colab/my_dataset/12MBID_Proyecto_Programacion_Entrega1.pdf
```

CONCLUCIONES

MODELO PROPIO

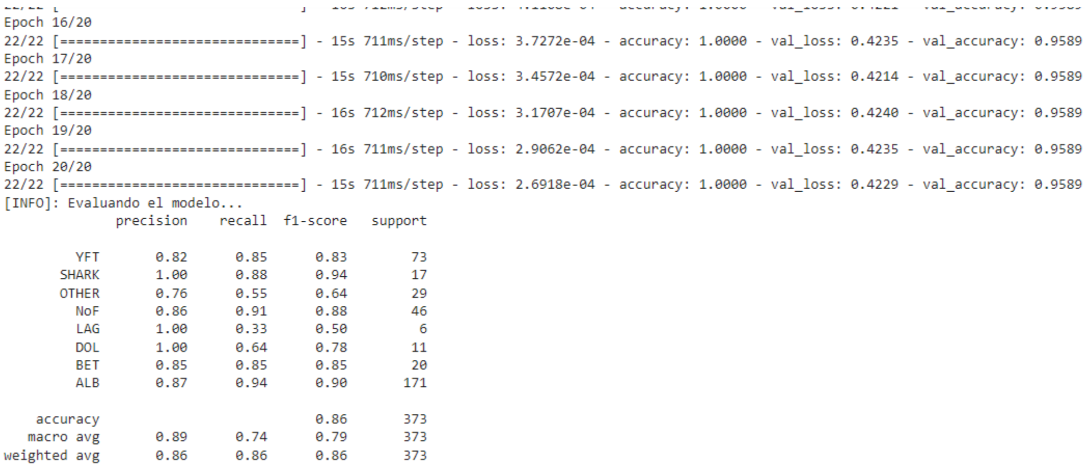
Se observa Overfitting con una precisión de por debajo del 50%, para mejorar se debe usar optimizadores y regularizaciones para reducir el overfitting, por el tiempo corto y las limitaciones del gpu las mejoras lo realizamos en la segunda parte en el cual entrenamos con modelos pre-entrenados y optimizando con data aumentation obteniendo mejores resultados.

```
<matplotlib.legend.Legend at 0x7f0ef8ecb410>
```

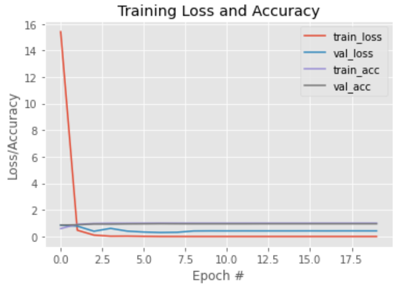


Desarrollo de la **arquitectura** de red neuronal y **entrenamiento** de la solución:

Se compilo la mejora teniendo una métrica de 86% de precisión mejorando el modelo propio menor del 50%.

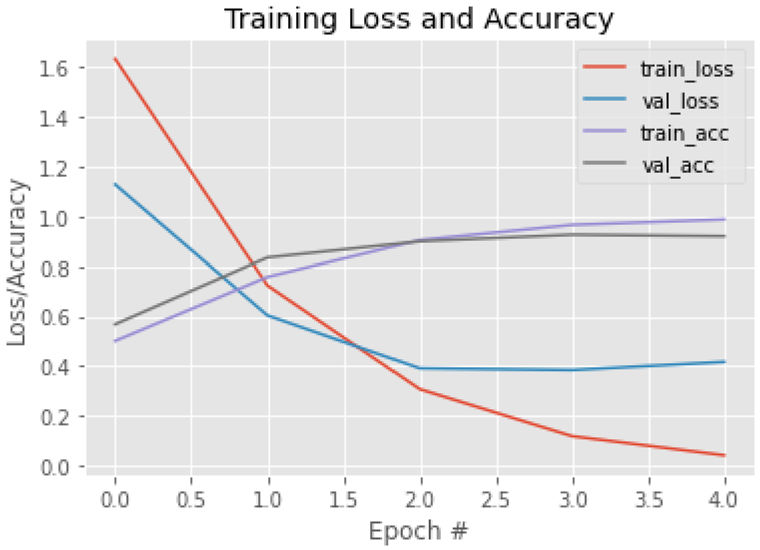


```
Epoch 16/20
22/22 [==============================] - 15s 711ms/step - loss: 3.7272e-04 - accuracy: 1.0000 - val_loss: 0.4235 - val_accuracy: 0.9589
Epoch 17/20
22/22 [==============================] - 15s 710ms/step - loss: 3.4572e-04 - accuracy: 1.0000 - val_loss: 0.4214 - val_accuracy: 0.9589
Epoch 18/20
22/22 [==============================] - 16s 712ms/step - loss: 3.1707e-04 - accuracy: 1.0000 - val_loss: 0.4240 - val_accuracy: 0.9589
Epoch 19/20
22/22 [==============================] - 16s 711ms/step - loss: 2.9062e-04 - accuracy: 1.0000 - val_loss: 0.4235 - val_accuracy: 0.9589
Epoch 20/20
22/22 [==============================] - 15s 711ms/step - loss: 2.6918e-04 - accuracy: 1.0000 - val_loss: 0.4229 - val_accuracy: 0.9589
[INFO]: Evaluando el modelo...
              precision    recall  f1-score   support

         YFT       0.82      0.85      0.83        73
       SHARK       1.00      0.88      0.94        17
       OTHER       0.76      0.55      0.64        29
         NoF       0.86      0.91      0.88        46
         LAG       1.00      0.33      0.50         6
         DOL       1.00      0.64      0.78        11
         BET       0.85      0.85      0.85        20
         ALB       0.87      0.94      0.90       171

    accuracy                           0.86       373
   macro avg       0.89      0.74      0.79       373
weighted avg       0.86      0.86      0.86       373
```



Nota: este grafico inicial se realizó con 5 épocas, lo cual puede ser engañoso los resultados, son insuficientes como para tomar una decisión por tal razón se volvió a compilar con 20 épocas y los resultados se mostraron al inicio donde se observa overffiting.

```
<matplotlib.legend.Legend at 0x7f27b00e0e90>
```



Modelo propio

Mejorando el Modelo Mediante Data Aumentación

La precisión baja drásticamente, debido a la complejidad de las imágenes el tratamiento de las imágenes necesitan pasar primero por una modelo de detección para detectar cuerpos que se desea clasificar y filtrar estas imágenes para tener un objetivo más claro como etiqueta de salida, es un proceso que ya no llegamos a realizarlo.

Otra razón puede ser que el tiempo de entrenamiento se debió hacer con más épocas ya que se vio que el acurusary mejoraba mientras más épocas tenia, los recursos de gpu se nos limitaron en colab.

Otra observación es que la gráfica de las métricas muestra la tendencia de la curva muy parejas entre la perdida y la validación