PROYECTO DE TRATAMIENTO DE DATOS

1. En la esta sección se importa las librerías necesarias para la ejecución del proyecto.

```
[105]: #Importacion de Paquetes
import os
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
import keras
#Exportacion desde las librerias
################## KERAS ######################
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Conv2D
from keras.layers import Dropout
from keras.models import load_model
from keras.models import Model

################## SKLEARN ######################
from sklearn import tree
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

################## TENSORFLOW y MLXTEND #########
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from mlxtend.plotting import plot_confusion_matrix
```

Se trabaja con las siguientes librerías principales.

- Keras
- tensorflow
- numpy
- sklearn
- mlxtend

2. En esta sección se definen rutas y variables del programa

```
[106]: #Definicion de Variables de programa
#Path de las imagenes C:\Users\ESPOIR\Documents\Proyecto_Final_Tr_Datos
path = 'C:/Users/ESPOIR/Documents/PROYECTO_FINAL/Carnes'
#Definicion de ruta de imagenes para entrenamiento
path_train = path + '/train'
#Definicion de ruta de imagenes para test o validación
path_test = path + '/test'
#print(path_test)
#################################################################
# Variables Globales
epocas = 60
carga = 100
pixeles_ancho = 300
pixeles_alto  = 300
```

Aquí se especifican rutas de las imágenes para que el modelo este en la capacidad de ser entrenado y posterior validar.
Además se definen algunas variables que se usan en el programa, como: épocas, cargas, tamaño de imágenes.

3. Obtención de la información de los datos

```
[107]: #Obtencion de los datos y definición
#path_train la ruta de las imagenes
#validation_split=fraccion de datos a reservar para validar
#subset = subconjuntos de datos a devolver (validacion/entrenamiento/ambos)
#seed = semilla aleatoria para barajar y transformaciones
#image_size = tamaño de imagen (alto, ancho)
train_ds = tf.keras.utils.image_dataset_from_directory(path_train, validation_split = 0.3,
                                                       subset = "training", seed = 123,
                                                       image_size = [pixeles_alto,pixeles_ancho])
#etiquetas de las clases en las subcarpeta de train
class_names = train_ds.class_names
print(class_names)
```

```
Found 1634 files belonging to 8 classes.
Using 1144 files for training.
['CLASS_01', 'CLASS_02', 'CLASS_03', 'CLASS_04', 'CLASS_05', 'CLASS_06', 'CLASS_07', 'CLASS_08']
```

En esta sesión se obtienen datos de las imágenes como las clases que se disponen en las rutas, ejemplo la de entrenamiento.

4. Graficar imágenes de las carpetas de entrenamiento

```
[108]: ################### GRAFICAR IMAGENES Y DEFINICIONES ############################
#Se define el tamaño por imagen para ser vizualizada
plt.figure(figsize=[20,20])

#Se recorre el arreglo para poder visualizar imagenes de las diferentes clases
for images, labels in train_ds.take(1):
    #vamos a obtener para la grafica un rango de 25
    for i in range(25):
        #definimos que se dibuje en un plot de 5x5
        ax = plt.subplot(5, 5, i+1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
```

Aquí definimos el tamaño de las imágenes que vamos a graficar y la cantidad para visualizar previo a la generación del modelo

5. Definición del modelo y características

```
[109]: ##########################################################################################################################
#Se definen las epocas para el aprendizaje del modelo y parametros del modelo
num_classes = len(class_names)

modelo_imagenes = tf.keras.Sequential(
    [
        tf.keras.layers.Rescaling(1./255, input_shape=(pixeles_alto, pixeles_ancho, 3)),
        tf.keras.layers.Conv2D(16, 3, padding='same', activation='relu'),

        tf.keras.layers.MaxPooling2D(),
        tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu'),

        tf.keras.layers.MaxPooling2D(),
        tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu'),

        tf.keras.layers.MaxPooling2D(),
        tf.keras.layers.Flatten(), tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(num_classes)
])
print("##########################################################################################################################")
print("El modelo se ha generado")
print("##########################################################################################################################")

##########################################################################################################################
El modelo se ha generado
##########################################################################################################################
```

6. Cómpile de modelo

7. Entrenamiento del modelo

Para el entrenamiento del modelo se definieron 60 épocas, con una carga de 100.

```
[111]: ######################### MODELO FIT #################################################################
modelo_clasificador = modelo_imagenes.fit(train_ds,epochs=epocas, batch_size=carga)
print("Modelo Entrenado")
print("##########################################################################################################################")

Epoch 1/60
36/36 [==============================] - 26s 704ms/step - loss: 3.1077 - accuracy: 0.5490
Epoch 2/60
36/36 [==============================] - 26s 727ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 3/60
36/36 [==============================] - 26s 731ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 4/60
36/36 [==============================] - 27s 731ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 5/60
36/36 [==============================] - 27s 759ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 6/60
36/36 [==============================] - 27s 743ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 7/60
36/36 [==============================] - 27s 749ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 8/60
36/36 [==============================] - 28s 760ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 9/60
36/36 [==============================] - 27s 752ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 10/60
36/36 [==============================] - 28s 784ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 11/60
36/36 [==============================] - 29s 788ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 12/60
36/36 [==============================] - 29s 791ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 13/60
36/36 [==============================] - 29s 793ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 14/60
36/36 [==============================] - 29s 790ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 15/60
36/36 [==============================] - 29s 791ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 16/60
```

```
Epoch 44/60
36/36 [==============================] - 30s 820ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 45/60
36/36 [==============================] - 30s 824ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 46/60
36/36 [==============================] - 29s 801ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 47/60
36/36 [==============================] - 29s 805ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 48/60
36/36 [==============================] - 29s 805ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 49/60
36/36 [==============================] - 29s 802ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 50/60
36/36 [==============================] - 29s 801ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 51/60
36/36 [==============================] - 29s 809ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 52/60
36/36 [==============================] - 29s 796ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 53/60
36/36 [==============================] - 29s 801ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 54/60
36/36 [==============================] - 29s 807ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 55/60
36/36 [==============================] - 29s 800ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 56/60
36/36 [==============================] - 29s 799ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 57/60
36/36 [==============================] - 29s 796ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 58/60
36/36 [==============================] - 29s 797ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 59/60
36/36 [==============================] - 29s 802ms/step - loss: 2.8798 - accuracy: 0.5664
Epoch 60/60
36/36 [==============================] - 30s 816ms/step - loss: 2.8798 - accuracy: 0.5664
Modelo Entrenado
```

8. Evaluación de modelo

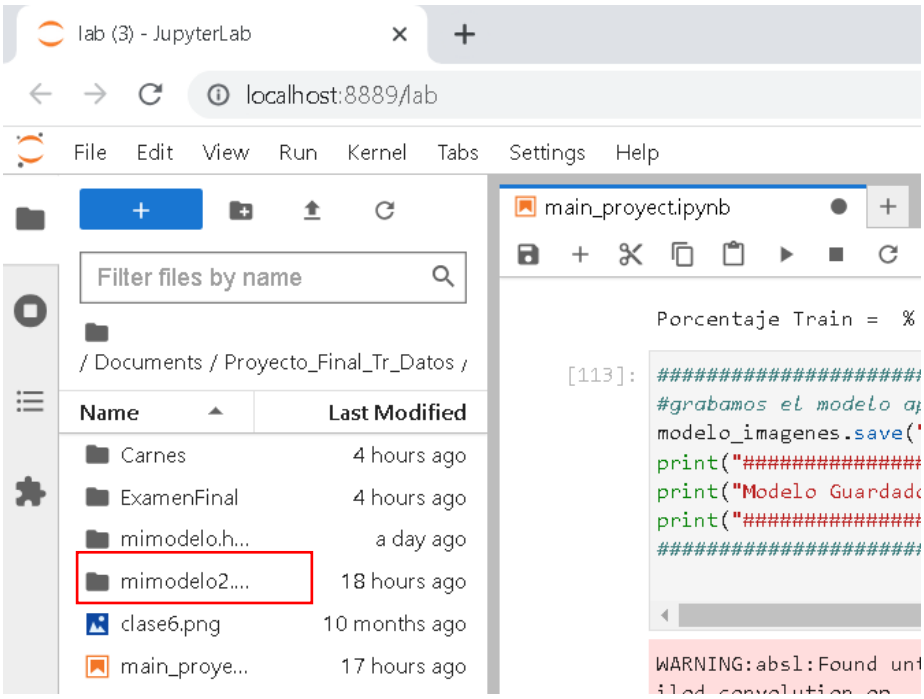
```
112]: ##########################################################################################################################
print("test_ds, train_eval, test_val")
#test_ds
#path_test ruta de las imagenes de entrenamiento
#seed = semilla aleatoria para barajar y transformaciones
#image_size = tamaño de imagen (alto, ancho)
test_ds = tf.keras.utils.image_dataset_from_directory(path_test,
                                                      seed = 123,
                                                      image_size= [pixeles_alto,pixeles_ancho]
                                                     )
#modelo_clasificador.evaluate recibe dos parametros train_ds o test_ds y verbose (valor 0=silencio,
#1=barra progreso, 2=linea unica)
evaluacion_train = modelo_imagenes.evaluate(train_ds, verbose=1)
evaluacion_test = modelo_imagenes.evaluate(test_ds, verbose=1)
porcentaje_test = evaluacion_test[1]*100
porcentaje_train = evaluacion_train[1]*100
print("\n")
print("Porcentaje Test =  %", round(porcentaje_test,2))
print("\n")
print("Porcentaje Train =  %", round(porcentaje_train,2))
```

```
test_ds, train_eval, test_val
Found 810 files belonging to 8 classes.
36/36 [==============================] - 8s 211ms/step - loss: 2.8798 - accuracy: 0.5664
26/26 [==============================] - 5s 197ms/step - loss: 3.2767 - accuracy: 0.5667


Porcentaje Test =  % 56.67
```

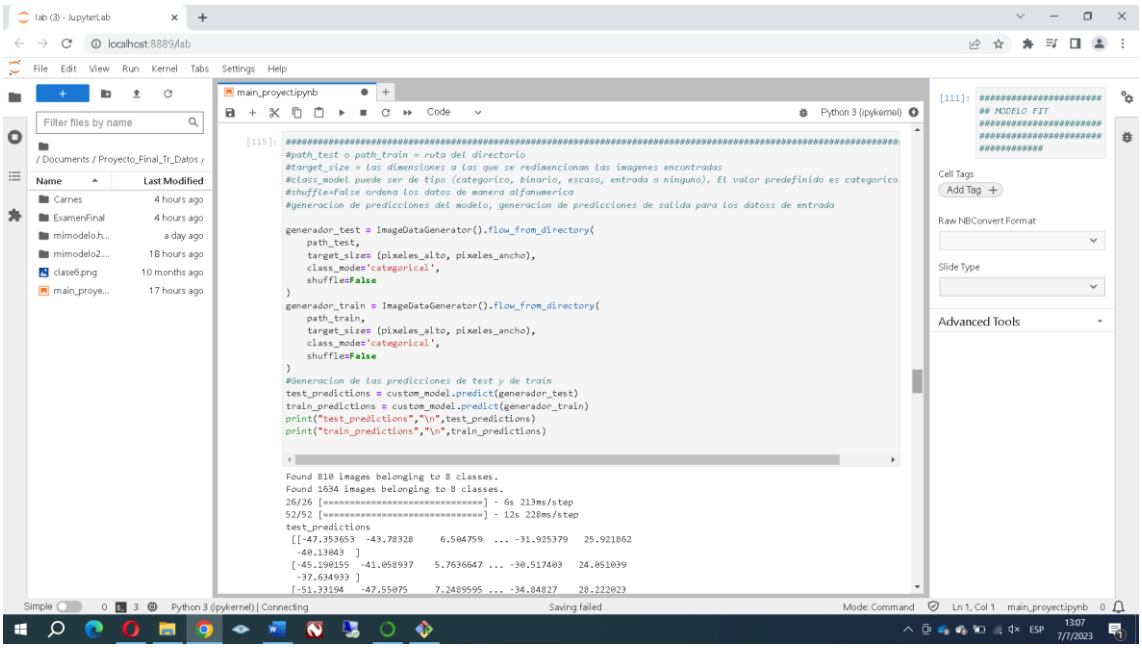
9. Grabar modelo

```
113]: ##########################################################################################################################
#grabamos el modelo aprendido para trabajarlo
modelo_imagenes.save("mimodelo2.h5py")
print("##########################################################################################################################")
print("Modelo Guardado")
print("##########################################################################################################################")
```

Cuando el modelo se ha grabado se crea el modelo en la carpeta raíz del proyecto de Jupyter Lab

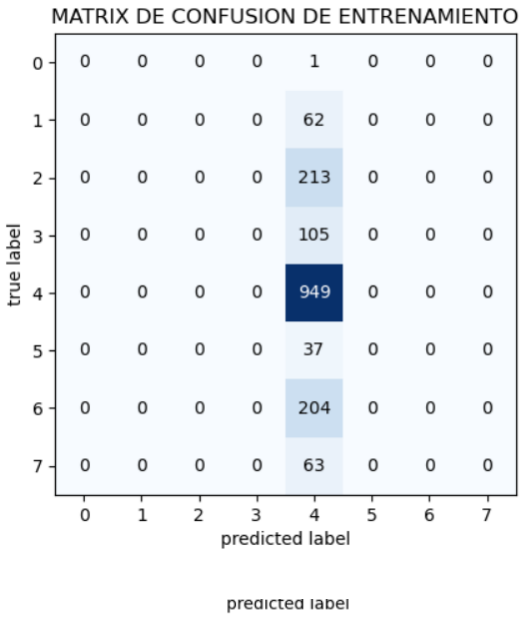


10.

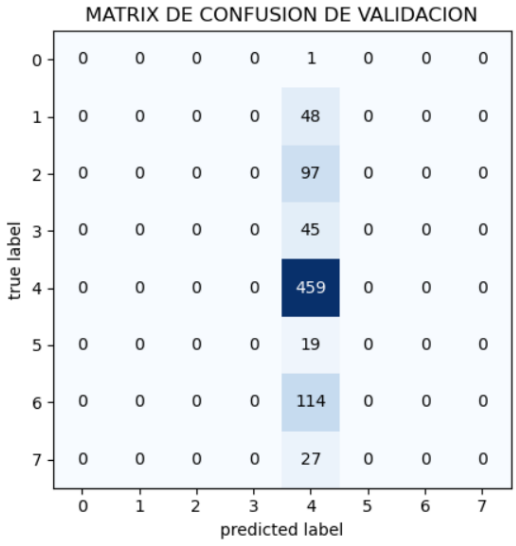


11. Plot de matrices de Convolucion

[120]:
```
#########################################################################################################################
#np.argmax recibe las predicciones y el axis
y_salida = np.argmax(test_predictions, axis=1)
#y entrada recibe las clases de test_generator
y_entrada = generador_test.classes

#np.argmax recibe las predicciones y el axis
x_salida = np.argmax(train_predictions, axis=1)
#X_entrada recibe las clases de train_generator
x_entrada = generador_train.classes

#Matrices de confusion de Validacion y de Entrenamiento
matrix_test=confusion_matrix(y_entrada, y_salida)
matrix_train=confusion_matrix(x_entrada, x_salida)

#Grafico de la matrix de confusion de entrenamiento
plot_confusion_matrix(matrix_train)
plt.title('MATRIX DE CONFUSION DE ENTRENAMIENTO')

#Grafico de la matrix de confusion de entrenamient
plot_confusion_matrix(matrix_test)
plt.title('MATRIX DE CONFUSION DE VALIDACION')
plt.tight_layout()
print("#########################################################################################################################
```

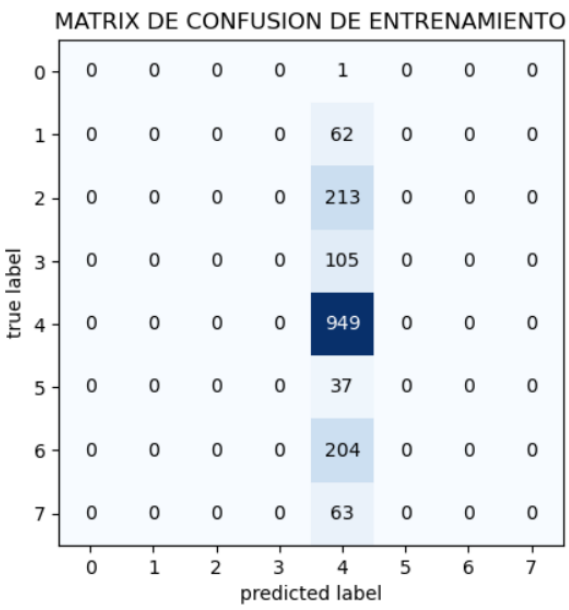


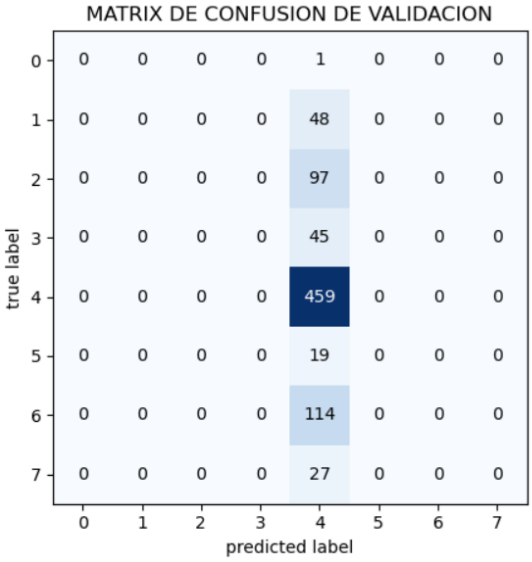
predicted label



12. Matrices

```
#Matrices de confusion de Validacion y de Entrenamiento
matrix_test=confusion_matrix(y_entrada, y_salida)
matrix_train=confusion_matrix(x_entrada, x_salida)

#Grafico de la matrix de confusion de entrenamiento
plot_confusion_matrix(matrix_train)
plt.title('MATRIX DE CONFUSION DE ENTRENAMIENTO')

#Grafico de la matrix de confusion de entrenamient
plot_confusion_matrix(matrix_test)
plt.title('MATRIX DE CONFUSION DE VALIDACION')
plt.tight_layout()
print("##########################################################################################################################################################################################")
```





```
Testing Set TEST:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00        48
           2       0.00      0.00      0.00        97
           3       0.00      0.00      0.00        45
           4       0.57      1.00      0.72       459
           5       0.00      0.00      0.00        19
           6       0.00      0.00      0.00       114
           7       0.00      0.00      0.00        27

    accuracy                           0.57       810
   macro avg       0.07      0.12      0.09       810
weighted avg       0.32      0.57      0.41       810

Testing Set TRAIN:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00        62
           2       0.00      0.00      0.00       213
           3       0.00      0.00      0.00       105
           4       0.58      1.00      0.73       949
           5       0.00      0.00      0.00        37
           6       0.00      0.00      0.00       204
           7       0.00      0.00      0.00        63

    accuracy                           0.58      1634
   macro avg       0.07      0.12      0.09      1634
weighted avg       0.34      0.58      0.43      1634

##############################################
```

13. Validación del modelo

```python
image_path = 'C:/Users/ESPOIR/Documents/Proyecto_Final_Tr_Datos/clase6.png'
print(image_path)

image = tf.keras.preprocessing.image.load_img(image_path).resize([pixeles_alto,pixeles_ancho])
input_arr = tf.keras.preprocessing.image.img_to_array(image)
input_arr = np.array([input_arr])
predictions = modelo_imagenes.predict(input_arr)
score = tf.nn.softmax(predictions[0])

print(" {} con un {:.2f}% de exatitud".format(class_names[np.argmax(score)],100 * np.max(score)))
```

```
C:/Users/ESPOIR/Documents/Proyecto_Final_Tr_Datos/clase6.png
1/1 [==============================] - 0s 80ms/step
 CLASS_05 con un 100.00% de exatitud
```