Unzipde Imágenes

In [1]:

```
from zipfile import ZipFile

dataset = "datasetmodelo.zip"

with ZipFile(dataset, 'r') as zip:
    zip.printdir()
    zip.extractall()
```

```
File Name                                             Modified             Si
ze
datasetmodelo/Bengin cases/                    2022-07-12 20:26:16
0
datasetmodelo/Bengin cases/Bengin case (100).jpg 2021-12-03 15:52:34       14
9360
datasetmodelo/Bengin cases/Bengin case (101).jpg 2021-12-03 15:52:34       30
7805
datasetmodelo/Bengin cases/Bengin case (102).jpg 2021-12-03 15:52:34       31
2811
datasetmodelo/Bengin cases/Bengin case (103).jpg 2021-12-03 15:52:34       30
1345
datasetmodelo/Bengin cases/Bengin case (104).jpg 2021-12-03 15:52:34       30
3198
datasetmodelo/Bengin cases/Bengin case (105).jpg 2021-12-03 15:52:34       30
3198
datasetmodelo/Bengin cases/Bengin case (106).jpg 2021-12-03 15:52:36       30
9777
datasetmodelo/Bengin cases/Bengin case (107).jpg 2021-12-03 15:52:36       30
9777
```

Importar librerías

In [2]:

```
import os

import tensorflow as tf
from tensorflow import keras
import argparse
from imutils import paths
import cv2
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
matplotlib.use("Agg")

# Importar los paquetes de keras y tensorflow
from tensorflow.keras import backend as K
from tensorflow.keras import utils as np_utils
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Conv2D, MaxPooling2D, ZeroPadding
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import  ReduceLROnPlateau
from tensorflow.keras.regularizers import l2
from keras import regularizers
from tensorflow.keras.optimizers import Adam, RMSprop, SGD

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

Carga de Imágenes from_directory

In [3]:

```
#Importando Imágenes con from_directory - mejores resultados para modelo

from tensorflow.keras.preprocessing.image import ImageDataGenerator

dataset_datagen = ImageDataGenerator(rescale=1./255,
                                    zoom_range=0.3,
                                    horizontal_flip=True)
dataset_set=dataset_datagen.flow_from_directory('datasetmodelo',
                                            batch_size=8,
                                            target_size=(512,512),
                                            shuffle=False,
                                            color_mode="grayscale",class_mode='ca
```

```
Found 878 images belonging to 3 classes.
```

In [4]:

```
dataset_set.class_indices
```

Out[4]:

```
{'Bengin cases': 0, 'Malignant cases': 1, 'Normal cases': 2}
```

Carga de Imágenes 1 x 1 y Label Encoder

In [5]:

```
#Importando Imágenes 1 x 1 para Matriz de Confusión y Reporte de Clasificación -

import glob
import tensorflow as tf
import numpy as np

bengin = glob.glob('datasetmodelo/Bengin cases/*.*')
malignant = glob.glob('datasetmodelo/Malignant cases/*.*')
normal = glob.glob('datasetmodelo/Normal cases/*.*')
data = []
labels = []
for i in bengin:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='grayscale',target_
    image=np.array(image)
    data.append(image)
    labels.append('Bengin')
for i in malignant:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='grayscale',target_
    image=np.array(image)
    data.append(image)
    labels.append('Malignant')
for i in normal:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='grayscale',target_
    image=np.array(image)
    data.append(image)
    labels.append('Normal')

set_data = np.array(data)
set_labels = np.array(labels)
set_data.shape
```

Out[5]:

```
(878, 512, 512)
```

In [6]:

```
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils

set_data = set_data.astype('float32')
set_data /= 255

lb = LabelEncoder()
y_set = np_utils.to_categorical(lb.fit_transform(set_labels))
```

In [7]:

```
(trainX, testX, trainY, testY) = train_test_split(set_data, y_set, test_size=0.20
testY.shape
```

Out[7]:

```
(176, 3)
```

In [8]:

```
#Parametros que dependen de la capacidad computacional
INIT_LR = 0
BS = 8
EPOCHS = 20

Hg = 512
Lng = 512
```

Modelo

In [9]:

```
inputs = tf.keras.Input(shape=(512, 512, 1), name='input')

conv1 = tf.keras.layers.Conv2D(filters=16, kernel_size=3, name='conv1')(inputs)
maxpool1 = tf.keras.layers.MaxPooling2D(name='maxpool1')(conv1)

conv2 = tf.keras.layers.Conv2D(filters=32, kernel_size=3, name='conv2')(maxpool1)
maxpool2 = tf.keras.layers.MaxPooling2D(name='maxpool2')(conv2)

conv3 = tf.keras.layers.Conv2D(filters=64, kernel_size=3, name='conv3')(maxpool2)
maxpool3 = tf.keras.layers.MaxPooling2D(name='maxpool3')(conv3)

avgpool = tf.keras.layers.GlobalAveragePooling2D(name='avgpool')(maxpool3)

outputs = tf.keras.layers.Dense(3, activation='softmax', name='output')(avgpool)


model = tf.keras.Model(inputs=inputs, outputs=outputs)
```

In [10]:
```
model.summary()
```

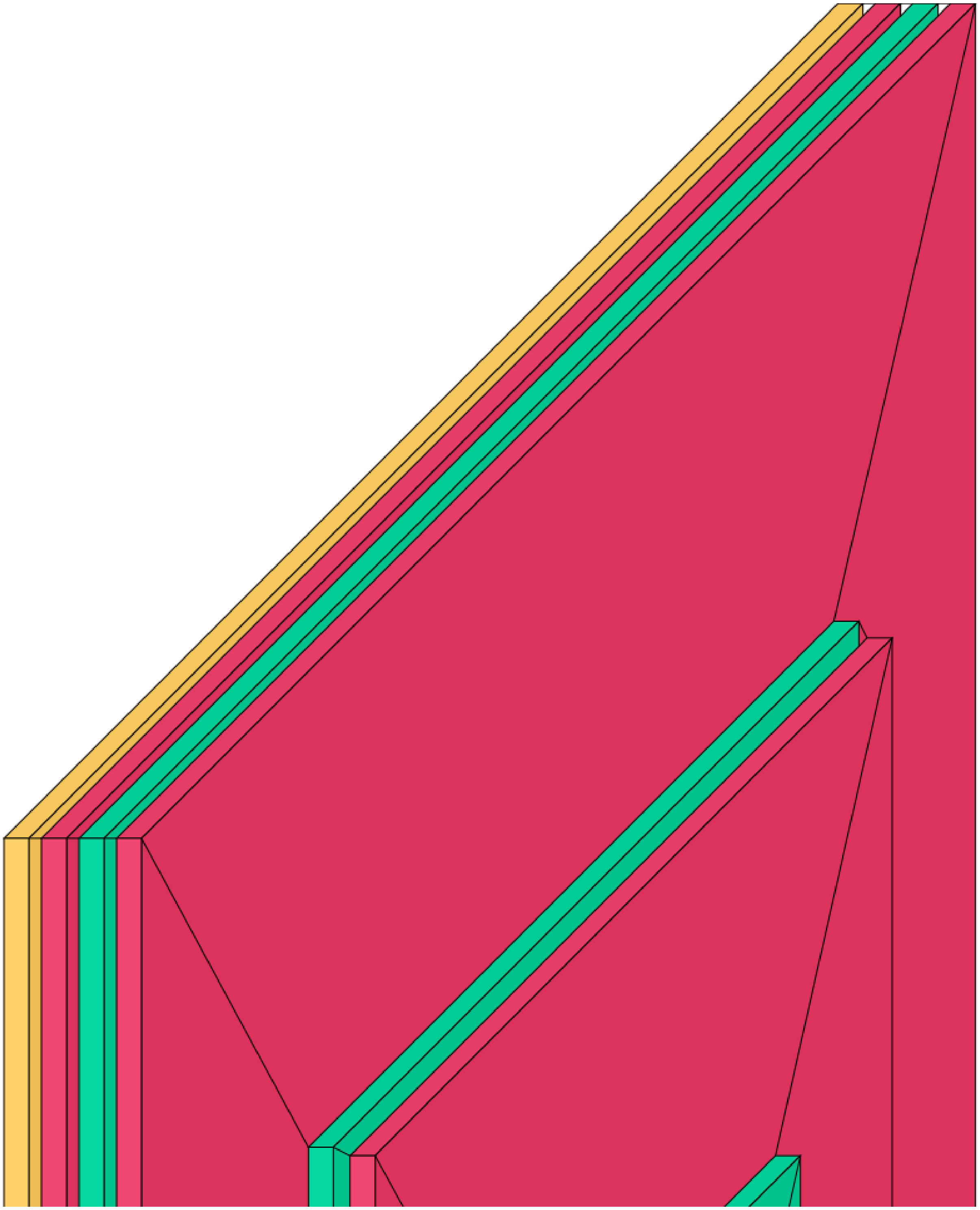
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input (InputLayer)          [(None, 512, 512, 1)]     0         
                                                                 
 conv1 (Conv2D)              (None, 510, 510, 16)      160       
                                                                 
 maxpool1 (MaxPooling2D)     (None, 255, 255, 16)      0         
                                                                 
 conv2 (Conv2D)              (None, 253, 253, 32)      4640      
                                                                 
 maxpool2 (MaxPooling2D)     (None, 126, 126, 32)      0         
                                                                 
 conv3 (Conv2D)              (None, 124, 124, 64)      18496     
                                                                 
 maxpool3 (MaxPooling2D)     (None, 62, 62, 64)        0         
                                                                 
 avgpool (GlobalAveragePooli  (None, 64)               0         
 ng2D)                                                           
                                                                 
 output (Dense)              (None, 3)                 195       
                                                                 
=================================================================
Total params: 23,491
Trainable params: 23,491
Non-trainable params: 0
_________________________________________________________________
```

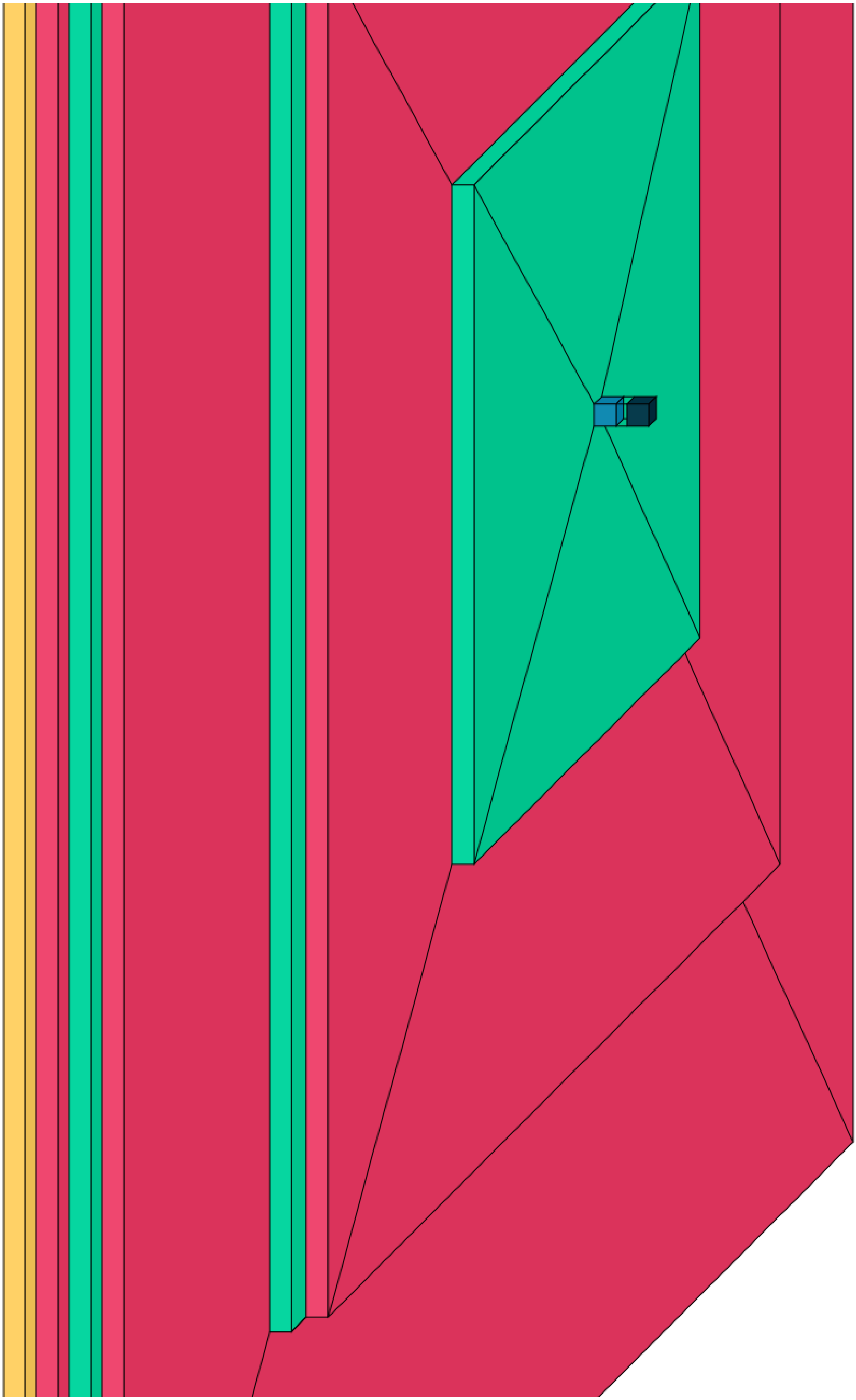
In [11]:

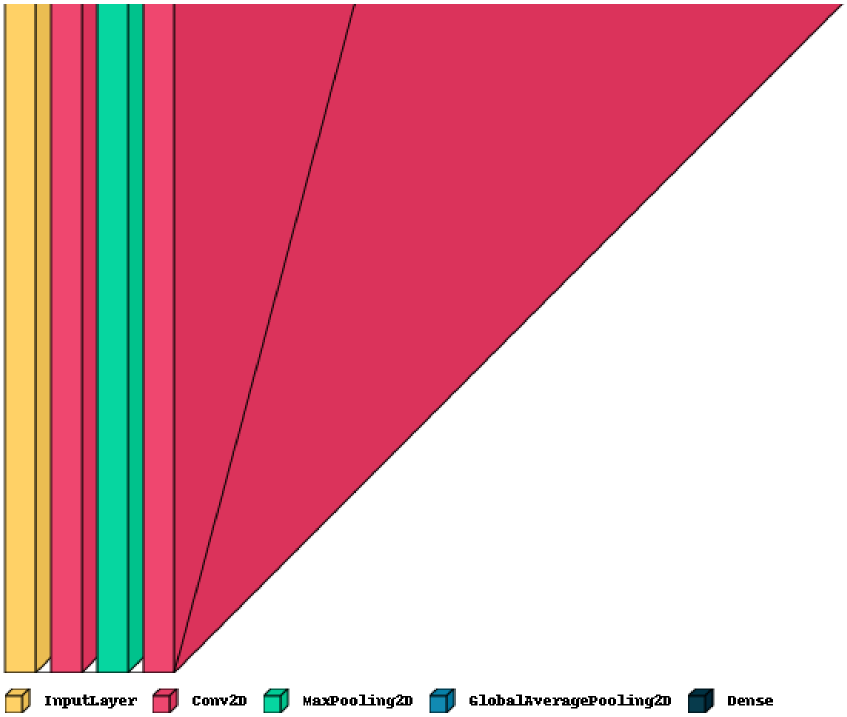
```
# Visualización de Modelo
!pip install visualkeras
import visualkeras
visualkeras.layered_view(model, scale_xy=10, legend=True)
```

```
Requirement already satisfied: visualkeras in c:\users\gueva\anaconda3\envs\dl
\lib\site-packages (0.0.2)
Requirement already satisfied: numpy>=1.18.1 in c:\users\gueva\anaconda3\envs\d
l\lib\site-packages (from visualkeras) (1.22.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\gueva\anaconda3\envs\d
l\lib\site-packages (from visualkeras) (9.0.1)
Requirement already satisfied: aggdraw>=1.3.11 in c:\users\gueva\anaconda3\envs
\dl\lib\site-packages (from visualkeras) (1.3.15)
```

Out[11]:







Entrenamiento del Modelo

In [12]:

```
model.compile(loss="categorical_crossentropy", optimizer= "Adam", metrics=["acc"]

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=
# Train
print("[INFO] Tr {} epochs...".format(EPOCHS))
history = model.fit(trainX, trainY, batch_size=BS, validation_data=(testX, testY)
                                    epochs=EPOCHS)
```

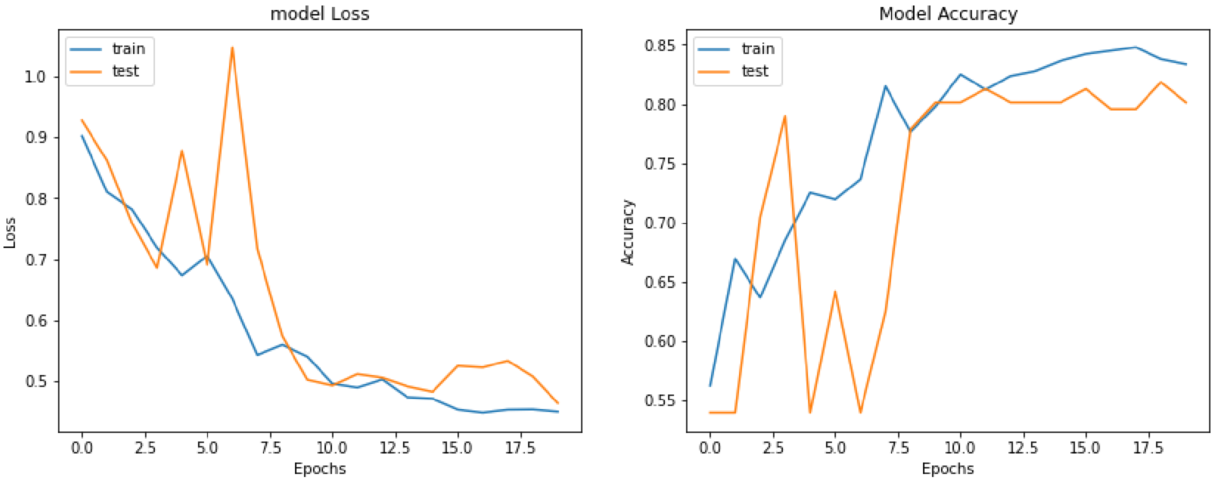
```
[INFO] Tr 20 epochs...
Epoch 1/20
88/88 [==============================] - 108s 1s/step - loss: 0.9021 - acc: 0.5
627 - val_loss: 0.9274 - val_acc: 0.5398 - lr: 0.0010
Epoch 2/20
88/88 [==============================] - 107s 1s/step - loss: 0.8104 - acc: 0.6
695 - val_loss: 0.8620 - val_acc: 0.5398 - lr: 0.0010
Epoch 3/20
88/88 [==============================] - 102s 1s/step - loss: 0.7818 - acc: 0.6
368 - val_loss: 0.7594 - val_acc: 0.7045 - lr: 0.0010
Epoch 4/20
88/88 [==============================] - 111s 1s/step - loss: 0.7191 - acc: 0.6
852 - val_loss: 0.6858 - val_acc: 0.7898 - lr: 0.0010
Epoch 5/20
88/88 [==============================] - 111s 1s/step - loss: 0.6736 - acc: 0.7
251 - val_loss: 0.8781 - val_acc: 0.5398 - lr: 0.0010
Epoch 6/20
88/88 [==============================] - 108s 1s/step - loss: 0.7057 - acc: 0.7
194 - val_loss: 0.6908 - val_acc: 0.6420 - lr: 0.0010
Epoch 7/20
88/88 [==============================] - 108s 1s/step - loss: 0.6354 - acc: 0.7
365 - val_loss: 1.0476 - val_acc: 0.5398 - lr: 0.0010
Epoch 8/20
88/88 [==============================] - 108s 1s/step - loss: 0.5432 - acc: 0.8
148 - val_loss: 0.7173 - val_acc: 0.6250 - lr: 0.0010
Epoch 9/20
88/88 [==============================] - 87s 981ms/step - loss: 0.5601 - acc:
0.7764 - val_loss: 0.5740 - val_acc: 0.7784 - lr: 0.0010
Epoch 10/20
88/88 [==============================] - 91s 1s/step - loss: 0.5405 - acc: 0.79
77 - val_loss: 0.5025 - val_acc: 0.8011 - lr: 0.0010
Epoch 11/20
88/88 [==============================] - 93s 1s/step - loss: 0.4963 - acc: 0.82
48 - val_loss: 0.4934 - val_acc: 0.8011 - lr: 0.0010
Epoch 12/20
88/88 [==============================] - 99s 1s/step - loss: 0.4899 - acc: 0.81
20 - val_loss: 0.5119 - val_acc: 0.8125 - lr: 0.0010
Epoch 13/20
88/88 [==============================] - 97s 1s/step - loss: 0.5033 - acc: 0.82
34 - val_loss: 0.5060 - val_acc: 0.8011 - lr: 0.0010
Epoch 14/20
88/88 [==============================] - 92s 1s/step - loss: 0.4738 - acc: 0.82
76 - val_loss: 0.4919 - val_acc: 0.8011 - lr: 0.0010
Epoch 15/20
88/88 [==============================] - 94s 1s/step - loss: 0.4716 - acc: 0.83
62 - val_loss: 0.4830 - val_acc: 0.8011 - lr: 0.0010
Epoch 16/20
88/88 [==============================] - 92s 1s/step - loss: 0.4536 - acc: 0.84
19 - val_loss: 0.5255 - val_acc: 0.8125 - lr: 0.0010
```

```
Epoch 17/20
88/88 [==============================] - 94s 1s/step - loss: 0.4486 - acc: 0.84
47 - val_loss: 0.5230 - val_acc: 0.7955 - lr: 0.0010
Epoch 18/20
88/88 [==============================] - 92s 1s/step - loss: 0.4535 - acc: 0.84
76 - val_loss: 0.5336 - val_acc: 0.7955 - lr: 0.0010
Epoch 19/20
88/88 [==============================] - 95s 1s/step - loss: 0.4539 - acc: 0.83
76 - val_loss: 0.5080 - val_acc: 0.8182 - lr: 0.0010
Epoch 20/20
88/88 [==============================] - 95s 1s/step - loss: 0.4502 - acc: 0.83
33 - val_loss: 0.4644 - val_acc: 0.8011 - lr: 0.0010
```

In [13]:
```
model.save('Modelval_Cancer2.h5')
```

Loss y Accuracy

In [31]:
```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt

plt.figure(figsize=(14,5))
plt.subplot(1,2,2)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(['train', 'test'], loc='upper left')

plt.subplot(1,2,1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



Evaluación del modelo en set de prueba

In [19]:
```python
from zipfile import ZipFile

dataset = "datasetvalidacion.zip"

with ZipFile(dataset, 'r') as zip:
    zip.printdir()
    zip.extractall()
```

```
datasetvalidacion/Malignant cases/Malignant case (2).jpg 2021-12-03 15:52:50
126631
datasetvalidacion/Malignant cases/Malignant case (20).jpg 2021-12-03 15:52:50
122063
datasetvalidacion/Malignant cases/Malignant case (21).jpg 2021-12-03 15:52:52
124696
datasetvalidacion/Malignant cases/Malignant case (22).jpg 2021-12-03 15:52:52
123945
datasetvalidacion/Malignant cases/Malignant case (23).jpg 2021-12-03 15:52:54
123287
datasetvalidacion/Malignant cases/Malignant case (24).jpg 2021-12-03 15:52:54
123605
datasetvalidacion/Malignant cases/Malignant case (25).jpg 2021-12-03 15:52:54
124477
datasetvalidacion/Malignant cases/Malignant case (26).jpg 2021-12-03 15:52:56
122871
datasetvalidacion/Malignant cases/Malignant case (27).jpg 2021-12-03 15:52:56
118672
datasetvalidacion/Malignant cases/Malignant case (28).jpg 2021-12-03 15:52:56
119375
```

In [20]:

```
#Importando Imágenes 1 x 1 para Matriz de Confusión y Reporte de Clasificación -

import glob
import tensorflow as tf
import numpy as np

bengin = glob.glob('datasetvalidacion/Bengin cases/*.*')
malignant = glob.glob('datasetvalidacion/Malignant cases/*.*')
normal = glob.glob('datasetvalidacion/Normal cases/*.*')
dataval = []
labelsval = []
for i in bengin:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='grayscale',target_
    image=np.array(image)
    dataval.append(image)
    labelsval.append('Bengin')
for i in malignant:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='grayscale',target_
    image=np.array(image)
    dataval.append(image)
    labelsval.append('Malignant')
for i in normal:
    image=tf.keras.preprocessing.image.load_img(i, color_mode='grayscale',target_
    image=np.array(image)
    dataval.append(image)
    labelsval.append('Normal')

setval_data = np.array(dataval)
setval_labels = np.array(labelsval)
setval_data.shape
```

Out[20]:

```
(219, 512, 512)
```

In [21]:

```
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils

setval_data =setval_data.astype('float32')
setval_data /= 255

lb = LabelEncoder()
y_setval = np_utils.to_categorical(lb.fit_transform(setval_labels))
y_setval
print(y_setval.shape)
print(setval_data.shape)
```

```
(219, 3)
(219, 512, 512)
```

In [25]:
```
train_loss, train_accu = model.evaluate(trainX,trainY)
test_loss, test_accu = model.evaluate(testX,testY)
print("final train accuracy = {:.2f} , validation accuracy = {:.2f}".format(trai
```

```
22/22 [==============================] - 27s 1s/step - loss: 0.4225 - acc: 0.84
62
6/6 [==============================] - 7s 1s/step - loss: 0.4644 - acc: 0.8011
final train accuracy = 84.62 , validation accuracy = 80.11
```

In [26]:
```
y_pred = model.predict(setval_data)
y_pred = np.argmax(y_pred, axis=1)
y_pred
```

Out[26]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2,
       2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2],
      dtype=int64)
```

In [27]:
```
y_pred.shape
```

Out[27]:
```
(219,)
```

In [28]:
```
class_labels = dataset_set.class_indices
class_labels = {v:k for k,v in class_labels.items()}
class_labels
```

Out[28]:
```
{0: 'Bengin cases', 1: 'Malignant cases', 2: 'Normal cases'}
```

In [29]:
```
#Convertir test Y a labels
import pandas as pd
y_setval
y_setvalDF=pd.DataFrame(y_setval)
y_setvalDF
y_setvalDF["Label"]=0
y_setvalDF
y_setvalDF.loc[y_setvalDF[0] == 1, "Label"] =0
y_setvalDF.loc[y_setvalDF[1] == 1, "Label"] =1
y_setvalDF.loc[y_setvalDF[2] == 1, "Label"] =2
y_setvalDF
y_setvalDF["Label"]
```

Out[29]:
```
0      0
1      0
2      0
3      0
4      0
      ..
214    2
215    2
216    2
217    2
218    2
Name: Label, Length: 219, dtype: int64
```

Matriz de Confusión y Reporte de Clasificación en Set de Prueba

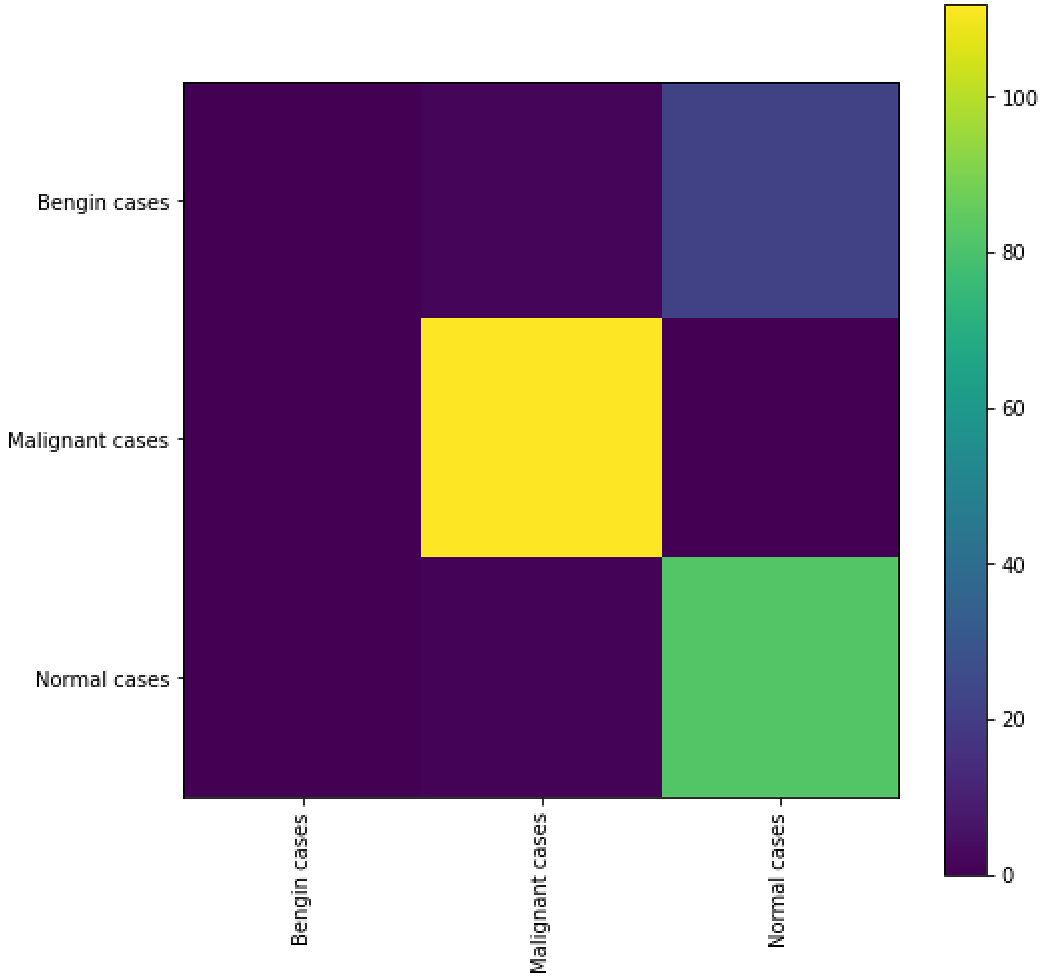
In [30]:

```
from sklearn.metrics import classification_report, confusion_matrix
%matplotlib inline

cm_test = confusion_matrix(y_setvalDF["Label"], y_pred)
print('Confusion Matrix')
print(cm_test)
print('Classification Report')
target_names = list(class_labels.values())
print(classification_report(y_setvalDF["Label"], y_pred, target_names=target_name

plt.figure(figsize=(8,8))
plt.imshow(cm_test, interpolation='nearest')
plt.colorbar()
tick_mark = np.arange(len(target_names))
_ = plt.xticks(tick_mark, target_names, rotation=90)
_ = plt.yticks(tick_mark, target_names)
plt
```

```
Confusion Matrix
[[  0   2  22]
 [  0 112   0]
 [  0   1  82]]
Classification Report
                 precision    recall  f1-score   support

   Bengin cases       0.00      0.00      0.00        24
Malignant cases       0.97      1.00      0.99       112
   Normal cases       0.79      0.99      0.88        83

       accuracy                           0.89       219
      macro avg       0.59      0.66      0.62       219
   weighted avg       0.80      0.89      0.84       219


C:\Users\gueva\anaconda3\envs\DL\lib\site-packages\sklearn\metrics\_classificat
ion.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` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\gueva\anaconda3\envs\DL\lib\site-packages\sklearn\metrics\_classificat
ion.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` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\gueva\anaconda3\envs\DL\lib\site-packages\sklearn\metrics\_classificat
ion.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` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[30]:

```
<module 'matplotlib.pyplot' from 'C:\\Users\\gueva\\anaconda3\\envs\\DL\\lib\\s
ite-packages\\matplotlib\\pyplot.py'>
```



In [32]:
```
hist_df=pd.DataFrame(history.history)
hist_df.to_csv('HistoryModelo.csv')
```