IMPORTAR LIBRERÍAS Y DATASETS

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import zipfile
import cv2
from skimage import io
import tensorflow as tf
from tensorflow.python.keras import Sequential
from tensorflow.keras import layers, optimizers
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping, ModelCheckpoint, LearningRateScheduler
from IPython.display import display
from tensorflow.keras import backend as K
from sklearn.preprocessing import StandardScaler, normalize
import os
from google.colab import files
%matplotlib inline
```

In [2]:
```
# Deberás montar tu unidad usando los siguientes comandos:
# Para obtener más información sobre el montaje, consulta en: https://stackoverflow.com/questions/46986398/imp...

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

In [3]:
```
# Datos que contienen imágenes con defectos con máscara de segmentación
defect_class_mask_df = pd.read_csv('/content/drive/MyDrive/01_Oriol/01_Machine Learning/06_Data Science For Bus...
```

In [4]:
```
# Datos que contienen imágenes con y sin defectos
all_images_df = pd.read_csv('/content/drive/MyDrive/01_Oriol/01_Machine Learning/06_Data Science For Business-m...
```

In [5]:
```
defect_class_mask_df
```

Out[5]:

	ImageId	ClassId	EncodedPixels
0	d2291de5c.jpg	1	147963 3 148213 9 148461 18 148711 24 148965 2...
1	78416c3d0.jpg	3	54365 3 54621 7 54877 10 55133 12 55388 14 556...
2	2283f2183.jpg	3	201217 43 201473 128 201729 213 201985 5086 20...
3	f0dc068a8.jpg	3	159207 26 159412 77 159617 128 159822 179 1600...
4	00d639396.jpg	3	229356 17 229595 34 229850 36 230105 37 230360...
...	...	...	...
5743	c12842f5e.jpg	3	88 23 342 29 596 34 850 39 1105 44 1361 46 161...
5744	2222a03b3.jpg	3	63332 4 63587 11 63841 20 64096 27 64351 35 64...
5745	b43ea2c01.jpg	1	185024 7 185279 11 185535 12 185790 13 186045 ...
5746	1bc37a6f4.jpg	3	303867 1 304122 3 304376 6 304613 3 304630 9 3...
5747	f4413e172.jpg	3	254911 3 255165 8 255419 12 255672 18 255926 2...

5748 rows × 3 columns

In [6]:
```
all_images_df
```

Out[6]:

	ImageID	label
0	0002cc93b.jpg	1
1	0007a71bf.jpg	1
2	000a4bcdd.jpg	1
3	000f6bf48.jpg	1
4	0014fce06.jpg	1
...	...	...
12992	0482ee1d6.jpg	0
12993	04802a6c2.jpg	0
12994	03ae2bc91.jpg	0
12995	04238d7e3.jpg	0
12996	023353d24.jpg	0

12997 rows × 2 columns

VISUALIZAR Y EXPLORAR EL DATASET

In [7]:
```
defect_class_mask_df['mask'] = defect_class_mask_df['ClassId'].map(lambda x: 1)
```

In [8]:
```
defect_class_mask_df.head(50)
```

Out[8]:

	ImageId	ClassId	EncodedPixels	mask
0	d2291de5c.jpg	1	147963 3 148213 9 148461 18 148711 24 148965 2...	1
1	78416c3d0.jpg	3	54365 3 54621 7 54877 10 55133 12 55388 14 556...	1
2	2283f2183.jpg	3	201217 43 201473 128 201729 213 201985 5086 20...	1
3	f0dc068a8.jpg	3	159207 26 159412 77 159617 128 159822 179 1600...	1
4	00d639396.jpg	3	229356 17 229595 34 229850 36 230105 37 230360...	1
5	17d02873a.jpg	3	254980 43 255236 127 255492 211 255748 253 256...	1
6	47b5ab1bd.jpg	3	128976 8 129230 12 129484 16 129739 23 129995 ...	1
7	a6ecee828.jpg	3	179011 27 179126 73 179259 39 179375 80 179497...	1
8	11aaf18e2.jpg	3	303235 2 303489 7 303743 9 303997 11 304181 2 ...	1
9	cdf669a1f.jpg	4	310246 11 310499 25 310753 28 311007 31 311262...	1
10	fb9558035.jpg	4	159233 1 159489 2 159745 4 160001 5 160257 6 1...	1
11	9fac588ab.jpg	3	68321 32 68513 96 68706 159 68930 191 69186 19...	1
12	83d9b39c8.jpg	3	175089 15 175313 47 175538 78 175762 110 17598...	1
13	749407e33.jpg	3	15704 3 15960 8 16216 13 16471 19 16727 23 169...	1
14	e2bdd4236.jpg	3	17490 175 17746 175 18002 175 18258 175 18514 ...	1
15	8bab4626b.jpg	3	37390 2 37644 5 37898 7 38151 11 38405 13 3865...	1
16	3bde297da.jpg	3	154381 5 154635 17 154889 27 155143 36 155397 ...	1
17	ff5483763.jpg	3	168785 7 169034 20 169284 33 169533 46 169779 ...	1
18	a369c5c1f.jpg	3	18358 11 18606 32 18854 53 19102 73 19225 6 19...	1
19	d62e553a8.jpg	3	11453 1 11709 2 11964 4 12220 5 12475 7 12731 ...	1
20	ceccb1eef.jpg	1	361364 18 361613 42 361862 55 362112 67 362337...	1
21	eda5114ee.jpg	3	38877 2 39129 6 39381 10 39633 14 39885 18 401...	1
22	23c450c03.jpg	1	9251 24 9505 29 9759 32 10013 36 10267 39 1032...	1
23	ab6afa374.jpg	3	65986 39 66165 116 66344 193 66561 232 66817 2...	1
24	a0906d0b3.jpg	4	213842 5 214096 9 214351 11 214605 15 214860 1...	1
25	5562229c3.jpg	3	22966 17 23189 49 23412 82 23636 113 23859 145...	1
26	2365be47a.jpg	3	31096 3 31352 7 31608 12 31863 17 32119 21 323...	1
27	737ae5c95.jpg	4	50890 4 51146 6 51401 8 51657 9 51912 11 52048...	1
28	f89ce1e24.jpg	3	325112 9 325352 25 325592 41 325832 57 326071 ...	1
29	a239718e1.jpg	3	322214 4 322470 12 322726 20 322982 28 323238 ...	1
30	2694c98fb.jpg	3	212692 11 212928 31 213164 51 213400 71 213636...	1
31	a9108753d.jpg	3	3244 4 3494 10 3743 18 3993 24 4245 28 4501 29...	1
32	c4f5ebbb2.jpg	4	229758 5 230006 13 230254 21 230502 29 230750 ...	1
33	75361926d.jpg	4	144404 7 144652 17 144906 20 145160 24 145414 ...	1
34	fc8cb11db.jpg	1	271869 4 272115 14 272358 27 272601 40 272845 ...	1
35	9f054c54f.jpg	1	191060 15 191309 24 191563 29 191818 32 191889...	1
36	faea44200.jpg	3	308123 102 308379 102 308635 102 308891 102 30...	1
37	9b72243dc.jpg	3	207915 9 208167 28 208385 2 208420 44 208641 6...	1
38	10bbf7cb3.jpg	3	324770 1 325024 5 325278 8 325533 11 325787 14...	1
39	d1cd969d5.jpg	3	307684 6 307916 7 307937 11 308167 13 308191 1...	1
40	1082cfe08.jpg	4	240140 9 240395 27 240650 46 240905 64 241160 ...	1
41	927be944d.jpg	1	26369 15 26625 30 26881 30 27137 30 27393 31 2...	1
42	0518e79e9.jpg	3	154369 38 154625 42 154881 46 155137 50 155393...	1
43	64934ac51.jpg	3	357377 28 357633 83 357889 130 358145 169 3584...	1
44	26b0e74fe.jpg	3	156299 6 156545 16 156791 25 157037 35 157283 ...	1
45	7b2257638.jpg	3	77828 64 78084 190 78340 253 78596 253 78852 2...	1
46	975f12b62.jpg	3	185857 11 186113 31 186369 51 186625 72 186881...	1
47	92932546c.jpg	3	53236 13 53468 37 53700 61 53931 86 54163 110 ...	1
48	464a009f9.jpg	3	81550 10 81792 31 82034 52 82277 73 82519 94 8...	1
49	24db6ba0d.jpg	1	217946 7 218143 4 218198 21 218373 2 218394 13...	1

In [9]:
```
plt.figure(figsize=(10,10))
sns.countplot(defect_class_mask_df['ClassId'])
plt.ylabel('Number of images per defect')
plt.xlabel('ClassID')
plt.title('Number of images per class')
```
```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[9]:
```
Text(0.5, 1.0, 'Number of images per class')
```

In [11]:
```
# Algunas imágenes están clasificadas con más de un defecto
# tenemos una imagen con 3 tipos de defectos
# tenemos 272 imágenes con 2 tipos de defectos
# tenemos 5201 imágenes con 1 tipo de defecto
defect_type = defect_class_mask_df.groupby(['ImageId'])['mask'].sum()
```

In [12]:
```
defect_type
```

Out[12]:
```
ImageId
0002cc93b.jpg    1
0007a71bf.jpg    1
000a4bcdd.jpg    1
000f6bf48.jpg    1
0014fce06.jpg    1
                ..
ffcf72ecf.jpg    1
fff02e9c5.jpg    1
fffe98443.jpg    1
ffff4eaa8.jpg    1
ffffd67df.jpg    1
Name: mask, Length: 5474, dtype: int64
```

In [13]:
```
defect_type.value_counts()
```

Out[13]:
```
1    5201
2     272
3       1
Name: mask, dtype: int64
```

In [14]:
```
plt.figure(figsize=(10,10))
sns.barplot(x = defect_type.value_counts().index, y = defect_type.value_counts() )
plt.xlabel('ClassID')
plt.title('Number of defects in image')
```

Out[14]:
```
Text(0.5, 1.0, 'Number of defects in image')
```

In [15]:
```
defect_class_mask_df.shape
```

Out[15]:
```
(5748, 4)
```

In [16]:
```
all_images_df.shape
```

Out[16]:
```
(12997, 2)
```

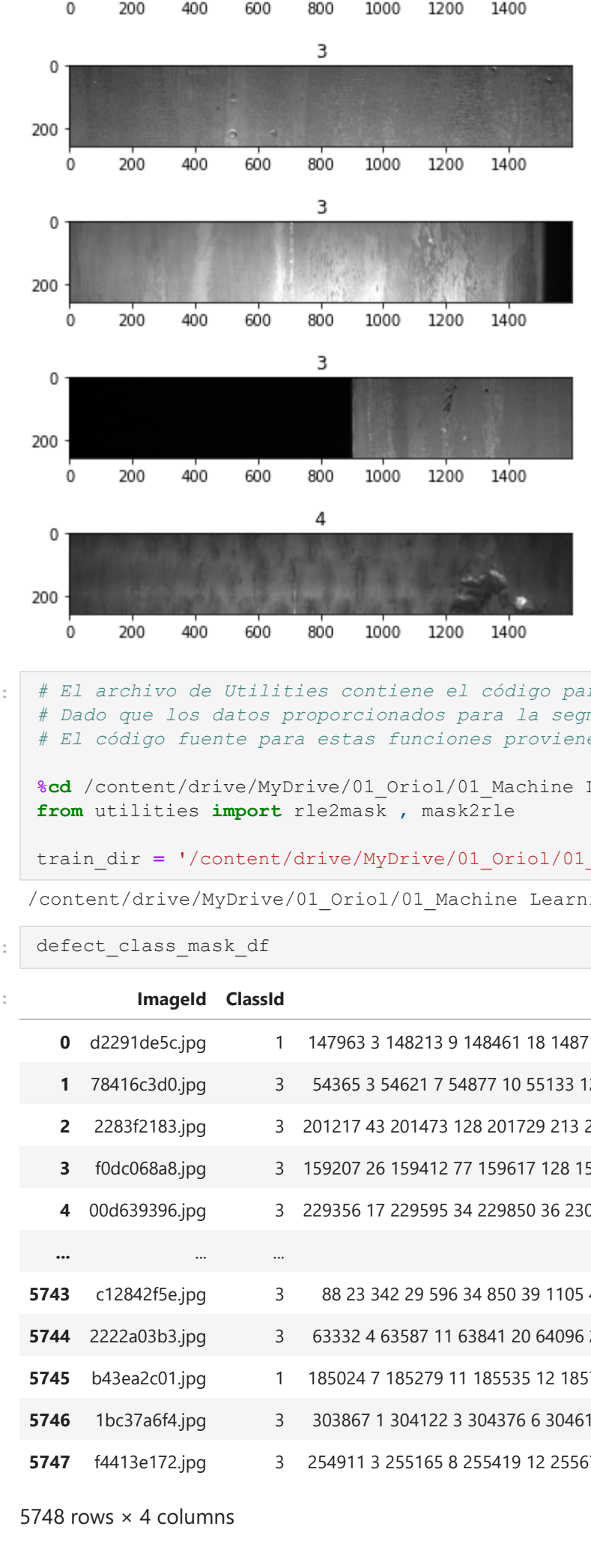
In [18]:
```
# Recuento imágenes con y sin defectos
all_images_df.label.value_counts()
```

Out[18]:
```
1    7095
0    5902
Name: label, dtype: int64
```

In [19]:
```
plt.figure(figsize=(10,10))
sns.barplot(x = all_images_df.label.value_counts().index, y = all_images_df.label.value_counts() )
plt.ylabel('Number of images ')
plt.xlabel('0 - Non-defect             1- Defect')
plt.title('Defect and non-defect images')
```

Out[19]:
```
Text(0.5, 1.0, 'Defect and non-defect images')
```

In [22]:
```
# Visualizar imágenes con defectos junto con sus etiquetas correspondientes
# Las imágenes son de 256 x 1600
from google.colab.patches import cv2_imshow
train_dir = '/content/drive/MyDrive/01_Oriol/01_Machine Learning/06_Data Science For Business-main/6. Maintenan...

for i in range(10):
  # Se especifica el path de las imágenes a partir de su ID de imagen
  img = io.imread(os.path.join(train_dir, defect_class_mask_df.ImageId[i]))
  plt.figure()
  plt.title(defect_class_mask_df.ClassId[i])
  plt.imshow(img)
```



In [24]:
```
# El archivo de Utilities contiene el código para rle2mask, mask2rle, función de pérdida personalizada y generac...
# Dado que los datos proporcionados para la segmentación están en formato RLE (run length encoded), usaremos l...
# El código fuente para estas funciones proviene de: https://www.kaggle.com/paulorzp/rle-functions-run-lenght-...

%cd /content/drive/MyDrive/01_Oriol/01_Machine Learning/06_Data Science For Business-main/6. Maintenance Depart...
from utilities import rle2mask , mask2rle

train_dir = '/content/drive/MyDrive/01_Oriol/01_Machine Learning/06_Data Science For Business-main/6. Maintenan...
```
```
/content/drive/MyDrive/01_Oriol/01_Machine Learning/06_Data Science For Business-main/6. Maintenance Department
```

In [25]:
```
defect_class_mask_df
```

Out[25]:

	ImageId	ClassId	EncodedPixels	mask
0	d2291de5c.jpg	1	147963 3 148213 9 148461 18 148711 24 148965 2...	1
1	78416c3d0.jpg	3	54365 3 54621 7 54877 10 55133 12 55388 14 556...	1
2	2283f2183.jpg	3	201217 43 201473 128 201729 213 201985 5086 20...	1
3	f0dc068a8.jpg	3	159207 26 159412 77 159617 128 159822 179 1600...	1
4	00d639396.jpg	3	229356 17 229595 34 229850 36 230105 37 230360...	1
...	...	...	...	...
5743	c12842f5e.jpg	3	88 23 342 29 596 34 850 39 1105 44 1361 46 161...	1
5744	2222a03b3.jpg	3	63332 4 63587 11 63841 20 64096 27 64351 35 64...	1
5745	b43ea2c01.jpg	1	185024 7 185279 11 185535 12 185790 13 186045 ...	1
5746	1bc37a6f4.jpg	3	303867 1 304122 3 304376 6 304613 3 304630 9 3...	1
5747	f4413e172.jpg	3	254911 3 255165 8 255419 12 255672 18 255926 2...	1

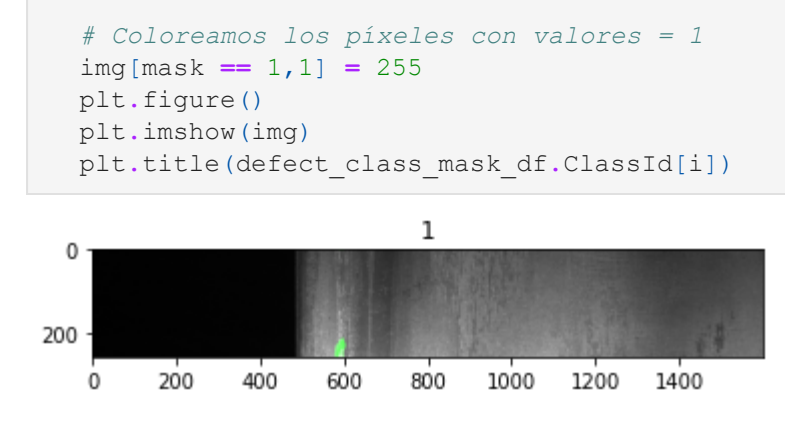
5748 rows × 4 columns

In [26]:
```
# Se prueba rle2mask en un imagen de prueba
image_index = 20 #20 40
mask = rle2mask(defect_class_mask_df.EncodedPixels[image_index], img.shape[0], img.shape[1])
mask.shape
```

Out[26]:
```
(256, 1600)
```

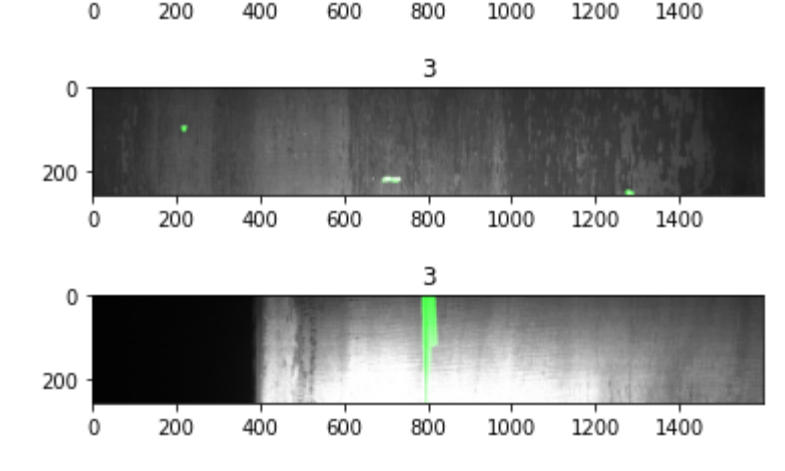
In [27]:
```
# Observamos la máscara
plt.imshow(mask)
```

Out[27]:
```
<matplotlib.image.AxesImage at 0x7f9b22f95710>
```



In [28]:
```
img = io.imread(os.path.join(train_dir, defect_class_mask_df.ImageId[image_index]))
plt.imshow(img)
img.shape
```

Out[28]:
```
(256, 1600, 3)
```



In [30]:
```
for i in range(10):

  # Se leen las imágenes utilizando opencv y las convertimos a formato rgb
  img = io.imread(os.path.join(train_dir, defect_class_mask_df.ImageId[i]))
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

  # Se obtiene la máscara de las imágenes de RLE
  mask = rle2mask(defect_class_mask_df.EncodedPixels[i], img.shape[0], img.shape[1])

  # Coloreamos los píxeles con valores = 1
  img[mask == 1,1] = 255
  plt.figure()
  plt.imshow(img)
  plt.title(defect_class_mask_df.ClassId[i])
```

CONSTRUIR Y ENTRENAR UN MODELO DE DEEP LEARNING PARA DETECTAR PRESENCIA O AUSENCIA DE DEFECTO EN UNA IMAGEN

In [31]:
```
all_images_df
```

Out[31]:

	ImageID	label
0	0002cc93b.jpg	1
1	0007a71bf.jpg	1
2	000a4bcdd.jpg	1
3	000f6bf48.jpg	1
4	0014fce06.jpg	1
...	...	...
12992	0482ee1d6.jpg	0
12993	04802a6c2.jpg	0
12994	03ae2bc91.jpg	0
12995	04238d7e3.jpg	0
12996	023353d24.jpg	0

12997 rows × 2 columns

In [32]:
```
# Dividir el dataset (defective and non defective)  en entrenamiento y testing
from sklearn.model_selection import train_test_split
train, test = train_test_split(all_images_df, test_size=0.15)
```

In [33]:
```
train.shape
```

Out[33]:
```
(11047, 2)
```

In [34]:
```
test.shape
```

Out[34]:
```
(1950, 2)
```

In [35]:
```
train_dir = '/content/drive/MyDrive/01_Oriol/01_Machine Learning/06_Data Science For Business-main/6. Maintenan...
```

In [36]:
```
# Se crea un generador de imágenes para el conjunto de datos de entrenamiento y validación
# Dividiremos los datos en entrenamiento, validación y testing.
# Training = 9390
# validation = 1657
# testing = 1950

from keras_preprocessing.image import ImageDataGenerator

# Crear un generador de datos que escale los datos de 0 a 1 y hacer una división de validación de 0,15
datagen = ImageDataGenerator(rescale=1./255., validation_split = 0.15)

train_generator = datagen.flow_from_dataframe(
dataframe = train,
directory = train_dir,
x_col = "ImageID",
y_col = "label",
subset = "training",
batch_size = 16,
shuffle = True,
class_mode = "other",
target_size = (256, 256))


valid_generator = datagen.flow_from_dataframe(
dataframe = train,
directory = train_dir,
x_col = "ImageID",
y_col = "label",
subset = "validation",
batch_size = 16,
shuffle = True,
class_mode = "other",
target_size = (256, 256))
```
```
Found 9390 validated image filenames.
Found 1657 validated image filenames.
```

In [37]:
```
# Crear un generador de datos para imágenes de test
test_datagen = ImageDataGenerator(rescale=1./255.)

test_generator = test_datagen.flow_from_dataframe(
dataframe = test,
directory = train_dir,
x_col = "ImageID",
y_col = None,
batch_size = 16,
shuffle = False,
class_mode = None,
target_size = (256, 256))
```
```
Found 1950 validated image filenames.
```

In [38]:
```
basemodel = ResNet50(weights = 'imagenet', include_top = False, input_tensor = Input(shape=(256,256,3)))
```
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 1s 0us/step
```

In [39]:
```
basemodel.summary()
```

```
Model: "resnet50"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 256, 256, 3) 0
__________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 262, 262, 3)  0           input_1[0][0]
__________________________________________________________________________________________________
conv1_conv (Conv2D)             (None, 128, 128, 64) 9472        conv1_pad[0][0]
__________________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, 128, 128, 64) 256         conv1_conv[0][0]
__________________________________________________________________________________________________
conv1_relu (Activation)         (None, 128, 128, 64) 0           conv1_bn[0][0]
__________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, 130, 130, 64) 0           conv1_relu[0][0]
__________________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, 64, 64, 64)   0           pool1_pad[0][0]
__________________________________________________________________________________________________
conv2_block1_1_conv (Conv2D)    (None, 64, 64, 64)   4160        pool1_pool[0][0]
__________________________________________________________________________________________________
conv2_block1_1_bn (BatchNormali (None, 64, 64, 64)   256         conv2_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_1_relu (Activation (None, 64, 64, 64)   0           conv2_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_2_conv (Conv2D)    (None, 64, 64, 64)   36928       conv2_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block1_2_bn (BatchNormali (None, 64, 64, 64)   256         conv2_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_2_relu (Activation (None, 64, 64, 64)   0           conv2_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_0_conv (Conv2D)    (None, 64, 64, 256)  16640       pool1_pool[0][0]
__________________________________________________________________________________________________
conv2_block1_3_conv (Conv2D)    (None, 64, 64, 256)  16640       conv2_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block1_0_bn (BatchNormali (None, 64, 64, 256)  1024        conv2_block1_0_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_3_bn (BatchNormali (None, 64, 64, 256)  1024        conv2_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_add (Add)          (None, 64, 64, 256)  0           conv2_block1_0_bn[0][0]
                                                                 conv2_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_out (Activation)   (None, 64, 64, 256)  0           conv2_block1_add[0][0]
__________________________________________________________________________________________________
conv2_block2_1_conv (Conv2D)    (None, 64, 64, 64)   16448       conv2_block1_out[0][0]
__________________________________________________________________________________________________
conv2_block2_1_bn (BatchNormali (None, 64, 64, 64)   256         conv2_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_1_relu (Activation (None, 64, 64, 64)   0           conv2_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_2_conv (Conv2D)    (None, 64, 64, 64)   36928       conv2_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block2_2_bn (BatchNormali (None, 64, 64, 64)   256         conv2_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_2_relu (Activation (None, 64, 64, 64)   0           conv2_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_3_conv (Conv2D)    (None, 64, 64, 256)  16640       conv2_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block2_3_bn (BatchNormali (None, 64, 64, 256)  1024        conv2_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_add (Add)          (None, 64, 64, 256)  0           conv2_block1_out[0][0]
                                                                 conv2_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_out (Activation)   (None, 64, 64, 256)  0           conv2_block2_add[0][0]
__________________________________________________________________________________________________
conv2_block3_1_conv (Conv2D)    (None, 64, 64, 64)   16448       conv2_block2_out[0][0]
__________________________________________________________________________________________________
conv2_block3_1_bn (BatchNormali (None, 64, 64, 64)   256         conv2_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block3_1_relu (Activation (None, 64, 64, 64)   0           conv2_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block3_2_conv (Conv2D)    (None, 64, 64, 64)   36928       conv2_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block3_2_bn (BatchNormali (None, 64, 64, 64)   256         conv2_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block3_2_relu (Activation (None, 64, 64, 64)   0           conv2_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block3_3_conv (Conv2D)    (None, 64, 64, 256)  16640       conv2_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block3_3_bn (BatchNormali (None, 64, 64, 256)  1024        conv2_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block3_add (Add)          (None, 64, 64, 256)  0           conv2_block2_out[0][0]
                                                                 conv2_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block3_out (Activation)   (None, 64, 64, 256)  0           conv2_block3_add[0][0]
__________________________________________________________________________________________________
conv3_block1_1_conv (Conv2D)    (None, 32, 32, 128)  32896       conv2_block3_out[0][0]
__________________________________________________________________________________________________
conv3_block1_1_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block1_1_relu (Activation (None, 32, 32, 128)  0           conv3_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block1_2_conv (Conv2D)    (None, 32, 32, 128)  147584      conv3_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block1_2_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block1_2_relu (Activation (None, 32, 32, 128)  0           conv3_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block1_0_conv (Conv2D)    (None, 32, 32, 512)  131584      conv2_block3_out[0][0]
__________________________________________________________________________________________________
conv3_block1_3_conv (Conv2D)    (None, 32, 32, 512)  66048       conv3_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block1_0_bn (BatchNormali (None, 32, 32, 512)  2048        conv3_block1_0_conv[0][0]
__________________________________________________________________________________________________
conv3_block1_3_bn (BatchNormali (None, 32, 32, 512)  2048        conv3_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block1_add (Add)          (None, 32, 32, 512)  0           conv3_block1_0_bn[0][0]
                                                                 conv3_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block1_out (Activation)   (None, 32, 32, 512)  0           conv3_block1_add[0][0]
__________________________________________________________________________________________________
conv3_block2_1_conv (Conv2D)    (None, 32, 32, 128)  65664       conv3_block1_out[0][0]
__________________________________________________________________________________________________
conv3_block2_1_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block2_1_relu (Activation (None, 32, 32, 128)  0           conv3_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block2_2_conv (Conv2D)    (None, 32, 32, 128)  147584      conv3_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block2_2_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block2_2_relu (Activation (None, 32, 32, 128)  0           conv3_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block2_3_conv (Conv2D)    (None, 32, 32, 512)  66048       conv3_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block2_3_bn (BatchNormali (None, 32, 32, 512)  2048        conv3_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block2_add (Add)          (None, 32, 32, 512)  0           conv3_block1_out[0][0]
                                                                 conv3_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block2_out (Activation)   (None, 32, 32, 512)  0           conv3_block2_add[0][0]
__________________________________________________________________________________________________
conv3_block3_1_conv (Conv2D)    (None, 32, 32, 128)  65664       conv3_block2_out[0][0]
__________________________________________________________________________________________________
conv3_block3_1_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block3_1_relu (Activation (None, 32, 32, 128)  0           conv3_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block3_2_conv (Conv2D)    (None, 32, 32, 128)  147584      conv3_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block3_2_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block3_2_relu (Activation (None, 32, 32, 128)  0           conv3_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block3_3_conv (Conv2D)    (None, 32, 32, 512)  66048       conv3_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block3_3_bn (BatchNormali (None, 32, 32, 512)  2048        conv3_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block3_add (Add)          (None, 32, 32, 512)  0           conv3_block2_out[0][0]
                                                                 conv3_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block3_out (Activation)   (None, 32, 32, 512)  0           conv3_block3_add[0][0]
__________________________________________________________________________________________________
conv3_block4_1_conv (Conv2D)    (None, 32, 32, 128)  65664       conv3_block3_out[0][0]
__________________________________________________________________________________________________
conv3_block4_1_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block4_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block4_1_relu (Activation (None, 32, 32, 128)  0           conv3_block4_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block4_2_conv (Conv2D)    (None, 32, 32, 128)  147584      conv3_block4_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block4_2_bn (BatchNormali (None, 32, 32, 128)  512         conv3_block4_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block4_2_relu (Activation (None, 32, 32, 128)  0           conv3_block4_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block4_3_conv (Conv2D)    (None, 32, 32, 512)  66048       conv3_block4_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block4_3_bn (BatchNormali (None, 32, 32, 512)  2048        conv3_block4_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block4_add (Add)          (None, 32, 32, 512)  0           conv3_block3_out[0][0]
                                                                 conv3_block4_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block4_out (Activation)   (None, 32, 32, 512)  0           conv3_block4_add[0][0]
__________________________________________________________________________________________________
conv4_block1_1_conv (Conv2D)    (None, 16, 16, 256)  131328      conv3_block4_out[0][0]
__________________________________________________________________________________________________
conv4_block1_1_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_1_relu (Activation (None, 16, 16, 256)  0           conv4_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block1_2_conv (Conv2D)    (None, 16, 16, 256)  590080      conv4_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block1_2_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_2_relu (Activation (None, 16, 16, 256)  0           conv4_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block1_0_conv (Conv2D)    (None, 16, 16, 1024) 525312      conv3_block4_out[0][0]
__________________________________________________________________________________________________
conv4_block1_3_conv (Conv2D)    (None, 16, 16, 1024) 263168      conv4_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block1_0_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block1_0_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_3_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_add (Add)          (None, 16, 16, 1024) 0           conv4_block1_0_bn[0][0]
                                                                 conv4_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block1_out (Activation)   (None, 16, 16, 1024) 0           conv4_block1_add[0][0]
__________________________________________________________________________________________________
conv4_block2_1_conv (Conv2D)    (None, 16, 16, 256)  262400      conv4_block1_out[0][0]
__________________________________________________________________________________________________
conv4_block2_1_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block2_1_relu (Activation (None, 16, 16, 256)  0           conv4_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block2_2_conv (Conv2D)    (None, 16, 16, 256)  590080      conv4_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block2_2_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block2_2_relu (Activation (None, 16, 16, 256)  0           conv4_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block2_3_conv (Conv2D)    (None, 16, 16, 1024) 263168      conv4_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block2_3_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block2_add (Add)          (None, 16, 16, 1024) 0           conv4_block1_out[0][0]
                                                                 conv4_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block2_out (Activation)   (None, 16, 16, 1024) 0           conv4_block2_add[0][0]
__________________________________________________________________________________________________
conv4_block3_1_conv (Conv2D)    (None, 16, 16, 256)  262400      conv4_block2_out[0][0]
__________________________________________________________________________________________________
conv4_block3_1_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block3_1_relu (Activation (None, 16, 16, 256)  0           conv4_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block3_2_conv (Conv2D)    (None, 16, 16, 256)  590080      conv4_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block3_2_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block3_2_relu (Activation (None, 16, 16, 256)  0           conv4_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block3_3_conv (Conv2D)    (None, 16, 16, 1024) 263168      conv4_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block3_3_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block3_add (Add)          (None, 16, 16, 1024) 0           conv4_block2_out[0][0]
                                                                 conv4_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block3_out (Activation)   (None, 16, 16, 1024) 0           conv4_block3_add[0][0]
__________________________________________________________________________________________________
conv4_block4_1_conv (Conv2D)    (None, 16, 16, 256)  262400      conv4_block3_out[0][0]
__________________________________________________________________________________________________
conv4_block4_1_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block4_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block4_1_relu (Activation (None, 16, 16, 256)  0           conv4_block4_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block4_2_conv (Conv2D)    (None, 16, 16, 256)  590080      conv4_block4_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block4_2_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block4_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block4_2_relu (Activation (None, 16, 16, 256)  0           conv4_block4_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block4_3_conv (Conv2D)    (None, 16, 16, 1024) 263168      conv4_block4_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block4_3_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block4_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block4_add (Add)          (None, 16, 16, 1024) 0           conv4_block3_out[0][0]
                                                                 conv4_block4_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block4_out (Activation)   (None, 16, 16, 1024) 0           conv4_block4_add[0][0]
__________________________________________________________________________________________________
conv4_block5_1_conv (Conv2D)    (None, 16, 16, 256)  262400      conv4_block4_out[0][0]
__________________________________________________________________________________________________
conv4_block5_1_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block5_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block5_1_relu (Activation (None, 16, 16, 256)  0           conv4_block5_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block5_2_conv (Conv2D)    (None, 16, 16, 256)  590080      conv4_block5_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block5_2_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block5_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block5_2_relu (Activation (None, 16, 16, 256)  0           conv4_block5_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block5_3_conv (Conv2D)    (None, 16, 16, 1024) 263168      conv4_block5_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block5_3_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block5_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block5_add (Add)          (None, 16, 16, 1024) 0           conv4_block4_out[0][0]
                                                                 conv4_block5_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block5_out (Activation)   (None, 16, 16, 1024) 0           conv4_block5_add[0][0]
__________________________________________________________________________________________________
conv4_block6_1_conv (Conv2D)    (None, 16, 16, 256)  262400      conv4_block5_out[0][0]
__________________________________________________________________________________________________
conv4_block6_1_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block6_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block6_1_relu (Activation (None, 16, 16, 256)  0           conv4_block6_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block6_2_conv (Conv2D)    (None, 16, 16, 256)  590080      conv4_block6_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block6_2_bn (BatchNormali (None, 16, 16, 256)  1024        conv4_block6_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block6_2_relu (Activation (None, 16, 16, 256)  0           conv4_block6_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block6_3_conv (Conv2D)    (None, 16, 16, 1024) 263168      conv4_block6_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block6_3_bn (BatchNormali (None, 16, 16, 1024) 4096        conv4_block6_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block6_add (Add)          (None, 16, 16, 1024) 0           conv4_block5_out[0][0]
                                                                 conv4_block6_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block6_out (Activation)   (None, 16, 16, 1024) 0           conv4_block6_add[0][0]
__________________________________________________________________________________________________
conv5_block1_1_conv (Conv2D)    (None, 8, 8, 512)    524800      conv4_block6_out[0][0]
__________________________________________________________________________________________________
conv5_block1_1_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block1_1_relu (Activation (None, 8, 8, 512)    0           conv5_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block1_2_conv (Conv2D)    (None, 8, 8, 512)    2359808     conv5_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block1_2_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block1_2_relu (Activation (None, 8, 8, 512)    0           conv5_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block1_0_conv (Conv2D)    (None, 8, 8, 2048)   2099200     conv4_block6_out[0][0]
__________________________________________________________________________________________________
conv5_block1_3_conv (Conv2D)    (None, 8, 8, 2048)   1050624     conv5_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block1_0_bn (BatchNormali (None, 8, 8, 2048)   8192        conv5_block1_0_conv[0][0]
__________________________________________________________________________________________________
conv5_block1_3_bn (BatchNormali (None, 8, 8, 2048)   8192        conv5_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block1_add (Add)          (None, 8, 8, 2048)   0           conv5_block1_0_bn[0][0]
                                                                 conv5_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block1_out (Activation)   (None, 8, 8, 2048)   0           conv5_block1_add[0][0]
__________________________________________________________________________________________________
conv5_block2_1_conv (Conv2D)    (None, 8, 8, 512)    1049088     conv5_block1_out[0][0]
__________________________________________________________________________________________________
conv5_block2_1_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block2_1_relu (Activation (None, 8, 8, 512)    0           conv5_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_2_conv (Conv2D)    (None, 8, 8, 512)    2359808     conv5_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block2_2_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block2_2_relu (Activation (None, 8, 8, 512)    0           conv5_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_3_conv (Conv2D)    (None, 8, 8, 2048)   1050624     conv5_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block2_3_bn (BatchNormali (None, 8, 8, 2048)   8192        conv5_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block2_add (Add)          (None, 8, 8, 2048)   0           conv5_block1_out[0][0]
                                                                 conv5_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_out (Activation)   (None, 8, 8, 2048)   0           conv5_block2_add[0][0]
__________________________________________________________________________________________________
conv5_block3_1_conv (Conv2D)    (None, 8, 8, 512)    1049088     conv5_block2_out[0][0]
__________________________________________________________________________________________________
conv5_block3_1_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_1_relu (Activation (None, 8, 8, 512)    0           conv5_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_2_conv (Conv2D)    (None, 8, 8, 512)    2359808     conv5_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_2_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_2_relu (Activation (None, 8, 8, 512)    0           conv5_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_3_conv (Conv2D)    (None, 8, 8, 2048)   1050624     conv5_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, 8, 8, 2048)   8192        conv5_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_add (Add)          (None, 8, 8, 2048)   0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_out (Activation)   (None, 8, 8, 2048)   0           conv5_block3_add[0][0]
==================================================================================================
Total params: 23,587,712
Trainable params: 23,534,592
Non-trainable params: 53,120
__________________________________________________________________________________________________
```

In [40]:
```
# Se congelan los pesos del modelo
for layer in basemodel.layers:
  layers.trainable = False
```

In [41]:
```
headmodel = basemodel.output
headmodel = AveragePooling2D(pool_size = (4,4))(headmodel)
headmodel = Flatten(name= 'flatten')(headmodel)
headmodel = Dense(256, activation = "relu")(headmodel)
headmodel = Dropout(0.3)(headmodel)
headmodel = Dense(1, activation = 'sigmoid')(headmodel)

model = Model(inputs = basemodel.input, outputs = headmodel)
```

In [42]:
```
model.compile(loss = 'binary_crossentropy', optimizer='Nadam', metrics= ["accuracy"])
```

In [43]:
```
# Se usa la parada temprana para parar el entrenamiento si la pérdida en validación no baja después de un ciert
earlystopping = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)

# Guardamos el modelo con menor error de validación
checkpointer = ModelCheckpoint(filepath="resnet-weights.hdf5", verbose=1, save_best_only=True)
```

In [ ]:
```
history = model.fit_generator(train_generator, steps_per_epoch= train_generator.n // 16, epochs = 40, validatio
```

```
WARNING:tensorflow:From <ipython-input-38-0017a398a809>:2: Model.fit_generator (from tensorflow.python.keras.en
gine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/40
586/586 [==============================] - ETA: 0s - loss: 0.5105 - accuracy: 0.7823
Epoch 00001: val_loss improved from inf to 1.10059, saving model to resnet-weights.hdf5
586/586 [==============================] - 6741s 12s/step - loss: 0.5105 - accuracy: 0.7823 - val_loss: 1.1006
- val_accuracy: 0.4478
Epoch 2/40
586/586 [==============================] - ETA: 0s - loss: 0.3659 - accuracy: 0.8392
Epoch 00002: val_loss improved from 1.10059 to 0.47946, saving model to resnet-weights.hdf5
586/586 [==============================] - 153s 262ms/step - loss: 0.3659 - accuracy: 0.8392 - val_loss: 0.4795
- val_accuracy: 0.7743
Epoch 3/40
586/586 [==============================] - ETA: 0s - loss: 0.3148 - accuracy: 0.8667
Epoch 00003: val_loss improved from 0.47946 to 0.42682, saving model to resnet-weights.hdf5
586/586 [==============================] - 154s 262ms/step - loss: 0.3148 - accuracy: 0.8667 - val_loss: 0.4268
- val_accuracy: 0.8004
Epoch 4/40
586/586 [==============================] - ETA: 0s - loss: 0.2766 - accuracy: 0.8883
Epoch 00004: val_loss did not improve from 0.42682
586/586 [==============================] - 149s 255ms/step - loss: 0.2766 - accuracy: 0.8883 - val_loss: 0.7513
- val_accuracy: 0.6566
Epoch 5/40
586/586 [==============================] - ETA: 0s - loss: 0.2442 - accuracy: 0.8972
Epoch 00005: val_loss did not improve from 0.42682
586/586 [==============================] - 149s 254ms/step - loss: 0.2442 - accuracy: 0.8972 - val_loss: 0.4715
- val_accuracy: 0.8313
Epoch 6/40
586/586 [==============================] - ETA: 0s - loss: 0.2219 - accuracy: 0.9121
Epoch 00006: val_loss did not improve from 0.42682
586/586 [==============================] - 150s 255ms/step - loss: 0.2219 - accuracy: 0.9121 - val_loss: 0.6994
- val_accuracy: 0.7894
Epoch 7/40
586/586 [==============================] - ETA: 0s - loss: 0.2144 - accuracy: 0.9175
Epoch 00007: val_loss did not improve from 0.42682
586/586 [==============================] - 148s 253ms/step - loss: 0.2144 - accuracy: 0.9175 - val_loss: 1.2568
- val_accuracy: 0.6911
Epoch 8/40
586/586 [==============================] - ETA: 0s - loss: 0.1845 - accuracy: 0.9250
Epoch 00008: val_loss did not improve from 0.42682
586/586 [==============================] - 146s 249ms/step - loss: 0.1845 - accuracy: 0.9250 - val_loss: 0.6313
- val_accuracy: 0.8186
Epoch 9/40
586/586 [==============================] - ETA: 0s - loss: 0.1748 - accuracy: 0.9316
Epoch 00009: val_loss did not improve from 0.42682
586/586 [==============================] - 147s 251ms/step - loss: 0.1748 - accuracy: 0.9316 - val_loss: 0.7022
- val_accuracy: 0.7676
Epoch 10/40
586/586 [==============================] - ETA: 0s - loss: 0.1606 - accuracy: 0.9384
Epoch 00010: val_loss did not improve from 0.42682
586/586 [==============================] - 148s 252ms/step - loss: 0.1606 - accuracy: 0.9384 - val_loss: 1.9122
- val_accuracy: 0.5789
Epoch 11/40
586/586 [==============================] - ETA: 0s - loss: 0.1526 - accuracy: 0.9411
Epoch 00011: val_loss did not improve from 0.42682
586/586 [==============================] - 144s 246ms/step - loss: 0.1526 - accuracy: 0.9411 - val_loss: 1.2133
- val_accuracy: 0.7306
Epoch 12/40
586/586 [==============================] - ETA: 0s - loss: 0.1321 - accuracy: 0.9487
Epoch 00012: val_loss did not improve from 0.42682
586/586 [==============================] - 143s 244ms/step - loss: 0.1321 - accuracy: 0.9487 - val_loss: 0.4608
- val_accuracy: 0.8374
Epoch 13/40
586/586 [==============================] - ETA: 0s - loss: 0.1285 - accuracy: 0.9519
Epoch 00013: val_loss did not improve from 0.42682
586/586 [==============================] - 145s 248ms/step - loss: 0.1285 - accuracy: 0.9519 - val_loss: 3.5264
- val_accuracy: 0.6141
Epoch 14/40
586/586 [==============================] - ETA: 0s - loss: 0.1322 - accuracy: 0.9490
Epoch 00014: val_loss did not improve from 0.42682
586/586 [==============================] - 145s 247ms/step - loss: 0.1322 - accuracy: 0.9490 - val_loss: 0.7503
- val_accuracy: 0.7549
Epoch 15/40
586/586 [==============================] - ETA: 0s - loss: 0.1028 - accuracy: 0.9591
Epoch 00015: val_loss did not improve from 0.42682
586/586 [==============================] - 143s 244ms/step - loss: 0.1028 - accuracy: 0.9591 - val_loss: 1.3928
- val_accuracy: 0.6426
Epoch 16/40
586/586 [==============================] - ETA: 0s - loss: 0.0977 - accuracy: 0.9642
Epoch 00016: val_loss did not improve from 0.42682
586/586 [==============================] - 145s 247ms/step - loss: 0.0977 - accuracy: 0.9642 - val_loss: 4.6727
- val_accuracy: 0.6481
Epoch 17/40
586/586 [==============================] - ETA: 0s - loss: 0.0788 - accuracy: 0.9703
Epoch 00017: val_loss did not improve from 0.42682
586/586 [==============================] - 143s 244ms/step - loss: 0.0788 - accuracy: 0.9703 - val_loss: 0.5990
- val_accuracy: 0.8368
Epoch 18/40
586/586 [==============================] - ETA: 0s - loss: 0.0813 - accuracy: 0.9699
Epoch 00018: val_loss did not improve from 0.42682
586/586 [==============================] - 143s 243ms/step - loss: 0.0813 - accuracy: 0.9699 - val_loss: 3.2607
- val_accuracy: 0.6408
Epoch 19/40
586/586 [==============================] - ETA: 0s - loss: 0.0798 - accuracy: 0.9707
Epoch 00019: val_loss did not improve from 0.42682
586/586 [==============================] - 143s 243ms/step - loss: 0.0798 - accuracy: 0.9707 - val_loss: 2.5021
- val_accuracy: 0.7427
Epoch 20/40
586/586 [==============================] - ETA: 0s - loss: 0.0945 - accuracy: 0.9670
Epoch 00020: val_loss did not improve from 0.42682
586/586 [==============================] - 142s 242ms/step - loss: 0.0945 - accuracy: 0.9670 - val_loss: 1.4716
- val_accuracy: 0.8022
Epoch 21/40
586/586 [==============================] - ETA: 0s - loss: 0.0596 - accuracy: 0.9787
Epoch 00021: val_loss did not improve from 0.42682
586/586 [==============================] - 141s 241ms/step - loss: 0.0596 - accuracy: 0.9787 - val_loss: 1.3433
- val_accuracy: 0.7816
Epoch 22/40
586/586 [==============================] - ETA: 0s - loss: 0.0570 - accuracy: 0.9806
Epoch 00022: val_loss did not improve from 0.42682
586/586 [==============================] - 140s 239ms/step - loss: 0.0570 - accuracy: 0.9806 - val_loss: 0.6523
- val_accuracy: 0.8568
Epoch 23/40
586/586 [==============================] - ETA: 0s - loss: 0.0609 - accuracy: 0.9787
Epoch 00023: val_loss did not improve from 0.42682
586/586 [==============================] - 141s 241ms/step - loss: 0.0609 - accuracy: 0.9787 - val_loss: 0.8560
- val_accuracy: 0.7985
Epoch 00023: early stopping
```

In [44]:
```
# Guardamos la arquitectura del modelo entrenado para el futuro

model_json = model.to_json()
with open("resnet-classifier-model.json","w") as json_file:
  json_file.write(model_json)
```

EVALUAR LA EFICACIA DEL MODELO

In [45]:
```
with open('/content/drive/MyDrive/01_Oriol/01_Machine Learning/06_Data Science For Business-main/6. Maintenance
    json_savedModel= json_file.read()
# Se carga el modelo
model = tf.keras.models.model_from_json(json_savedModel)
model.load_weights('/content/drive/MyDrive/01_Oriol/01_Machine Learning/06_Data Science For Business-main/6. Ma
model.compile(loss = 'binary_crossentropy', optimizer='Nadam', metrics= ["accuracy"])
```

In [47]:
```
# Se efectúa la predicción

from keras_preprocessing.image import ImageDataGenerator

test_predict = model.predict(test_generator, steps = test_generator.n // 16, verbose =1)
```

```
  1/121 [..............................] - ETA: 0sWARNING:tensorflow:Callbacks method `on_predict_batch_end` is
slow compared to the batch time (batch time: 0.0126s vs `on_predict_batch_end` time: 0.0520s). Check your callb
acks.
121/121 [==============================] - 564s 5s/step
```

In [48]:
```
test_predict
```

Out[48]:
```
array([[1.4639436e-02],
       [9.9567324e-01],
       [9.9938178e-01],
       ...,
       [4.3552849e-01],
       [1.5520183e-05],
       [9.9994826e-01]], dtype=float32)
```

In [49]:
```
# Se hace uso de la función de activación sigmoidea, obteniendo el resultado con valores continuos de 0 a 1.
# La red se usa inicialmente para clasificar si la imagen tiene defecto o no
# Luego, estas imágenes (defectuosas) se pasan por la red de segmentación para obtener la ubicación y el tipo d
# Vamos a elegir 0.01, para asegurarnos de que omitimos las imágenes para que no pasen por la red de segmentaci
# estemos muy seguros de que no tiene ningún defecto y si no estamos seguros, podemos pasar esta imagen a trave


predict = []

for i in test_predict:
  if i < 0.01: #0.5
    predict.append(0)
  else:
    predict.append(1)

predict = np.asarray(predict)
```

In [50]:
```
len(predict)
```

Out[50]: 1936

In [51]:
```
# como se usa el generador de prueba, limito las imágenes a 1936, debido al tamaño del lote

original = np.asarray(test.label)[:1936]
len(original)
```

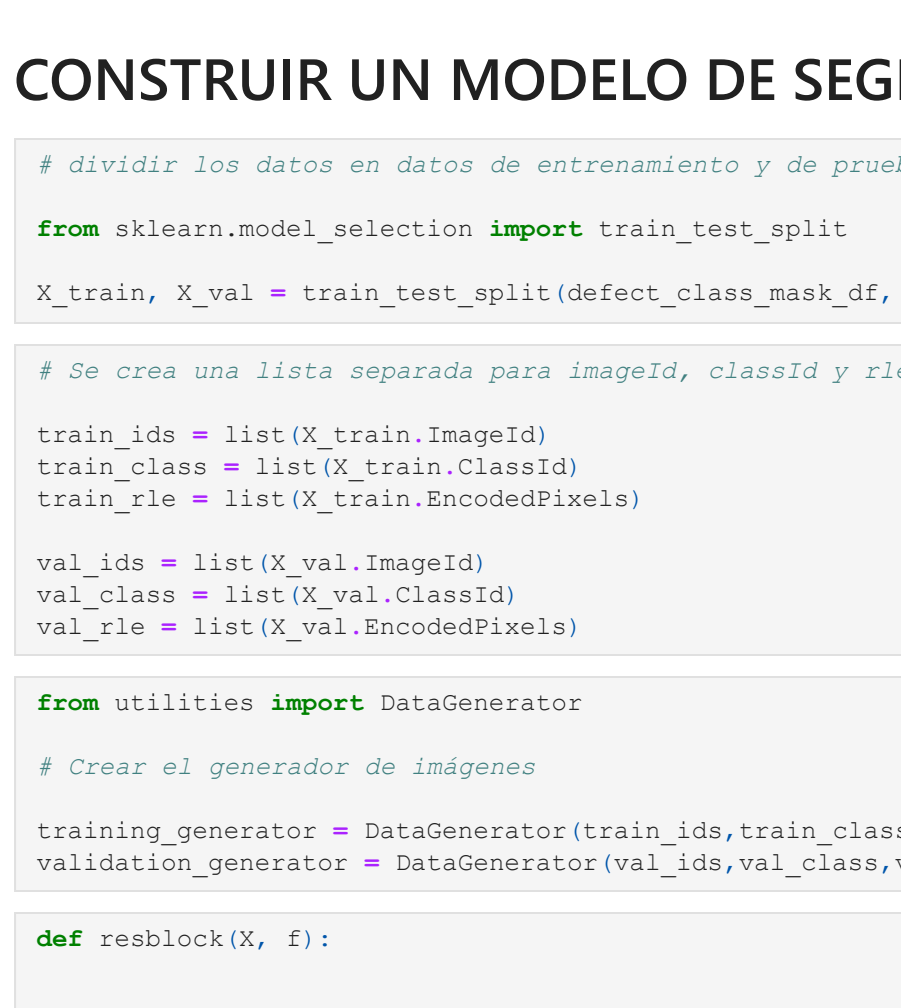
Out[51]: 1936

In [52]:
```
# Se busca la accuracy del modelo
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(original, predict)
accuracy
```

Out[52]: 0.8868801652892562

In [53]:
```
# Se representa la matriz de confusión
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(original, predict)
plt.figure(figsize = (7,7))
sns.heatmap(cm, annot=True)
```

Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3b2356c3c8>



In [54]:
```
# Reporte de clasificación

from sklearn.metrics import classification_report

report = classification_report(original,predict, labels = [0,1])
print(report)
```

```
              precision    recall  f1-score   support

           0       1.00      0.75      0.86       879
           1       0.83      1.00      0.91      1057

    accuracy                           0.89      1936
   macro avg       0.91      0.88      0.88      1936
weighted avg       0.91      0.89      0.88      1936
```

CONSTRUIR UN MODELO DE SEGMENTACIÓN CON RESUNET

In [55]:
```
# dividir los datos en datos de entrenamiento y de prueba

from sklearn.model_selection import train_test_split

X_train, X_val = train_test_split(defect_class_mask_df, test_size=0.2)
```

In [56]:
```
# Se crea una lista separada para imageId, classId y rle para pasar al generador

train_ids = list(X_train.ImageId)
train_class = list(X_train.ClassId)
train_rle = list(X_train.EncodedPixels)

val_ids = list(X_val.ImageId)
val_class = list(X_val.ClassId)
val_rle = list(X_val.EncodedPixels)
```

In [57]:
```
from utilities import DataGenerator

# Crear el generador de imágenes

training_generator = DataGenerator(train_ids,train_class, train_rle, train_dir)
validation_generator = DataGenerator(val_ids,val_class,val_rle, train_dir)
```

In [58]:
```
def resblock(X, f):


  # Hacemos una copia de la entrada
  X_copy = X

  # Path principal

  X = Conv2D(f, kernel_size = (1,1), strides = (1,1), kernel_initializer ='he_normal')(X)
  X = BatchNormalization()(X)
  X = Activation('relu')(X)

  X = Conv2D(f, kernel_size = (3,3), strides =(1,1), padding = 'same', kernel_initializer ='he_normal')(X)
  X = BatchNormalization()(X)

  # Path corto

  X_copy = Conv2D(f, kernel_size = (1,1), strides =(1,1), kernel_initializer ='he_normal')(X_copy)
  X_copy = BatchNormalization()(X_copy)

  # Añadimos el fichero de salida a partir de la combinación de path principal y el corto

  X = Add()([X,X_copy])
  X = Activation('relu')(X)

  return X
```

In [59]:
```
# Función para upscale y concatenar los valores

def upsample_concat(x, skip):
  x = UpSampling2D((2,2))(x)
  merge = Concatenate()([x, skip])

  return merge
```

In [60]:
```
input_shape = (256,256,1)

#Input tensor shape
X_input = Input(input_shape)

#Stage 1
conv1_in = Conv2D(16,3,activation= 'relu', padding = 'same', kernel_initializer ='he_normal')(X_input)
conv1_in = BatchNormalization()(conv1_in)
conv1_in = Conv2D(16,3,activation= 'relu', padding = 'same', kernel_initializer ='he_normal')(conv1_in)
conv1_in = BatchNormalization()(conv1_in)
pool_1 = MaxPool2D(pool_size = (2,2))(conv1_in)

#Stage 2
conv2_in = resblock(pool_1, 32)
pool_2 = MaxPool2D(pool_size = (2,2))(conv2_in)

#Stage 3
conv3_in = resblock(pool_2, 64)
pool_3 = MaxPool2D(pool_size = (2,2))(conv3_in)

#Stage 4
conv4_in = resblock(pool_3, 128)
pool_4 = MaxPool2D(pool_size = (2,2))(conv4_in)

#Stage 5
conv5_in = resblock(pool_4, 256)

#Upscale stage 1
up_1 = upsample_concat(conv5_in, conv4_in)
up_1 = resblock(up_1, 128)

#Upscale stage 2
up_2 = upsample_concat(up_1, conv3_in)
up_2 = resblock(up_2, 64)

#Upscale stage 3
up_3 = upsample_concat(up_2, conv2_in)
up_3 = resblock(up_3, 32)

#Upscale stage 4
up_4 = upsample_concat(up_3, conv1_in)
up_4 = resblock(up_4, 16)

#Final Output
output = Conv2D(4, (1,1), padding = "same", activation = "sigmoid")(up_4)

model_seg = Model(inputs = X_input, outputs = output )
```

Función de pérdidas:

Necesitamos una función de pérdida personalizada para entrenar a este ResUNet: https://github.com/nabsabraham/focal-tversky-unet/blob/master/losses.py

@article{focal-unet, title={A novel Focal Tversky loss function with improved Attention U-Net for lesion segmentation}, author={Abraham, Nabila and Khan, Naimul Mefraz}, journal={arXiv preprint arXiv:1810.07842}, year={2018} }

In [61]:
```
%cd /content/drive/MyDrive/01_Oriol/01_Machine Learning/06_Data Science For Business-main/6. Maintenance Depart

from utilities import focal_tversky, tversky_loss, tversky
```

```
/content/drive/MyDrive/01_Oriol/01_Machine Learning/06_Data Science For Business-main/6. Maintenance Department
```

In [62]:
```
adam = tf.keras.optimizers.Adam(lr = 0.05, epsilon = 0.1)
model_seg.compile(optimizer = adam, loss = focal_tversky, metrics = [tversky])
```

In [63]:
```
# Se usa la parada temprana para salir del entrenamiento si la pérdida de validación no disminuye incluso despu
earlystopping = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)
# guardar el mejor modelo con menor pérdida de validación
checkpointer = ModelCheckpoint(filepath="resnet-segmentation-weights.hdf5", verbose=1, save_best_only=True)
```

In [ ]:
```
history = model_seg.fit_generator(training_generator, epochs = 40, validation_data= validation_generator, calli
```

In [ ]:
```
# Guardamos el modelo para uso futuro
model_json = model_seg.to_json()
with open("resnet-segmentation-model.json","w") as json_file:
  json_file.write(model_json)
```

EVALUAR LA EFICACIA DEL MODELO DE SEGMENTACIÓN ENTRENADO

In [66]:
```
from utilities import focal_tversky, tversky_loss, tversky

with open('resnet-segmentation-model.json', 'r') as json_file:
    json_savedModel= json_file.read()

# Cargar el modelo
model_seg = tf.keras.models.model_from_json(json_savedModel)
model_seg.load_weights('weights_seg.hdf5')
adam = tf.keras.optimizers.Adam(lr = 0.05, epsilon = 0.1)
model_seg.compile(optimizer = adam, loss = focal_tversky, metrics = [tversky])
```

In [67]:
```
# Datos del conjunto de test para la tarea de segmentación

test_df = pd.read_csv('test.csv')
```

In [68]:
```
test_df
```

Out[68]:

	ImageId	ClassId	EncodedPixels
0	0ca915b9f.jpg	3	188383 3 188637 5 188892 6 189148 5 1894...
1	7773445b7.jpg	3	75789 33 76045 97 76300 135 76556 143 76...
2	5e0744d4b.jpg	3	120323 91 120579 182 120835 181 121091 1...
3	6ccde604d.jpg	3	295905 32 296098 95 296290 159 296483 22...
4	16aabaf79.jpg	1	352959 24 353211 28 353465 31 353719 33 3...
...	...	...	...
633	a4334d7da.jpg	4	11829 7 12073 20 12317 32 12566 40 12821 41 13...
634	418e47222.jpg	3	46340 43 46596 127 46852 211 47108 253 47364 2...
635	817a545aa.jpg	3	206529 64 206657 4518 211201 179 211457 128 21...
636	caad490a5.jpg	3	59631 10 59887 30 60103 50 60339 69 60585 79 6...
637	a5e9195b6.jpg	3	321 51 424 43 577 51 641 82 833 51 897 82 1089...

638 rows × 3 columns

In [69]:
```
test_df.ImageId
```

Out[69]:
```
0      0ca915b9f.jpg
1      7773445b7.jpg
2      5e0744d4b.jpg
3      6ccde604d.jpg
4      16aabaf79.jpg
           ...
633    a4334d7da.jpg
634    418e47222.jpg
635    817a545aa.jpg
636    caad490a5.jpg
637    a5e9195b6.jpg
Name: ImageId, Length: 638, dtype: object
```

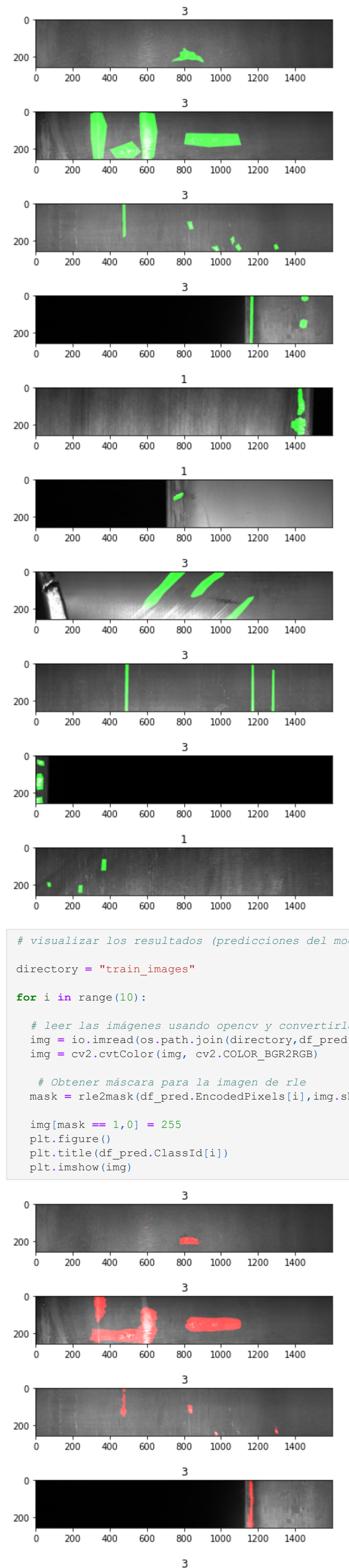
In [70]:
```
from utilities import prediction

# Hacemos la predicción
image_id, defect_type, mask = prediction(test_df, model, model_seg)
```

In [71]:
```
# Se crea el dataframe para el resultado
df_pred= pd.DataFrame({'ImageId': image_id,'EncodedPixels': mask,'ClassId': defect_type})
df_pred.head()
```

Out[71]:

	ImageId	EncodedPixels	ClassId
0	0ca915b9f.jpg	151421 1 151423 2 151677 1 151679 2 151933 1 1...	3
1	7773445b7.jpg	72927 2 73183 2 73439 2 73695 2 73951 2 74207 ...	3
2	5e0744d4b.jpg	116095 2 116351 2 116607 2 116863 2 117119 2 1...	3
3	6ccde604d.jpg	290305 4 290561 4 290817 4 291073 4 291329 4 2...	3
4	16aabaf79.jpg	352937 24 353193 24 353449 24 353705 24 353961...	3

In [72]:
```
# Se muestran las imágenes junto con sus máscaras originales (ground truth)
for i in range(10):

  # leer las imágenes usando opencv y convertirlas a formato rgb
  img = io.imread(os.path.join(train_dir,test_df.ImageId[i]))
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

  # Obtener máscara para la imagen de rle
  mask = rle2mask(test_df.EncodedPixels[i],img.shape[0],img.shape[1])

  img[mask == 1] = 255
  plt.figure()
  plt.title(test_df.ClassId[i])
  plt.imshow(img)
```



In [74]:
```python
# visualizar los resultados (predicciones del modelo)

directory = "train_images"

for i in range(10):

  # leer las imágenes usando opencv y convertirlas a formato rgb
  img = io.imread(os.path.join(directory,df_pred.ImageId[i]))
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

   # Obtener máscara para la imagen de rle
  mask = rle2mask(df_pred.EncodedPixels[i],img.shape[0],img.shape[1])

  img[mask == 1,0] = 255
  plt.figure()
  plt.title(df_pred.ClassId[i])
  plt.imshow(img)
```