In [ ]:
```
import numpy as np
import keras
keras.__version__
import tensorflow
from tensorflow.keras.applications.resnet50 import ResNet50
#from tf.keras.applications.resnet50 import ResNet50
```

In [ ]:
```
import os
data_folder_path = 'G:\Mi unidad\Demat\8 semestre\ML\Tarea 4\Datos\kaggle_3m'
data_folders = os.listdir(data_folder_path)  # leo las carpetas de los datos.

data_folders.remove('data.csv')  # Quitamos lo que son datos
data_folders.remove('README.md')
```

In [ ]:
```
len(data_folders)
```

Out[ ]:
```
110
```

In [ ]:
```
#os.mkdir('imagenes') #creo las carpetas imagenes y masks
#os.mkdir('mascaras')
```

In [ ]:
```
images_folder = './imagenes/'
masks_folder = './mascaras/'
```

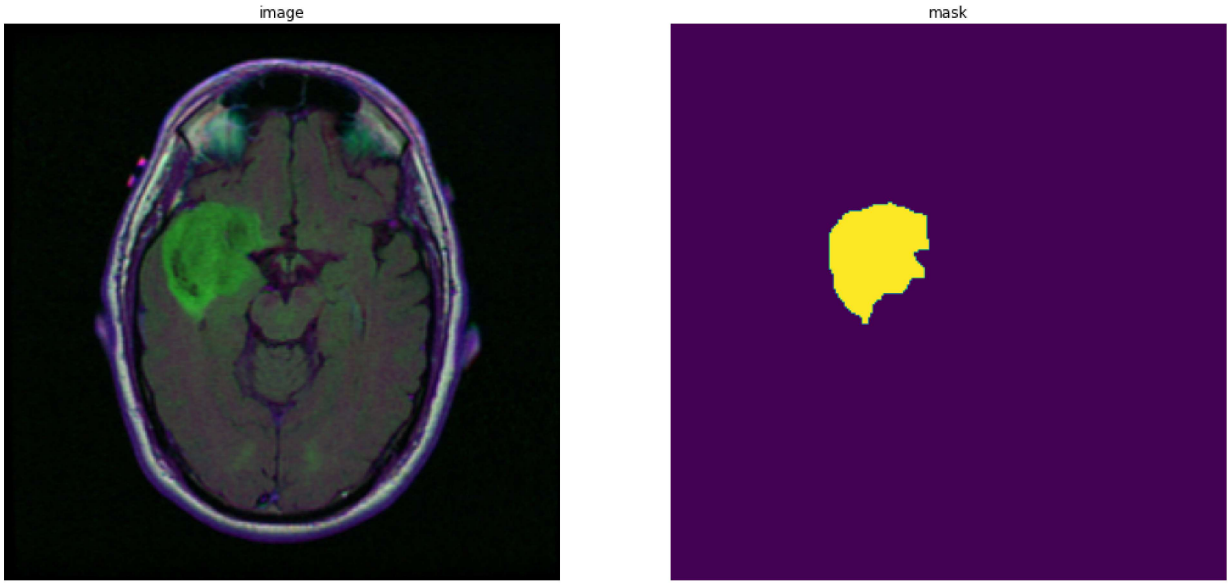
In [ ]:
```
import PIL
import matplotlib
import matplotlib.pyplot as plt
from PIL import Image


def read_and_save_images():
  c=0
  for folder in data_folders: #get each single folder
    files = os.listdir(data_folder_path+'/'+folder) #list all the files and masks in t
    for i in range(1, len(files)//2): #get single file
      img=Image.open(data_folder_path+'/'+folder+'/'+folder+'_'+str(i)+'.tif')
      img.save(images_folder+str(c)+'.tif')
      mask=Image.open(data_folder_path+'/'+folder+'/'+folder+'_'+str(i)+'_mask.tif')
      mask.save(masks_folder+str(c)+'.tif')
      c += 1
      print(c)
    print(folder)
```

In [ ]:
```
#read_and_save_images()
```

In [ ]:
```
import random
images = os.listdir(images_folder)
masks = os.listdir(masks_folder)
rand_index = random.randint(0, len(images))
img = plt.imread(images_folder+images[rand_index])
mask = plt.imread(masks_folder+masks[rand_index])
plt.figure(figsize=(18,18))
plt.subplot(121)
```

```
plt.imshow(img)
plt.title('image')
plt.axis('off')
plt.subplot(122)
plt.imshow(mask)
plt.title('mask')
plt.axis('off')
```

Out[ ]: (-0.5, 255.5, 255.5, -0.5)

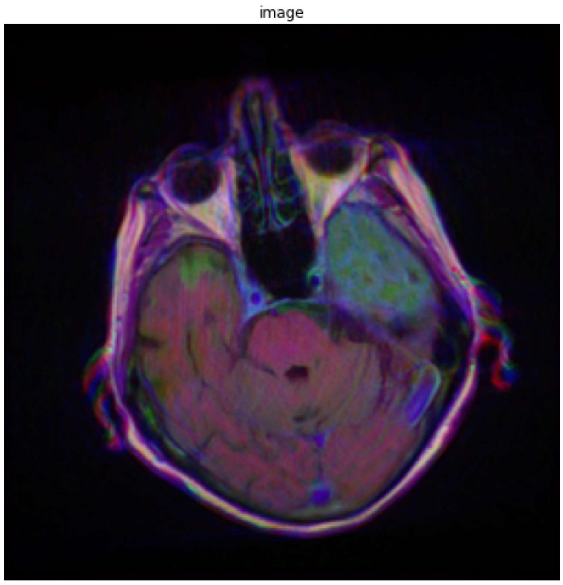


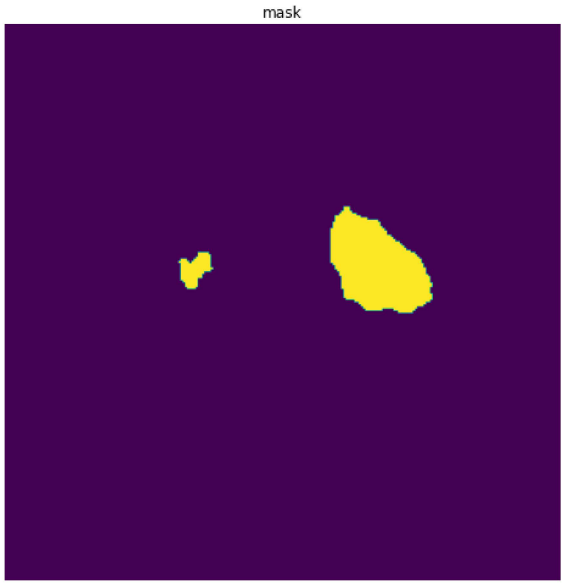
In [ ]:
```
from sklearn.model_selection import train_test_split
train_images, test_images, train_masks, test_masks = train_test_split(images, masks, t
print(f"train image length: {len(train_images)} \ntrain masks length: {len(train_masks
```

```
train image length: 3055
train masks length: 3055
test images length: 764
test masks length: 764
```

In [ ]:
```
rand_index = random.randint(0, len(train_images))
img = plt.imread(images_folder+train_images[rand_index])
mask = plt.imread(masks_folder+train_masks[rand_index])
print(img.shape, mask.shape)
plt.figure(figsize=(18,18))
plt.subplot(121)
plt.imshow(img)
plt.title('image')
plt.axis('off')
plt.subplot(122)
plt.imshow(mask)
plt.title('mask')
plt.axis('off')
```

Out[ ]:
```
(256, 256, 3) (256, 256)
(-0.5, 255.5, 255.5, -0.5)
```





In [ ]:
```
# creo los directorios de los datos de entrenamiento y prueba
'''
if not os.path.exists("./Data/"):
  os.mkdir("./Data/")
if not os.path.exists("./Data/train_images/"):
  os.mkdir("./Data/train_images/")
if not os.path.exists("./Data/val_images/"):
  os.mkdir("./Data/val_images/")
if not os.path.exists("./Data/val_masks/"):
  os.mkdir("./Data/val_masks/")
if not os.path.exists("./Data/train_masks/"):
  os.mkdir("./Data/train_masks/")
'''
```

Out[ ]:
```
' \nif not os.path.exists("./Data/"):\n  os.mkdir("./Data/")\nif not os.path.exists
("./Data/train_images/"):\n  os.mkdir("./Data/train_images/")\nif not os.path.exists
("./Data/val_images/"):\n  os.mkdir("./Data/val_images/")\nif not os.path.exists("./D
ata/val_masks/"):\n  os.mkdir("./Data/val_masks/")\nif not os.path.exists("./Data/tra
in_masks/"):\n  os.mkdir("./Data/train_masks/")\n'
```

In [ ]:
```
'''
if not os.path.exists("Data/train_images/train"):
  os.mkdir("Data/train_images/train")
if not os.path.exists("Data/val_images/val"):
  os.mkdir("Data/val_images/val")
if not os.path.exists("Data/val_masks/val"):
  os.mkdir("Data/val_masks/val")
if not os.path.exists("Data/train_masks/train"):
  os.mkdir("Data/train_masks/train")
'''
```

Out[ ]:
```
'\nif not os.path.exists("Data/train_images/train"):\n  os.mkdir("Data/train_images/t
rain")\nif not os.path.exists("Data/val_images/val"):\n  os.mkdir("Data/val_images/va
l")\nif not os.path.exists("Data/val_masks/val"):\n  os.mkdir("Data/val_masks/val")\n
if not os.path.exists("Data/train_masks/train"):\n  os.mkdir("Data/train_masks/trai
n")\n'
```

In [ ]:
```
train_images_path = './Data/train_images/train/'
train_masks_path = './Data/train_masks/train/'
val_images_path = './Data/val_images/val/'
val_masks_path = './Data/val_masks/val/'
```

In [ ]:
```
#save images and masks in folders
'''
for x in train_images:
  img = Image.open(images_folder+x)
  img.save(train_images_path+x)
  mask = Image.open(masks_folder+x)
  mask.save(train_masks_path+x)
'''
```

Out[ ]:
```
'\nfor x in train_images:\n  img = Image.open(images_folder+x)\n  img.save(train_imag
es_path+x)\n  mask = Image.open(masks_folder+x)\n  mask.save(train_masks_path+x)\n'
```

In [ ]:
```
'''
for x in test_images:
  img = Image.open(images_folder+x)
  img.save(val_images_path+x)
  mask = Image.open(masks_folder+x)
  mask.save(val_masks_path+x)
'''
```

Out[ ]:
```
'\nfor x in test_images:\n  img = Image.open(images_folder+x)\n  img.save(val_images_
path+x)\n  mask = Image.open(masks_folder+x)\n  mask.save(val_masks_path+x)\n'
```

In [ ]:
```
train_i = os.listdir(train_images_path)
train_m = os.listdir(train_masks_path)
val_i = os.listdir(val_images_path)
val_m = os.listdir(val_masks_path)
len(train_i), len(train_m), len(val_i), len(val_m)
```

Out[ ]:
```
(3055, 3055, 764, 764)
```

In [ ]:
```
train_images_path = './Data/train_images/'
train_masks_path = './Data/train_masks/'
val_images_path = './Data/val_images/'
val_masks_path = './Data/val_masks/'
```

In [ ]:
```
import tensorflow as tf
from tensorflow import keras
from keras.preprocessing.image import ImageDataGenerator
tf.random.set_seed(54)
img_data_gen_args = dict(rotation_range=90,
                     width_shift_range=0.3,
                     height_shift_range=0.3,
                     shear_range=0.5,
                     zoom_range=0.3,
                     horizontal_flip=True,
                     vertical_flip=True,
                     fill_mode='reflect')
mask_data_gen_args = dict(rotation_range=90,
                     width_shift_range=0.3,
                     height_shift_range=0.3,
                     shear_range=0.5,
                     zoom_range=0.3,
                     horizontal_flip=True,
                     vertical_flip=True,
                     fill_mode='reflect',
                     preprocessing_function = lambda x: np.where(x>0, 1, 0).astype(x.d
```

In [ ]:
```
#create dataset
#for training
image_data_generator = ImageDataGenerator(**img_data_gen_args, rescale=1.0/255.0)
train_image_generator = image_data_generator.flow_from_directory(train_images_path,
                                                    batch_size=4,
                                                    class_mode=None,
                                                    seed=54)
masks_data_generator = ImageDataGenerator(**mask_data_gen_args)
train_mask_generator = masks_data_generator.flow_from_directory(train_masks_path,
                                                      class_mode=None,
                                                      seed=54,
                                                      batch_size=4)
#for validation
image_data_generator = ImageDataGenerator(rescale=1.0/255.0)
test_image_generator=image_data_generator.flow_from_directory(val_images_path,
                                                    batch_size=4,
                                                    seed=54,
                                                    class_mode=None)
test_masks_generator = image_data_generator.flow_from_directory(val_masks_path,
                                                      batch_size=4,
                                                      seed=54,
                                                      class_mode=None)
```

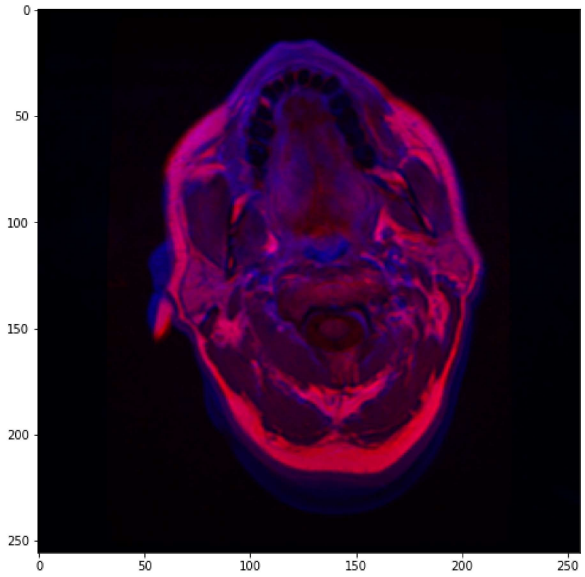
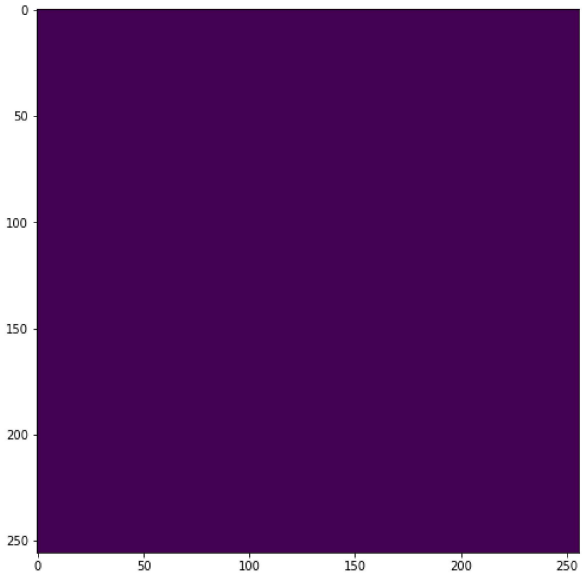
```
Found 3055 images belonging to 1 classes.
Found 3055 images belonging to 1 classes.
Found 764 images belonging to 1 classes.
Found 764 images belonging to 1 classes.
```

In [ ]:
```
#create a function
import numpy as np
def image_mask_generator(image_generator, mask_generator):
  train_generator = zip(image_generator, mask_generator)
  for (img, mask) in train_generator:
    mask=mask[:,:,:,0]
    mask=np.expand_dims(mask, axis=3)
    yield (img, mask)
```

In [ ]:
```
train_datagen = image_mask_generator(train_image_generator, train_mask_generator)
test_datagen  = image_mask_generator(test_image_generator, test_masks_generator)
```

In [ ]:
```
import matplotlib.pyplot as plt
#x, y = train_datagen.__next__()
x, y = test_datagen.__next__()
plt.figure(figsize=(18,18))
for i in range(0,1):
    image = x[i]
    mask = y[i]
    print(image.shape, mask.shape)
    plt.subplot(1,2,1)
    plt.imshow(image)
    plt.subplot(1,2,2)
    plt.imshow(mask[:,:,0])
    plt.show()
```

```
(256, 256, 3) (256, 256, 1)
```

In [ ]:
```
image.max(), mask.max(), image.min(), mask.min(), image.shape, mask.shape
```

Out[ ]:
```
(0.9725491, 0.0, 0.0, 0.0, (256, 256, 3), (256, 256, 1))
```

In [ ]:
```
IMG_HEIGHT = image.shape[0]
IMG_WIDTH  = image.shape[1]
IMG_CHANNELS = image.shape[2]
IMG_WIDTH, IMG_HEIGHT, IMG_CHANNELS #Dimenciones de los datos.
```

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

In [ ]:
```
# para no sobre ajustar el modelo
import tensorflow as tf
early_stopping=tf.keras.callbacks.EarlyStopping(
    monitor="accuracy",
    patience=3,
    verbose=0,
    mode="auto",
    baseline=None,
    restore_best_weights=True
)


from keras.models import Model
from keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D, concatenate, Conv2
from tensorflow.keras.optimizers import Adam
from keras.metrics import MeanIoU

kernel_initializer =  'he_uniform'  # also try 'he_normal' but model not converging...
```

MODELO DE EJEMPLO

In [ ]:

```
#################################################################
def simple_unet_model(IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS):
#Build the model
    inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
    #s = Lambda(lambda x: x / 255)(inputs)   #No need for this if we normalize our inp
    s = inputs

    #Contraction path
    c1 = Conv2D(16, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    c1 = Dropout(0.1)(c1)
    c1 = Conv2D(16, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    p1 = MaxPooling2D((2, 2))(c1)

    c2 = Conv2D(32, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    c2 = Dropout(0.1)(c2)
    c2 = Conv2D(32, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    p2 = MaxPooling2D((2, 2))(c2)

    c3 = Conv2D(64, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    c3 = Dropout(0.2)(c3)
    c3 = Conv2D(64, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    p3 = MaxPooling2D((2, 2))(c3)

    c4 = Conv2D(128, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    c4 = Dropout(0.2)(c4)
    c4 = Conv2D(128, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    p4 = MaxPooling2D(pool_size=(2, 2))(c4)

    c5 = Conv2D(256, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    c5 = Dropout(0.3)(c5)
    c5 = Conv2D(256, (3, 3), activation='relu', kernel_initializer=kernel_initializer,

    #Expansive path
    u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(c5)
    u6 = concatenate([u6, c4])
    c6 = Conv2D(128, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    c6 = Dropout(0.2)(c6)
    c6 = Conv2D(128, (3, 3), activation='relu', kernel_initializer=kernel_initializer,

    u7 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(c6)
    u7 = concatenate([u7, c3])
    c7 = Conv2D(64, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    c7 = Dropout(0.2)(c7)
    c7 = Conv2D(64, (3, 3), activation='relu', kernel_initializer=kernel_initializer,

    u8 = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(c7)
    u8 = concatenate([u8, c2])
    c8 = Conv2D(32, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    c8 = Dropout(0.1)(c8)
    c8 = Conv2D(32, (3, 3), activation='relu', kernel_initializer=kernel_initializer,

    u9 = Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same')(c8)
    u9 = concatenate([u9, c1], axis=3)
    c9 = Conv2D(16, (3, 3), activation='relu', kernel_initializer=kernel_initializer,
    c9 = Dropout(0.1)(c9)
    c9 = Conv2D(16, (3, 3), activation='relu', kernel_initializer=kernel_initializer,

    outputs = Conv2D(1, (1, 1), activation='sigmoid')(c9)

    model = Model(inputs=[inputs], outputs=[outputs])
```

```
    model.compile(optimizer=Adam(lr = 1e-3), loss='binary_crossentropy', metrics=['acc
    #model.compile(optimizer=Adam(lr = 1e-3), loss='binary_crossentropy', metrics=[Mea
    model.summary()

    return model
```

In [ ]:

```
def get_model():
    return simple_unet_model(IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS)
model_ejemplo = get_model()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0         []

 conv2d (Conv2D)             (None, 256, 256, 16)      448       ['input_1[0][0]']

 dropout (Dropout)           (None, 256, 256, 16)      0         ['conv2d[0][0]']

 conv2d_1 (Conv2D)           (None, 256, 256, 16)      2320      ['dropout[0][0]']

 max_pooling2d (MaxPooling2D)  (None, 128, 128, 16)    0         ['conv2d_1[0][0]']

 conv2d_2 (Conv2D)           (None, 128, 128, 32)      4640      ['max_pooling2d[0][0]']

 dropout_1 (Dropout)         (None, 128, 128, 32)      0         ['conv2d_2[0][0]']

 conv2d_3 (Conv2D)           (None, 128, 128, 32)      9248      ['dropout_1[0][0]']

 max_pooling2d_1 (MaxPooling2D)  (None, 64, 64, 32)    0         ['conv2d_3[0][0]']

 conv2d_4 (Conv2D)           (None, 64, 64, 64)        18496     ['max_pooling2d_1[0][0]']

 dropout_2 (Dropout)         (None, 64, 64, 64)        0         ['conv2d_4[0][0]']

 conv2d_5 (Conv2D)           (None, 64, 64, 64)        36928     ['dropout_2[0][0]']

 max_pooling2d_2 (MaxPooling2D)  (None, 32, 32, 64)    0         ['conv2d_5[0][0]']

 conv2d_6 (Conv2D)           (None, 32, 32, 128)       73856     ['max_pooling2d_2[0][0]']

 dropout_3 (Dropout)         (None, 32, 32, 128)       0         ['conv2d_6[0][0]']

 conv2d_7 (Conv2D)           (None, 32, 32, 128)       147584    ['dropout_3[0][0]']

 max_pooling2d_3 (MaxPooling2D)  (None, 16, 16, 128)   0         ['conv2d_7[0][0]']

 conv2d_8 (Conv2D)           (None, 16, 16, 256)       295168    ['max_pooling2d_3[0][0]']

 dropout_4 (Dropout)         (None, 16, 16, 256)       0         ['conv2d_8[0][0]']

 conv2d_9 (Conv2D)           (None, 16, 16, 256)       590080    ['dropout_4[0][0]']

 conv2d_transpose (Conv2DTranspose)  (None, 32, 32, 128)  131200  ['conv2d_9[0][0]']
```

```
 concatenate (Concatenate)      (None, 32, 32, 256)  0           ['conv2d_transpose
[0][0]',
                                                                  'conv2d_7[0][0]']

 conv2d_10 (Conv2D)             (None, 32, 32, 128)  295040      ['concatenate[0]
[0]']

 dropout_5 (Dropout)            (None, 32, 32, 128)  0           ['conv2d_10[0][0]']

 conv2d_11 (Conv2D)             (None, 32, 32, 128)  147584      ['dropout_5[0][0]']

 conv2d_transpose_1 (Conv2DTran  (None, 64, 64, 64)  32832       ['conv2d_11[0][0]']
 spose)

 concatenate_1 (Concatenate)    (None, 64, 64, 128)  0           ['conv2d_transpose_1
[0][0]',
                                                                  'conv2d_5[0][0]']

 conv2d_12 (Conv2D)             (None, 64, 64, 64)   73792       ['concatenate_1[0]
[0]']

 dropout_6 (Dropout)            (None, 64, 64, 64)   0           ['conv2d_12[0][0]']

 conv2d_13 (Conv2D)             (None, 64, 64, 64)   36928       ['dropout_6[0][0]']

 conv2d_transpose_2 (Conv2DTran  (None, 128, 128, 32  8224       ['conv2d_13[0][0]']
 spose)                         )

 concatenate_2 (Concatenate)    (None, 128, 128, 64  0           ['conv2d_transpose_2
[0][0]',
                                )                                 'conv2d_3[0][0]']

 conv2d_14 (Conv2D)             (None, 128, 128, 32  18464       ['concatenate_2[0]
[0]']
                                )

 dropout_7 (Dropout)            (None, 128, 128, 32  0           ['conv2d_14[0][0]']
                                )

 conv2d_15 (Conv2D)             (None, 128, 128, 32  9248        ['dropout_7[0][0]']
                                )

 conv2d_transpose_3 (Conv2DTran  (None, 256, 256, 16  2064       ['conv2d_15[0][0]']
 spose)                         )

 concatenate_3 (Concatenate)    (None, 256, 256, 32  0           ['conv2d_transpose_3
[0][0]',
                                )                                 'conv2d_1[0][0]']

 conv2d_16 (Conv2D)             (None, 256, 256, 16  4624        ['concatenate_3[0]
[0]']
                                )

 dropout_8 (Dropout)            (None, 256, 256, 16  0           ['conv2d_16[0][0]']
                                )

 conv2d_17 (Conv2D)             (None, 256, 256, 16  2320        ['dropout_8[0][0]']
                                )
```

```
 conv2d_18 (Conv2D)             (None, 256, 256, 1)  17          ['conv2d_17[0][0]']

==================================================================================
=============
Total params: 1,941,105
Trainable params: 1,941,105
Non-trainable params: 0
__________________________________________________________________________________
_____________
```

```
c:\Users\ASUS\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\optimize
r_v2\adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning_rate` i
nstead.
  super(Adam, self).__init__(name, **kwargs)
```

In [ ]:
```
model_ejemplo.save('G:/Mi unidad/Demat/8 semestre/ML/Tarea 3/EJEMPLO.h5')   # Gardo mi
```

In [ ]:
```

```

In [ ]:
```
import os
num_imges=len(os.listdir(train_images_path+"train/"))
num_vals=len(os.listdir(val_images_path+"val/"))

steps_per_epoch=num_imges//16
validation_steps=int(0.1 * num_vals//16)
steps_per_epoch, validation_steps
```

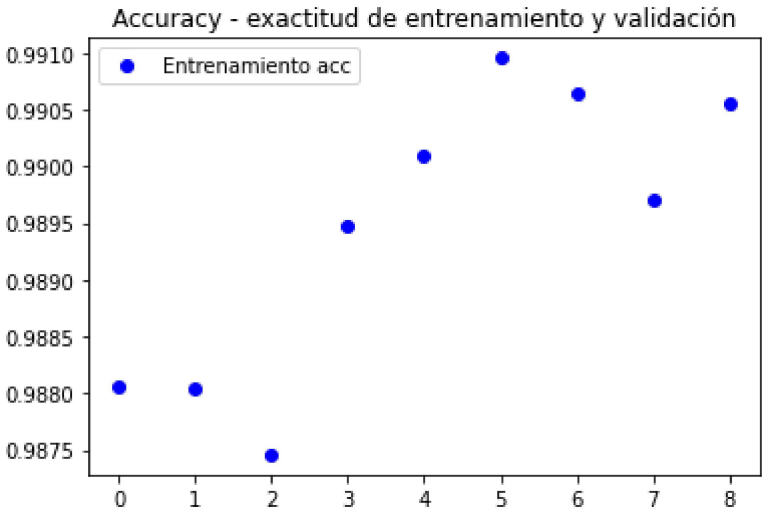
Out[ ]:
```
(190, 4)
```

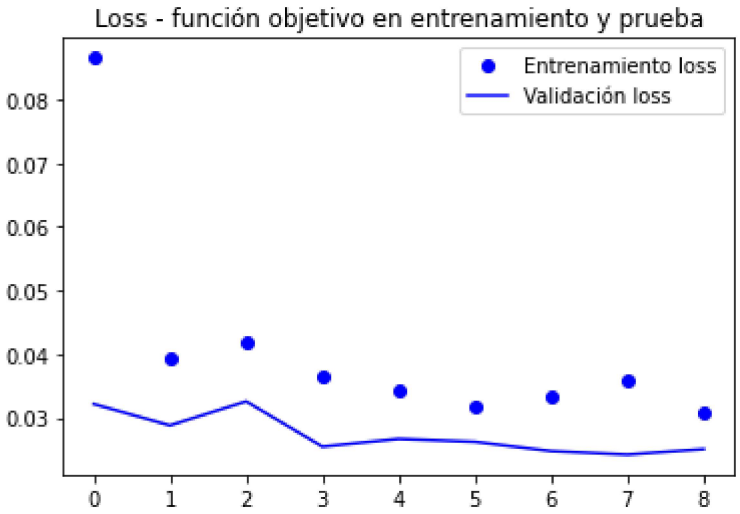
In [ ]:
```
#Fit the model
history = model_ejemplo.fit(train_datagen, validation_data=test_datagen, steps_per_epo
                    callbacks=early_stopping)
model_ejemplo.save('G:/Mi unidad/Demat/8 semestre/ML/Tarea 3/EJEMPLO_fit.h5')   # Gard
```

```
Epoch 1/15
190/190 [==============================] - 351s 2s/step - loss: 0.0865 - accuracy: 0.
9881 - val_loss: 0.0322 - val_accuracy: 0.9897
Epoch 2/15
190/190 [==============================] - 355s 2s/step - loss: 0.0392 - accuracy: 0.
9880 - val_loss: 0.0288 - val_accuracy: 0.9901
Epoch 3/15
190/190 [==============================] - 344s 2s/step - loss: 0.0417 - accuracy: 0.
9875 - val_loss: 0.0326 - val_accuracy: 0.9909
Epoch 4/15
190/190 [==============================] - 353s 2s/step - loss: 0.0364 - accuracy: 0.
9895 - val_loss: 0.0255 - val_accuracy: 0.9926
Epoch 5/15
190/190 [==============================] - 356s 2s/step - loss: 0.0344 - accuracy: 0.
9901 - val_loss: 0.0267 - val_accuracy: 0.9923
Epoch 6/15
190/190 [==============================] - 346s 2s/step - loss: 0.0317 - accuracy: 0.
9910 - val_loss: 0.0262 - val_accuracy: 0.9921
Epoch 7/15
190/190 [==============================] - 347s 2s/step - loss: 0.0334 - accuracy: 0.
9907 - val_loss: 0.0248 - val_accuracy: 0.9927
Epoch 8/15
190/190 [==============================] - 347s 2s/step - loss: 0.0358 - accuracy: 0.
9897 - val_loss: 0.0242 - val_accuracy: 0.9928
Epoch 9/15
190/190 [==============================] - 343s 2s/step - loss: 0.0309 - accuracy: 0.
9905 - val_loss: 0.0251 - val_accuracy: 0.9929
```

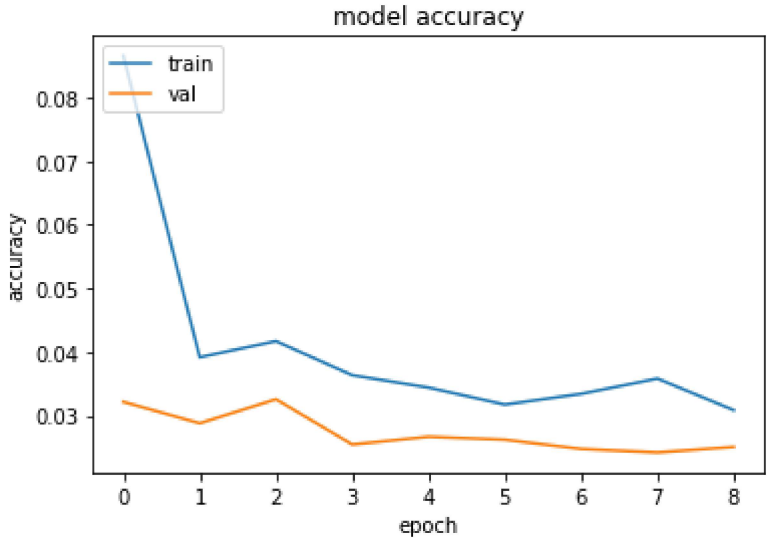
In [ ]:

```python
import matplotlib.pyplot as plt

#1850

acc      = history.history['accuracy']
loss     = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'bo', label='Entrenamiento acc')
plt.title('Accuracy - exactitud de entrenamiento y validación')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Entrenamiento loss')
plt.plot(epochs, val_loss, 'b', label='Validación loss')
plt.title('Loss - función objetivo en entrenamiento y prueba')
plt.legend()

plt.show()
```





In [ ]:
```
import keras
from matplotlib import pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



In [ ]:
```
model_ejemplo  = keras.models.load_model('G:/Mi unidad/Demat/8 semestre/ML/Tarea 3/EJE
```

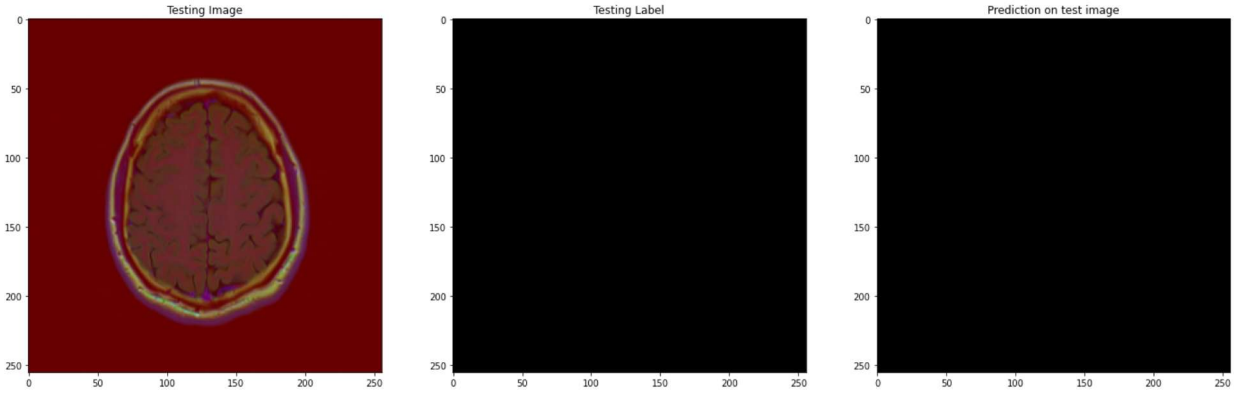
In [ ]:
```
model_ejemplo.compile(optimizer=Adam(lr = 1e-3), loss='binary_crossentropy', metrics=[
```

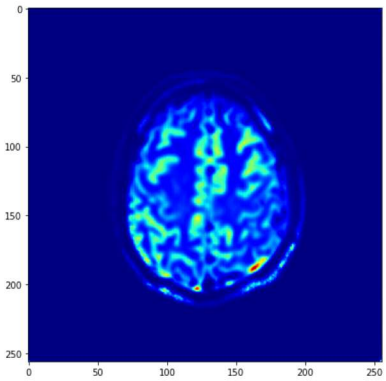
```
c:\Users\ASUS\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\optimize
r_v2\adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning_rate` i
nstead.
  super(Adam, self).__init__(name, **kwargs)
```

In [ ]:
```
#making predictions
threshold = 0.5
#test_img_number = random.randint(0, num_vals-1)
x, y=test_datagen.__next__()
for i in range(0,1):
  test_img=x[i]
  ground_truth=y[i]
  print(ground_truth.shape)

# test_img = X_test[test_img_number]
# ground_truth=y_test[test_img_number]
test_img_input=np.expand_dims(test_img, 0)
print(test_img_input.shape)
prediction = (model_ejemplo.predict(test_img_input)[0,:,:,:] > 0.2).astype(np.uint8)
#prediction = (model.predict(test_img_input)[0,:,:,:])
print(prediction.shape)

plt.figure(figsize=(24, 24))
plt.subplot(231)
plt.title('Testing Image')
plt.imshow(test_img[:,:,:])
plt.subplot(232)
plt.title('Testing Label')
plt.imshow(ground_truth[:,:,0], cmap='gray')
plt.subplot(233)
plt.title('Prediction on test image')
plt.imshow(prediction[:,:,0], cmap='gray')
plt.subplot(234)

img = np.array(model_ejemplo.predict(test_img_input)[0,:,:,:] )
```

```
plt.imshow(img, 'jet')
plt.show()
```

```
(256, 256, 1)
(1, 256, 256, 3)
(256, 256, 1)
```





El modelo del ejemplo tiene un buen desempeño aun que en algunos casos no segmenta todo el tumor.

MI MODELO Intent implementar un modelo tratando de colar un decoder despues de la MobileNetV2 y un clasificador aprecido al del ejemplo despues del encode, pero no funciono.

In [ ]:
```
IMG_SHAPE = (IMG_WIDTH, IMG_HEIGHT, IMG_CHANNELS)

# Create the base model from the pre-trained model MobileNet V2
all_model = None
all_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96,
128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the defaul
t.
```

lo intent con la funcion de activation linear y relu

In [ ]:
```
from keras import models
model = None
#model = models.Sequential()

c1 = None
c1 = layers.Conv2DTranspose(1280, (1, 1), strides=2, activation="linear", padding="sam
c1 = layers.Conv2DTranspose(320, (1, 1), strides=2, activation="linear", padding="same
#c1 = layers.Conv2DTranspose(960, (1, 1), strides=2, activation="linear", padding="sam
#c1 = layers.Conv2DTranspose(160, (1, 1), strides=2, activation="linear", padding="sam
c1 = layers.Conv2DTranspose(144, (1, 1), strides=2, activation="linear", padding="same
#c1 = layers.Conv2DTranspose(24, (1, 1), strides=2, activation="linear", padding="same
c1 = layers.Conv2DTranspose(24, (1, 1), strides=2, activation="linear", padding="same'
#c1 = layers.Conv2DTranspose(92, (1, 1), strides=2, activation="linear", padding="same
#c1 = layers.Conv2DTranspose(16, (1, 1), strides=2, activation="linear", padding="same
c1 = layers.Conv2DTranspose(32, (3, 3), strides=2, activation="linear", padding="same'
c1 = layers.Conv2D(1, (3, 3), activation="linear", padding="same")(c1)



c1 = Conv2D(16, (2, 2), activation='linear', kernel_initializer=kernel_initializer, pa
c1 = (Dropout(0.1))(c1)
c1 = Conv2D(16, (3, 3), activation='linear', kernel_initializer=kernel_initializer, pa
c1 = Conv2D(1, (1, 1), activation='linear', kernel_initializer=kernel_initializer, pad

model = tf.keras.Model(inputs=all_model.input, outputs=c1)
#all_model.trainable= False
model.compile(optimizer=Adam(lr = 1e-3), loss='binary_crossentropy', metrics=['accurac
#model.output

# INTENTE IMPLEMENTAR UN DECODE PERO NO ME FUNCIONO :c
```

In [ ]:
```
history = model.fit(train_datagen, validation_data=test_datagen, steps_per_epoch=steps
                    )
model.save('G:/Mi unidad/Demat/8 semestre/ML/Tarea 4/modelo_base_decode_clas_linear.h5
```

```
190/190 [==============================] - 433s 2s/step - loss: 0.0972 - accuracy: 0.
9807 - val_loss: 0.0364 - val_accuracy: 0.9903
```

Red base fija

In [ ]:
```
model_pix2pix  = keras.models.load_model('G:/Mi unidad/Demat/8 semestre/ML/Tarea 4/Mod
model_pix2pix.compile(optimizer=Adam(lr = 1e-3), loss='binary_crossentropy', metrics=[
#model.output
model_pix2pix.summary()
```

```
Model: "model_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_16 (InputLayer)       [(None, 256, 256, 3)]     0         []

 model_9 (Functional)        [(None, 128, 128, 96),    1841984   ['input_16[0][0]']
                              (None, 64, 64, 144),
                              (None, 32, 32, 192),
                              (None, 16, 16, 576),
                              (None, 8, 8, 320)]

 sequential_20 (Sequential)  (None, 16, 16, 512)       165888    ['model_9[0][4]']

 concatenate_36 (Concatenate (None, 16, 16, 1088)      0         ['sequential_20[0][0]',
 )                                                                'model_9[0][3]']

 sequential_21 (Sequential)  (None, 32, 32, 256)       279552    ['concatenate_36[0][0]']

 concatenate_37 (Concatenate (None, 32, 32, 448)       0         ['sequential_21[0][0]',
 )                                                                'model_9[0][2]']

 sequential_22 (Sequential)  (None, 64, 64, 128)       57856     ['concatenate_37[0][0]']

 concatenate_38 (Concatenate (None, 64, 64, 272)       0         ['sequential_22[0][0]',
 )                                                                'model_9[0][1]']

 sequential_23 (Sequential)  (None, 128, 128, 64)      17664     ['concatenate_38[0][0]']

 concatenate_39 (Concatenate (None, 128, 128, 160)     0         ['sequential_23[0][0]',
 )                                                                'model_9[0][0]']

 conv2d_transpose_33 (Conv2DTra  (None, 256, 256, 3)   4323      ['concatenate_39[0][0]']
 nspose)

 conv2d_1 (Conv2D)           (None, 256, 256, 1)       28        ['conv2d_transpose_33[0][0]']

 conv2d_2 (Conv2D)           (None, 256, 256, 16)      80        ['conv2d_1[0][0]']

 dropout (Dropout)           (None, 256, 256, 16)      0         ['conv2d_2[0][0]']
```

```
 conv2d_3 (Conv2D)              (None, 256, 256, 16  2320        ['dropout[0][0]']
                                )

 conv2d_4 (Conv2D)              (None, 256, 256, 1)  17          ['conv2d_3[0][0]']

==================================================================================================
Total params: 2,369,712
Trainable params: 525,808
Non-trainable params: 1,843,904
__________________________________________________________________________________________________
```

In [ ]:

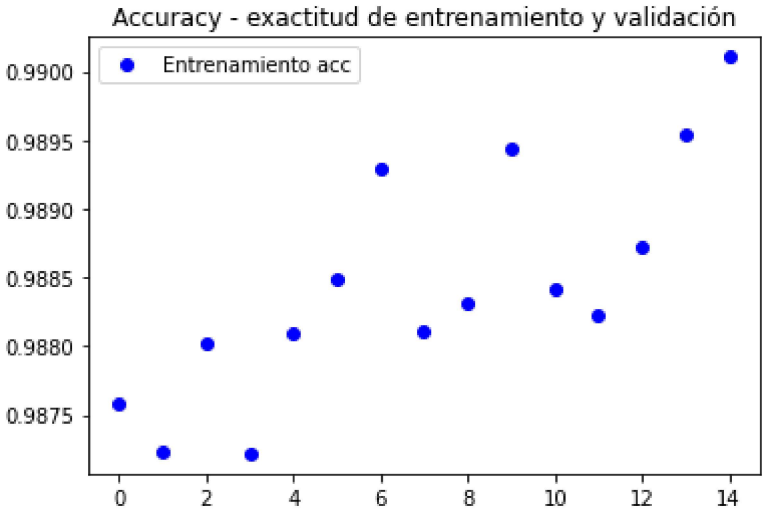
```
import os
num_imges=len(os.listdir(train_images_path+"train/"))
num_vals=len(os.listdir(val_images_path+"val/"))

steps_per_epoch=num_imges//16
validation_steps=int(0.1 * num_vals//16)
steps_per_epoch, validation_steps

history = model_pix2pix.fit(train_datagen, validation_data=test_datagen, steps_per_epc
                    )
model_pix2pix.save('G:/Mi unidad/Demat/8 semestre/ML/Tarea 4/Modelos/pix2pix_red_base_
```

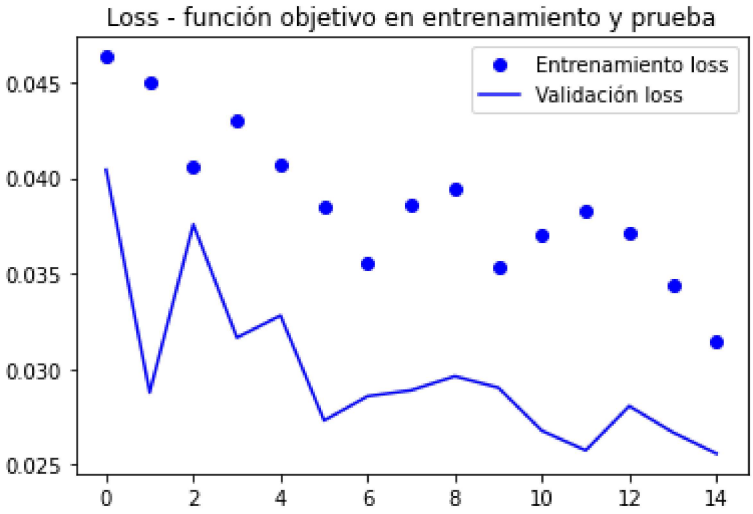
```
Epoch 1/15
190/190 [==============================] - 211s 1s/step - loss: 0.0463 - accuracy: 0.
9876 - val_loss: 0.0404 - val_accuracy: 0.9889
Epoch 2/15
190/190 [==============================] - 209s 1s/step - loss: 0.0450 - accuracy: 0.
9872 - val_loss: 0.0288 - val_accuracy: 0.9908
Epoch 3/15
190/190 [==============================] - 213s 1s/step - loss: 0.0406 - accuracy: 0.
9880 - val_loss: 0.0376 - val_accuracy: 0.9894
Epoch 4/15
190/190 [==============================] - 187s 988ms/step - loss: 0.0430 - accuracy:
0.9872 - val_loss: 0.0316 - val_accuracy: 0.9894
Epoch 5/15
190/190 [==============================] - 172s 902ms/step - loss: 0.0407 - accuracy:
0.9881 - val_loss: 0.0328 - val_accuracy: 0.9902
Epoch 6/15
190/190 [==============================] - 192s 1s/step - loss: 0.0385 - accuracy: 0.
9885 - val_loss: 0.0273 - val_accuracy: 0.9916
Epoch 7/15
190/190 [==============================] - 199s 1s/step - loss: 0.0355 - accuracy: 0.
9893 - val_loss: 0.0286 - val_accuracy: 0.9909
Epoch 8/15
190/190 [==============================] - 195s 1s/step - loss: 0.0386 - accuracy: 0.
9881 - val_loss: 0.0289 - val_accuracy: 0.9909
Epoch 9/15
190/190 [==============================] - 194s 1s/step - loss: 0.0395 - accuracy: 0.
9883 - val_loss: 0.0296 - val_accuracy: 0.9910
Epoch 10/15
190/190 [==============================] - 236s 1s/step - loss: 0.0353 - accuracy: 0.
9894 - val_loss: 0.0290 - val_accuracy: 0.9908
Epoch 11/15
190/190 [==============================] - 198s 1s/step - loss: 0.0370 - accuracy: 0.
9884 - val_loss: 0.0268 - val_accuracy: 0.9915
Epoch 12/15
190/190 [==============================] - 203s 1s/step - loss: 0.0383 - accuracy: 0.
9882 - val_loss: 0.0257 - val_accuracy: 0.9917
Epoch 13/15
190/190 [==============================] - 191s 1s/step - loss: 0.0372 - accuracy: 0.
9887 - val_loss: 0.0281 - val_accuracy: 0.9909
Epoch 14/15
190/190 [==============================] - 185s 975ms/step - loss: 0.0344 - accuracy:
0.9895 - val_loss: 0.0267 - val_accuracy: 0.9916
Epoch 15/15
190/190 [==============================] - 173s 911ms/step - loss: 0.0314 - accuracy:
0.9901 - val_loss: 0.0256 - val_accuracy: 0.9917
```

In [ ]:

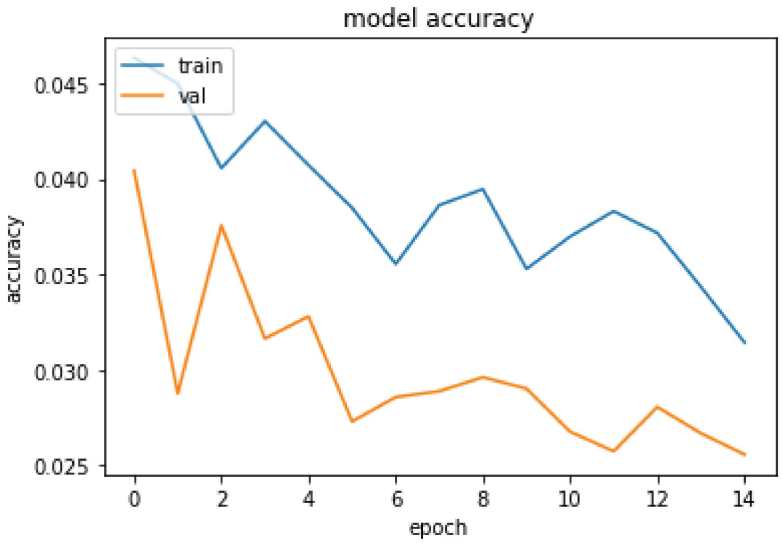
```python
import matplotlib.pyplot as plt

#1850

acc      = history.history['accuracy']
loss     = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'bo', label='Entrenamiento acc')
plt.title('Accuracy - exactitud de entrenamiento y validación')
plt.legend()
```

```
plt.figure()

plt.plot(epochs, loss, 'bo', label='Entrenamiento loss')
plt.plot(epochs, val_loss, 'b', label='Validación loss')
plt.title('Loss - función objetivo en entrenamiento y prueba')
plt.legend()

plt.show()
```



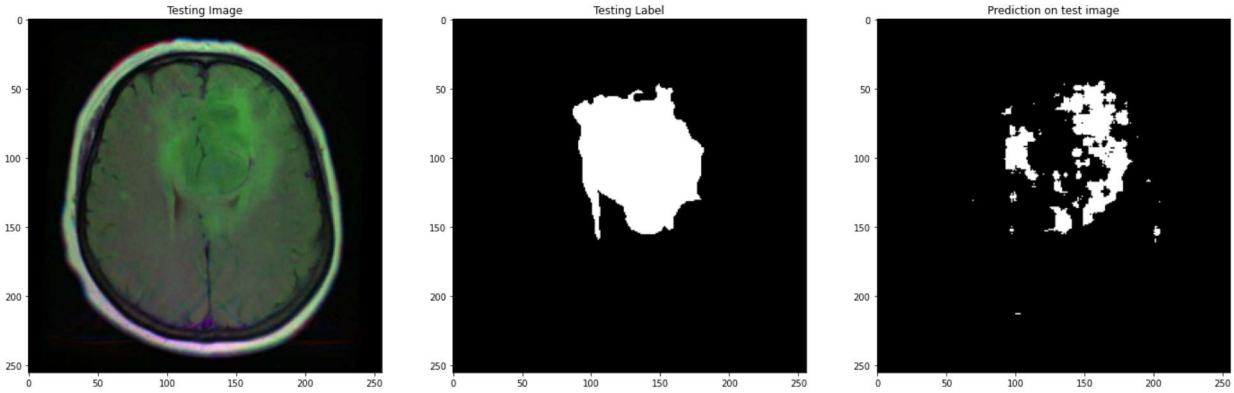


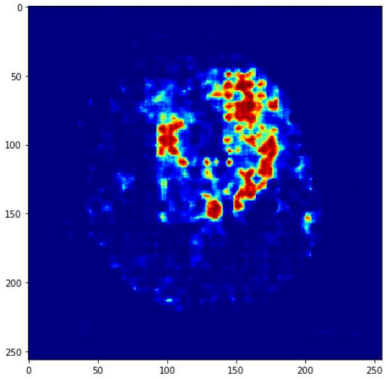
In [ ]:
```
import keras
from matplotlib import pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



In [ ]:
```
#making predictions
threshold = 0.5
#test_img_number = random.randint(0, num_vals-1)
x, y=test_datagen.__next__()
for i in range(0,1):
  test_img=x[i]
  ground_truth=y[i]
  print(ground_truth.shape)

# test_img = X_test[test_img_number]
# ground_truth=y_test[test_img_number]
test_img_input=np.expand_dims(test_img, 0)
print(test_img_input.shape)
prediction = (model_pix2pix2.predict(test_img_input)[0,:,:,:] > 0.3).astype(np.uint8)
#prediction = (model.predict(test_img_input)[0,:,:,:])
print(prediction.shape)

plt.figure(figsize=(24, 24))
plt.subplot(231)
plt.title('Testing Image')
plt.imshow(test_img[:,:,:])
plt.subplot(232)
plt.title('Testing Label')
plt.imshow(ground_truth[:,:,0], cmap='gray')
plt.subplot(233)
plt.title('Prediction on test image')
plt.imshow(prediction[:,:,0], cmap='gray')
plt.subplot(234)

img = np.array(model_pix2pix2.predict(test_img_input)[0,:,:,:] )

plt.imshow(img, 'jet')
plt.show()
```

```
(256, 256, 1)
(1, 256, 256, 3)
(256, 256, 1)
```





In [ ]:
```
np.min(img)
```

Out[ ]:
```
5.326367e-13
```

Aun que la red detecta el tumor, no lo segmenta todo y tiene algunos guecos.

RED BASE SEMI LIBRE (fine tuning).

In [ ]:
```
model_pix2pix_semi  = keras.models.load_model('G:/Mi unidad/Demat/8 semestre/ML/Tarea
model_pix2pix_semi.compile(optimizer=Adam(lr = 1e-3), loss='binary_crossentropy', metr
#model.output
model_pix2pix_semi.summary()
```

```
WARNING:tensorflow:No training configuration found in the save file, so the model was
*not* compiled. Compile it manually.
Model: "model_38"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================
 input_39 (InputLayer)       [(None, 256, 256, 3)]     0         []

 model_36 (Functional)       [(None, 128, 128, 96),    1841984   ['input_39[0][0]']
                              (None, 64, 64, 144),
                              (None, 32, 32, 192),
                              (None, 16, 16, 576),
                              (None, 8, 8, 320)]

 sequential_48 (Sequential)  (None, 16, 16, 512)       165888    ['model_36[0][4]']

 concatenate_64 (Concatenate (None, 16, 16, 1088)      0         ['sequential_48[0][0]',
                                                                  'model_36[0][3]']

 sequential_49 (Sequential)  (None, 32, 32, 256)       279552    ['concatenate_64[0][0]']

 concatenate_65 (Concatenate (None, 32, 32, 448)       0         ['sequential_49[0][0]',
                                                                  'model_36[0][2]']

 sequential_50 (Sequential)  (None, 64, 64, 128)       57856     ['concatenate_65[0][0]']

 concatenate_66 (Concatenate (None, 64, 64, 272)       0         ['sequential_50[0][0]',
                                                                  'model_36[0][1]']

 sequential_51 (Sequential)  (None, 128, 128, 64)      17664     ['concatenate_66[0][0]']

 concatenate_67 (Concatenate (None, 128, 128, 160)     0         ['sequential_51[0][0]',
                                                                  'model_36[0][0]']

 conv2d_transpose_68 (Conv2DTra  (None, 256, 256, 3)   4323      ['concatenate_67[0][0]']
 nspose)

 conv2d_29 (Conv2D)          (None, 256, 256, 1)       28        ['conv2d_transpose_68[0][0]']

 conv2d_30 (Conv2D)          (None, 256, 256, 16)      80        ['conv2d_29[0][0]']

 dropout_7 (Dropout)         (None, 256, 256, 16)      0         ['conv2d_30[0][0]']
```

```
                                )

 conv2d_31 (Conv2D)             (None, 256, 256, 16  2320        ['dropout_7[0][0]']
                                )

 conv2d_32 (Conv2D)             (None, 256, 256, 1)  17          ['conv2d_31[0][0]']

==================================================================================
=============
Total params: 2,369,712
Trainable params: 1,543,600
Non-trainable params: 826,112
__________________________________________________________________________________
_____________
```

In [ ]:
```
import os
num_imges=len(os.listdir(train_images_path+"train/"))
num_vals=len(os.listdir(val_images_path+"val/"))

steps_per_epoch=num_imges//16
validation_steps=int(0.1 * num_vals//16)
steps_per_epoch, validation_steps

history_semi = model_pix2pix_semi.fit(train_datagen, validation_data=test_datagen, ste
                    )
model_pix2pix_semi.save('G:/Mi unidad/Demat/8 semestre/ML/Tarea 4/Modelos/pix2pix_red_
```

```
Epoch 1/15
190/190 [==============================] - 329s 2s/step - loss: 0.0465 - accuracy: 0.
9880 - val_loss: 0.0521 - val_accuracy: 0.9897
Epoch 2/15
190/190 [==============================] - 316s 2s/step - loss: 0.0316 - accuracy: 0.
9880 - val_loss: 0.0966 - val_accuracy: 0.9897
Epoch 3/15
190/190 [==============================] - 308s 2s/step - loss: 0.0309 - accuracy: 0.
9899 - val_loss: 0.2764 - val_accuracy: 0.9566
Epoch 4/15
190/190 [==============================] - 300s 2s/step - loss: 0.0287 - accuracy: 0.
9915 - val_loss: 0.1436 - val_accuracy: 0.9877
Epoch 5/15
190/190 [==============================] - 298s 2s/step - loss: 0.0257 - accuracy: 0.
9918 - val_loss: 0.0873 - val_accuracy: 0.9854
Epoch 6/15
190/190 [==============================] - 301s 2s/step - loss: 0.0211 - accuracy: 0.
9934 - val_loss: 0.2218 - val_accuracy: 0.9692
Epoch 7/15
190/190 [==============================] - 299s 2s/step - loss: 0.0240 - accuracy: 0.
9919 - val_loss: 0.0866 - val_accuracy: 0.9861
Epoch 8/15
190/190 [==============================] - 293s 2s/step - loss: 0.0238 - accuracy: 0.
9922 - val_loss: 0.4850 - val_accuracy: 0.7548
Epoch 9/15
190/190 [==============================] - 292s 2s/step - loss: 0.0204 - accuracy: 0.
9930 - val_loss: 0.0784 - val_accuracy: 0.9895
Epoch 10/15
190/190 [==============================] - 291s 2s/step - loss: 0.0201 - accuracy: 0.
9933 - val_loss: 0.1712 - val_accuracy: 0.9746
Epoch 11/15
190/190 [==============================] - 292s 2s/step - loss: 0.0207 - accuracy: 0.
9928 - val_loss: 0.8327 - val_accuracy: 0.4941
Epoch 12/15
190/190 [==============================] - 292s 2s/step - loss: 0.0228 - accuracy: 0.
9923 - val_loss: 0.0539 - val_accuracy: 0.9901
Epoch 13/15
190/190 [==============================] - 290s 2s/step - loss: 0.0203 - accuracy: 0.
9930 - val_loss: 0.2899 - val_accuracy: 0.9095
Epoch 14/15
190/190 [==============================] - 289s 2s/step - loss: 0.0189 - accuracy: 0.
9933 - val_loss: 0.0543 - val_accuracy: 0.9896
Epoch 15/15
190/190 [==============================] - 289s 2s/step - loss: 0.0170 - accuracy: 0.
9940 - val_loss: 0.0625 - val_accuracy: 0.9896
```

In [ ]:

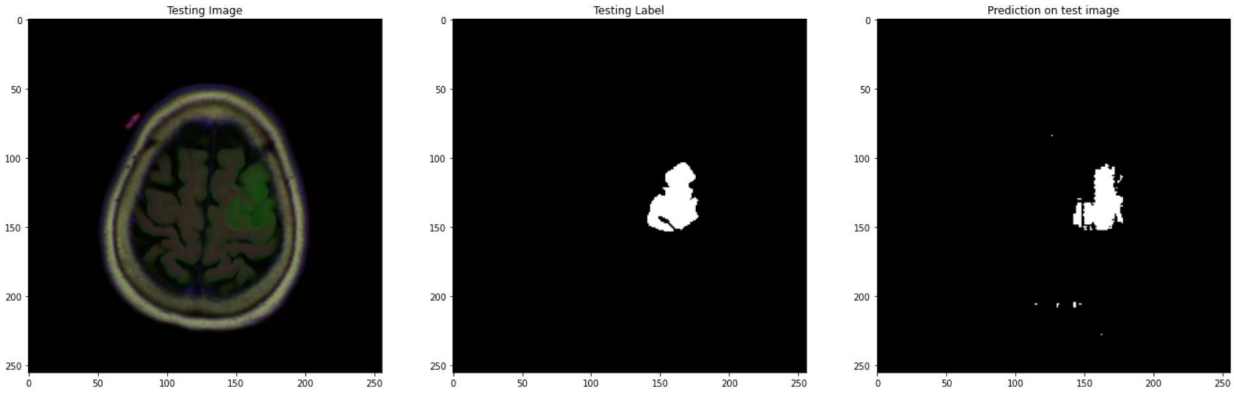
```python
import matplotlib.pyplot as plt

#1850

acc      = history_semi.history['accuracy']
loss     = history_semi.history['loss']
val_loss = history_semi.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'bo', label='Entrenamiento acc')
plt.title('Accuracy - exactitud de entrenamiento y validación')
plt.legend()
```

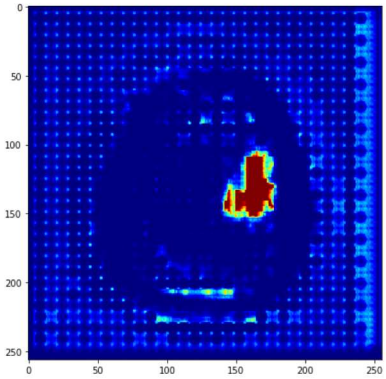
```
plt.figure()

plt.plot(epochs, loss, 'bo', label='Entrenamiento loss')
plt.plot(epochs, val_loss, 'b', label='Validación loss')
plt.title('Loss - función objetivo en entrenamiento y prueba')
plt.legend()

plt.show()
```

In [ ]:

```
#making predictions
threshold = 0.5
#test_img_number = random.randint(0, num_vals-1)
x, y=test_datagen.__next__()
for i in range(0,1):
  test_img=x[i]
  ground_truth=y[i]
  print(ground_truth.shape)

# test_img = X_test[test_img_number]
# ground_truth=y_test[test_img_number]
test_img_input=np.expand_dims(test_img, 0)
print(test_img_input.shape)
prediction = (model_pix2pix_semi.predict(test_img_input)[0,:,:,:] > 0.5).astype(np.uir
#prediction = (model.predict(test_img_input)[0,:,:,:])
print(prediction.shape)

plt.figure(figsize=(24, 24))
plt.subplot(231)
plt.title('Testing Image')
plt.imshow(test_img[:,:,:])
plt.subplot(232)
plt.title('Testing Label')
plt.imshow(ground_truth[:,:,0], cmap='gray')
plt.subplot(233)
plt.title('Prediction on test image')
plt.imshow(prediction[:,:,0], cmap='gray')
plt.subplot(234)

img = np.array(model_pix2pix_semi.predict(test_img_input)[0,:,:,:] )

plt.imshow(img, 'jet')
plt.show()
```

```
(256, 256, 1)
(1, 256, 256, 3)
(256, 256, 1)
```





Tiende a dar valores muy grandes en la mascara y por lo mismo suele dar faltos positivos.

FULL TUNING

In [ ]:
```
model_pix2pix_FULL  = keras.models.load_model('G:/Mi unidad/Demat/8 semestre/ML/Tarea
model_pix2pix_FULL.compile(optimizer=Adam(lr = 1e-3), loss='binary_crossentropy', metr
#model.output
model_pix2pix_FULL.summary()
```

```
WARNING:tensorflow:No training configuration found in the save file, so the model was
*not* compiled. Compile it manually.
Model: "model_45"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================
 input_43 (InputLayer)       [(None, 256, 256, 3)]     0         []

 model_43 (Functional)       [(None, 128, 128, 96),    1841984   ['input_43[0][0]']
                              (None, 64, 64, 144),
                              (None, 32, 32, 192),
                              (None, 16, 16, 576),
                              (None, 8, 8, 320)]

 sequential_56 (Sequential)  (None, 16, 16, 512)       165888    ['model_43[0][4]']

 concatenate_72 (Concatenate (None, 16, 16, 1088)      0         ['sequential_56[0][0]',
 )                                                                'model_43[0][3]']

 sequential_57 (Sequential)  (None, 32, 32, 256)       279552    ['concatenate_72[0][0]']

 concatenate_73 (Concatenate (None, 32, 32, 448)       0         ['sequential_57[0][0]',
 )                                                                'model_43[0][2]']

 sequential_58 (Sequential)  (None, 64, 64, 128)       57856     ['concatenate_73[0][0]']

 concatenate_74 (Concatenate (None, 64, 64, 272)       0         ['sequential_58[0][0]',
 )                                                                'model_43[0][1]']

 sequential_59 (Sequential)  (None, 128, 128, 64)      17664     ['concatenate_74[0][0]']

 concatenate_75 (Concatenate (None, 128, 128, 160)     0         ['sequential_59[0][0]',
 )                                                                'model_43[0][0]']

 conv2d_transpose_78 (Conv2DTra  (None, 256, 256, 3)   4323      ['concatenate_75[0][0]']
 nspose)

 conv2d_37 (Conv2D)          (None, 256, 256, 1)       28        ['conv2d_transpose_78[0][0]']

 conv2d_38 (Conv2D)          (None, 256, 256, 16)      80        ['conv2d_37[0][0]']

 dropout_9 (Dropout)         (None, 256, 256, 16)      0         ['conv2d_38[0][0]']
```

```
                                )

 conv2d_39 (Conv2D)             (None, 256, 256, 16  2320        ['dropout_9[0][0]']
                                )

 conv2d_40 (Conv2D)             (None, 256, 256, 1)  17          ['conv2d_39[0][0]']

==================================================================================
=============
Total params: 2,369,712
Trainable params: 2,336,880
Non-trainable params: 32,832
__________________________________________________________________________________
_____________
```

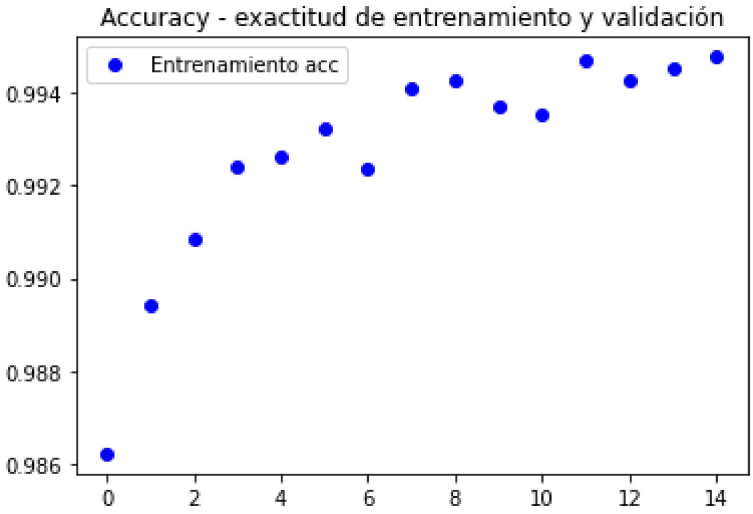
In [ ]:
```
import os
num_imges=len(os.listdir(train_images_path+"train/"))
num_vals=len(os.listdir(val_images_path+"val/"))

steps_per_epoch=num_imges//16
validation_steps=int(0.1 * num_vals//16)
steps_per_epoch, validation_steps

history_FULL = model_pix2pix_FULL.fit(train_datagen, validation_data=test_datagen, ste
                    )
model_pix2pix_FULL.save('G:/Mi unidad/Demat/8 semestre/ML/Tarea 4/Modelos/pix2pix_FULL
```

```
Epoch 1/15
190/190 [==============================] - 306s 2s/step - loss: 0.0457 - accuracy: 0.
9862 - val_loss: 0.0472 - val_accuracy: 0.9897
Epoch 2/15
190/190 [==============================] - 301s 2s/step - loss: 0.0348 - accuracy: 0.
9894 - val_loss: 0.2742 - val_accuracy: 0.9197
Epoch 3/15
190/190 [==============================] - 302s 2s/step - loss: 0.0295 - accuracy: 0.
9908 - val_loss: 0.1739 - val_accuracy: 0.9635
Epoch 4/15
190/190 [==============================] - 301s 2s/step - loss: 0.0240 - accuracy: 0.
9924 - val_loss: 0.1715 - val_accuracy: 0.9615
Epoch 5/15
190/190 [==============================] - 301s 2s/step - loss: 0.0234 - accuracy: 0.
9926 - val_loss: 0.2009 - val_accuracy: 0.9264
Epoch 6/15
190/190 [==============================] - 300s 2s/step - loss: 0.0211 - accuracy: 0.
9932 - val_loss: 0.0437 - val_accuracy: 0.9905
Epoch 7/15
190/190 [==============================] - 309s 2s/step - loss: 0.0243 - accuracy: 0.
9924 - val_loss: 0.3853 - val_accuracy: 0.8187
Epoch 8/15
190/190 [==============================] - 305s 2s/step - loss: 0.0178 - accuracy: 0.
9941 - val_loss: 0.3306 - val_accuracy: 0.8462
Epoch 9/15
190/190 [==============================] - 308s 2s/step - loss: 0.0171 - accuracy: 0.
9942 - val_loss: 0.4645 - val_accuracy: 0.7717
Epoch 10/15
190/190 [==============================] - 301s 2s/step - loss: 0.0180 - accuracy: 0.
9937 - val_loss: 0.1938 - val_accuracy: 0.9117
Epoch 11/15
190/190 [==============================] - 301s 2s/step - loss: 0.0191 - accuracy: 0.
9935 - val_loss: 0.4201 - val_accuracy: 0.7991
Epoch 12/15
190/190 [==============================] - 301s 2s/step - loss: 0.0157 - accuracy: 0.
9947 - val_loss: 0.2844 - val_accuracy: 0.8900
Epoch 13/15
190/190 [==============================] - 300s 2s/step - loss: 0.0165 - accuracy: 0.
9943 - val_loss: 0.0436 - val_accuracy: 0.9837
Epoch 14/15
190/190 [==============================] - 302s 2s/step - loss: 0.0160 - accuracy: 0.
9945 - val_loss: 0.0337 - val_accuracy: 0.9883
Epoch 15/15
190/190 [==============================] - 300s 2s/step - loss: 0.0159 - accuracy: 0.
9948 - val_loss: 0.0267 - val_accuracy: 0.9922
```

In [ ]:

```
import matplotlib.pyplot as plt

#1850

acc      = history_FULL.history['accuracy']
loss     = history_FULL.history['loss']
val_loss = history_FULL.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'bo', label='Entrenamiento acc')
plt.title('Accuracy - exactitud de entrenamiento y validación')
plt.legend()
```

```
plt.figure()

plt.plot(epochs, loss, 'bo', label='Entrenamiento loss')
plt.plot(epochs, val_loss, 'b', label='Validación loss')
plt.title('Loss - función objetivo en entrenamiento y prueba')
plt.legend()

plt.show()
```

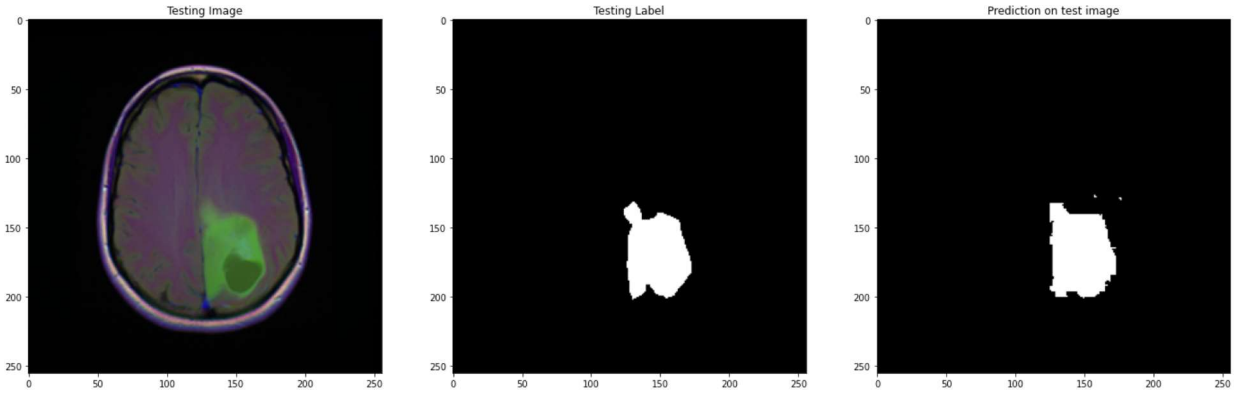


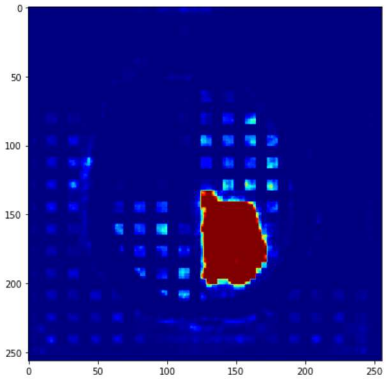
In [ ]:

```
#making predictions
threshold = 0.5
#test_img_number = random.randint(0, num_vals-1)
x, y=test_datagen.__next__()
for i in range(0,1):
  test_img=x[i]
  ground_truth=y[i]
  print(ground_truth.shape)

# test_img = X_test[test_img_number]
# ground_truth=y_test[test_img_number]
test_img_input=np.expand_dims(test_img, 0)
print(test_img_input.shape)
prediction = (model_pix2pix_FULL.predict(test_img_input)[0,:,:,:] > 0.5).astype(np.uir
#prediction = (model.predict(test_img_input)[0,:,:,:])
print(prediction.shape)
```

```
plt.figure(figsize=(24, 24))
plt.subplot(231)
plt.title('Testing Image')
plt.imshow(test_img[:,:,:])
plt.subplot(232)
plt.title('Testing Label')
plt.imshow(ground_truth[:,:,0], cmap='gray')
plt.subplot(233)
plt.title('Prediction on test image')
plt.imshow(prediction[:,:,0], cmap='gray')
plt.subplot(234)

img = np.array(model_pix2pix_FULL.predict(test_img_input)[0,:,:,:] )

plt.imshow(img, 'jet')
plt.show()
```

```
(256, 256, 1)
(1, 256, 256, 3)
(256, 256, 1)
```





Igual que antes suele dar valores mas grandes y se cuadricula el resultado. Aun que segmenta mejor los tumores.

El problema del "cuadriculado" que tiene los 3 casos puede que sea por el desencode que usamos que no coincide del todo con el encode que tiene la red base.