```python
#importing all necessary libraries
import os
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use("ggplot")
%matplotlib inline

from tqdm import tqdm_notebook, tnrange
from itertools import chain
from skimage.io import imread, imshow, concatenate_images
from skimage.transform import resize
from skimage.morphology import label
from sklearn.model_selection import train_test_split

import tensorflow as tf

from keras.models import Model, load_model
from keras.layers import Input, BatchNormalization, Activation, Dense, Dropout
from keras.layers.core import Lambda, RepeatVector, Reshape
from keras.layers.convolutional import Conv2D, Conv2DTranspose
from keras.layers.pooling import MaxPooling2D, GlobalMaxPool2D
```

```python
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
```

```python
#set some params , image width, height, border of image
im_width = 128
im_height = 128
border = 5
```

```python
#print the no.of images in training dataset
ids = next(os.walk("/content/drive/My Drive/pancreas/train/img-panc"))[2]
print("No. of images = ", len(ids))
```

```
No. of images =  6029
```

```python
#initialize array X, y with size of image height, width
X = np.zeros((len(ids), im_height, im_width, 1), dtype=np.float32)
y = np.zeros((len(ids), im_height, im_width, 1), dtype=np.float32)
```

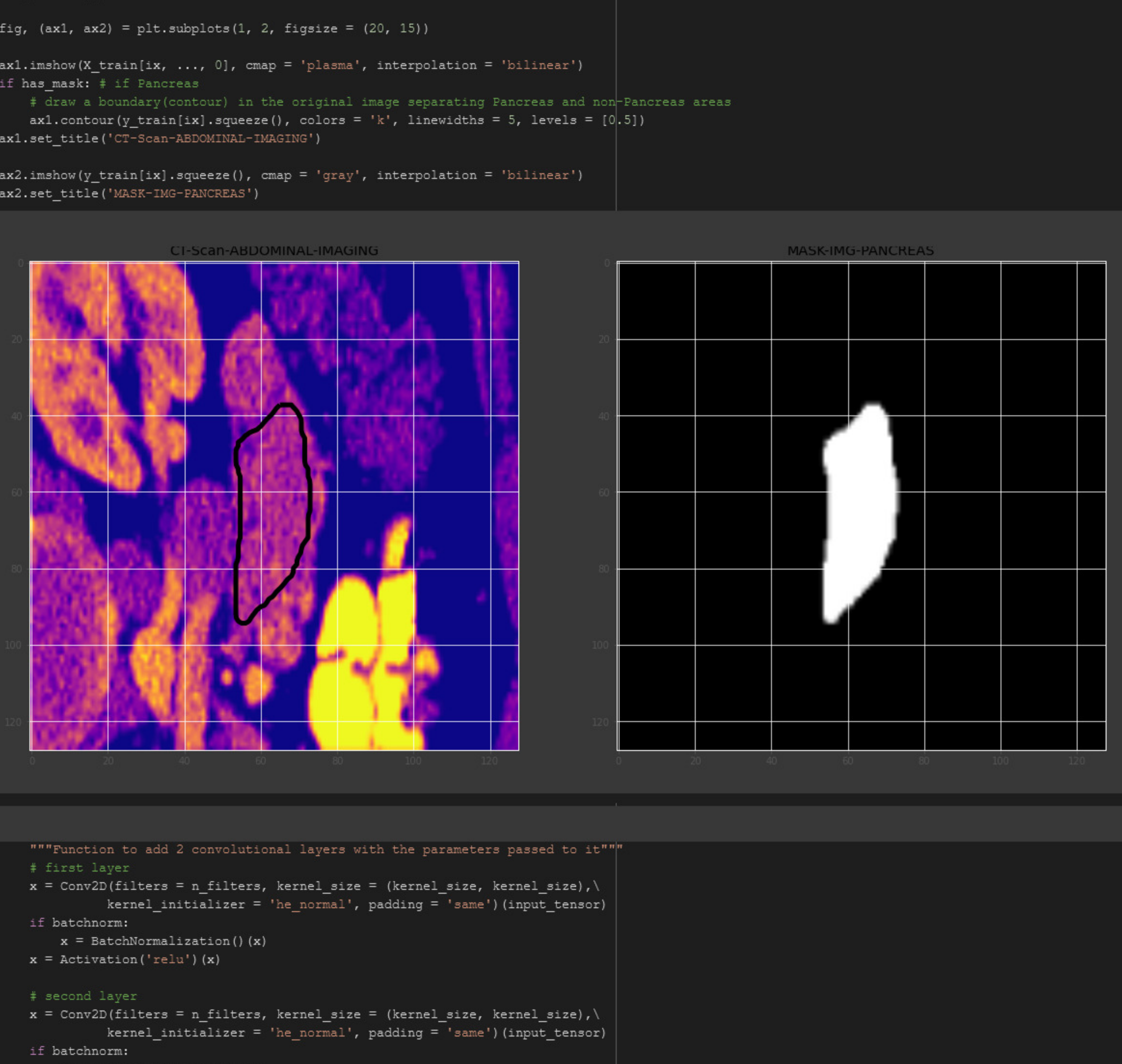
```python
# tqdm is used to display the progress bar
for n, id_ in tqdm_notebook(enumerate(ids), total=len(ids)):
    try:
      # Load images
      img = load_img("/content/drive/My Drive/pancreas/train/img-panc/{}".format(id_), color_mode="grayscale")
      x_img = img_to_array(img)
      x_img = resize(x_img, (128, 128, 1), mode = 'constant', preserve_range = True)
      # Load masks
      mask = img_to_array(load_img("/content/drive/My Drive/pancreas/train/mask-panc/{}".format(id_), color_mode="grayscale"))
```

```python
      X[n] = x_img/255.0
      y[n] = mask/255.0
    except:
      continue
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`

100% 6029/6029 [2:38:33<00:00, 1.58s/it]
```

```python
# Split train and valid
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.1, random_state=42)
```

```python
# Visualize any randome image along with the mask
ix = random.randint(0, len(X_train))
has_mask = y_train[ix].max() > 0 # Pancreas indicator

fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (20, 15))

ax1.imshow(X_train[ix, ..., 0], cmap = 'plasma', interpolation = 'bilinear')
if has_mask: # if Pancreas
    # draw a boundary(contour) in the original image separating Pancreas and non-Pancreas areas
    ax1.contour(y_train[ix].squeeze(), colors = 'k', linewidths = 5, levels = [0.5])
ax1.set_title('CT-Scan-ABDOMINAL-IMAGING')

ax2.imshow(y_train[ix].squeeze(), cmap = 'gray', interpolation = 'bilinear')
ax2.set_title('MASK-IMG-PANCREAS')
```




```python
    """Function to add 2 convolutional layers with the parameters passed to it"""
    # first layer
    x = Conv2D(filters = n_filters, kernel_size = (kernel_size, kernel_size),\
              kernel_initializer = 'he_normal', padding = 'same')(input_tensor)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # second layer
    x = Conv2D(filters = n_filters, kernel_size = (kernel_size, kernel_size),\
              kernel_initializer = 'he_normal', padding = 'same')(input_tensor)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)

    return x
```

```python
#standard u-net arch
def get_unet(input_img, n_filters = 16, dropout = 0.1, batchnorm = True):
    """Function to define the UNET Model"""
    # Contracting Path
    c1 = conv2d_block(input_img, n_filters * 1, kernel_size = 3, batchnorm = batchnorm)
    p1 = MaxPooling2D((2, 2))(c1)
    p1 = Dropout(dropout)(p1)

    c2 = conv2d_block(p1, n_filters * 2, kernel_size = 3, batchnorm = batchnorm)
    p2 = MaxPooling2D((2, 2))(c2)
    p2 = Dropout(dropout)(p2)

    c3 = conv2d_block(p2, n_filters * 4, kernel_size = 3, batchnorm = batchnorm)
```

```python
    c4 = conv2d_block(p3, n_filters * 8, kernel_size = 3, batchnorm = batchnorm)
    p4 = MaxPooling2D((2, 2))(c4)
    p4 = Dropout(dropout)(p4)

    c5 = conv2d_block(p4, n_filters = n_filters * 16, kernel_size = 3, batchnorm = batchnorm)

    # Expansive Path
    u6 = Conv2DTranspose(n_filters * 8, (3, 3), strides = (2, 2), padding = 'same')(c5)
    u6 = concatenate([u6, c4])
    u6 = Dropout(dropout)(u6)
    c6 = conv2d_block(u6, n_filters * 8, kernel_size = 3, batchnorm = batchnorm)

    u7 = Conv2DTranspose(n_filters * 4, (3, 3), strides = (2, 2), padding = 'same')(c6)
    u7 = concatenate([u7, c3])
    u7 = Dropout(dropout)(u7)
    c7 = conv2d_block(u7, n_filters * 4, kernel_size = 3, batchnorm = batchnorm)

    u8 = Conv2DTranspose(n_filters * 2, (3, 3), strides = (2, 2), padding = 'same')(c7)
    u8 = concatenate([u8, c2])
    u8 = Dropout(dropout)(u8)
    c8 = conv2d_block(u8, n_filters * 2, kernel_size = 3, batchnorm = batchnorm)

    u9 = Conv2DTranspose(n_filters * 1, (3, 3), strides = (2, 2), padding = 'same')(c8)
    u9 = concatenate([u9, c1])
    u9 = Dropout(dropout)(u9)
    c9 = conv2d_block(u9, n_filters * 1, kernel_size = 3, batchnorm = batchnorm)

    outputs = Conv2D(1, (1, 1), activation='sigmoid')(c9)
    model = Model(inputs=[input_img], outputs=[outputs])
```

```python
#compiling the model
input_img = Input((im_height, im_width, 1), name='img')
model = get_unet(input_img, n_filters=16, dropout=0.05, batchnorm=True)
model.compile(optimizer=Adam(), loss="binary_crossentropy", metrics=["accuracy"])
```

```python
model.summary()
```

```
conv2d_transpose (Conv2DTranspo (None, 16, 16, 128)  295040      activation_9[0][0]
concatenate (Concatenate)       (None, 16, 16, 256)  0           conv2d_transpose[0][0]
                                                                 activation_7[0][0]
dropout_4 (Dropout)             (None, 16, 16, 256)  0           concatenate[0][0]
conv2d_11 (Conv2D)              (None, 16, 16, 128)  295040      dropout_4[0][0]
batch_normalization_11 (BatchNo (None, 16, 16, 128)  512         conv2d_11[0][0]
activation_11 (Activation)      (None, 16, 16, 128)  0           batch_normalization_11[0][0]
conv2d_transpose_1 (Conv2DTrans (None, 32, 32, 64)   73792       activation_11[0][0]
concatenate_1 (Concatenate)     (None, 32, 32, 128)  0           conv2d_transpose_1[0][0]
                                                                 activation_5[0][0]
dropout_5 (Dropout)             (None, 32, 32, 128)  0           concatenate_1[0][0]
conv2d_13 (Conv2D)              (None, 32, 32, 64)   73792       dropout_5[0][0]
batch_normalization_13 (BatchNo (None, 32, 32, 64)   256         conv2d_13[0][0]
concatenate_2 (Concatenate)     (None, 64, 64, 64)   0           conv2d_transpose_2[0][0]
                                                                 activation_3[0][0]
dropout_6 (Dropout)             (None, 64, 64, 64)   0           concatenate_2[0][0]
conv2d_15 (Conv2D)              (None, 64, 64, 32)   18464       dropout_6[0][0]
batch_normalization_15 (BatchNo (None, 64, 64, 32)   128         conv2d_15[0][0]
activation_15 (Activation)      (None, 64, 64, 32)   0           batch_normalization_15[0][0]
conv2d_transpose_3 (Conv2DTrans (None, 128, 128, 16) 4624        activation_15[0][0]
concatenate_3 (Concatenate)     (None, 128, 128, 32) 0           conv2d_transpose_3[0][0]
                                                                 activation_1[0][0]
dropout_7 (Dropout)             (None, 128, 128, 32) 0           concatenate_3[0][0]
conv2d_17 (Conv2D)              (None, 128, 128, 16) 4624        dropout_7[0][0]
batch_normalization_17 (BatchNo (None, 128, 128, 16) 64          conv2d_17[0][0]
activation_17 (Activation)      (None, 128, 128, 16) 0           batch_normalization_17[0][0]
conv2d_18 (Conv2D)              (None, 128, 128, 1)  17          activation_17[0][0]
==================================================================================================
Total params: 1,179,121
Trainable params: 1,177,649
Non-trainable params: 1,472
```

```python
#creating a checkpoint to save the best model
```

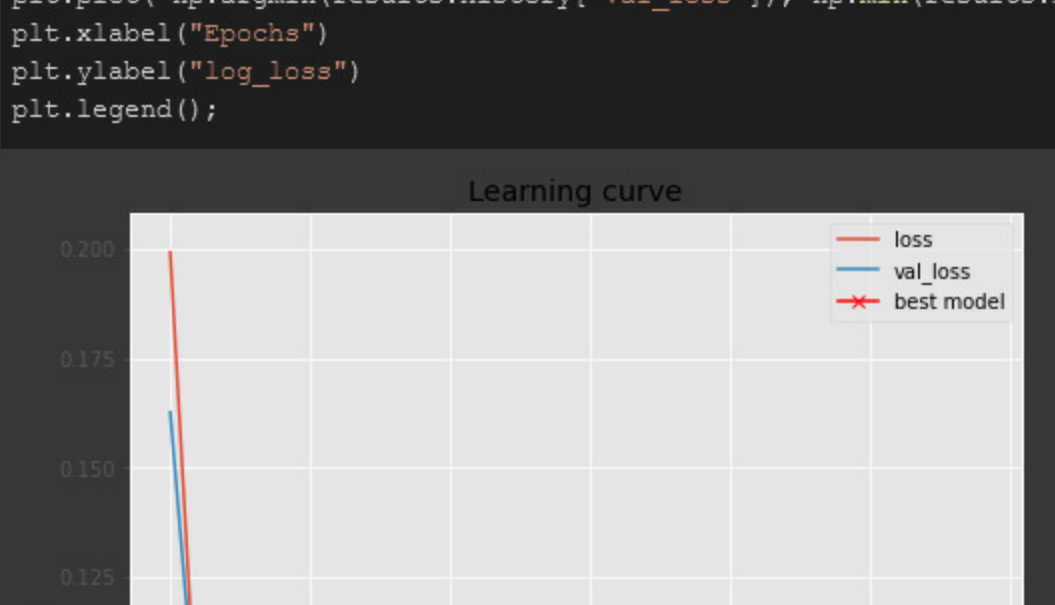
```python
    ReduceLROnPlateau(factor=0.1, patience=5, min_lr=0.00001, verbose=1),
    ModelCheckpoint('model-tgs-pancreas.h5', verbose=1, save_best_only=True, save_weights_only=True)
]
```

```python
#fit the data to the model by training it
results = model.fit(X_train, y_train, batch_size=32, epochs=30, callbacks=callbacks,\
                    validation_data=(X_valid, y_valid))
```

```
Epoch 00017: val_loss improved from 0.03349 to 0.03197, saving model to model-tgs-pancreas.h5
Epoch 18/30
170/170 [==============================] - 635s 4s/step - loss: 0.0235 - accuracy: 0.9750 - val_loss: 0.0326 - val_accuracy: 0.9719

Epoch 00018: val_loss did not improve from 0.03197
Epoch 19/30
170/170 [==============================] - 635s 4s/step - loss: 0.0232 - accuracy: 0.9751 - val_loss: 0.0334 - val_accuracy: 0.9721

Epoch 00019: val_loss did not improve from 0.03197
Epoch 20/30
170/170 [==============================] - 634s 4s/step - loss: 0.0227 - accuracy: 0.9752 - val_loss: 0.0345 - val_accuracy: 0.9719

Epoch 00020: val_loss did not improve from 0.03197
Epoch 21/30
170/170 [==============================] - 634s 4s/step - loss: 0.0232 - accuracy: 0.9748 - val_loss: 0.0346 - val_accuracy: 0.9717

Epoch 00021: val_loss did not improve from 0.03197
Epoch 22/30
170/170 [==============================] - 634s 4s/step - loss: 0.0222 - accuracy: 0.9753 - val_loss: 0.0332 - val_accuracy: 0.9720

Epoch 00022: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.

Epoch 00023: val_loss improved from 0.03197 to 0.03185, saving model to model-tgs-pancreas.h5
Epoch 24/30
170/170 [==============================] - 633s 4s/step - loss: 0.0196 - accuracy: 0.9760 - val_loss: 0.0319 - val_accuracy: 0.9727

Epoch 00024: val_loss did not improve from 0.03185
Epoch 25/30
170/170 [==============================] - 633s 4s/step - loss: 0.0193 - accuracy: 0.9761 - val_loss: 0.0322 - val_accuracy: 0.9727

Epoch 00025: val_loss did not improve from 0.03185
Epoch 26/30
170/170 [==============================] - 632s 4s/step - loss: 0.0188 - accuracy: 0.9764 - val_loss: 0.0320 - val_accuracy: 0.9727

Epoch 00026: val_loss did not improve from 0.03185
Epoch 27/30
170/170 [==============================] - 632s 4s/step - loss: 0.0188 - accuracy: 0.9763 - val_loss: 0.0320 - val_accuracy: 0.9727

Epoch 00027: val_loss did not improve from 0.03185
Epoch 28/30
170/170 [==============================] - 632s 4s/step - loss: 0.0186 - accuracy: 0.9764 - val_loss: 0.0324 - val_accuracy: 0.9727

Epoch 00028: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.

Epoch 00028: val_loss did not improve from 0.03185
Epoch 29/30
170/170 [==============================] - 633s 4s/step - loss: 0.0187 - accuracy: 0.9761 - val_loss: 0.0323 - val_accuracy: 0.9727

Epoch 00029: val_loss did not improve from 0.03185
Epoch 30/30
170/170 [==============================] - 632s 4s/step - loss: 0.0183 - accuracy: 0.9764 - val_loss: 0.0324 - val_accuracy: 0.9727

Epoch 00030: val_loss did not improve from 0.03185
```

```
drive  model-tgs-pancreas.h5  sample_data
```

```python
#plotting the models loss , validation loss
plt.figure(figsize=(8, 8))
plt.title("Learning curve")
plt.plot(results.history["loss"], label="loss")
plt.plot(results.history["val_loss"], label="val_loss")
plt.plot( np.argmin(results.history["val_loss"]), np.min(results.history["val_loss"]), marker="x", color="r", label="best model")
plt.xlabel("Epochs")
plt.ylabel("log_loss")
plt.legend();
```




```python
#saving the model to access frequently
model.save('drive/My Drive/pancreas/train/pancreas_model.h5')
```

```python
#loading the model to use it in the process
model = load_model('drive/My Drive/pancreas/train/pancreas_model.h5')
```

```python
# load the best model
model.load_weights('drive/My Drive/pancreas/train/model-tgs-pancreas.h5')
```

```python
# Evaluate on validation set (this must be equals to the best log_loss)
model.evaluate(X_valid, y_valid, verbose=1)
```

```
19/19 [==============================] - 16s 815ms/step - loss: 0.0319 - accuracy: 0.9726
[0.03185439109802246, 0.9725990891456604]
```

```python
# Predict on train, val and test
preds_train = model.predict(X_train, verbose=1)
preds_val = model.predict(X_valid, verbose=1)
```

```
170/170 [==============================] - 140s 823ms/step
```

```python
# Threshold predictions
preds_train_t = (preds_train > 0.5).astype(np.uint8)
preds_val_t = (preds_val > 0.5).astype(np.uint8)
```

```python
#function to plot the data and to check the predictions.
def plot_sample(X, y, preds, binary_preds, ix=None):
    """Function to plot the results"""
    if ix is None:
        ix = random.randint(0, len(X))

    has_mask = y[ix].max() > 0

    fig, ax = plt.subplots(1, 4, figsize=(20, 10))
    ax[0].imshow(X[ix, ..., 0], cmap='seismic')
    if has_mask:
        ax[0].contour(y[ix].squeeze(), colors='k', levels=[0.5])
    ax[0].set_title('Seismic - CT_Scan-IMG-PANCREAS')

    ax[1].imshow(y[ix].squeeze())
    ax[1].set_title('PANCREAS')

    ax[2].imshow(preds[ix].squeeze(), vmin=0, vmax=1)
    if has_mask:
        ax[2].contour(y[ix].squeeze(), colors='k', levels=[0.5])
    ax[2].set_title('PANCREAS Segmentation Prediction')

    ax[3].imshow(binary_preds[ix].squeeze(), vmin=0, vmax=1)
    if has_mask:
        ax[3].contour(y[ix].squeeze(), colors='k', levels=[0.5])
```

```python
# Check if training data looks all right
plot_sample(X_train, y_train, preds_train, preds_train_t, ix=14)
```


```python
# Check if valid data looks all right
plot_sample(X_valid, y_valid, preds_val, preds_val_t, ix=19)
```


```python
#ploting a random sample from validation set
plot_sample(X_valid, y_valid, preds_val, preds_val_t)
```