blood-cancer-semantic-segment-tf-unet(G)

March 7, 2023

[2]:
```
#Libraries------------------
import os
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
import gc
from tqdm.notebook import trange, tqdm
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 glob
import cv2
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator,␣
 ↪array_to_img, img_to_array, load_img
from tensorflow.keras.layers import Conv2D, Input, MaxPooling2D, Dropout,␣
 ↪concatenate, UpSampling2D
from tensorflow.keras.models import load_model, Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint,␣
 ↪ReduceLROnPlateau, TensorBoard
from tensorflow.keras import backend as K
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import (BatchNormalization, Conv2DTranspose,
                                     SeparableConv2D, MaxPooling2D, Activation,␣
 ↪Flatten, Dropout, Dense)
from tensorflow.keras.preprocessing.image import load_img, array_to_img,␣
 ↪img_to_array
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Conv2D, LeakyReLU, BatchNormalization,␣
 ↪MaxPool2D,Conv2DTranspose, concatenate,Input
from tensorflow.keras.callbacks import CSVLogger
import warnings
```

```
warnings.filterwarnings("ignore")
K.clear_session()
warnings.filterwarnings('ignore')
plt.style.use("ggplot")
get_ipython().run_line_magic('matplotlib', 'inline')
```

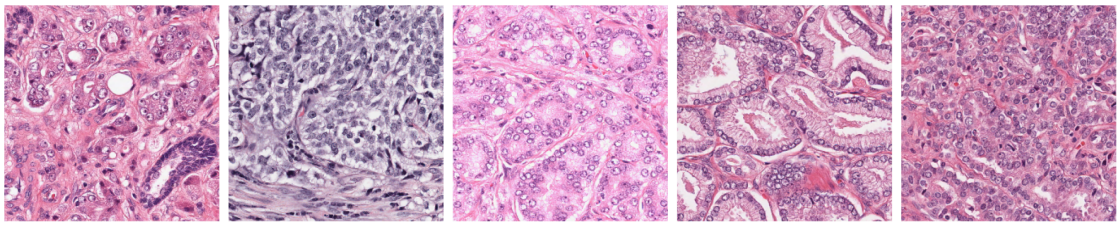
[3]:
```
def load_image(image, SIZE):
    return np.round(tfi.resize(img_to_array(load_img(image))/255.,(SIZE,␣
 ↪SIZE)),4)

def load_images(image_paths, SIZE, mask=False, trim=None):
    if trim is not None:
        image_paths = image_paths[:trim]

    if mask:
        images = np.zeros(shape=(len(image_paths), SIZE, SIZE, 1))
    else:
        images = np.zeros(shape=(len(image_paths), SIZE, SIZE, 3))

    for i,image in enumerate(image_paths):
        img = load_image(image,SIZE)
        if mask:
            images[i] = img[:,:,:1]
        else:
            images[i] = img

    return images
```

[8]:
```
#Load image data-------------------
w,h,ch=256,256,3
def load_img(path):
    img= cv2.imread(path)
    img= cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img=cv2.resize(img,(w,h))
    return img
```

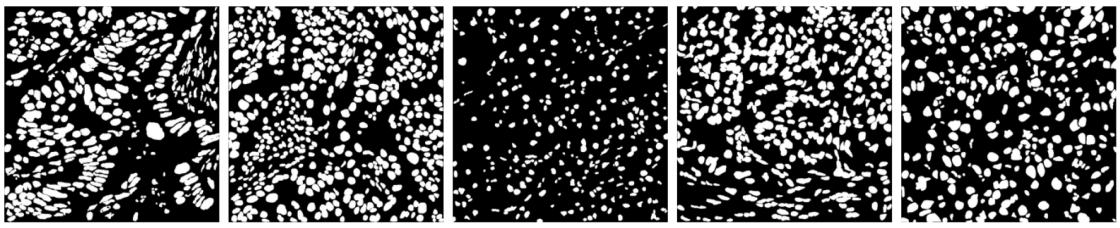
[9]:
```
#Load data----------------------
BASE_DIR="Monuseg/train/"
img_path= os.listdir(BASE_DIR+'images')
mask_path= os.listdir(BASE_DIR+'masks')
```

[10]:
```
#plot sample images--------------
fig, ax= plt.subplots(1,5, figsize=(20, 10))
for i in range(5):
    path= BASE_DIR + 'images/'
    ax[i].imshow(load_img(path + img_path[i]))
    ax[i].set_xticks([]); ax[i].set_yticks([])
```

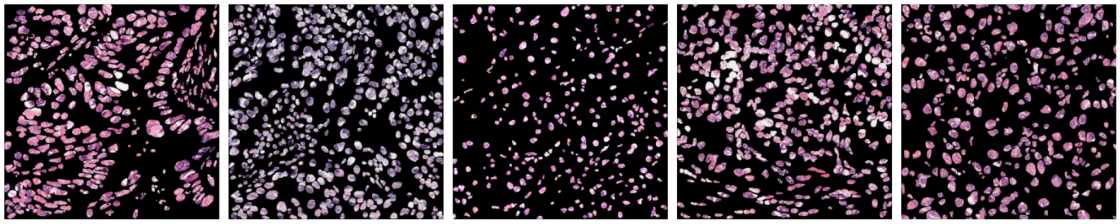
```
fig.tight_layout()
plt.show()
```



[11]:
```
#plot sample masks--------------
fig, ax= plt.subplots(1,5, figsize=(20, 10))
for i in range(5):
    path= BASE_DIR + 'masks/'
    ax[i].imshow(load_img(path + mask_path[i])[:, :, 0], 'gray')
    ax[i].set_xticks([]); ax[i].set_yticks([])

fig.tight_layout()
plt.show()
```



[12]:
```
#plot sample images--with blended mask ------------
fig, ax= plt.subplots(1,5, figsize=(20, 10))
for i in range(5):
    path1= BASE_DIR + 'images/'
    ax[i].imshow((load_img(path1 + img_path[i])/255) * (load_img(path +␣
 ↪mask_path[i])/255))
    ax[i].set_xticks([]); ax[i].set_yticks([])

fig.tight_layout()
plt.show()
```



[13]:
```
#data  preparation
X_train, X_test, y_train, y_test = train_test_split(img_path, mask_path,␣
 ↪test_size=0.2, random_state=22)
len(X_train), len(X_test)
```

[13]:
```
(19, 5)
```

[14]:
```
#batch generation----------------------
def load_data(path_list, gray=False):
    data=[]
    for path in tqdm(path_list):
        img= load_img(path)
        if gray:
            img= img[:, :, 0:1]
        img= cv2.resize(img, (w, h))
        data.append(img)
    return np.array(data)
```

[15]:
```
#train data generation---------------------
X_train= load_data([BASE_DIR + 'images/' + x for x in X_train])/255.0
X_test= load_data([BASE_DIR + 'images/' + x for x in X_test])/255.0

X_train.shape, X_test.shape
```

```
  0%|          | 0/19 [00:00<?, ?it/s]
  0%|          | 0/5 [00:00<?, ?it/s]
```

[15]:
```
((19, 256, 256, 3), (5, 256, 256, 3))
```

[16]:
```
##test data generation---------------------
Y_train= load_data([BASE_DIR + 'masks/' + x for x in y_train], gray=True)/255.0
Y_test= load_data([BASE_DIR + 'masks/' + x for x in y_test], gray=True)/255.0
Y_train= Y_train.reshape(-1, w, h, 1)
Y_test= Y_test.reshape(-1, w, h, 1)

Y_train.shape, Y_test.shape
```

```
  0%|          | 0/19 [00:00<?, ?it/s]
  0%|          | 0/5 [00:00<?, ?it/s]
```

[16]:
```
((19, 256, 256, 1), (5, 256, 256, 1))
```

[17]:
```
# Convolutional blocks-----------------

def conv_block(inputs,n_filters,max_pool=True):

    x =␣
 ↪Conv2D(n_filters,3,padding='same',kernel_initializer='he_normal',use_bias=False)(inputs)
    x = BatchNormalization()(x)
    x = LeakyReLU(0.2)(x)
    x =␣
 ↪Conv2D(n_filters,3,padding='same',kernel_initializer='he_normal',use_bias=False)(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(0.2)(x)
    skip = x

    if max_pool:
        next_layer = MaxPool2D()(x)
    else:
        next_layer = x
    return next_layer, skip
```

[18]:
```
# upsampling block
def up_block(reg_inputs,skip_inputs,n_filters):

    x = Conv2DTranspose(n_filters,3,2,padding='same')(reg_inputs)
    x = concatenate([x,skip_inputs],axis=3)
    x = conv_block(x,n_filters,max_pool=False)[0]

    return x
```

[19]:
```
# Unet model---------------------

def unet(input_size=(h,w,ch),number_of_classes=1):
    inputs = Input(shape=input_size)
    cb1 = conv_block(inputs,32)
    cb2 = conv_block(cb1[0],64)
    cb3 = conv_block(cb2[0],128)
    cb4 = conv_block(cb3[0],256)
    cb5 = conv_block(cb4[0],512,max_pool=False)

    up1 = up_block(cb5[0],cb4[1],256)
    up2 = up_block(up1,cb3[1],128)
    up3 = up_block(up2,cb2[1],64)
```

```
    up4 = up_block(up3,cb1[1],32)

    conv1 =␣
 ↪Conv2D(32,3,padding='same',kernel_initializer='he_normal',use_bias=False)(up4)
    bn = BatchNormalization()(conv1)
    lrl = LeakyReLU(0.2)(bn)
    outputs =␣
 ↪Conv2D(number_of_classes,1,padding='same',activation='sigmoid')(lrl)

    unet = Model(inputs=inputs,outputs=outputs)
    return unet
```

[20]:
```
# IoU loss-------------

def jaccard_coef(y_true,y_pred,smooth=1):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f*y_pred_f)
    return (intersection+smooth)/(K.sum(y_true_f)+K.
 ↪sum(y_pred_f)-intersection+smooth)

# Jaccard loss-----------------
def jaccard_loss(y_true,y_pred,smooth=1):

    return -jaccard_coef(y_true,y_pred,smooth)
```

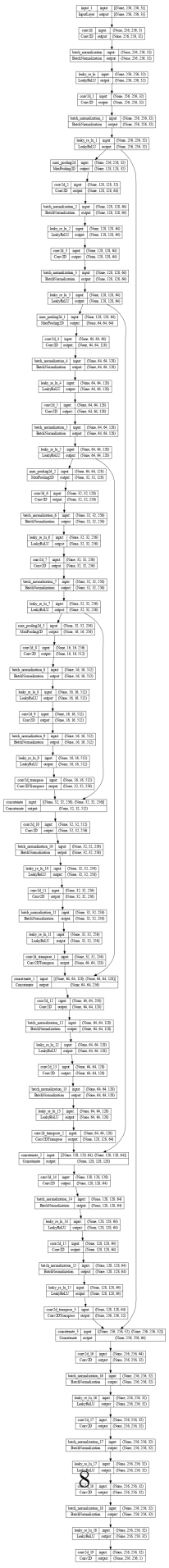
[21]:
```
# Defining our model----------------
model = unet()
```

```
2023-03-07 04:35:04.440022: E
tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:267] failed call to
cuInit: CUDA_ERROR_NO_DEVICE: no CUDA-capable device is detected
2023-03-07 04:35:04.451467: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_diagnostics.cc:156] kernel
driver does not appear to be running on this host (picox):
/proc/driver/nvidia/version does not exist
2023-03-07 04:35:04.506887: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

[22]:
```
# Plotting  model---------------------------
tf.keras.utils.plot_model(
    model,
    to_file="model.png",
```

```
    show_shapes=True,
    show_layer_names=True,
    dpi=60
)
```

[22]:


input_1 InputLayer input: [(None, 256, 256, 3)] output: [(None, 256, 256, 3)]
conv2d Conv2D input: (None, 256, 256, 3) output: (None, 254, 254, 32)
batch_normalization BatchNormalization input: (None, 254, 254, 32) output: (None, 254, 254, 32)
leaky_re_lu LeakyReLU input: (None, 254, 254, 32) output: (None, 254, 254, 32)
conv2d_1 Conv2D input: (None, 254, 254, 32) output: (None, 254, 254, 32)
batch_normalization_1 BatchNormalization input: (None, 254, 254, 32) output: (None, 254, 254, 32)
leaky_re_lu_1 LeakyReLU input: (None, 254, 254, 32) output: (None, 254, 254, 32)
max_pooling2d MaxPooling2D input: (None, 254, 254, 32) output: (None, 128, 128, 32)
conv2d_2 Conv2D input: (None, 128, 128, 32) output: (None, 128, 128, 64)
batch_normalization_2 BatchNormalization input: (None, 128, 128, 64) output: (None, 128, 128, 64)
leaky_re_lu_2 LeakyReLU input: (None, 128, 128, 64) output: (None, 128, 128, 64)
conv2d_3 Conv2D input: (None, 128, 128, 64) output: (None, 128, 128, 64)
batch_normalization_3 BatchNormalization input: (None, 128, 128, 64) output: (None, 128, 128, 64)
leaky_re_lu_3 LeakyReLU input: (None, 128, 128, 64) output: (None, 128, 128, 64)
max_pooling2d_1 MaxPooling2D input: (None, 128, 128, 64) output: (None, 64, 64, 64)
conv2d_4 Conv2D input: (None, 64, 64, 64) output: (None, 64, 64, 128)
batch_normalization_4 BatchNormalization input: (None, 64, 64, 128) output: (None, 64, 64, 128)
leaky_re_lu_4 LeakyReLU input: (None, 64, 64, 128) output: (None, 64, 64, 128)
conv2d_5 Conv2D input: (None, 64, 64, 128) output: (None, 64, 64, 128)
batch_normalization_5 BatchNormalization input: (None, 64, 64, 128) output: (None, 64, 64, 128)
leaky_re_lu_5 LeakyReLU input: (None, 64, 64, 128) output: (None, 64, 64, 128)
max_pooling2d_2 MaxPooling2D input: (None, 64, 64, 128) output: (None, 32, 32, 128)
conv2d_6 Conv2D input: (None, 32, 32, 128) output: (None, 32, 32, 256)
batch_normalization_6 BatchNormalization input: (None, 32, 32, 256) output: (None, 32, 32, 256)
leaky_re_lu_6 LeakyReLU input: (None, 32, 32, 256) output: (None, 32, 32, 256)
conv2d_7 Conv2D input: (None, 32, 32, 256) output: (None, 32, 32, 256)
batch_normalization_7 BatchNormalization input: (None, 32, 32, 256) output: (None, 32, 32, 256)
leaky_re_lu_7 LeakyReLU input: (None, 32, 32, 256) output: (None, 32, 32, 256)
max_pooling2d_3 MaxPooling2D input: (None, 32, 32, 256) output: (None, 16, 16, 256)
conv2d_8 Conv2D input: (None, 16, 16, 256) output: (None, 16, 16, 512)
batch_normalization_8 BatchNormalization input: (None, 16, 16, 512) output: (None, 16, 16, 512)
leaky_re_lu_8 LeakyReLU input: (None, 16, 16, 512) output: (None, 16, 16, 512)
conv2d_9 Conv2D input: (None, 16, 16, 512) output: (None, 16, 16, 512)
batch_normalization_9 BatchNormalization input: (None, 16, 16, 512) output: (None, 16, 16, 512)
leaky_re_lu_9 LeakyReLU input: (None, 16, 16, 512) output: (None, 16, 16, 512)
conv2d_transpose Conv2DTranspose input: (None, 16, 16, 512) output: (None, 32, 32, 256)
concatenate Concatenate input: [(None, 32, 32, 256), (None, 32, 32, 256)] output: (None, 32, 32, 512)
conv2d_10 Conv2D input: (None, 32, 32, 512) output: (None, 32, 32, 256)
batch_normalization_10 BatchNormalization input: (None, 32, 32, 256) output: (None, 32, 32, 256)
leaky_re_lu_10 LeakyReLU input: (None, 32, 32, 256) output: (None, 32, 32, 256)
conv2d_11 Conv2D input: (None, 32, 32, 256) output: (None, 32, 32, 256)
batch_normalization_11 BatchNormalization input: (None, 32, 32, 256) output: (None, 32, 32, 256)
leaky_re_lu_11 LeakyReLU input: (None, 32, 32, 256) output: (None, 32, 32, 256)
conv2d_transpose_1 Conv2DTranspose input: (None, 32, 32, 256) output: (None, 64, 64, 128)
concatenate_1 Concatenate input: [(None, 64, 64, 128), (None, 64, 64, 128)] output: (None, 64, 64, 256)
conv2d_12 Conv2D input: (None, 64, 64, 256) output: (None, 64, 64, 128)
batch_normalization_12 BatchNormalization input: (None, 64, 64, 128) output: (None, 64, 64, 128)
leaky_re_lu_12 LeakyReLU input: (None, 64, 64, 128) output: (None, 64, 64, 128)
conv2d_13 Conv2D input: (None, 64, 64, 128) output: (None, 64, 64, 128)
batch_normalization_13 BatchNormalization input: (None, 64, 64, 128) output: (None, 64, 64, 128)
leaky_re_lu_13 LeakyReLU input: (None, 64, 64, 128) output: (None, 64, 64, 128)
conv2d_transpose_2 Conv2DTranspose input: (None, 64, 64, 128) output: (None, 128, 128, 64)
concatenate_2 Concatenate input: [(None, 128, 128, 64), (None, 128, 128, 64)] output: (None, 128, 128, 128)
conv2d_14 Conv2D input: (None, 128, 128, 128) output: (None, 128, 128, 64)
batch_normalization_14 BatchNormalization input: (None, 128, 128, 64) output: (None, 128, 128, 64)
leaky_re_lu_14 LeakyReLU input: (None, 128, 128, 64) output: (None, 128, 128, 64)
conv2d_15 Conv2D input: (None, 128, 128, 64) output: (None, 128, 128, 64)
batch_normalization_15 BatchNormalization input: (None, 128, 128, 64) output: (None, 128, 128, 64)
leaky_re_lu_15 LeakyReLU input: (None, 128, 128, 64) output: (None, 128, 128, 64)
conv2d_transpose_3 Conv2DTranspose input: (None, 128, 128, 64) output: (None, 256, 256, 32)
concatenate_3 Concatenate input: [(None, 256, 256, 32), (None, 256, 256, 32)] output: (None, 256, 256, 64)
conv2d_16 Conv2D input: (None, 256, 256, 64) output: (None, 256, 256, 32)
batch_normalization_16 BatchNormalization input: (None, 256, 256, 32) output: (None, 256, 256, 32)
leaky_re_lu_16 LeakyReLU input: (None, 256, 256, 32) output: (None, 256, 256, 32)
conv2d_17 Conv2D input: (None, 256, 256, 32) output: (None, 256, 256, 32)
batch_normalization_17 BatchNormalization input: (None, 256, 256, 32) output: (None, 256, 256, 32)
leaky_re_lu_17 LeakyReLU input: (None, 256, 256, 32) output: (None, 256, 256, 32)
conv2d_18 Conv2D input: (None, 256, 256, 32) output: (None, 256, 256, 32)
batch_normalization_18 BatchNormalization input: (None, 256, 256, 32) output: (None, 256, 256, 32)
leaky_re_lu_18 LeakyReLU input: (None, 256, 256, 32) output: (None, 256, 256, 32)
conv2d_19 Conv2D input: (None, 256, 256, 32) output: (None, 256, 256, 3)

[23]:
```
#Summary of model------------------
model.summary()
```

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

 conv2d (Conv2D)                (None, 256, 256, 32  864
['input_1[0][0]']
                                )

 batch_normalization (BatchNorm  (None, 256, 256, 32  128
['conv2d[0][0]']
 alization)                     )

 leaky_re_lu (LeakyReLU)        (None, 256, 256, 32  0
['batch_normalization[0][0]']
                                )

 conv2d_1 (Conv2D)              (None, 256, 256, 32  9216
['leaky_re_lu[0][0]']
                                )

 batch_normalization_1 (BatchNo  (None, 256, 256, 32  128
['conv2d_1[0][0]']
 rmalization)                   )

 leaky_re_lu_1 (LeakyReLU)      (None, 256, 256, 32  0
['batch_normalization_1[0][0]']
                                )

 max_pooling2d (MaxPooling2D)   (None, 128, 128, 32  0
['leaky_re_lu_1[0][0]']
                                )

 conv2d_2 (Conv2D)              (None, 128, 128, 64  18432
['max_pooling2d[0][0]']
                                )

 batch_normalization_2 (BatchNo  (None, 128, 128, 64  256
```

```
['conv2d_2[0][0]']
 rmalization)                   )

 leaky_re_lu_2 (LeakyReLU)      (None, 128, 128, 64  0
['batch_normalization_2[0][0]']
                                )

 conv2d_3 (Conv2D)              (None, 128, 128, 64  36864
['leaky_re_lu_2[0][0]']
                                )

 batch_normalization_3 (BatchNo  (None, 128, 128, 64  256
['conv2d_3[0][0]']
 rmalization)                   )

 leaky_re_lu_3 (LeakyReLU)      (None, 128, 128, 64  0
['batch_normalization_3[0][0]']
                                )

 max_pooling2d_1 (MaxPooling2D)  (None, 64, 64, 64)  0
['leaky_re_lu_3[0][0]']

 conv2d_4 (Conv2D)              (None, 64, 64, 128)  73728
['max_pooling2d_1[0][0]']

 batch_normalization_4 (BatchNo  (None, 64, 64, 128)  512
['conv2d_4[0][0]']
 rmalization)

 leaky_re_lu_4 (LeakyReLU)      (None, 64, 64, 128)  0
['batch_normalization_4[0][0]']

 conv2d_5 (Conv2D)              (None, 64, 64, 128)  147456
['leaky_re_lu_4[0][0]']

 batch_normalization_5 (BatchNo  (None, 64, 64, 128)  512
['conv2d_5[0][0]']
 rmalization)

 leaky_re_lu_5 (LeakyReLU)      (None, 64, 64, 128)  0
['batch_normalization_5[0][0]']

 max_pooling2d_2 (MaxPooling2D)  (None, 32, 32, 128)  0
['leaky_re_lu_5[0][0]']

 conv2d_6 (Conv2D)              (None, 32, 32, 256)  294912
['max_pooling2d_2[0][0]']
```

```
 batch_normalization_6 (BatchNo  (None, 32, 32, 256)  1024
['conv2d_6[0][0]']
 rmalization)

 leaky_re_lu_6 (LeakyReLU)      (None, 32, 32, 256)  0
['batch_normalization_6[0][0]']

 conv2d_7 (Conv2D)              (None, 32, 32, 256)  589824
['leaky_re_lu_6[0][0]']

 batch_normalization_7 (BatchNo  (None, 32, 32, 256)  1024
['conv2d_7[0][0]']
 rmalization)

 leaky_re_lu_7 (LeakyReLU)      (None, 32, 32, 256)  0
['batch_normalization_7[0][0]']

 max_pooling2d_3 (MaxPooling2D)  (None, 16, 16, 256)  0
['leaky_re_lu_7[0][0]']

 conv2d_8 (Conv2D)              (None, 16, 16, 512)  1179648
['max_pooling2d_3[0][0]']

 batch_normalization_8 (BatchNo  (None, 16, 16, 512)  2048
['conv2d_8[0][0]']
 rmalization)

 leaky_re_lu_8 (LeakyReLU)      (None, 16, 16, 512)  0
['batch_normalization_8[0][0]']

 conv2d_9 (Conv2D)              (None, 16, 16, 512)  2359296
['leaky_re_lu_8[0][0]']

 batch_normalization_9 (BatchNo  (None, 16, 16, 512)  2048
['conv2d_9[0][0]']
 rmalization)

 leaky_re_lu_9 (LeakyReLU)      (None, 16, 16, 512)  0
['batch_normalization_9[0][0]']

 conv2d_transpose (Conv2DTransp  (None, 32, 32, 256)  1179904
['leaky_re_lu_9[0][0]']
 ose)

 concatenate (Concatenate)      (None, 32, 32, 512)  0
['conv2d_transpose[0][0]',
'leaky_re_lu_7[0][0]']
```

```
 conv2d_10 (Conv2D)             (None, 32, 32, 256)  1179648
['concatenate[0][0]']

 batch_normalization_10 (BatchN  (None, 32, 32, 256)  1024
['conv2d_10[0][0]']
 ormalization)

 leaky_re_lu_10 (LeakyReLU)     (None, 32, 32, 256)  0
['batch_normalization_10[0][0]']

 conv2d_11 (Conv2D)             (None, 32, 32, 256)  589824
['leaky_re_lu_10[0][0]']

 batch_normalization_11 (BatchN  (None, 32, 32, 256)  1024
['conv2d_11[0][0]']
 ormalization)

 leaky_re_lu_11 (LeakyReLU)     (None, 32, 32, 256)  0
['batch_normalization_11[0][0]']

 conv2d_transpose_1 (Conv2DTran  (None, 64, 64, 128)  295040
['leaky_re_lu_11[0][0]']
 spose)

 concatenate_1 (Concatenate)    (None, 64, 64, 256)  0
['conv2d_transpose_1[0][0]',
'leaky_re_lu_5[0][0]']

 conv2d_12 (Conv2D)             (None, 64, 64, 128)  294912
['concatenate_1[0][0]']

 batch_normalization_12 (BatchN  (None, 64, 64, 128)  512
['conv2d_12[0][0]']
 ormalization)

 leaky_re_lu_12 (LeakyReLU)     (None, 64, 64, 128)  0
['batch_normalization_12[0][0]']

 conv2d_13 (Conv2D)             (None, 64, 64, 128)  147456
['leaky_re_lu_12[0][0]']

 batch_normalization_13 (BatchN  (None, 64, 64, 128)  512
['conv2d_13[0][0]']
 ormalization)

 leaky_re_lu_13 (LeakyReLU)     (None, 64, 64, 128)  0
['batch_normalization_13[0][0]']
```

```
 conv2d_transpose_2 (Conv2DTran  (None, 128, 128, 64  73792
['leaky_re_lu_13[0][0]']
 spose)                         )

 concatenate_2 (Concatenate)    (None, 128, 128, 12  0
['conv2d_transpose_2[0][0]',
                                8)
'leaky_re_lu_3[0][0]']

 conv2d_14 (Conv2D)             (None, 128, 128, 64  73728
['concatenate_2[0][0]']
                                )

 batch_normalization_14 (BatchN  (None, 128, 128, 64  256
['conv2d_14[0][0]']
 ormalization)                  )

 leaky_re_lu_14 (LeakyReLU)     (None, 128, 128, 64  0
['batch_normalization_14[0][0]']
                                )

 conv2d_15 (Conv2D)             (None, 128, 128, 64  36864
['leaky_re_lu_14[0][0]']
                                )

 batch_normalization_15 (BatchN  (None, 128, 128, 64  256
['conv2d_15[0][0]']
 ormalization)                  )

 leaky_re_lu_15 (LeakyReLU)     (None, 128, 128, 64  0
['batch_normalization_15[0][0]']
                                )

 conv2d_transpose_3 (Conv2DTran  (None, 256, 256, 32  18464
['leaky_re_lu_15[0][0]']
 spose)                         )

 concatenate_3 (Concatenate)    (None, 256, 256, 64  0
['conv2d_transpose_3[0][0]',
                                )
'leaky_re_lu_1[0][0]']

 conv2d_16 (Conv2D)             (None, 256, 256, 32  18432
['concatenate_3[0][0]']
                                )

 batch_normalization_16 (BatchN  (None, 256, 256, 32  128
['conv2d_16[0][0]']
```

```
 ormalization)                  )

 leaky_re_lu_16 (LeakyReLU)     (None, 256, 256, 32  0
['batch_normalization_16[0][0]']
                                )

 conv2d_17 (Conv2D)             (None, 256, 256, 32  9216
['leaky_re_lu_16[0][0]']
                                )

 batch_normalization_17 (BatchN  (None, 256, 256, 32  128
['conv2d_17[0][0]']
 ormalization)                  )

 leaky_re_lu_17 (LeakyReLU)     (None, 256, 256, 32  0
['batch_normalization_17[0][0]']
                                )

 conv2d_18 (Conv2D)             (None, 256, 256, 32  9216
['leaky_re_lu_17[0][0]']
                                )

 batch_normalization_18 (BatchN  (None, 256, 256, 32  128
['conv2d_18[0][0]']
 ormalization)                  )

 leaky_re_lu_18 (LeakyReLU)     (None, 256, 256, 32  0
['batch_normalization_18[0][0]']
                                )

 conv2d_19 (Conv2D)             (None, 256, 256, 1)  33
['leaky_re_lu_18[0][0]']

==================================================================================
==================
Total params: 8,648,673
Trainable params: 8,642,721
Non-trainable params: 5,952
----------------------------------------------------------------------------------
------------------
```

[24]:
```
#compile model--------------------
model.compile(optimizer=tf.keras.optimizers.
  ↪Adam(1e-3),loss=[jaccard_loss],metrics=[jaccard_coef])
```

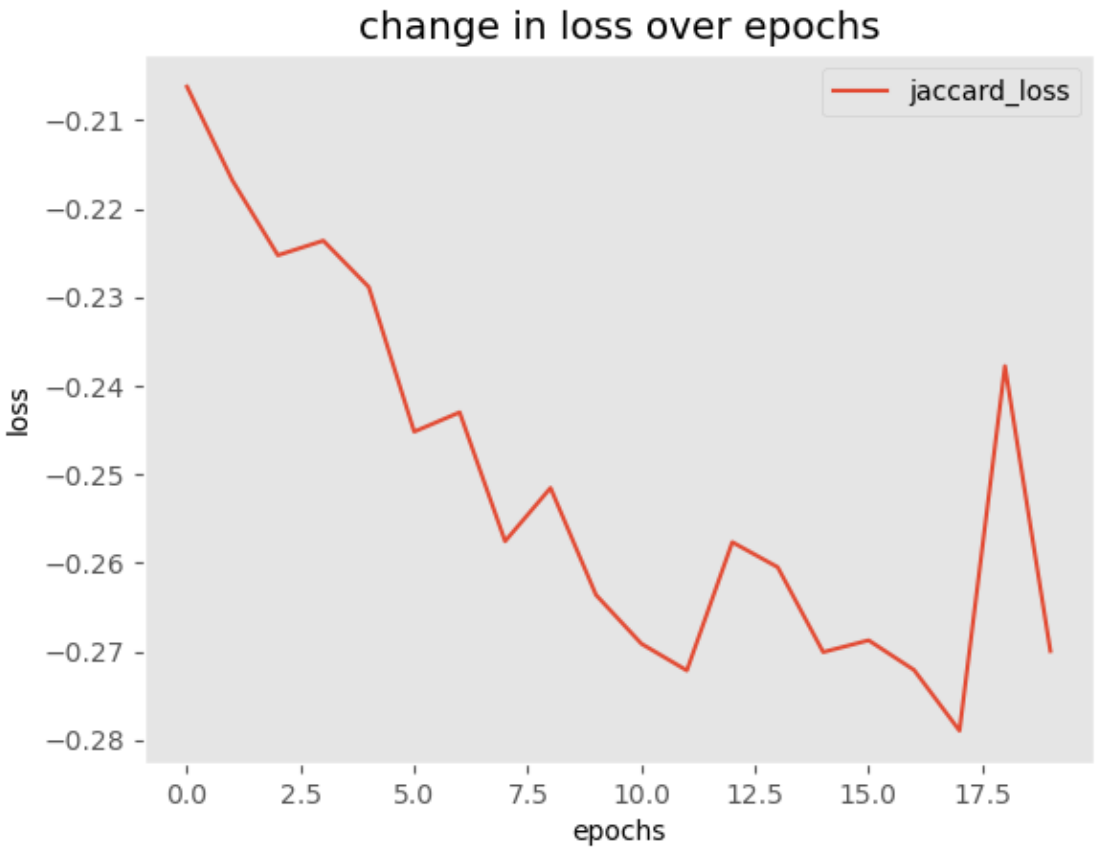
[25]:
```
#Train model---------------------------
nbatch_size=8
```

```
nepochs=20
history = model.
  ↪fit(X_train,Y_train,batch_size=nbatch_size,epochs=nepochs,validation_split=0.
  ↪2)
```

```
Epoch 1/20
2/2 [==============================] - 101s 48s/step - loss: -0.2062 -
jaccard_coef: 0.2081 - val_loss: -0.1875 - val_jaccard_coef: 0.1875
Epoch 2/20
2/2 [==============================] - 110s 85s/step - loss: -0.2168 -
jaccard_coef: 0.2180 - val_loss: -0.1316 - val_jaccard_coef: 0.1316
Epoch 3/20
2/2 [==============================] - 126s 106s/step - loss: -0.2253 -
jaccard_coef: 0.2250 - val_loss: -0.0389 - val_jaccard_coef: 0.0389
Epoch 4/20
2/2 [==============================] - 63s 46s/step - loss: -0.2236 -
jaccard_coef: 0.2247 - val_loss: -0.0033 - val_jaccard_coef: 0.0033
Epoch 5/20
2/2 [==============================] - 69s 46s/step - loss: -0.2288 -
jaccard_coef: 0.2264 - val_loss: -5.7392e-04 - val_jaccard_coef: 5.7392e-04
Epoch 6/20
2/2 [==============================] - 74s 51s/step - loss: -0.2452 -
jaccard_coef: 0.2440 - val_loss: -1.7949e-04 - val_jaccard_coef: 1.7949e-04
Epoch 7/20
2/2 [==============================] - 110s 75s/step - loss: -0.2430 -
jaccard_coef: 0.2415 - val_loss: -4.6241e-05 - val_jaccard_coef: 4.6241e-05
Epoch 8/20
2/2 [==============================] - 62s 35s/step - loss: -0.2576 -
jaccard_coef: 0.2601 - val_loss: -2.4360e-05 - val_jaccard_coef: 2.4360e-05
Epoch 9/20
2/2 [==============================] - 55s 27s/step - loss: -0.2515 -
jaccard_coef: 0.2513 - val_loss: -2.4811e-05 - val_jaccard_coef: 2.4811e-05
Epoch 10/20
2/2 [==============================] - 46s 27s/step - loss: -0.2636 -
jaccard_coef: 0.2637 - val_loss: -2.2980e-05 - val_jaccard_coef: 2.2980e-05
Epoch 11/20
2/2 [==============================] - 45s 25s/step - loss: -0.2691 -
jaccard_coef: 0.2684 - val_loss: -2.2001e-05 - val_jaccard_coef: 2.2001e-05
Epoch 12/20
2/2 [==============================] - 46s 22s/step - loss: -0.2722 -
jaccard_coef: 0.2717 - val_loss: -1.6303e-05 - val_jaccard_coef: 1.6303e-05
Epoch 13/20
2/2 [==============================] - 46s 26s/step - loss: -0.2577 -
jaccard_coef: 0.2542 - val_loss: -1.9636e-05 - val_jaccard_coef: 1.9636e-05
Epoch 14/20
2/2 [==============================] - 47s 25s/step - loss: -0.2605 -
jaccard_coef: 0.2621 - val_loss: -1.6618e-05 - val_jaccard_coef: 1.6618e-05
```

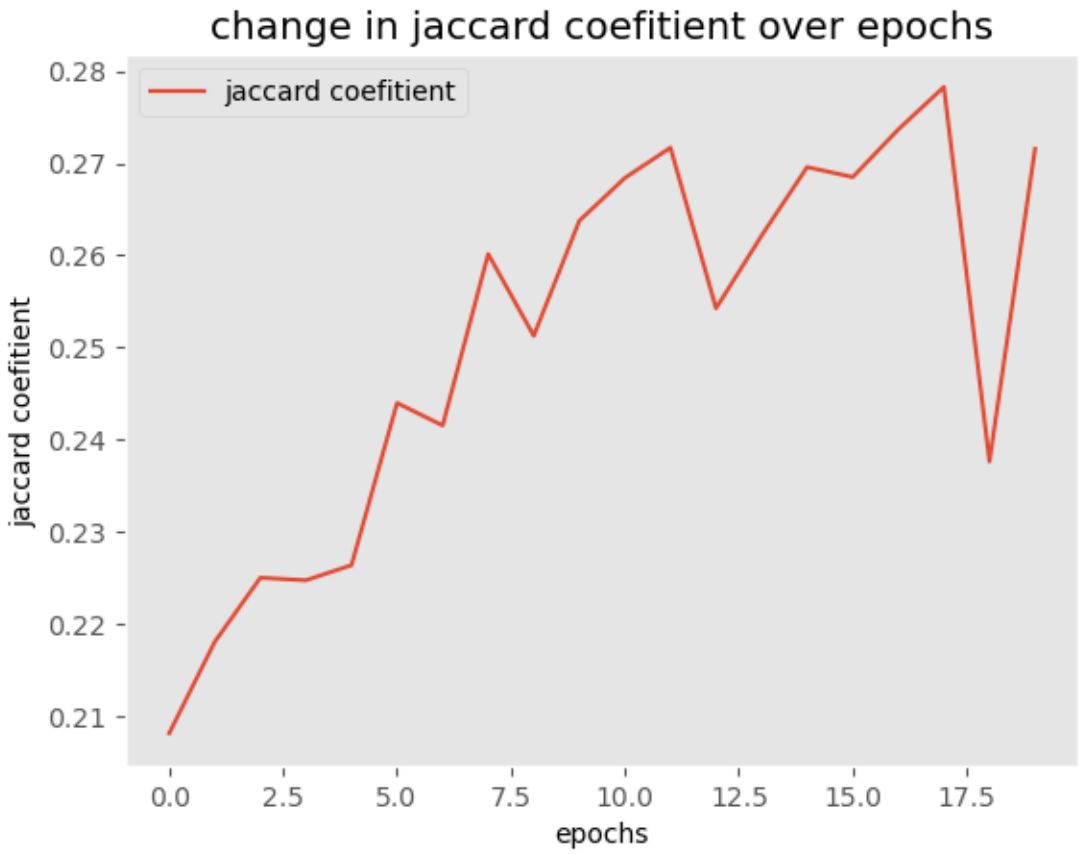
```
Epoch 15/20
2/2 [==============================] - 91s 64s/step - loss: -0.2701 -
jaccard_coef: 0.2696 - val_loss: -1.5582e-05 - val_jaccard_coef: 1.5582e-05
Epoch 16/20
2/2 [==============================] - 36s 13s/step - loss: -0.2688 -
jaccard_coef: 0.2685 - val_loss: -1.5417e-05 - val_jaccard_coef: 1.5417e-05
Epoch 17/20
2/2 [==============================] - 43s 24s/step - loss: -0.2721 -
jaccard_coef: 0.2737 - val_loss: -0.0075 - val_jaccard_coef: 0.0075
Epoch 18/20
2/2 [==============================] - 56s 37s/step - loss: -0.2790 -
jaccard_coef: 0.2783 - val_loss: -0.2489 - val_jaccard_coef: 0.2489
Epoch 19/20
2/2 [==============================] - 76s 37s/step - loss: -0.2378 -
jaccard_coef: 0.2376 - val_loss: -0.2462 - val_jaccard_coef: 0.2462
Epoch 20/20
2/2 [==============================] - 110s 78s/step - loss: -0.2700 -
jaccard_coef: 0.2716 - val_loss: -0.0068 - val_jaccard_coef: 0.0068
```

[26]:
```python
# Plotting loss change over epochs---------------
nrange=nepochs
x = [i for i in range(nrange)]
plt.plot(x,history.history['loss'])
plt.title('change in loss over epochs')
plt.legend(['jaccard_loss'])
plt.xlabel('epochs')
plt.ylabel('loss')
#plt.axis('off')
plt.grid(None)
plt.show()
plt.tight_layout()
```




```
<Figure size 640x480 with 0 Axes>
```

[27]:
```
# Plotting IoU change over epochs----------------------
x = [i for i in range(nrange)]
plt.plot(x,history.history['jaccard_coef'])
plt.title('change in jaccard coefitient over epochs')
plt.legend(['jaccard coefitient'])
plt.xlabel('epochs')
plt.ylabel('jaccard coefitient')
plt.grid(None)
plt.show()
plt.tight_layout()
```




```
<Figure size 640x480 with 0 Axes>
```

[28]:
```
# Creating predictions on our test set-----------------
predictions = model.predict(X_test)
```

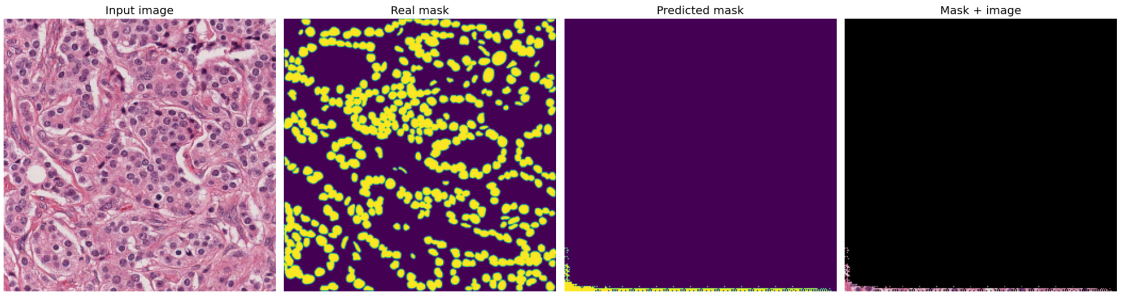
```
1/1 [==============================] - 26s 26s/step
```

[29]:
```
# create predictes mask--------------

def create_mask(predictions,input_shape=(w,h,1)):
    mask = np.zeros(input_shape)
    mask[predictions>0.5] = 1
    return mask
```

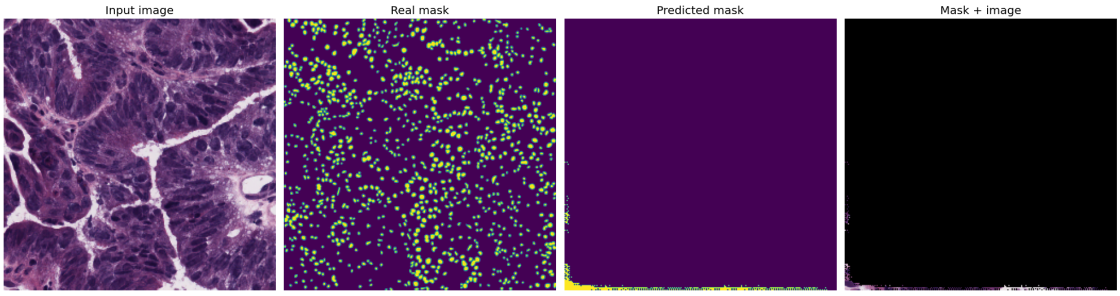
[30]:
```
# Ploting results for one image

def plot_results_for_one_sample(sample_index):

    mask = create_mask(predictions[sample_index])
```

```
    fig = plt.figure(figsize=(20,20))
    fig.add_subplot(1,4,1)
    plt.title('Input image')
    plt.imshow(X_test[sample_index])
    plt.axis('off')
    plt.grid(None)

    fig.add_subplot(1,4,2)
    plt.title('Real mask')
    plt.imshow(Y_test[sample_index])
    plt.axis('off')
    plt.grid(None)

    fig.add_subplot(1,4,3)
    plt.title('Predicted mask')
    plt.imshow(mask)
    plt.axis('off')
    plt.grid(None)

    fig.add_subplot(1,4,4)
    plt.title("Mask + image")
    plt.imshow(X_test[sample_index]*mask)
    plt.grid(None)
    plt.axis('off')
    fig.tight_layout()
```

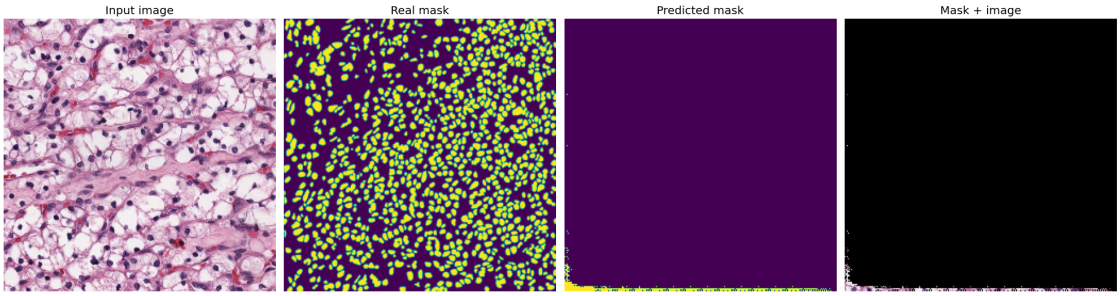
[31]:
```
#Show predicted result---------------
plot_results_for_one_sample(0)
```



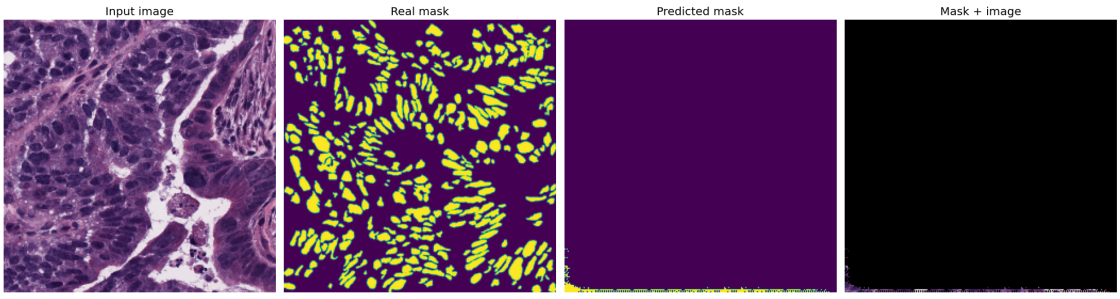
[32]:
```
#Show predicted result---------------
plot_results_for_one_sample(1)
```



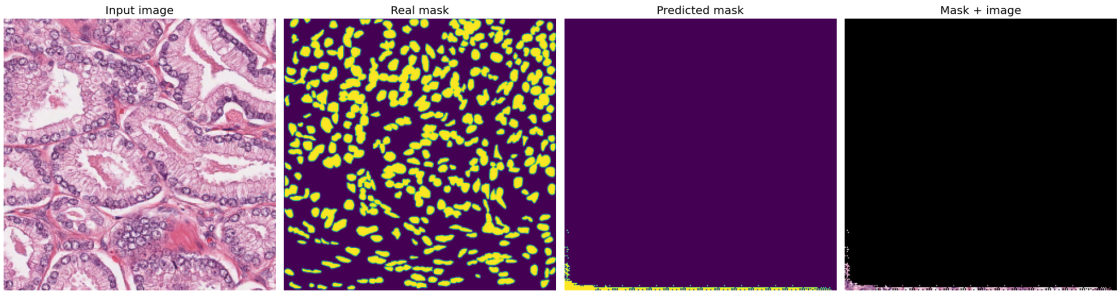
[33]:
```
#Show predicted result---------------
plot_results_for_one_sample(2)
```



[36]:
```
#Show predicted result---------------
plot_results_for_one_sample(3)
```



[34]:
```
#Show predicted result---------------
plot_results_for_one_sample(4)
```


Input image
Real mask
Predicted mask
Mask + image

[ ]: