unet5

April 3, 2023

[26]:
```
from tensorflow.keras.utils import normalize
import tensorflow as tf
import os
import cv2
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.optimizers import Adam
import glob
```

[27]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[28]:
```
image_directory = '/content/drive/MyDrive/Colab Notebooks/lung_colon_image_set/
 ↪lung_image_sets'
mask_directory = '/content/drive/MyDrive/Colab Notebooks/lung_colon_image_set/
 ↪lung_image_sets'
```

[29]:
```
SIZE = 256
num_images = 15000
```

Load images and masks in order so they match

[30]:
```
image_names = glob.glob("/content/drive/MyDrive/Colab Notebooks/
 ↪lung_colon_image_set/lung_image_sets")
print(image_names)
```

```
['/content/drive/MyDrive/Colab Notebooks/lung_colon_image_set/lung_image_sets']
```

[31]:
```
image_names.sort()
print(image_names)
```

```
['/content/drive/MyDrive/Colab Notebooks/lung_colon_image_set/lung_image_sets']
```

[32]:
```
image_names_subset = image_names[0:num_images]
```

[33]:
```
images = [cv2.imread(img, 0) for img in image_names_subset]
```

[34]:
```
image_dataset = np.array(images)
image_dataset = np.expand_dims(image_dataset, axis = 1)
```

Read masks the same way.

[35]:
```
mask_names = glob.glob("/content/drive/MyDrive/Colab Notebooks/
 ↪lung_colon_image_set/lung_image_sets")
mask_names.sort()
mask_names_subset = mask_names[0:num_images]
masks = [cv2.imread(mask, 0) for mask in mask_names_subset]
mask_dataset = np.array(masks)
mask_dataset = np.expand_dims(mask_dataset, axis = 1)
```

[36]:
```
print("Image data shape is: ", image_dataset.shape)
print("Mask data shape is: ", mask_dataset.shape)
print("Max pixel value in image is: ", image_dataset.max())
print("Labels in the mask are : ", np.unique(mask_dataset))
```

```
Image data shape is:  (1, 1)
Mask data shape is:  (1, 1)
Max pixel value in image is:  None
Labels in the mask are :  [None]
```

[37]:
```
#scaler = MinMaxScaler()
```

[38]:
```
#test_image_data=scaler.fit_transform(image_dataset_uint8.reshape(-1,␣
 ↪image_dataset_uint8.shape[-1])).reshape(image_dataset_uint8.shape)
```

[46]:
```
SIZE_X = SIZE_Y = 128
#80% train data 20% test data
datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/
 ↪255,zoom_range=0.2,horizontal_flip=True,validation_split = 0.2)

train_set = datagen.flow_from_directory(image_directory,
                                        class_mode = "categorical",
                                        target_size = (SIZE_X,SIZE_Y),
                                        batch_size = 16,
                                        subset='training',
                                        seed = 42,
                                        )
```

```
Found 12174 images belonging to 3 classes.
```

```
[69]: #Sanity check, view few mages
import random

image_number = random.randint(0,len(SIZE_X)-1)
plt.figure(figsize=(12, 6))
plt.subplot(121)
plt.imshow(SIZE_X[image_number,:,:,0], cmap='gray')
plt.subplot(122)
plt.imshow(SIZE_Y[image_number,:,:,0], cmap='gray')
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-69-1450d1e3b970> in <cell line: 4>()
      2 import random
      3
----> 4 image_number = random.randint(0,len(SIZE_X)-1)
      5 plt.figure(figsize=(12, 6))
      6 plt.subplot(121)

TypeError: object of type 'int' has no len()
```

```
[60]: # Building Unet by dividing encoder and decoder into blocks

from keras.models import Model
from keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D,␣
 ↪concatenate, Conv2DTranspose, BatchNormalization, Dropout, Lambda
from keras.optimizers import Adam
from keras.layers import Activation, MaxPool2D, Concatenate


def conv_block(input, num_filters):
    x = Conv2D(num_filters, 3, padding="same")(input)
    x = BatchNormalization()(x)   #Not in the original network.
    x = Activation("relu")(x)

    x = Conv2D(num_filters, 3, padding="same")(x)
    x = BatchNormalization()(x)  #Not in the original network
    x = Activation("relu")(x)

    return x

#Encoder block: Conv block followed by maxpooling


def encoder_block(input, num_filters):
    x = conv_block(input, num_filters)
    p = MaxPool2D((2, 2))(x)
    return x, p

#Decoder block
#skip features gets input from encoder for concatenation

def decoder_block(input, skip_features, num_filters):
    x = Conv2DTranspose(num_filters, (2, 2), strides=2, padding="same")(input)
    x = Concatenate()([x, skip_features])
    x = conv_block(x, num_filters)
    return x

#Build Unet using the blocks
def build_unet(input_shape, n_classes):
    inputs = Input(input_shape)

    s1, p1 = encoder_block(inputs, 64)
    s2, p2 = encoder_block(p1, 128)
    s3, p3 = encoder_block(p2, 256)
    s4, p4 = encoder_block(p3, 512)

    b1 = conv_block(p4, 1024) #Bridge

    d1 = decoder_block(b1, s4, 512)
    d2 = decoder_block(d1, s3, 256)
    d3 = decoder_block(d2, s2, 128)
    d4 = decoder_block(d3, s1, 64)

    if n_classes == 1:  #Binary
      activation = 'sigmoid'
    else:
      activation = 'softmax'

    outputs = Conv2D(n_classes, 1, padding="same", activation=activation)(d4) 
 ↪#Change the activation based on n_classes
    print(activation)

    model = Model(inputs, outputs, name="U-Net")
    return model
```

[64]:
```
IMG_HEIGHT = image_dataset.shape[1]
IMG_WIDTH  = image_dataset.shape[1]
IMG_CHANNELS = image_dataset.shape[1]

input_shape = (IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS)
```

```
[65]: model = build_unet(input_shape, n_classes=1)
      model.compile(optimizer=Adam(learning_rate = 1e-3), loss='binary_crossentropy',␣
       ↪metrics=['accuracy'])
      model.summary()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-65-f2242abc72ac> in <cell line: 1>()
----> 1 model = build_unet(input_shape, n_classes=1)
      2 model.compile(optimizer=Adam(learning_rate = 1e-3),␣
 ↪loss='binary_crossentropy', metrics=['accuracy'])
      3 model.summary()

<ipython-input-60-9aeba7489b1f> in build_unet(input_shape, n_classes)
     39     inputs = Input(input_shape)
     40
---> 41     s1, p1 = encoder_block(inputs, 64)
     42     s2, p2 = encoder_block(p1, 128)
     43     s3, p3 = encoder_block(p2, 256)

<ipython-input-60-9aeba7489b1f> in encoder_block(input, num_filters)
     23 def encoder_block(input, num_filters):
     24     x = conv_block(input, num_filters)
---> 25     p = MaxPool2D((2, 2))(x)
     26     return x, p
     27

/usr/local/lib/python3.9/dist-packages/keras/utils/traceback_utils.py in␣
 ↪error_handler(*args, **kwargs)
     68             # To get the full stack trace, call:
     69             # `tf.debugging.disable_traceback_filtering()`
---> 70             raise e.with_traceback(filtered_tb) from None
     71         finally:
     72             del filtered_tb

/usr/local/lib/python3.9/dist-packages/tensorflow/python/framework/ops.py in␣
 ↪_create_c_op(graph, node_def, inputs, control_inputs, op_def,␣
 ↪extract_traceback)
   1971   except errors.InvalidArgumentError as e:
   1972     # Convert to ValueError for backwards compatibility.
-> 1973     raise ValueError(e.message)
   1974
   1975   # Record the current Python stack trace as the creating stacktrace of
 ↪this

ValueError: Exception encountered when calling layer "max_pooling2d" (type␣
 ↪MaxPooling2D).
```

```
Negative dimension size caused by subtracting 2 from 1 for '{{node max_pooling2d/
↪MaxPool}} = MaxPool[T=DT_FLOAT, data_format="NHWC", explicit_paddings=[],␣
↪ksize=[1, 2, 2, 1], padding="VALID", strides=[1, 2, 2, 1]](Placeholder)' with␣
↪input shapes: [?,1,1,64].

Call arguments received by layer "max_pooling2d" (type MaxPooling2D):
  • inputs=tf.Tensor(shape=(None, 1, 1, 64), dtype=float32)
```

[ ]:
```
history = model.fit(X_train, y_train,
                    batch_size = 16,
                    verbose=1,
                    epochs=25,
                    validation_data=(X_test, y_test),
                    shuffle=False)
```

[ ]:
```
#Save the model for future use
model.save('/content/drive/MyDrive/Colab Notebooks/saved_models/
 ↪tutorial118_mitochondria_25epochs.hdf5')
```

[ ]:
```
#plot the training and validation accuracy and loss at each epoch
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
plt.plot(epochs, acc, 'y', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

[ ]:
```
#Load previously saved model
from keras.models import load_model
model = load_model("/content/drive/MyDrive/Colab Notebooks/saved_models/
 ↪tutorial118_mitochondria_25epochs.hdf5", compile=False)
```

```
[ ]: #IOU
     y_pred=model.predict(X_test)
     y_pred_thresholded = y_pred > 0.5
```

```
[ ]: from tensorflow.keras.metrics import MeanIoU
```

```
[ ]: n_classes = 2
     IOU_keras = MeanIoU(num_classes=n_classes)
     IOU_keras.update_state(y_pred_thresholded, y_test)
     print("Mean IoU =", IOU_keras.result().numpy())
```

```
[ ]: threshold = 0.5
     test_img_number = random.randint(0, len(X_test)-1)
     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.predict(test_img_input)[0,:,:,0] > 0.5).astype(np.uint8)
     print(prediction.shape)

     plt.figure(figsize=(16, 8))
     plt.subplot(231)
     plt.title('Testing Image')
     plt.imshow(test_img[:,:,0], cmap='gray')
     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, cmap='gray')

     plt.show()
```