Dataset Tertiary Pre-Processing

Author: Alexander Goudemond, Student Number: 219030365

In the previous notebooks, the author investigated several processing techniques to try and prepare the data for Training.

However, After trying to train a Neural Network, the author has realized the following:

- The processed images have a lot of information present in them. This may lead to the network getting confused
- The datasets have varying dimensions, and are not always a regular square in shape. The smallest image is (512 x 512) and the largest is (1010 x 1010). This means that if we attempt to resize our images, we lose a lot of information
- to avoid pixelation and loss of detail as a result of resizing, we can cut the images into small slices and crop them
- If we cut the images into smaller pieces, of some ideal patch size (using Patchify), we can transform a dataset with a small amount of training data (2 images and 2 masks) into a larger collection. However, large collections of data will grow in size as well
- The Unet model I have been using comes from Tensorflow, and is pre-trained. Because it is pre-built, the number of available input sizes is fixed. The user can choose between (96, 128, 160, 192, 224) as possible square input sizes
- The model's performance has been poor. The author will investigate whether they can produce a more desirable processed image (perhaps one that contains less details in the image) and will then attemp to train on that network, if successful. If the author is unable to do this, they will exhaust training with the Morphological Processed images

In this notebook, the author is going to do the following:

- confirm the unique file dimensions across the datasets
- re-examine techniques to try remove unwanted details in the images

Imports

In [1]:
```
from os import remove, mkdir, walk, getcwd
from os.path import exists, join

from shutil import move # moves and replaces files

import cv2
from PIL.Image import fromarray
```

```
import numpy as np
import matplotlib.pyplot as plt
```

Checking File Sizes

In this section of the notebook, we will investigate the size of both the image and the mask from the dataset

In [2]:
```
desired_directory = "Comp700_DataSets"
data_sets = getcwd() + "\\..\\..\\" + desired_directory

sample_image_paths = []

for root, dirs, files in walk(data_sets):
    if (len(files) != 0):
        # only append images
        if (".zip" not in files[0]):
            if (".txt" not in files[0]):
                sample_image_paths.append( root + "\\" + files[0] )
            else:
                sample_image_paths.append( root + "\\" + files[1] )

print(sample_image_paths[0:3])
print("Number of images:", len(sample_image_paths))
```

```
['c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_DataSets\\Extracted\\BF-C2D
L-HSC\\BF-C2DL-HSC\\01\\t0000.tif', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700
\\..\\..\\Comp700_DataSets\\Extracted\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG\\man_seg005
8.tif', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_DataSets\\Extracted
\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\TRA\\man_track0000.tif']
Number of images: 96
```

We know from previous tutorials that the dimensions of the images do not change in the folders, so if we read the dimensions of 1 image, all the other images in that folder will match.

We can use this knowledge to generate a dictionary of unique dimensions:

In [13]:
```
def extractRightSubstring(path, symbol):
    right_most_index = path.rfind(symbol) # right find
    return (path[ right_most_index + len(symbol) : ])
###

def extractLeftSubstring(path, symbol):
    right_most_index = path.find(symbol) # left find
    return (path[ : right_most_index ])
###
```

In [4]:
```
unique_dimensions = []
unique_dimensions_name = []
temp = ""

for image in sample_image_paths:
    img = cv2.imread(image)
    (x, y, z) = img.shape
    temp = "(" + str(x) + "," + str(y) + "," + str(z) + ")"

    if (temp not in unique_dimensions):
        unique_dimensions.append(temp)
        unique_dimensions_name.append(extractRightSubstring(image, "Extracted\\"))

for i in range(len(unique_dimensions)):
    print(unique_dimensions_name[i], "-", unique_dimensions[i])
```

```
BF-C2DL-HSC\BF-C2DL-HSC\01\t0000.tif - (1010,1010,3)
BF-C2DL-MuSC\BF-C2DL-MuSC\01\t0000.tif - (1036,1070,3)
DIC-C2DH-HeLa\DIC-C2DH-HeLa\01\t000.tif - (512,512,3)
Fluo-C2DL-Huh7\Fluo-C2DL-Huh7\01\t000.tif - (1024,1024,3)
Fluo-C2DL-MSC\Fluo-C2DL-MSC\01\t000.tif - (832,992,3)
Fluo-C2DL-MSC\Fluo-C2DL-MSC\02\t000.tif - (782,1200,3)
Fluo-N2DH-SIM+\Fluo-N2DH-SIM+\01\t000.tif - (690,628,3)
Fluo-N2DH-SIM+\Fluo-N2DH-SIM+\02\t000.tif - (773,739,3)
Fluo-N2DH-SIM+ (1)\Fluo-N2DH-SIM+ (1)\01\t000.tif - (718,660,3)
Fluo-N2DH-SIM+ (1)\Fluo-N2DH-SIM+ (1)\02\t000.tif - (790,664,3)
Fluo-N2DL-HeLa\Fluo-N2DL-HeLa\01\t000.tif - (700,1100,3)
PhC-C2DH-U373\PhC-C2DH-U373\01\t000.tif - (520,696,3)
PhC-C2DL-PSC\PhC-C2DL-PSC\01\t000.tif - (576,720,3)
```

We can marry this information with the directories we have as well:

In [25]:
```
# get a collection of directories:

directories = []

for path in sample_image_paths:
    temp = extractRightSubstring(path, "Extracted\\")
    temp = extractLeftSubstring(temp, "\\") # should have a directory name here

    if (temp not in directories):
        directories.append( temp )

# should have 20 directories
len(directories)
```

Out[25]:
```
20
```

In [24]:
```
# get a list of the dimensions of each folder:

picture_dimensions = []
visited_directories = []
directory_index = -1
temp = ""
next_directory = ""

for image in sample_image_paths:
    # initialize
    if (directory_index == -1):
        directory_index += 1
        next_directory = directories[directory_index]

    if (next_directory in image):
        # update
        visited_directories.append( next_directory )
        if (directory_index != len(directories)-1):
            directory_index += 1
            next_directory = directories[directory_index]
        else:
            directory_index = -1

        # read image dimensions
        img = cv2.imread(image)
        (x, y, z) = img.shape
        temp = "(" + str(x) + "," + str(y) + "," + str(z) + ")"

        # save dimensions
        picture_dimensions.append( temp )

len(picture_dimensions)
```

Out[24]: 20

Below is a detailed description of the file sizes:

In [26]:
```
for i in range(len(picture_dimensions)):
    print(directories[i], "-->", picture_dimensions[i])
```

```
BF-C2DL-HSC --> (1010,1010,3)
BF-C2DL-HSC (1) --> (1010,1010,3)
BF-C2DL-MuSC --> (1036,1070,3)
BF-C2DL-MuSC (1) --> (1036,1070,3)
DIC-C2DH-HeLa --> (512,512,3)
DIC-C2DH-HeLa (1) --> (512,512,3)
Fluo-C2DL-Huh7 --> (1024,1024,3)
Fluo-C2DL-Huh7 (1) --> (1024,1024,3)
Fluo-C2DL-MSC --> (832,992,3)
Fluo-C2DL-MSC (1) --> (832,992,3)
Fluo-N2DH-GOWT1 --> (1024,1024,3)
Fluo-N2DH-GOWT1 (1) --> (1024,1024,3)
Fluo-N2DH-SIM+ --> (690,628,3)
Fluo-N2DH-SIM+ (1) --> (718,660,3)
Fluo-N2DL-HeLa --> (700,1100,3)
Fluo-N2DL-HeLa (1) --> (700,1100,3)
PhC-C2DH-U373 --> (520,696,3)
PhC-C2DH-U373 (1) --> (520,696,3)
PhC-C2DL-PSC --> (576,720,3)
PhC-C2DL-PSC (1) --> (576,720,3)
```