In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
# for dirname, _, filenames in os.walk('/kaggle/input'):
#     for filename in filenames:
#         print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

In [2]:
```
from PIL import Image
import tensorflow as tf
import tensorflow.keras.preprocessing as image
import tensorflow_io as tfio
from sklearn.model_selection import train_test_split
import cv2
import tifffile as tifi
import gc
import os
import openslide
from openslide import OpenSlide
import math
from keras.layers.merge import concatenate
from keras.layers import Input
import tensorflow as tf
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPooling2D,GlobalAveragePooling2D, Flatten, Dense, Dropout, BatchNormaliz
from tensorflow.keras import layers
from tensorflow.keras.applications.densenet import DenseNet169
from openslide import open_slide
from matplotlib import pyplot as plt
import pandas as pd
inp_size=512
```

In [3]:
```
def make_train_file(x):
    return "../input/mayo-clinic-strip-ai/train/" + x + ".tif"

def make_test_file(x):
    return x + ".tif"
```

In [4]:
```
train = pd.read_csv('../input/mayo-clinic-strip-ai/train.csv')
train.head()
train_data = pd.DataFrame({'image_id': train.image_id.apply(make_train_file), 'label': train.label})
```

In [5]:
```
train_data.head()
```

Out[5]:

	image_id	label
0	../input/mayo-clinic-strip-ai/train/006388_0.tif	CE
1	../input/mayo-clinic-strip-ai/train/008e5c_0.tif	CE
2	../input/mayo-clinic-strip-ai/train/00c058_0.tif	LAA
3	../input/mayo-clinic-strip-ai/train/01adc5_0.tif	LAA
4	../input/mayo-clinic-strip-ai/train/026c97_0.tif	CE

In [6]:
```
train_data.label.value_counts()
```

Out[6]:
```
CE     547
LAA    207
Name: label, dtype: int64
```

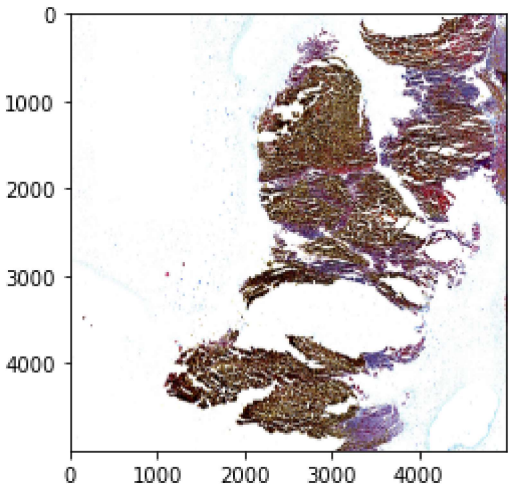
In [7]:
```
# path = '../input/mayo-clinic-strip-ai/train/'
```

In [8]:
```
id=0
sample = train_data.image_id[id]
label = train_data.label[id]
print(label)
slide = open_slide(sample)
print(slide.properties)
print(slide.dimensions)
print(slide.level_dimensions)
#in the slide object, we have only 1 level bcoz it is the way it was stored originally
#so that is why we can only view images of original resolution
slide=slide.read_region((1300,1900),0,(5000,5000))
slide=slide.convert('RGB')
```

```
slide=np.array(slide)
plt.imshow(slide)
```

```
CE
<_PropertyMap {'openslide.level-count': '1', 'openslide.level[0].downsample': '1', 'openslide.level[0].height': '6079
7', 'openslide.level[0].tile-height': '128', 'openslide.level[0].tile-width': '128', 'openslide.level[0].width': '3400
7', 'openslide.vendor': 'generic-tiff', 'tiff.ResolutionUnit': 'centimeter', 'tiff.XResolution': '10', 'tiff.YResolutio
n': '10'}>
(34007, 60797)
((34007, 60797),)
<matplotlib.image.AxesImage at 0x7fb0a33bef10>
```

Out[8]:



In [9]:

```
from openslide.deepzoom import DeepZoomGenerator
slide = open_slide(sample)
tiles=DeepZoomGenerator(slide,tile_size=inp_size,overlap=0,limit_bounds=False)
```

In [10]:

```
print("number of levels : " , tiles.level_count)
print("dimensions of each level : " , tiles.level_dimensions)
print('tot num of tiles: ' , tiles.tile_count)
print("get only last level  :--------- ")
print("grid size : " , tiles.level_tiles[tiles.level_count-1])
print("each tile dim : " , tiles.get_tile_dimensions(tiles.level_count-1,(1,4)))
```

```
number of levels :  17
dimensions of each level :  ((1, 1), (2, 2), (3, 4), (5, 8), (9, 15), (17, 30), (34, 60), (67, 119), (133, 238), (266,
475), (532, 950), (1063, 1900), (2126, 3800), (4251, 7600), (8502, 15200), (17004, 30399), (34007, 60797))
tot num of tiles:  10724
get only last level  :---------
grid size :  (67, 119)
each tile dim :  (512, 512)
```

In [11]:

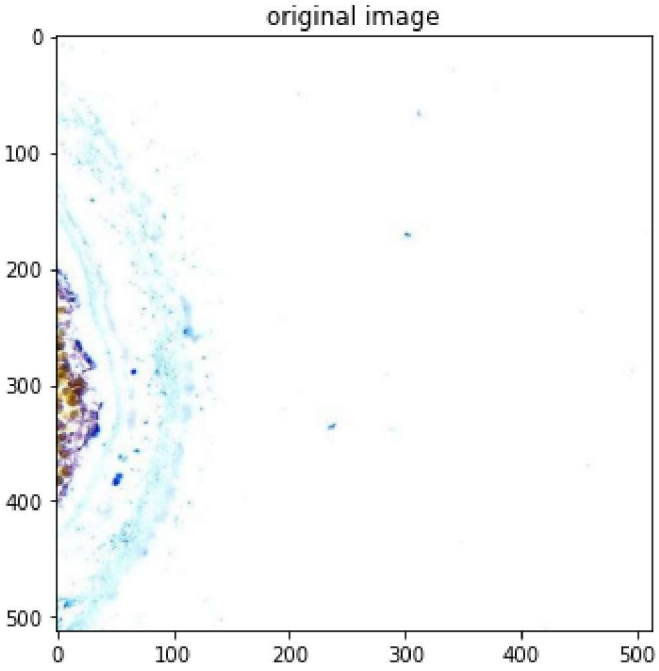
```python
def norm_HnE(img, Io=250, alpha=1, beta=0.15):


    ######## Step 1: Convert RGB to OD ###################
    ## reference H&E OD matrix.
    #Can be updated if you know the best values for your image.
    #Otherwise use the following default values.
    #Read the above referenced papers on this topic.
    HERef = np.array([[0.5626, 0.2159],
                      [0.7201, 0.8012],
                      [0.4062, 0.5581]])
    ### reference maximum stain concentrations for H&E
    maxCRef = np.array([1.9705, 1.0308])


    # extract the height, width and num of channels of image
    h, w, c = img.shape

    # reshape image to multiple rows and 3 columns.
    #Num of rows depends on the image size (wxh)
    img = img.reshape((-1,3))

    # calculate optical density
    # OD = -log10(I)
    #OD = -np.log10(img+0.004)  #Use this when reading images with skimage
    #Adding 0.004 just to avoid log of zero.

    OD = -np.log10((img.astype(np.float)+1)/Io) #Use this for opencv imread
    #Add 1 in case any pixels in the image have a value of 0 (log 0 is indeterminate)


    ############ Step 2: Remove data with OD intensity less than 0 #############
    # remove transparent pixels (clear region with no tissue)
    ODhat = OD[~np.any(OD < beta, axis=1)] #Returns an array where OD values are above beta
    #Check by printing ODhat.min()

    ############# Step 3: Calculate SVD on the OD tuples ######################
```

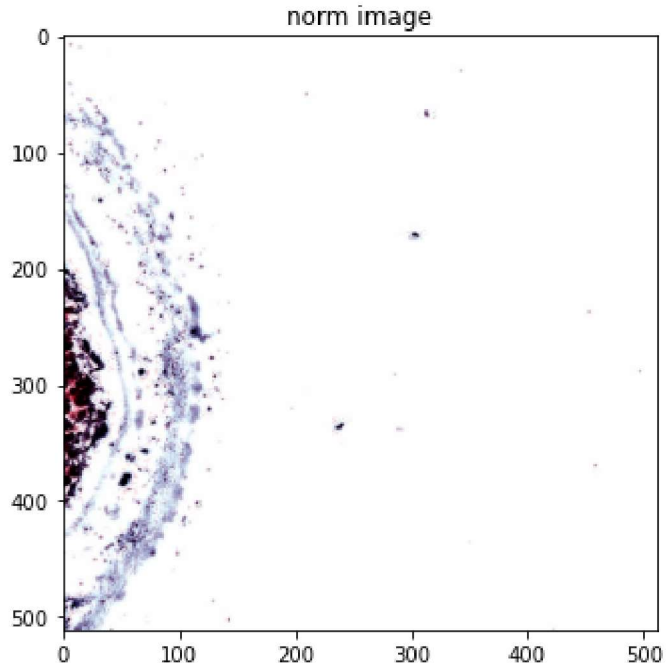
```
    #Estimate covariance matrix of ODhat (transposed)
    # and then compute eigen values & eigenvectors.
    eigvals, eigvecs = np.linalg.eigh(np.cov(ODhat.T))


    ######## Step 4: Create plane from the SVD directions with two largest values ######
    #project on the plane spanned by the eigenvectors corresponding to the two
    # largest eigenvalues
    That = ODhat.dot(eigvecs[:,1:3]) #Dot product

    ############### Step 5: Project data onto the plane, and normalize to unit length ###########
    ############## Step 6: Calculate angle of each point wrt the first SVD direction ########
    #find the min and max vectors and project back to OD space
    phi = np.arctan2(That[:,1],That[:,0])

    minPhi = np.percentile(phi, alpha)
    maxPhi = np.percentile(phi, 100-alpha)

    vMin = eigvecs[:,1:3].dot(np.array([(np.cos(minPhi), np.sin(minPhi))]).T)
    vMax = eigvecs[:,1:3].dot(np.array([(np.cos(maxPhi), np.sin(maxPhi))]).T)


    # a heuristic to make the vector corresponding to hematoxylin first and the
    # one corresponding to eosin second
    if vMin[0] > vMax[0]:
        HE = np.array((vMin[:,0], vMax[:,0])).T

    else:
        HE = np.array((vMax[:,0], vMin[:,0])).T


    # rows correspond to channels (RGB), columns to OD values
    Y = np.reshape(OD, (-1, 3)).T

    # determine concentrations of the individual stains
    C = np.linalg.lstsq(HE,Y, rcond=None)[0]

    # normalize stain concentrations
    maxC = np.array([np.percentile(C[0,:], 99), np.percentile(C[1,:],99)])
    tmp = np.divide(maxC,maxCRef)
    C2 = np.divide(C,tmp[:, np.newaxis])

    ###### Step 8: Convert extreme values back to OD space
    # recreate the normalized image using reference mixing matrix
```

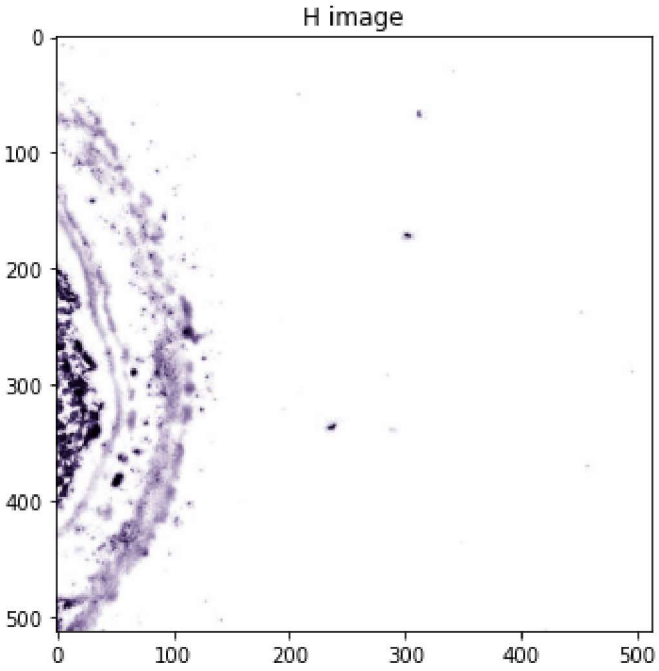
```
    Inorm = np.multiply(Io, np.exp(-HERef.dot(C2)))
    Inorm[Inorm>255] = 254
    Inorm = np.reshape(Inorm.T, (h, w, 3)).astype(np.uint8)

    # Separating H and E components

    H = np.multiply(Io, np.exp(np.expand_dims(-HERef[:,0], axis=1).dot(np.expand_dims(C2[0,:], axis=0))))
    H[H>255] = 254
    H = np.reshape(H.T, (h, w, 3)).astype(np.uint8)

    E = np.multiply(Io, np.exp(np.expand_dims(-HERef[:,1], axis=1).dot(np.expand_dims(C2[1,:], axis=0))))
    E[E>255] = 254
    E = np.reshape(E.T, (h, w, 3)).astype(np.uint8)

    return (Inorm, H, E)
```

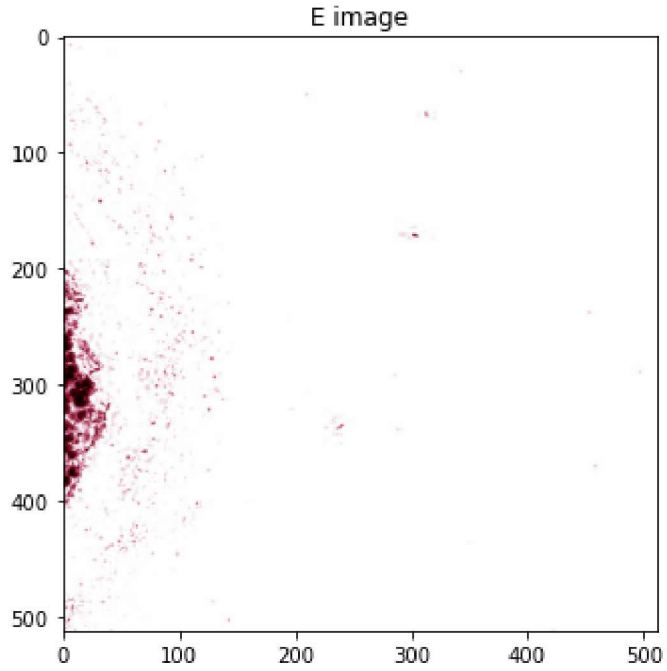
In [12]:
```
#get and print a single tile

tile=tiles.get_tile(tiles.level_count-1,(16,0))
tile=tile.convert("RGB")
tile=np.array(tile)
norm_img,H,E= norm_HnE(tile)
plt.figure(figsize=(12,12))
plt.subplot(221)
plt.title("original image")
plt.imshow(tile)

plt.subplot(222)
plt.title("norm image")
plt.imshow(norm_img)

plt.subplot(223)
plt.title("H image")
plt.imshow(H)

plt.subplot(224)
plt.title("E image")
plt.imshow(E)
print(tile.mean())
print(tile.std())
print(norm_img.mean())
print(norm_img.std())
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:28: DeprecationWarning: `np.float` is a deprecated alias f
or the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is
safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecatio
ns
252.0380541483561
15.59256399547555
247.23180770874023
31.01866130116241
```


original image
0
100
200
300
400
500
0
100
200
300
400
500


norm image
0
100
200
300
400
500
0
100
200
300
400
500


H image
0
100
200
300
400
500
0
100
200
300
400
500


E image
0
100
200
300
400
500
0
100
200
300
400
500

In [ ]:

In [13]:
```
# import shutil
# shutil.rmtree("/kaggle/working/")
# # os.remove("/kaggle/working/download.zip")
```

In [14]:
```
#save all tiles of a image in to directory


base_dir = '/kaggle/working/DATASET/'
CE_dir = 'ce/'
LAA_dir = 'laa/'

original_path = base_dir+'original/'
normalized_path= base_dir+'norm/'



if not os.path.exists(normalized_path+LAA_dir):
    os.makedirs(normalized_path+LAA_dir)

if not os.path.exists(normalized_path+CE_dir):
    os.makedirs(normalized_path+CE_dir)


if not os.path.exists(original_path+LAA_dir):
    os.makedirs(original_path+LAA_dir)

if not os.path.exists(original_path+CE_dir):
    os.makedirs(original_path+CE_dir)
```

In [15]:
```
import traceback
ce_imgs=0
laa_imgs=0

for x in range(int(train_data.shape[0])):
    img_path = train_data.image_id[x]
    label = train_data.label[x]
    print(x,label)
    label=label.lower()
    slide = open_slide(img_path)
    tiles=DeepZoomGenerator(slide,tile_size=inp_size,overlap=0,limit_bounds=False)
    cols,rows = tiles.level_tiles[tiles.level_count-1]
```

```
    count=0
    if label=='ce':
        thresh=6
    else:
        thresh=22

    for row in range(0,rows,5):
        for col in range(0,cols,5):
            file_name = str(x)+'_'+ str(col)+'_'+str(row)+'.png'
            tile=tiles.get_tile(tiles.level_count-1,(col,row))
            tile=tile.convert("RGB")
            tile=np.array(tile)
            try:
                if tile.mean()<180 and tile.std()>50:
                    norm_img,H,E= norm_HnE(tile)
                    plt.imsave(original_path+label+'/'+file_name,tile)
                    plt.imsave(normalized_path+label+'/'+file_name,norm_img)
                    if label=='ce':ce_imgs+=1
                    else:laa_imgs+=1
                    count+=1
                    if count>thresh:break

            except:
                print('exception')
                traceback.print_exc()
                plt.imshow(tile)
                plt.imshow(norm_img)
                pass
        if count>thresh:break
    print(x,ce_imgs,laa_imgs)
```

```
0 CE
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:28: DeprecationWarning: `np.float` is a deprecated alias f
or the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is
safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecatio
ns
```

```
0 7 0
1 CE
1 11 0
2 LAA
2 11 11
3 LAA
3 11 20
4 CE
4 13 20
5 LAA
5 13 43
6 CE
6 20 43
7 CE
7 22 43
8 CE
8 24 43
9 CE
9 31 43
10 LAA
10 31 48
11 CE
11 34 48
12 CE
12 41 48
13 CE
13 48 48
14 CE
14 55 48
15 CE
15 62 48
16 CE
16 69 48
17 CE
17 72 48
18 CE
18 75 48
19 CE
19 82 48
20 LAA
20 82 62
21 CE
21 89 62
22 LAA
22 89 85
```

```
23 CE
23 96 85
24 LAA
24 96 88
25 CE
25 103 88
26 CE
26 110 88
27 CE
27 117 88
28 CE
28 124 88
29 CE
29 127 88
30 LAA
30 127 100
31 LAA
31 127 123
32 LAA
32 127 146
33 CE
33 130 146
34 CE
34 134 146
35 CE
35 141 146
36 CE
36 148 146
37 CE
37 155 146
38 CE
38 157 146
39 CE
39 164 146
40 LAA
40 164 162
41 CE
41 171 162
42 CE
42 176 162
43 CE
43 179 162
44 CE
44 186 162
45 CE
```

```
45 193 162
46 CE
46 196 162
47 CE
47 203 162
48 CE
48 207 162
49 CE
49 214 162
50 CE
50 221 162
51 CE
51 223 162
52 CE
52 224 162
53 CE
53 226 162
54 LAA
54 226 174
55 LAA
55 226 179
56 LAA
56 226 193
57 LAA
57 226 216
58 CE
58 233 216
59 CE
59 233 216
60 CE
60 239 216
61 CE
61 246 216
62 LAA
62 246 217
63 CE
63 253 217
64 CE
64 260 217
65 CE
65 266 217
66 CE
66 273 217
67 CE
67 280 217
```

```
68 CE
68 287 217
69 CE
69 289 217
70 LAA
70 289 226
71 LAA
71 289 233
72 CE
72 294 233
73 CE
73 301 233
74 CE
74 308 233
75 CE
75 313 233
76 CE
76 320 233
77 CE
77 327 233
78 CE
78 334 233
79 CE
79 341 233
80 CE
80 342 233
81 CE
81 349 233
82 CE
82 350 233
83 CE
83 357 233
84 LAA
84 357 233
85 LAA
85 357 239
86 CE
86 364 239
87 CE
87 371 239
88 LAA
88 371 243
89 CE
89 378 243
90 CE
```

```
90 385 243
91 CE
91 392 243
92 LAA
92 392 259
93 CE
93 399 259
94 CE
94 405 259
95 CE
95 405 259
96 CE
96 412 259
97 CE
97 419 259
98 CE
98 422 259
99 CE
99 429 259
100 CE
100 432 259
101 CE
101 439 259
102 CE
102 439 259
103 CE
103 446 259
104 LAA
104 446 259
105 CE
105 453 259
106 CE
106 460 259
107 CE
107 467 259
108 CE
108 474 259
109 LAA
109 474 259
110 LAA
110 474 259
111 LAA
111 474 259
112 LAA
112 474 280
```

```
113 CE
113 481 280
114 CE
114 483 280
115 CE
115 489 280
116 CE
116 489 280
117 CE
117 496 280
118 CE
118 499 280
119 CE
119 499 280
120 CE
120 499 280
121 CE
121 505 280
122 CE
122 508 280
123 CE
123 515 280
124 CE
124 522 280
125 LAA
125 522 282
126 LAA
126 522 291
127 LAA
127 522 292
128 LAA
128 522 315
129 CE
129 522 315
130 LAA
130 522 324
131 LAA
131 522 334
132 LAA
132 522 347
133 LAA
133 522 365
134 LAA
134 522 388
135 LAA
```

```
135 522 403
136 CE
136 525 403
137 CE
137 532 403
138 CE
138 536 403
139 LAA
139 536 404
140 CE
140 540 404
141 LAA
141 540 407
142 CE
142 547 407
143 CE
143 553 407
144 CE
144 555 407
145 CE
145 562 407
146 LAA
146 562 430
147 CE
147 569 430
148 LAA
148 569 441
149 CE
149 569 441
150 CE
150 570 441
151 CE
151 575 441
152 CE
152 582 441
153 LAA
153 582 444
154 CE
154 582 444
155 CE
155 589 444
156 CE
156 596 444
157 CE
157 602 444
```

```
158 LAA
158 602 445
159 CE
159 609 445
160 LAA
160 609 453
161 CE
161 616 453
162 CE
162 617 453
163 LAA
163 617 465
164 CE
164 624 465
165 CE
165 624 465
166 CE
166 630 465
167 CE
167 631 465
168 CE
168 636 465
169 CE
169 643 465
170 CE
170 650 465
171 LAA
171 650 474
172 CE
172 652 474
173 CE
173 656 474
174 LAA
174 656 497
175 CE
175 663 497
176 CE
176 668 497
177 CE
177 668 497
178 CE
178 675 497
179 CE
179 682 497
180 CE
```

```
180 689 497
181 CE
181 692 497
182 CE
182 699 497
183 CE
183 699 497
184 CE
184 699 497
185 CE
185 699 497
186 CE
186 699 497
187 CE
187 701 497
188 CE
188 701 497
189 LAA
189 701 506
190 LAA
190 701 526
191 CE
191 701 526
192 CE
192 708 526
193 CE
193 715 526
194 CE
194 722 526
195 LAA
195 722 542
196 LAA
196 722 546
197 LAA
197 722 548
198 LAA
198 722 563
199 LAA
199 722 586
200 CE
200 729 586
201 CE
201 732 586
202 CE
202 739 586
```

```
203 CE
203 743 586
204 CE
204 743 586
205 CE
205 750 586
206 CE
206 751 586
207 CE
207 751 586
208 CE
208 756 586
209 CE
209 760 586
210 CE
210 762 586
211 CE
211 767 586
212 CE
212 773 586
213 CE
213 777 586
214 CE
214 784 586
215 CE
215 787 586
216 LAA
216 787 591
217 CE
217 794 591
218 CE
218 801 591
219 LAA
219 801 614
220 CE
220 808 614
221 LAA
221 808 619
222 CE
222 815 619
223 CE
223 822 619
224 CE
224 829 619
225 CE
```

```
225 836 619
226 CE
226 843 619
227 CE
227 850 619
228 CE
228 857 619
229 CE
229 857 619
230 LAA
230 857 626
231 CE
231 862 626
232 CE
232 863 626
233 CE
233 870 626
234 CE
234 871 626
235 CE
235 878 626
236 CE
236 885 626
237 CE
237 892 626
238 LAA
238 892 634
239 CE
239 894 634
240 CE
240 901 634
241 LAA
241 901 642
242 CE
242 908 642
243 LAA
243 908 657
244 CE
244 908 657
245 CE
245 915 657
246 CE
246 922 657
247 CE
247 922 657
```

```
248 CE
248 929 657
249 CE
249 936 657
250 CE
250 943 657
251 CE
251 943 657
252 CE
252 950 657
253 CE
253 954 657
254 CE
254 961 657
255 LAA
255 961 680
256 LAA
256 961 703
257 LAA
257 961 709
258 CE
258 963 709
259 CE
259 966 709
260 CE
260 973 709
261 LAA
261 973 718
262 CE
262 980 718
263 CE
263 984 718
264 CE
264 991 718
265 LAA
265 991 727
266 CE
266 991 727
267 CE
267 998 727
268 CE
268 998 727
269 CE
269 1002 727
270 CE
```

```
270 1002 727
271 CE
271 1002 727
272 CE
272 1002 727
273 CE
273 1009 727
274 CE
274 1013 727
275 CE
275 1020 727
276 CE
276 1024 727
277 CE
277 1027 727
278 CE
278 1034 727
279 CE
279 1039 727
280 CE
280 1046 727
281 LAA
281 1046 743
282 LAA
282 1046 755
283 CE
283 1050 755
284 CE
284 1057 755
285 LAA
285 1057 763
286 CE
286 1064 763
287 CE
287 1065 763
288 CE
288 1069 763
289 CE
289 1069 763
290 CE
290 1073 763
291 CE
291 1077 763
292 CE
292 1084 763
```

```
293 CE
293 1086 763
294 CE
294 1088 763
295 CE
295 1095 763
296 CE
296 1102 763
297 CE
297 1104 763
298 CE
298 1104 763
299 CE
299 1105 763
300 LAA
300 1105 766
301 LAA
301 1105 778
302 LAA
302 1105 796
303 CE
303 1112 796
304 LAA
304 1112 808
305 CE
305 1118 808
306 CE
306 1125 808
307 CE
307 1132 808
308 CE
308 1139 808
309 CE
309 1139 808
310 CE
310 1146 808
311 CE
311 1153 808
312 CE
312 1160 808
313 CE
313 1167 808
314 LAA
314 1167 813
315 CE
```

```
315 1170 813
316 CE
316 1174 813
317 CE
317 1181 813
318 CE
318 1184 813
319 LAA
319 1184 813
320 LAA
320 1184 823
321 LAA
321 1184 846
322 CE
322 1187 846
323 CE
323 1188 846
324 CE
324 1191 846
325 CE
325 1191 846
326 CE
326 1198 846
327 CE
327 1205 846
328 LAA
328 1205 858
329 LAA
329 1205 881
330 LAA
330 1205 904
331 LAA
331 1205 916
332 CE
332 1206 916
333 LAA
333 1206 919
334 CE
334 1206 919
335 CE
335 1207 919
336 LAA
336 1207 925
337 CE
337 1210 925
```

```
338 LAA
338 1210 925
339 LAA
339 1210 935
340 CE
340 1217 935
341 CE
341 1220 935
342 LAA
342 1220 939
343 LAA
343 1220 940
344 LAA
344 1220 943
345 CE
345 1227 943
346 CE
346 1234 943
347 CE
347 1241 943
348 CE
348 1248 943
349 LAA
349 1248 962
350 LAA
350 1248 985
351 CE
351 1252 985
352 CE
352 1255 985
353 LAA
353 1255 992
354 CE
354 1255 992
355 CE
355 1261 992
356 CE
356 1267 992
357 CE
357 1274 992
358 CE
358 1278 992
359 CE
359 1278 992
360 CE
```

```
360 1285 992
361 CE
361 1292 992
362 CE
362 1296 992
363 LAA
363 1296 1002
364 LAA
364 1296 1005
365 CE
365 1303 1005
366 LAA
366 1303 1007
367 CE
/opt/conda/lib/python3.7/site-packages/numpy/lib/function_base.py:380: RuntimeWarning: Mean of empty slice.
  avg = a.mean(axis)
/opt/conda/lib/python3.7/site-packages/numpy/core/_methods.py:182: RuntimeWarning: invalid value encountered in true_di
vide
  ret, rcount, out=ret, casting='unsafe', subok=False)
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:40: RuntimeWarning: Degrees of freedom <= 0 for slice
/opt/conda/lib/python3.7/site-packages/numpy/lib/function_base.py:2542: RuntimeWarning: divide by zero encountered in t
rue_divide
  c *= np.true_divide(1, fact)
/opt/conda/lib/python3.7/site-packages/numpy/lib/function_base.py:2542: RuntimeWarning: invalid value encountered in mu
ltiply
  c *= np.true_divide(1, fact)
Traceback (most recent call last):
  File "/tmp/ipykernel_23/1694351545.py", line 28, in <module>
    norm_img,H,E= norm_HnE(tile)
  File "/tmp/ipykernel_23/4084621158.py", line 40, in norm_HnE
    eigvals, eigvecs = np.linalg.eigh(np.cov(ODhat.T))
  File "<__array_function__ internals>", line 6, in eigh
  File "/opt/conda/lib/python3.7/site-packages/numpy/linalg/linalg.py", line 1470, in eigh
    w, vt = gufunc(a, signature=signature, extobj=extobj)
  File "/opt/conda/lib/python3.7/site-packages/numpy/linalg/linalg.py", line 94, in _raise_linalgerror_eigenvalues_nonc
onvergence
    raise LinAlgError("Eigenvalues did not converge")
numpy.linalg.LinAlgError: Eigenvalues did not converge
```

```
exception
367 1310 1007
368 LAA
368 1310 1028
369 CE
369 1317 1028
370 CE
370 1324 1028
371 LAA
371 1324 1031
372 LAA
372 1324 1032
373 CE
373 1324 1032
374 CE
374 1329 1032
375 CE
375 1330 1032
376 CE
376 1337 1032
377 CE
377 1341 1032
378 CE
378 1341 1032
379 CE
379 1348 1032
380 CE
380 1355 1032
381 CE
381 1362 1032
382 CE
382 1362 1032
383 CE
383 1362 1032
384 CE
384 1369 1032
385 CE
385 1376 1032
386 LAA
386 1376 1053
387 CE
387 1376 1053
388 CE
388 1383 1053
389 CE
```

```
389 1383 1053
390 CE
390 1390 1053
391 CE
391 1397 1053
392 CE
392 1404 1053
393 CE
393 1411 1053
394 CE
394 1412 1053
395 CE
395 1416 1053
396 CE
396 1423 1053
397 CE
397 1427 1053
398 LAA
398 1427 1076
399 CE
399 1427 1076
400 CE
400 1427 1076
401 CE
401 1427 1076
402 LAA
402 1427 1088
403 LAA
403 1427 1098
404 CE
404 1434 1098
405 CE
405 1441 1098
406 CE
406 1441 1098
407 CE
407 1448 1098
408 CE
408 1448 1098
409 CE
409 1454 1098
410 CE
410 1457 1098
411 CE
411 1464 1098
```

```
412 CE
412 1471 1098
413 LAA
413 1471 1102
414 LAA
414 1471 1105
415 CE
415 1471 1105
416 CE
416 1478 1105
417 CE
417 1485 1105
418 CE
418 1487 1105
419 LAA
419 1487 1110
420 CE
420 1494 1110
421 LAA
421 1494 1120
422 CE
422 1496 1120
423 CE
423 1503 1120
424 LAA
424 1503 1140
425 CE
425 1510 1140
426 CE
426 1511 1140
427 CE
427 1518 1140
428 LAA
428 1518 1151
429 CE
429 1525 1151
430 CE
430 1530 1151
431 CE
431 1533 1151
432 LAA
432 1533 1172
433 LAA
433 1533 1173
434 LAA
```

```
434 1533 1173
435 LAA
435 1533 1174
436 LAA
436 1533 1197
437 LAA
437 1533 1207
438 CE
438 1533 1207
439 CE
439 1538 1207
440 CE
440 1538 1207
441 LAA
441 1538 1230
442 CE
442 1545 1230
443 CE
443 1549 1230
444 CE
444 1556 1230
445 CE
445 1561 1230
446 CE
446 1568 1230
447 CE
447 1575 1230
448 LAA
448 1575 1238
449 CE
449 1582 1238
450 CE
450 1582 1238
451 CE
451 1589 1238
452 LAA
452 1589 1241
453 CE
453 1596 1241
454 CE
454 1603 1241
455 CE
455 1610 1241
456 CE
456 1617 1241
```

```
457 CE
457 1624 1241
458 CE
458 1626 1241
459 CE
459 1633 1241
460 LAA
460 1633 1241
461 LAA
461 1633 1245
462 LAA
462 1633 1258
463 CE
463 1640 1258
464 CE
464 1647 1258
465 CE
465 1649 1258
466 LAA
466 1649 1268
467 LAA
467 1649 1278
468 CE
468 1652 1278
469 CE
469 1654 1278
470 CE
470 1654 1278
471 CE
471 1661 1278
472 CE
472 1668 1278
473 LAA
473 1668 1285
474 LAA
474 1668 1302
475 CE
475 1669 1302
476 LAA
476 1669 1313
477 LAA
477 1669 1329
478 LAA
478 1669 1340
479 CE
```

```
479 1676 1340
480 LAA
480 1676 1340
481 CE
481 1676 1340
482 CE
482 1681 1340
483 CE
483 1688 1340
484 LAA
484 1688 1357
485 CE
485 1695 1357
486 CE
486 1699 1357
487 CE
487 1706 1357
488 CE
488 1713 1357
489 CE
489 1720 1357
490 CE
490 1727 1357
491 CE
491 1734 1357
492 LAA
492 1734 1358
493 LAA
493 1734 1372
494 CE
494 1738 1372
495 LAA
495 1738 1394
496 CE
496 1738 1394
497 CE
497 1738 1394
498 CE
498 1743 1394
499 CE
499 1750 1394
500 CE
500 1757 1394
501 CE
501 1759 1394
```

```
502 CE
502 1762 1394
503 CE
503 1763 1394
504 LAA
504 1763 1398
505 CE
505 1770 1398
506 LAA
506 1770 1401
507 LAA
507 1770 1410
508 LAA
508 1770 1419
509 CE
509 1777 1419
510 LAA
510 1777 1431
511 CE
511 1780 1431
512 CE
512 1780 1431
513 LAA
513 1780 1438
514 CE
514 1787 1438
515 LAA
515 1787 1461
516 LAA
516 1787 1484
517 CE
517 1791 1484
518 CE
518 1794 1484
519 CE
519 1801 1484
520 CE
520 1808 1484
521 CE
521 1810 1484
522 CE
522 1810 1484
523 CE
523 1817 1484
524 CE
```

```
524 1819 1484
525 CE
525 1826 1484
526 CE
526 1833 1484
527 LAA
527 1833 1500
528 LAA
528 1833 1502
529 LAA
529 1833 1514
530 CE
530 1840 1514
531 CE
531 1847 1514
532 CE
532 1854 1514
533 CE
533 1861 1514
534 CE
534 1864 1514
535 LAA
535 1864 1535
536 CE
536 1867 1535
537 CE
exception
Traceback (most recent call last):
  File "/tmp/ipykernel_23/1694351545.py", line 28, in <module>
    norm_img,H,E= norm_HnE(tile)
  File "/tmp/ipykernel_23/4084621158.py", line 40, in norm_HnE
    eigvals, eigvecs = np.linalg.eigh(np.cov(ODhat.T))
  File "<__array_function__ internals>", line 6, in eigh
  File "/opt/conda/lib/python3.7/site-packages/numpy/linalg/linalg.py", line 1470, in eigh
    w, vt = gufunc(a, signature=signature, extobj=extobj)
  File "/opt/conda/lib/python3.7/site-packages/numpy/linalg/linalg.py", line 94, in _raise_linalgerror_eigenvalues_nonc
onvergence
    raise LinAlgError("Eigenvalues did not converge")
numpy.linalg.LinAlgError: Eigenvalues did not converge
```

```
537 1869 1535
538 CE
538 1876 1535
539 LAA
539 1876 1541
540 CE
540 1881 1541
541 CE
541 1887 1541
542 CE
542 1889 1541
543 LAA
543 1889 1555
544 CE
544 1894 1555
545 LAA
545 1894 1559
546 CE
546 1901 1559
547 CE
547 1908 1559
548 CE
548 1915 1559
549 CE
549 1922 1559
550 CE
550 1922 1559
551 CE
551 1929 1559
552 CE
552 1929 1559
553 CE
553 1936 1559
554 LAA
554 1936 1563
555 CE
555 1938 1563
556 CE
556 1943 1563
557 LAA
557 1943 1563
558 CE
558 1950 1563
559 CE
559 1955 1563
```

```
560 CE
560 1957 1563
561 CE
561 1961 1563
562 CE
562 1968 1563
563 LAA
563 1968 1567
564 CE
564 1975 1567
565 CE
565 1982 1567
566 LAA
566 1982 1590
567 CE
567 1989 1590
568 CE
568 1989 1590
569 CE
569 1992 1590
570 CE
570 1999 1590
571 CE
571 2006 1590
572 CE
572 2012 1590
573 CE
573 2018 1590
574 CE
574 2021 1590
575 CE
575 2028 1590
576 CE
576 2031 1590
577 CE
577 2033 1590
578 LAA
578 2033 1613
579 LAA
579 2033 1628
580 CE
580 2035 1628
581 LAA
581 2035 1628
582 CE
```

```
582 2037 1628
583 LAA
583 2037 1651
584 CE
584 2040 1651
585 CE
585 2047 1651
586 LAA
586 2047 1674
587 LAA
587 2047 1684
588 CE
588 2051 1684
589 CE
589 2054 1684
590 CE
590 2055 1684
591 LAA
591 2055 1699
592 CE
592 2062 1699
593 CE
593 2065 1699
594 CE
594 2072 1699
595 CE
595 2073 1699
596 CE
596 2080 1699
597 CE
597 2081 1699
598 LAA
598 2081 1711
599 CE
599 2084 1711
600 CE
600 2091 1711
601 LAA
601 2091 1734
602 CE
602 2098 1734
603 CE
603 2104 1734
604 CE
604 2111 1734
```

```
605 LAA
605 2111 1742
606 LAA
606 2111 1757
607 CE
607 2114 1757
608 LAA
608 2114 1766
609 CE
609 2114 1766
610 CE
610 2114 1766
611 LAA
611 2114 1784
612 CE
612 2121 1784
613 CE
613 2128 1784
614 CE
614 2135 1784
615 LAA
615 2135 1807
616 LAA
616 2135 1827
617 CE
617 2142 1827
618 CE
618 2149 1827
619 LAA
619 2149 1832
620 CE
620 2151 1832
621 CE
621 2151 1832
622 LAA
622 2151 1832
623 CE
623 2158 1832
624 LAA
624 2158 1842
625 CE
625 2165 1842
626 CE
626 2172 1842
627 CE
```

```
627 2178 1842
628 CE
628 2180 1842
629 LAA
629 2180 1861
630 LAA
630 2180 1862
631 LAA
631 2180 1885
632 LAA
632 2180 1885
633 CE
633 2181 1885
634 LAA
634 2181 1900
635 CE
635 2188 1900
636 CE
636 2195 1900
637 CE
637 2202 1900
638 LAA
638 2202 1923
639 CE
639 2209 1923
640 CE
640 2216 1923
641 LAA
641 2216 1932
642 LAA
642 2216 1940
643 LAA
643 2216 1955
644 CE
644 2219 1955
645 CE
645 2222 1955
646 CE
646 2229 1955
647 CE
647 2231 1955
648 CE
648 2238 1955
649 CE
649 2245 1955
```

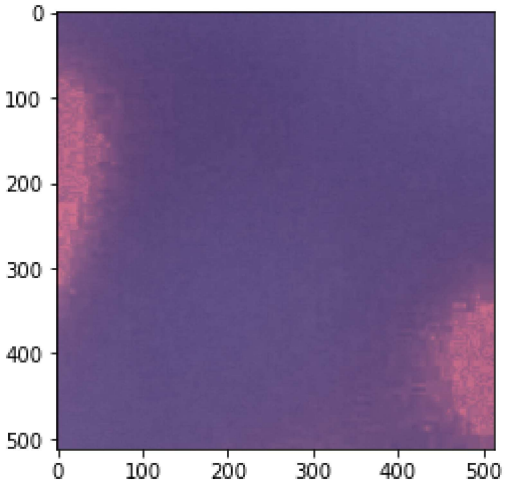
```
650 CE
650 2252 1955
651 CE
651 2255 1955
652 LAA
652 2255 1978
653 CE
653 2256 1978
654 LAA
654 2256 1978
655 LAA
655 2256 1982
656 CE
656 2263 1982
657 CE
657 2263 1982
658 CE
658 2266 1982
659 CE
659 2273 1982
660 LAA
660 2273 2002
661 CE
661 2280 2002
662 CE
662 2285 2002
663 CE
663 2292 2002
664 CE
664 2297 2002
665 CE
665 2303 2002
666 LAA
666 2303 2025
667 CE
667 2305 2025
668 CE
668 2311 2025
669 CE
669 2311 2025
670 CE
670 2318 2025
671 CE
671 2325 2025
672 CE
```

```
672 2327 2025
673 CE
673 2331 2025
674 CE
674 2333 2025
675 CE
675 2340 2025
676 CE
676 2340 2025
677 CE
677 2345 2025
678 CE
678 2352 2025
679 LAA
679 2352 2030
680 CE
680 2357 2030
681 CE
681 2364 2030
682 CE
682 2367 2030
683 CE
683 2374 2030
684 LAA
684 2374 2032
685 LAA
685 2374 2055
686 CE
686 2381 2055
687 CE
687 2388 2055
688 CE
688 2395 2055
689 CE
689 2395 2055
690 CE
690 2402 2055
691 CE
691 2406 2055
692 LAA
692 2406 2078
693 CE
693 2406 2078
694 CE
694 2407 2078
```

```
695 CE
695 2408 2078
696 CE
696 2408 2078
697 CE
697 2415 2078
698 CE
698 2416 2078
699 CE
699 2422 2078
700 CE
700 2429 2078
701 LAA
701 2429 2079
702 LAA
702 2429 2082
703 LAA
703 2429 2082
704 LAA
704 2429 2082
705 LAA
705 2429 2087
706 LAA
706 2429 2087
707 CE
707 2435 2087
708 CE
708 2442 2087
709 CE
709 2444 2087
710 CE
710 2446 2087
711 CE
711 2453 2087
712 CE
712 2460 2087
713 LAA
713 2460 2103
714 CE
714 2464 2103
715 CE
715 2464 2103
716 CE
716 2469 2103
717 LAA
```

```
717 2469 2110
718 LAA
718 2469 2133
719 CE
719 2472 2133
720 LAA
720 2472 2156
721 CE
721 2479 2156
722 CE
722 2486 2156
723 CE
723 2493 2156
724 CE
724 2500 2156
725 LAA
725 2500 2164
726 LAA
726 2500 2169
727 CE
727 2501 2169
728 CE
728 2505 2169
729 LAA
729 2505 2177
730 CE
730 2507 2177
731 CE
731 2507 2177
732 CE
732 2507 2177
733 CE
733 2514 2177
734 LAA
734 2514 2200
735 LAA
735 2514 2223
736 CE
736 2521 2223
737 CE
737 2521 2223
738 LAA
738 2521 2238
739 CE
739 2528 2238
```

```
740 LAA
740 2528 2261
741 CE
741 2533 2261
742 CE
742 2534 2261
743 CE
743 2534 2261
744 CE
744 2536 2261
745 LAA
745 2536 2263
746 CE
746 2543 2263
747 CE
747 2550 2263
748 CE
748 2555 2263
749 CE
749 2562 2263
750 LAA
750 2562 2277
751 CE
751 2569 2277
752 LAA
752 2569 2295
753 LAA
753 2569 2302
```



In [16]:
```
# import shutil
# shutil.make_archive("mean_180-std_50-20_gap_dataset", 'zip', '/kaggle/working/')

# !tar -czf dataset.tar.gz /kaggle/working
```

In [17]:
```
from IPython.display import FileLink
FileLink(r'/kaggle/working/dataset.tar.gz')
```

Out[17]: Path (/kaggle/working/dataset.tar.gz) doesn't exist. It may still be in the process of being generated, or you may have the incorrect path.

In [ ]:

In [ ]:

In [ ]: