```
import numpy as np
import os
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.image as immg
import cv2
from google.colab import drive
from google.colab.patches import cv2_imshow
drive.mount('/content/drive')

import pandas as pd
import tensorflow as tf
import keras as keras
from keras import layers

from skimage.util import random_noise
from skimage.filters import threshold_multiotsu
```

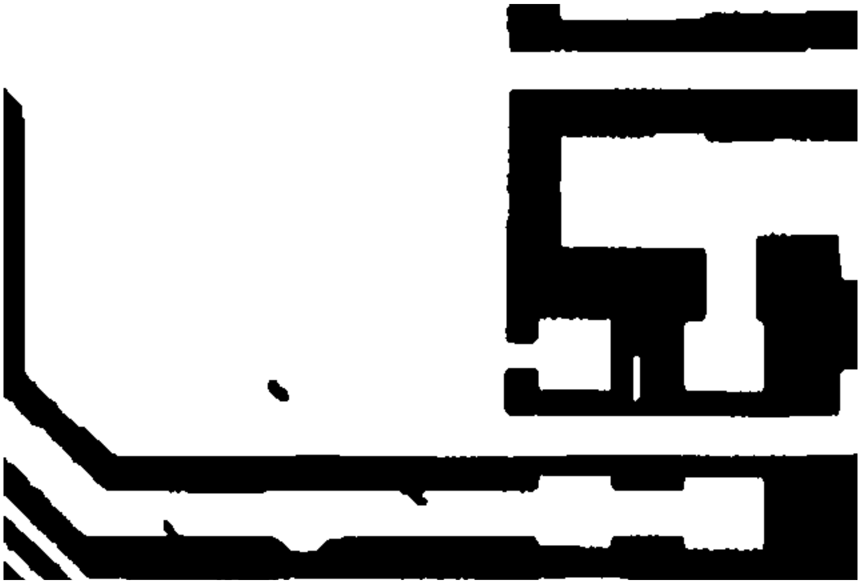
```
    Mounted at /content/drive
```

```
# function to add Gaussian noise to image
def addNoise(img, noiseFactor):
  h = len(img)
  w = len(img[0])
  noise_img = 255*random_noise(img, mode='s&p',amount=noiseFactor)
  return noise_img

def preprocess(img, noise=False, noiseFactor=None):
  rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  r,g,b = cv2.split(rgb_img)
  for i in range(len(g)):
    for j in range(len(g[i])):
      g[i][j] = 0

  rgb = np.dstack((b,g,r))

  hsv_img = cv2.cvtColor(rgb, cv2.COLOR_BGR2HSV)
  h,s,v = cv2.split(hsv_img)
  hsv_split = np.concatenate((h,s,v),axis=1)

  thresholds = threshold_multiotsu(v, classes=3)
  if noise: v = addNoise(v, noiseFactor)

  # Using the threshold values, we generate the three regions.
  regions = np.digitize(v, bins=thresholds)

  return regions

def cutImageUp(img, w, h):
  rowRange = range(0, len(img)//w * w, w)
  colRange = range(0, len(img[0])//h * h, h)
  cutup = np.zeros(((len(rowRange)) * (len(colRange)), w, h, 1))
  index = 0
  for (ri, i) in enumerate(rowRange):
    for (ci, j) in enumerate(colRange):
      cutup[index] = np.reshape(img[i : (i + w), j : (j + h)], (w, h, 1))
      index = index + 1
  return cutup

def stitchTogether(cutImg, w, h):
  dim = cutImg[0].shape
  w_i, h_i = dim[0], dim[1]
  n = len(cutImg)

  rangeW = w // w_i
  rangeH = h // h_i

  img_lst = []

  cnt = 0
  for j in range(0, rangeH):
    lst = []
    for i in range(0, rangeW):
      if cnt >= n:
         return cv2.vconcat(img_lst)
      lst.append(cutImg[cnt])
      lst.append(cutimg[cnt])
      cnt += 1
    img_lst.append(cv2.hconcat(lst))



  return cv2.vconcat(img_lst)
```

```
nodefect_autoencoder = keras.models.load_model('/content/drive/MyDrive/ENEE 439D Final Project/Aerospace PCB Our Pics/nodefectRec
```

```
def classify_regular_pcb(file_path, preprocessBool=True):
  img = cv2.imread(file_path)
  if not preprocessBool:
    img[img > 255/2] = 255.
    img[img < 255/2] = 0.
    img = (255 - img)/255
    r,g,b = cv2.split(img)
    img = r

  if preprocessBool:
    img = preprocess(img)
    img[img > 0.5] = 1.
    img[img < 0.5] = 0.
  cv2_imshow(img*255)

  cutup = cutImageUp(img, 80, 80)
  prediction = nodefect_autoencoder.predict(cutup, verbose=1)
  prediction = np.reshape(prediction, np.shape(prediction)[:-1])
  h, w = img.shape

  # scale up images
  prediction = [img for img in prediction]
  cutup = [img for img in cutup]

  pred_full = stitchTogether(prediction, w, h)
  #pred_full[pred_full > 0.9] = 1.
  #pred_full[pred_full < 0.9] = 0.
  cv2_imshow(pred_full*255)

  original_img = stitchTogether(cutup, w, h)
  difference = np.subtract(original_img, pred_full)
  cv2_imshow(difference*255)
```

```
classify_regular_pcb("/content/drive/MyDrive/ENEE 439D Final Project/Aerospace PCB Our Pics/defectComponent/deepPCB2.jpg", prepro
```



```
# classify_regular_pcb("/content/drive/MyDrive/ENEE 439D Final Project/Aerospace PCB Our Pics/defect_cropped/12.JPG")
classify_regular_pcb("/content/drive/MyDrive/ENEE 439D Final Project/Aerospace PCB Our Pics/defect/1.JPG")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-cf39f4eb859b> in <cell line: 2>()
      1 # classify_regular_pcb("/content/drive/MyDrive/ENEE 439D Final Project/Aerospace PCB Our Pics/defect_cropped/12.JPG"
----> 2 classify_regular_pcb("/content/drive/MyDrive/ENEE 439D Final Project/Aerospace PCB Our Pics/defect/1.JPG")

NameError: name 'classify_regular_pcb' is not defined
```

SEARCH STACK OVERFLOW

```
classify_regular_pcb("/content/drive/MyDrive/ENEE 439D Final Project/Aerospace PCB Our Pics/defect/8.JPG")
```

Model for Checking for False Positives

Loading in Dataset to train

```
folder = "/content/drive/MyDrive/ENEE 439D Final Project/Aerospace PCB Our Pics"
n = 49 * 2
train = np.zeros((n, 80, 80, 1))
test = np.zeros((n, 80, 80, 1))
for i in range(n):

  trainImg = cv2.imread(folder + "/defect_cropped/" + str((i%49)+1) + ".JPG")
  testImg = cv2.imread(folder + "/nodefect_cropped/" + str((i%49)+1) + ".JPG")

  defect = preprocess(trainImg, noise = True, noiseFactor = 0.1)
  #defect = cutImageUp(img, 80, 80)[0]/255.
  #defect = defect.astype('uint8')
  train[i] = np.reshape(defect, (80, 80, 1))

  nodefect = preprocess(testImg, noise = True, noiseFactor = 0.1)
  #nodefect = cutImageUp(img, 80, 80)[0]/255.
  #nodefect = nodefect.astype('uint8')
  test[i] = np.reshape(nodefect, (80, 80, 1))

# cv2_imshow(train[0]*255)

train_labels = np.ones((n,1))
test_labels = np.zeros((n,1))

total_imgs = np.concatenate((train, test))
total_labels = np.concatenate((train_labels, test_labels))

pred_out = nodefect_autoencoder.predict(total_imgs, verbose=1)

difference = np.zeros((2*n, 80, 80, 1))
for i in range(n):
  diff = np.abs(np.subtract(pred_out[i], total_imgs[i])) # want normalized pics
  difference[i] = np.reshape(diff, (80, 80, 1))

cv2_imshow(difference[20] * 255)

# toss out some non defects
difference = difference[:2*n-20]
total_labels = total_labels[:2*n-20]
print(difference.shape, total_labels.shape)


# shuffle
idx = np.random.permutation(2*n-20)
difference, total_labels = difference[idx], total_labels[idx]
```

```
7/7 [==============================] - 1s 117ms/step
```

```
(176, 80, 80, 1) (176, 1)
```

```
from tensorflow.keras import layers, models

# set up model
model = models.Sequential()
model.add(layers.Conv2D(5, (3, 3), activation='relu', input_shape=(80, 80, 1)))
model.add(layers.MaxPooling2D((2, 2)))
# model.add(layers.Conv2D(5, (3, 3), activation='relu'))
# model.add(layers.MaxPooling2D((2, 2)))
# model.add(layers.Conv2D(12, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(5, activation='relu'))
model.add(layers.Dense(1))
model.add(tf.keras.layers.ReLU(max_value=1.0))

model.summary()


model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

history = model.fit(difference, total_labels, epochs=3,
                    validation_data=(difference, total_labels))
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 78, 78, 5)         50        
                                                                 
 max_pooling2d (MaxPooling2D  (None, 39, 39, 5)        0         
 )                                                               
                                                                 
 flatten (Flatten)           (None, 7605)              0         
                                                                 
 dense (Dense)               (None, 5)                 38030     
                                                                 
 dense_1 (Dense)             (None, 1)                 6         
                                                                 
 re_lu (ReLU)                (None, 1)                 0         
                                                                 
=================================================================
Total params: 38,086
Trainable params: 38,086
Non-trainable params: 0
_________________________________________________________________
Epoch 1/3
6/6 [==============================] - 2s 119ms/step - loss: 0.8031 - accuracy: 0.8977 - val_loss: 0.0000e+00 - val_accuracy
Epoch 2/3
6/6 [==============================] - 0s 83ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accur
Epoch 3/3
6/6 [==============================] - 0s 80ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accur
```

```
test_loss, test_acc = model.evaluate(difference,  total_labels, verbose=2)
print(test_acc)

pred = model.predict(difference, verbose = 1)
pred = [round(x[0]) for x in pred]
print(pred)
```

```
6/6 - 0s - loss: 0.0000e+00 - accuracy: 1.0000 - 145ms/epoch - 24ms/step
1.0
6/6 [==============================] - 0s 24ms/step
[0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0,
```

```
# new detector test
def classify_regular_pcb_w_cnn(file_path):
  img = cv2.imread(file_path)
  img = preprocess(img)
  #cv2_imshow(img*255)
  cutup = cutImageUp(img, 80, 80)
  prediction = nodefect_autoencoder.predict(cutup, verbose=1)
  prediction = np.reshape(prediction, np.shape(prediction)[:-1])
  h, w = img.shape

  # scale up images
  prediction = [img * 255 for img in prediction]
  cutup = [img * 255 for img in cutup]

  # create diff array
  n = len(prediction)
  difference = np.zeros((n, 80, 80, 1))

  for i in range(0, n):
    difference[i] = np.reshape((np.abs(np.subtract(cutup[i], prediction[i]))), (80,80,1)) # not normalized
  #cv2_imshow(prediction[0])

  pred_full = stitchTogether(prediction, w, h)
  cv2_imshow(pred_full * 255)

  print(np.max(difference), np.min(difference))

  # cnn output
  cnn_pred = model.predict(difference, verbose = 1)
  cnn_pred = [round(x[0]) for x in cnn_pred]

  print(cnn_pred)

  # modified difference
  diff_mod = []
  for i in range(0, n):
    if cnn_pred[i] == 1:
      diff_mod.append(difference[i])
    else:
      diff_mod.append(np.zeros(80,80,1))


  diff_full = stitchTogether(diff_mod, w, h)
  cv2_imshow(diff_full)

  cv2_imshow(stitchTogether(difference, w, h))

  # cv2_imshow(prediction[0] * 255)
  # cv2_imshow(cutup[0] * 255)
  # difference = np.subtract(cutup[0], prediction[0]) * 255
  # cv2_imshow(difference)

classify_regular_pcb_w_cnn("/content/drive/MyDrive/ENEE 439D Final Project/Aerospace PCB Our Pics/defect/12.JPG")
```

```
30/30 [==============================] - 6s 204ms/step
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-9-309f960c2a0a> in <cell line: 53>()
     51   # cv2_imshow(difference)
     52 
---> 53 classify_regular_pcb_w_cnn("/content/drive/MyDrive/ENEE 439D Final Project/Aerospace PCB Our Pics/defect/12.JPG")

                    3 frames

/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py in _wrapfunc(obj, method, *args, **kwds)
     55 
     56     try:
---> 57         return bound(*args, **kwds)
     58     except TypeError:
     59         # A TypeError occurs if the object does have such a method in its

ValueError: cannot reshape array of size 512000 into shape (80,80,1)
```

SEARCH STACK OVERFLOW