CSE 353
HomeWork 1

JeongYoon Lee (114133199)

a) Introduction. Brief summary of what you think the assignment is about

This assignment is about building a Bayesian classifier. We are trying to get an algorithm that find the ground in the picture using some greenhouses pictures.

So first, we will train the picture which pixel value is ground part using labeled picture of greenhouse, and then test with other greenhouse picture and find precision, recall, and f-score and analyze it.

There are some hyper parameters that we can handle in this code like number of dimension or number of pictures when using training and testing, and I'll try to scale these parameters to find proper model.

b) Method. Brief outline of your (algorithmic) approach

Input x : a pixel's RGB values R|G|B = [0,255].

Output y : if this pixel is ground pixel, y =1
otherwise, y = 0

Pr(y) = Pr(y=1) , Pr(y=0)

Pr(y=1) + Pr(y=0) = 1

Pr(y|x) : if Pr(y =1|x) >0.5, x is a ground pixel.

In Bayes' Rule,

Posterior : Pr(y|x)

Likelihood : Pr(x|y)

Prior : Pr(y)

And we are trying to train using this theorem.

- For Training (Prior and likelihood related code)

```
for i in range(nrows):
  for j in range(ncols):
    r = origIm[i][j][0]
    g = origIm[i][j][1]
    b = origIm[i][j][2]

    if(labels[i][j] ==1):
      Pr_x_given_y_equalsTo_1[r][g][b] = Pr_x_given_y_equalsTo_1[r][g][b] +1
      N_GroundPixels = N_GroundPixels +1
    else:
      Pr_x_given_y_equalsTo_0[r][g][b] = Pr_x_given_y_equalsTo_0[r][g][b] +1
```

➔ Put each RGB values into r,g,b variables, and if that pixel is ground, Pr(x|y=1)++. And N_GroundPixels ++.

➔ If that pixel is not ground, Pr(x|y = 0) ++.

- For Testing (Infer posterior)

```
for i in range(nrows):
  for j in range(ncols):
    r = origIm[i][j][0]
    g = origIm[i][j][1]
    b = origIm[i][j][2]
    Pr_y_equalsTo_1_given_x[i][j] = Pr_x_given_y_equalsTo_1[r][g][b] * Pr_y_equalsTo_1
    if Pr_x_given_y_equalsTo_1[r][g][b] * Pr_y_equalsTo_1 > 0.5:
      detectedMask[i][j] = 1
```

➔ Similarly, put each RGB values of test picture into r,g,b variables, and if that pixel is ground, Pr(y=1|x) = Pr(x|y=1)*Pr(y=1), and if Pr(y=1|x) >0.5, put 1 into detectedMask array.

```
for i in range(nrows) :
  for j in range(ncols) :
    if (labels[i, j] == 1) and (detectedMask[i, j] == 1) :
      truePositives += 1;
    elif (labels[i, j] == 1) and (detectedMask[i, j] == 0) :
      falsePositives += 1;
    elif (labels[i, j] == 0) and (detectedMask[i, j] == 1) :
      falseNegatives += 1;
    elif (labels[i, j] == 0) and (detectedMask[i, j] == 0) :
      trueNegatives += 1;
print("TP : ",truePositives)
print("FP : ",falsePositives)
print("FN : ",falseNegatives)
print("TN : ",trueNegatives)
print()


precision = truePositives/(truePositives+falsePositives)
recall = truePositives/(truePositives+falseNegatives)
print("Precision: ", precision)
print("Recall: ", recall)
print("fscore: ", (2 * precision * recall) / (precision + recall))
```

We can also find TruePositives, FalsePositives, FalseNegatives, TrueNegatives through the detectedMask, and we can find precision, recall and f score value.

I also make chart of this confusion matrix to see better.

```
plt.clf()
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative','Positive']
plt.title('Confusion Matrix')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN','FP'], ['FN', 'TP']]
for i in range(2):
    for j in range(2):
        plt.text(j,i, str(s[i][j])+" = "+str(cm[i][j]))
plt.show()
```

c) Experiments. Tables and/or pictures of intermediate and final results that convince us that the program does what you think it does.

	CASE 1 Training : Tunnel-03 Testing Tunnel -01,02		**CASE 2** Training : Tunnel-02 Testing : Tunnel -01,03		**CASE 3** Training : Tunnel-01 Testing : Tunnel -02,03	
Testing	Tunnel-01	Tunnel-02	Tunnel-01	Tunnel-03	Tunnel-02	Tunnel-03
Precision	0.1333	0.23277	0.11415	0.1921	0.55390	0.03115
Recall	0.7716	0.53877	0.68175	0.2881	0.2897	0.14774
F-Score	0.2274	0.32509	0.1955	0.23051	0.09299	0.05145

The result shows that the F-score of 3 cases are quite low. When we see the table, we can find that recall value for case 1 is quite high. Recall value is the proportion that among the values I need to find, the value I actually found, and the precision is the percentage of correct answers among what I expected. So I thought that the reason why precision is quite low and recall value is quite high in Case 1 is that in the Tunnel-03 labeled picture, they have too many noise so they train both ground pixel and green plant pixel as ground label. So that's why the precision is low in case 1 since they predict ground colors and other colors as ground while testing, and the recall value is quite high since they already find all ground and additionally treat other colors as ground as well. So I think they need more sensitivity.

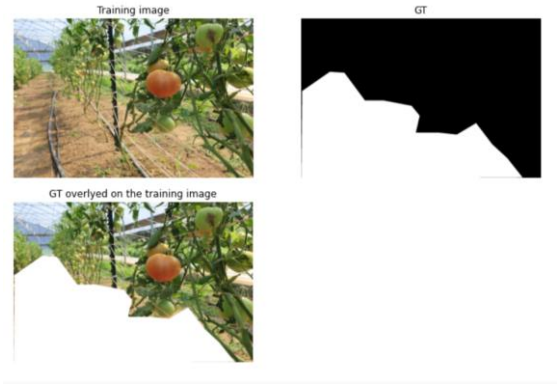
Also I thought Tunnel -03 is not proper for training picture in this model. Plus, I thought the number of Training data should bigger than the number of Testing data.

Here's some of my personal suggestion to improve this model.

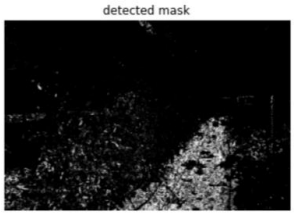
1. Labeled Data should have more sensitivity. They have too many noise, and they have low sensitivity that other colors except ground color are included and treated as ground part.
2. Train with more dataset. We usually train and test model with 7:3. So I think we need more train data. Also, we need more various ground colors for the training data.
3. We can reduce RGB color range when training. So we can try to reduce nDim in this code.

#Case 1 – Training Data : Tunnel-03 , Testing Data : Tunnel-01, Tunnel-02

-Training Data

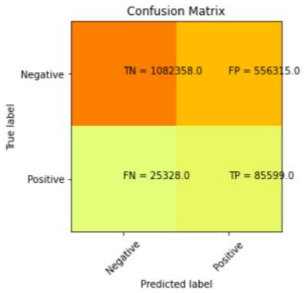


- Testing Result for Tunnel-01

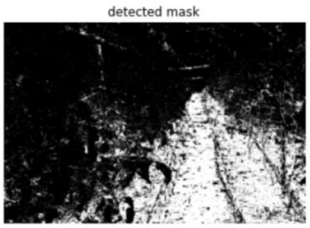



```
TP :  85599
FP :  556315
FN :  25328
TN :  1082358

Precision:  0.1333496387366532
Recall:  0.7716696556621021
fscore:  0.22740259895515788
```

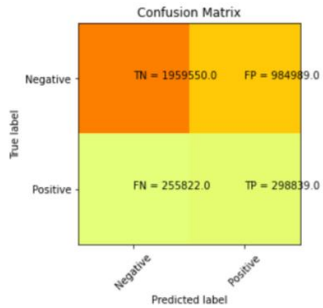


- Testing Result for Tunnel-02



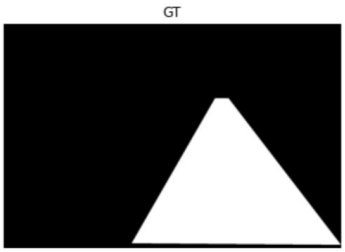

```
TP :  298839
FP :  984989
FN :  255822
TN :  1959550

Precision:  0.23277183547951907
Recall:  0.5387777399168141
fscore:  0.32509196410748176
```

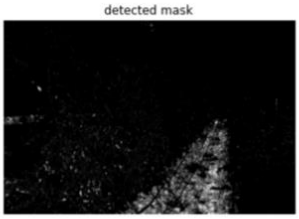


#Case 2 – Training Data : Tunnel-02 , Testing Data : Tunnel-01, Tunnel-03

-Training Data

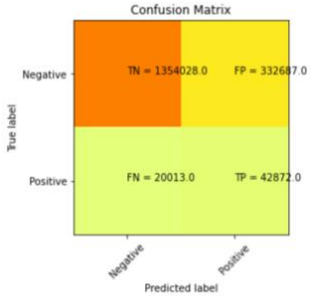


- Testing Result for Tunnel-01



TP : 42872
FP : 332687
FN : 20013
TN : 1354028

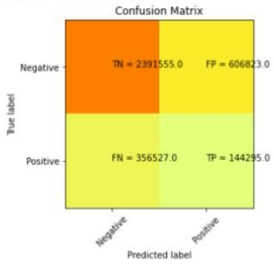
Precision: 0.11415516603250088
Recall: 0.6817524051840661
fscore: 0.19556431380062217



- Testing Result for Tunnel-03

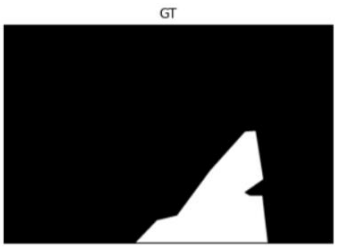

```
TP :  144295
FP :  606823
FN :  356527
TN :  2391555

Precision:  0.19210696588285728
Recall:  0.2881163367423955
fscore:  0.23051424189657665
```

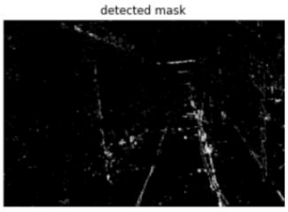


#Case 3 – Training Data : Tunnel-01 , Testing Data : Tunnel-02, Tunnel-03

-Training Data

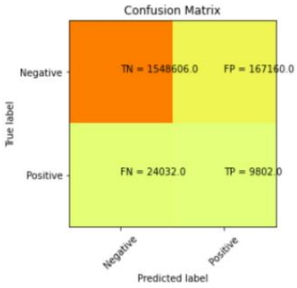


- Testing Result for Tunnel-02

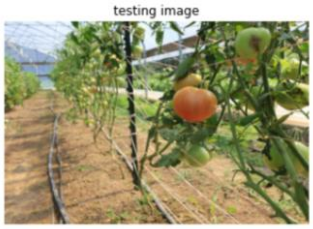



```
TP :  9802
FP :  167160
FN :  24032
TN :  1548606

Precision:  0.05539042280263559
Recall:  0.2897085771708932
fscore:  0.09299986717015504
```

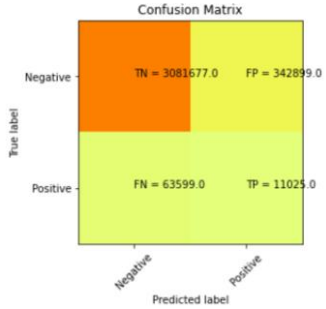


- Testing Result for Tunnel-03




```
TP :  11025
FP :  342899
FN :  63599
TN :  3081677

Precision:  0.031507555294357
Recall:  0.1477406732418525
fscore:  0.05145281275376387
```



d) Discussions. Any design decisions you had to make and your experimental observations. What do you observe about the behavior of your program when you run it? Does it seem to work the way you think it should? Play around a little with different setting to see what happens. Note, your open-ended exploration is highly valued.

As we discussed in question c), I will try to change some hyperparameter to scale this model.

Here are some ways that I used for scaling model.

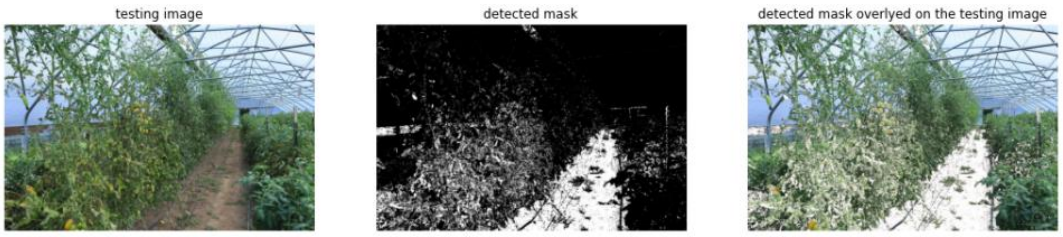
1. Increase the number of training data
2. Change the nDim parameter

1. Increase the number of training data

	CASE 1 Training : Tunnel-02,03 Testing Tunnel -01	**CASE 2** Training : Tunnel-01,03 Testing : Tunnel -02	**CASE 3** Training : Tunnel-01,02 Testing : Tunnel -03
Testing	Tunnel-01	Tunnel-02	Tunnel-03
Precision	0.38221	0.37800	0.36877
Recall	0.7509	0.461299	0.22554
F-Score	0.50658	0.41552	0.279899

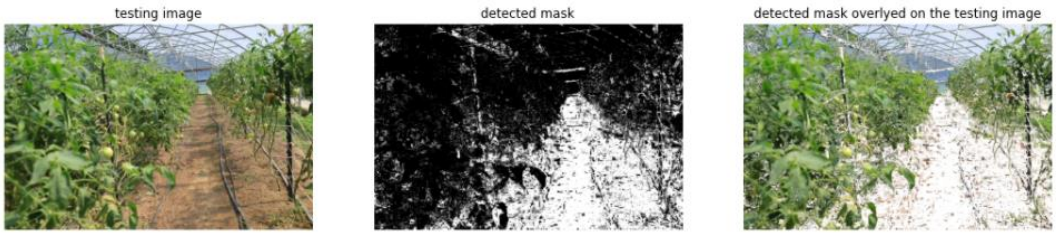
It seems that Case 1 has quite high f-score, but if you look at the testing result picture, the green color are all treated as ground part, so I think it is not worked well since there are some wrong label while training. So I decided the case 2 model is proper for scaling some nDim parameter below.

#Case 1 Testing Result



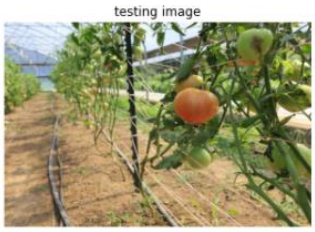
➔ A lot of green pixels are labeled as ground

#Case 2 Testing Result



➔ It seemed tested well since there are not many green part treated as ground, and F-score is quite high.

#Case 3 Testing Result



➔ F-score is too low, but since the labeled picture of Tunnel-03 has many noise, we cannot know this model is good or not.

2. Change the nDim parameter.

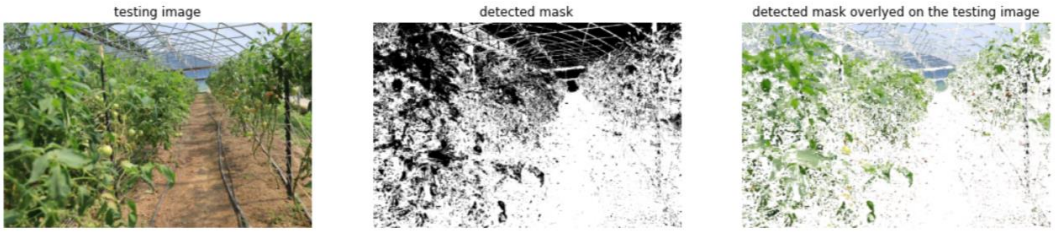
When changing the nDim parameter, we should change nDim and divided into (256/nDim) for each r,g,b variables to make range of color pixel properly. It is explained in the code.

	CASE 1 Training : Tunnel-01,03 Testing Tunnel -02	**CASE 2** Training : Tunnel-01,03 Testing : Tunnel -02
nDim	64	128
Precision	0.7818	0.482221
Recall	0.43485	0.45017
F-Score	0.55886	0.465646

When we changing nDim with smaller value than 256(default value is 256), the Precision, Recall, and F-score are quite high. But If you look at the testing result picture, you can see that the model treated almost all of colors are ground part. So I thought it has low sensitivity, so I tried with more bigger nDim = 128.

And after that, I can find that they find ground well, and the Precision, recall and F-score is okay. So I think this model is the best with these 3 pictures to recognize ground pixel.

#Case 1 Testing result picture (nDim = 64)



➔ Low sensitivity

#Case 2 Testing result picture (nDim = 128)

➔ Looks good! Select as final model.

So I found that the most proper model for this assignment is that when nDim = 128 with training data is Tunnel-01, Tunnel-03 and testing data is Tunnel-02. What I realized through this assignment is that there are no absolute correct index even it is F score or something, and the data is the most important.

In this case, our labeled data is quite vague and insensitive, so it is quite hard to train ground well. Even some model have high value of precision, recall or F-score, we should check the testing result that there's any problem in it. And if the data is bias, or have wrong labeled data, it is quite hard to make good classifier model.