Noisy image classification

Rebeca Tijiboy

Shrivatsa Neerkaje

Table of Contents

[Introduction 2](#_Toc418422660)

[Data Set 3](#_Toc418422661)

[Feature Set 3](#_Toc418422662)

[Bayes Classifier 4](#_Toc418422663)

[Neural Network 6](#_Toc418422664)

[Conclusion 8](#_Toc418422665)

[Appendix I-Sample Images 9](#_Toc418422666)

[Appendix II-Feature Vector Distribution 10](#_Toc418422667)

[Appendix III-Bayes Classifier MATLAB code 11](#_Toc418422668)

[Appendix IV-Neural Network Classifier MATLAB code 13](#_Toc418422669)

### Introduction

The purpose of this project is to develop classifiers to identify images with sparse noise and images with diffused noise (class 1 and class 2). Two classifiers were to be developed to find the most efficient classifier. The selected classifiers were the Bayes classifier and an Artificial Neural network (ANN) classifier. Bayes was attempted due to close Gaussian approximation of feature values. The reason ANN was chosen is because it is well suited for non-linear problems. The algorithm works by applying a two layer feed forward neural network directly to the input images. Each network is trained to output the type of noise present in the image.

### Data Set

The data set is made up of 578 images. The set was created from a set of 289 raw images to which noise was added in two different ways. The images are aerial landscapes. There a total of 289 images with noise added only to certain pixels, sparse noise. The other 289 images, are images with noise across the whole image, diffused noise. See Appendix I for sample images.

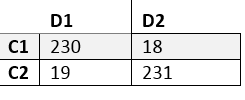
### Feature Set

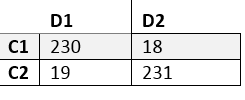
The proposed features included signal to noise ratio, local contrast, histogram standard deviation, number of object outlines and root mean squared standard deviation. The first four features mentioned were used in both classification methods. RMS standard deviation was not used due to correlation with local contrast. Signal to noise ratio was obtained using the raw image and each noisy image to find the average of the ratios between the two. The histogram standard deviation was calculated from the histogram for all of the grey scale pixels. The number of outlines were calculated by converting the images into binary images. This allowed for object detection to count the number of outlines using a built in Matlab function. Local contrast was calculated using the pixel standard deviation in the third order neighborhood. Then the average of these values was obtained since there was a value for each pixel of the image. This feature combination was found to be the most efficient for both classifiers. Each feature was assessed individually. The results show that global features have a better performance than the local feature. Also, among the global features, the number of outlines showed the best performance.

### Bayes Classifier

The feature set had a close Gaussian approximation (see Appendix II) so the Bayes classifier was selected. The Bayes classifier was implemented in two different ways. The first one separating the data into training and testing data. The training set was made up of 498 images, 249 of each class. The classifier was trained using Matlab’s built in function (see Appendix III for source code). The training was done using the previously mentioned feature vectors stored into separate columns of a matrix. The target class was also stored into a one column matrix. The first 249 rows of each matrix contained the data for class 1, sparse noise images; the rest contained class 2, diffused noise images. The classifier was trained and testing followed. Testing was done with the remaining 80 images, 40 of each class. The next method used was an iterative one, where the testing set was chosen to be 10% due to a large amount of data. The classifier was trained and tested ten times to construct a confusion matrix to evaluate the classifier.

Results

After testing, the classifier showed good performance. Class 1 is images with sparse noise, class 2 is images with diffused noise. The percent correct was 92.56%, a sensitivity of 92.7% and a specificity of 92.4%. The following confusion matrix shows the detailed results of the training.

Fig. 1. Bayes Testing Confusion Matrix

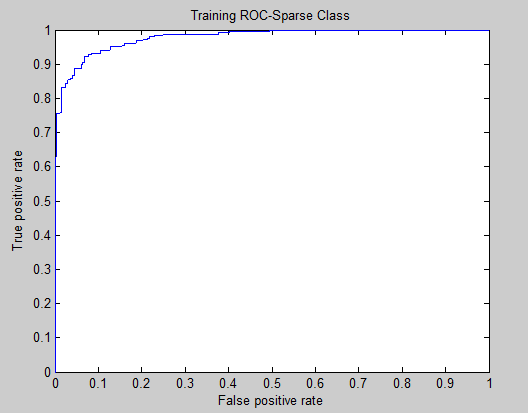
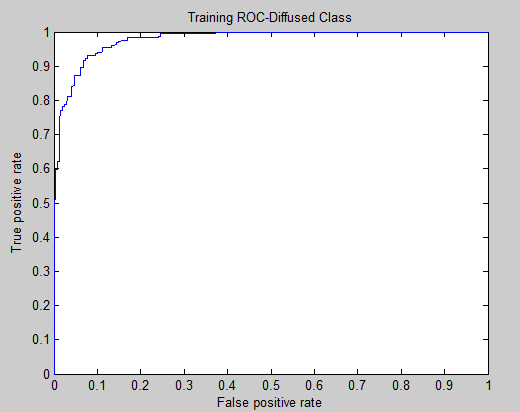
The performance of the classifier was also analyzed by plotting the Receiver-Operating Characteristic Curves for each class. These ROC curves reflect performance based on the selected threshold. Fig.2 represents class 1, sparse noise images; and Fig. 3 represents class 2, diffused noise images.

Fig. 2 Class 1 ROC curve Fig. 3 Class 2 ROC curve

From these performance measures we can see training yielded good results so testing was done with the data which was kept independent. The percent correct from testing was 90%, with a sensitivity of 95% and a sensibility of 85%. Fig. 4 shows more detailed results of the classification.

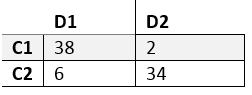


Fig. 4 Bayes Test Confusion Matrix

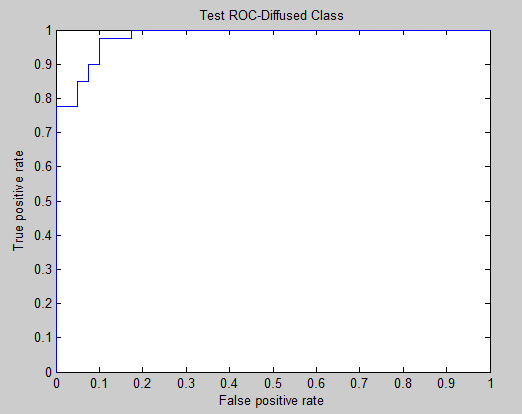
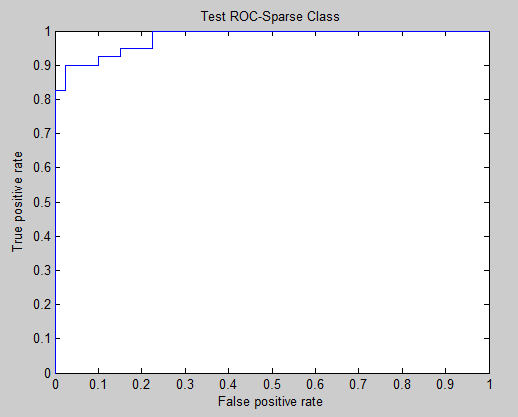
The classifier was also evaluated by plotting the ROC curves in the same way as for training, the results for class 1 and 2 can be seen in figures 5 and 6 respectively.

Fig. 5 ROC curve for Class 1 Fig. 6 ROC curve for Class 2

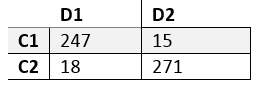
The best results within the Bayes classifier were given by the iterative training and testing method. Fig. 7 shows the confusion matrix class 1 being sparse noise and class 2 being diffused noise. This classifier 90.1 percent correct, with a sensitivity of 94% and a sensibility of 93.8%.

Fig. 7 Bayes Test, k=10% Confusion Matrix

### Neural Network

All four features where again used for this classification. The data used for classification was stored into two vectors, input matrix X and Target matrix T. All features were row ordered and stored in the vector X. The target class was stored into matrix T, where there was one row per class and a column per data point. Each image belonging to the sparse class will have a value ‘1’ in the 1st row and 0 in the 2nd row while the ones belonging to diffused class will have a value ‘1’ in the 2nd row and ‘0’ in the 1st  row of the corresponding columns of the T vector. For testing, the dataset consisted of 396 images. The size of the matrices X and T were 4x396 and 2X392 respectively.

The neural network was initialized with a set weight in order to avoid randomization and different results in each run. The weights were automatically adjusted during classifier training to find the most efficient weight. The ideal amount of neurons was found on a trial and error basis. The number of neurons was raised in increments of five until performance reached a maximum. Thirty was found to be the ideal amount, maximizing performance and minimizing the number of neurons.

The samples are automatically divided into training, validation and test sets. 198 images belonging to sparse class and 194 images belonging to diffused class were used for classifier training (see Appendix IV for source code).

Results

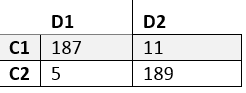
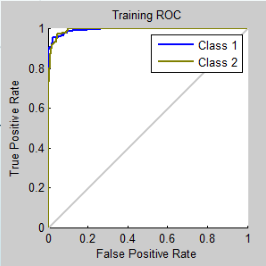
As it is observed in Fig. 8, the trained classifier resulted in 187 images from class sparse and 189 images from class diffused correctly classifier. The percent correct of this classier is 95.9%, the specificity is 97.4% and the sensitivity is 94.4%. The trained classifier showed good performance.

Fig. 8 Neural Net Training Confusion Matrix

An ROC curve was plotted as another measure of performance. The ROC curve for a Neural net is obtained by varying the threshold of the output from 0 to 1, which involves adjustment of the neuron weights. The higher the number of neurons, the more sensitive the classes are. Each class is represented by a different color line in Fig. 9.

Fig. 9 ROC curve for Neural Net Training

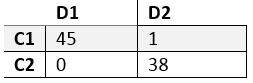
After training showed good results, the classifier was tested using the independent testing set. The results showed high performance with a 98.8 percent correct, a sensitivity of 97.8% and a specificity of 100%. More detailed results are presented on Fig. 10.

Fig. 10 Confusion Matrix for Neural Net Test

An ROC curve was also plotted for the test in the same was as for training. It is observed from Fig. 11 that the classifier model is efficient. The model shows a slightly better performance for classification of diffused noise images than sparse noise images.

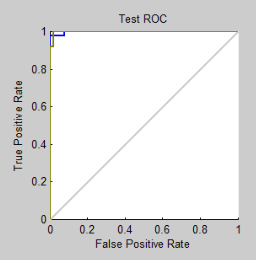


Fig. 11 ROC curve for Neural Net Test

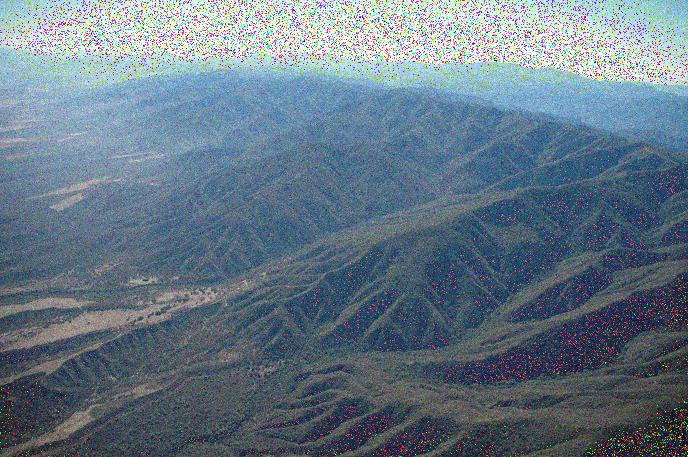
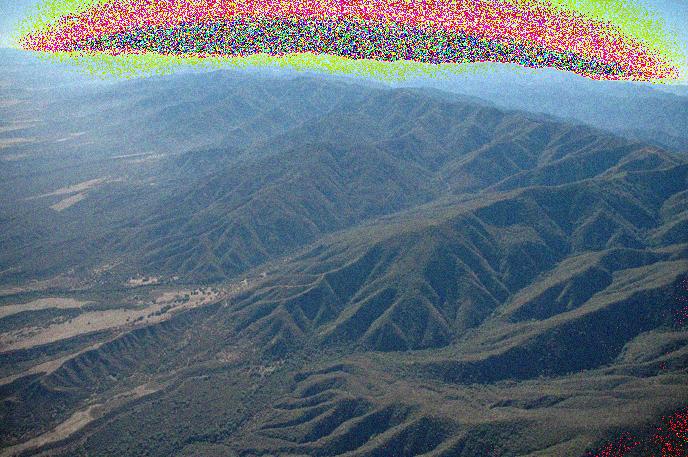
### Conclusion

From the results it can be concluded that for this specific data set, a neural network classifier works best compared to Bayes. The reason for that is the error that Bayes does not give a perfect classification due to the data having some overlap. Table 1 shows a comparison of all resulting performance measures. From this project it may also be concluded that for this set, global features where more efficient in separating the classes, the best feature being the number of outlines.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Bayes Classifier | Bayes Classifier k=10% | Artificial Neural Net |
| Percent Correct | 90 | 90.1 | 98.8 |
| Sensitivity | 95% | 94% | 97.8% |
| Specificity | 85% | 93.8% | 100% |

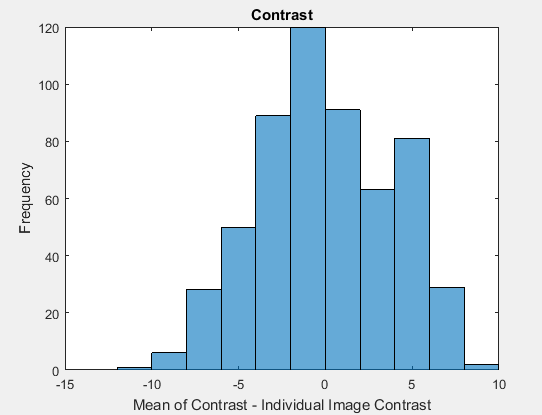
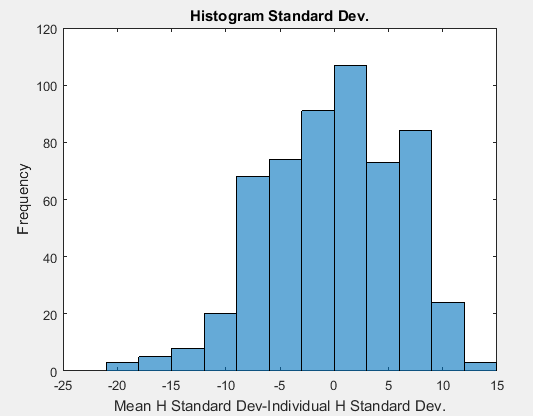
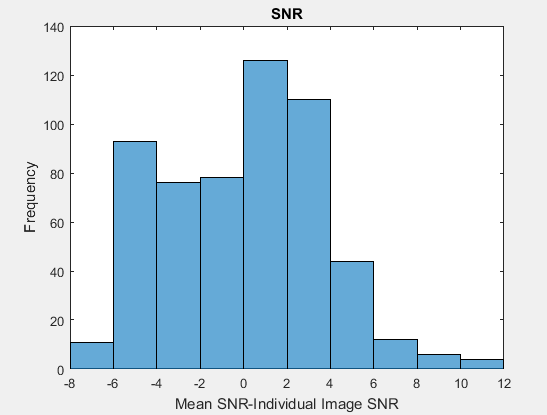
Table 1. Results

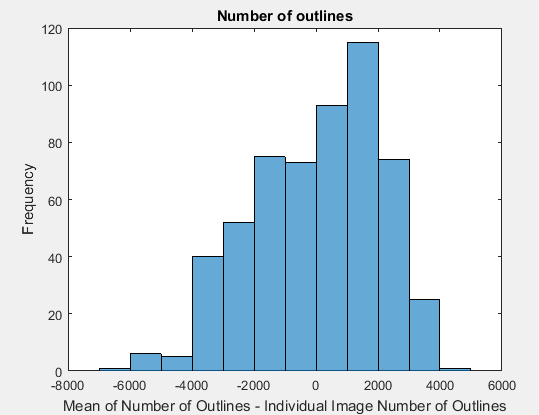
### Appendix I-Sample Images



Diffused Noise Sparse Noise

### Appendix II-Feature Vector Distribution





### Appendix III-Bayes Classifier MATLAB code

%This code trains and tests a Bayes Classifier for classification of noisy images. It extracts the following features: SNR, number of outlines, image contrast, and histogram standard deviation. It trains using the first 249 images of each class and tests on the remaining images. This code also, displays the confusion matrix and ROC curves both for training and testing.

clc

close all;

clear all;

sf=dir('Original\im\*'); %store raw image

for i = 1: length(sf)

fn=strcat(pwd,'\Original\', sf(i).name);

org{i}=imread(fn);

end

sf1 = dir('10NL\\*aA\*'); %store sparse noise images

for i = 1: length(sf)

fn1=strcat(pwd,'\10NL\',sf1(i).name);

loc{i}=imread(fn1);

end

sf2=dir('10NL\\*qA\*');%store diffused noise images

for i = 1 : length(sf2)

fn2=strcat(pwd,'\10NL\',sf2(i).name);

dif{i}=imread(fn2);

end

k1=249;%number of images per class for testing

for i=1:k1

x(i,1)= psnr(loc{i}, org{i});%snr of sparse noise

x(i,2)=std2(loc{i});%histogram standard dev. of sparse noise

I=im2bw(loc{i});%rgb image to binary

A=regionprops(I, 'Area');%counting number of objects

x(i,3)=size(A,1);%outlines feature of sparse noise

x(i,4)=mean(stdfilt(loc{i}(:)));%contrast of sparse noise

x(i+k1,1)= psnr(dif{i}, org{i});%snr feature of diffused noise

x(i+k1,2)=std2(dif{i});%histogram standard dev. of diffused noise

I1=im2bw(dif{i});

A1=regionprops(I1, 'Area');

x(i+k1,2)=size(A1,1);%outlines feature of diffused noise

x(i+k1,4)=mean(stdfilt(dif{i}(:)));%contrast of diffused noise

%filling up target class matix

y(i,1)=cellstr('Sparse');

y(i+k1,1)=cellstr('Diffused');

end

%train Bayes classifier

M1 = fitNaiveBayes(x,y);

predictLabels1 = predict(M1,x);

[ConfusionMat1,labels] = confusionmat(y,predictLabels1)%display Confusion Matrix

%plot ROC Curves

p = posterior(M1,x);

[X,Y] = perfcurve(y,p(:,1),'Sparse');

figure

plot(X,Y)

xlabel('False positive rate'); ylabel('True positive rate')

title('Training ROC-Sparse Class')

[X,Y] = perfcurve(y,p(:,2),'Diffused');

figure

plot(X,Y)

xlabel('False positive rate'); ylabel('True positive rate')

title('Training ROC-Diffused Class')

%Testing the Classifier

i=250;

%feature extraction for sparse noise to fill matrix

for k=1:20

test(k,1)= psnr(loc{i}, org{i});

test(k,2)=std2(loc{i});

I=im2bw(loc{i});

A=regionprops(I, 'Area');

test(k,3)=size(A,1);

test(k,4)=mean(stdfilt(loc{i}(:)));

ty(k,1)=cellstr('Sparse');

i=i+1;

end

i=250;

%feature extraction for diffused noise to fill matrix

for k=21:40

test(k,1)= psnr(dif{i}, org{i});

test(k,2)=std2(dif{i});%standard dev. of diffused noise

I1=im2bw(dif{i});

A1=regionprops(I1, 'Area');

test(k,3)=size(A1,1);

test(k,4)=mean(stdfilt(dif{i}(:)));

ty(k,1)=cellstr('Diffused');

i=i+1;

end

%test classifier

predictLabels2 = predict(M1,test)

[ConfusionMat2,labels] = confusionmat(ty,predictLabels2)%confusion matrix for test

%ROC curve for test

p = posterior(M1,test);

[X,Y] = perfcurve(ty,p(:,1),'Sparse');

figure

plot(X,Y)

xlabel('False positive rate'); ylabel('True positive rate')

title('Test ROC-Sparse Class')

[X,Y] = perfcurve(ty,p(:,2),'Diffused');

figure

plot(X,Y)

xlabel('False positive rate'); ylabel('True positive rate')

title('Test ROC-Diffused Class')

### Appendix IV-Neural Network Classifier MATLAB code

%This code trains and tests a Neural Net nClassifier for classification of noisy images. It extracts the following features: SNR, number of outlines, image contrast, and histogram standard deviation. It trains using a set of images of each class and tests on the remaining images. This code also, displays the confusion matrix and ROC curves both for training and testing.

clc

close all;

clear all;

% Reads in the set of 289 aerial images(without noise)

sf=dir('Original\im\*');

for i = 1 : length(sf)

fn=strcat(pwd,'\Original\',sf(i).name);

org{i}=imread(fn);

end

% Reads in the set of 289 aerial images(with sparse noise)

sf1 = dir('10NL\\*aA\*');

for i = 1 : length(sf)

fn1=strcat(pwd,'\10NL\',sf1(i).name);

loc{i}=imread(fn1);

end

% Reads in the set of 289 aerial images(with diffused noise)

sf2=dir('10NL\\*qA\*');

for i = 1 : length(sf2)

fn2=strcat(pwd,'\10NL\',sf2(i).name);

dif{i}=imread(fn2);

end

nImages=280;

% Loop to store the features into the rows of the Input vector

for i=1:nImages

x(1,i)= psnr(loc{i}, org{i}); %SNR of sparse noise

x(2,i)=std2(loc{i}); %histogram standard dev. of sparse noise

I=im2bw(loc{i});

A=regionprops(I, 'Area');

x(3,i)=size(A,1); %outline count of sparse noise

x(4,i)=mean(stdfilt(loc{i}(:))) ; %intensity of sparse noise

x(1,i+nImages)= psnr(dif{i}, org{i}); %SNR of diffused noise

x(2,i+nImages)=std2(dif{i}); %histogram standard dev. of diffused noise

I1=im2bw(dif{i});

A1=regionprops(I1, 'Area');

x(3,i+nImages)=size(A1,1); %outline count of diffused noise

x(4,i+nImages)=mean(stdfilt(dif{i}(:)));%intensity of diffused noise

y(1,i)=1; % The target matrix which is used as a reference to the to which class the corresponding column belomgs to

y(2,i)=0;

y(1,i+nImages)=0;

y(2,i+nImages,1)=1;

end

setdemorandstream(491218382); %Set a weight to the classifier to avoid randomness

net=patternnet(30); %Set the number of neutrons in the hidden layer to 30

[net tr]=train(net,x,y); % Train the classifier using the input and trget matrix

nntraintool