Abhay Doke

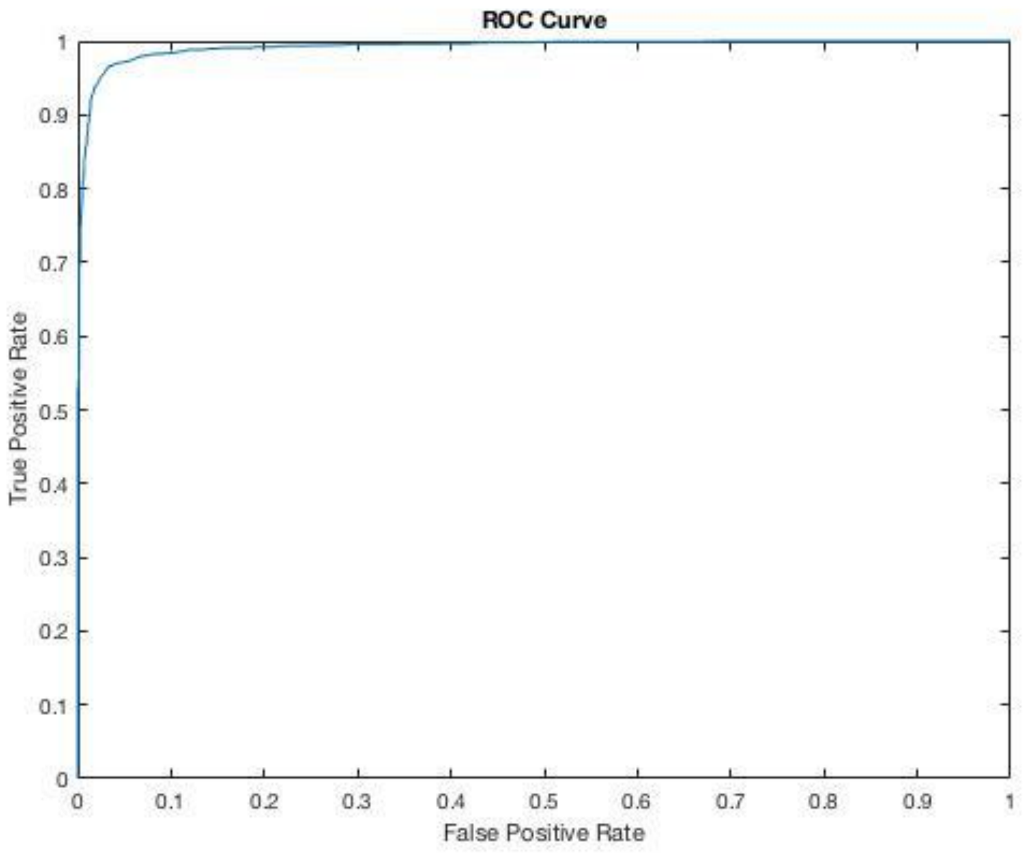
Intelligent Visual Computing
Assignment 2

1 Neural network with default hyper-parameters

a. Learning Rate = 0.5
b. Weight Decay = 0.001
c. Momentum = 0.9
d. Iterations = 1000
e. Number of Hidden Nodes Per Layer = [16 8 4]
f. Threshold = 0.99

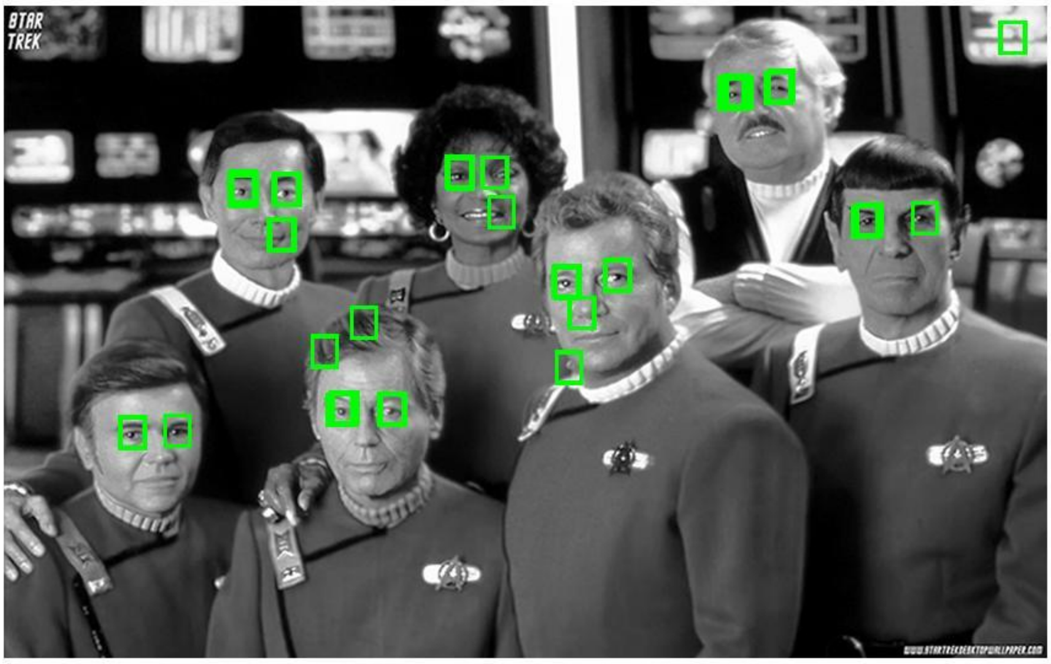
Note: For reporting all classification error, I used a threshold of 0.99. While detecting eyes I chose different threshold to study its effect

Test Error = 5.86%



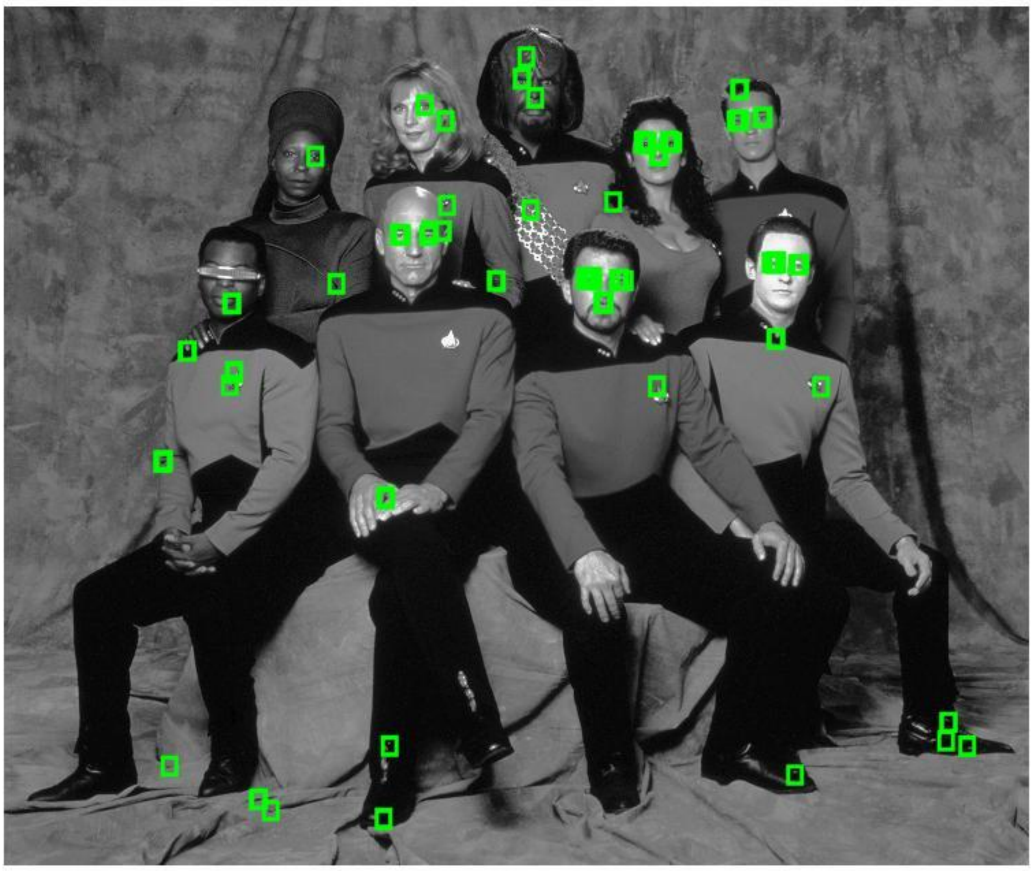
Choosing the Threshold:
Threshold value of 0.5 is not good enough for saying eye or not eye. Our trained model is more inclined toward positive labels. If we choose threshold as 0.5, there are too many false positives in the predictions. Hence we need to choose much higher threshold value to reduce false positive rate.

For star_trek1.pgm I chose the threshold value = 0.9984



Star_trek1 result

For star_trek2.pgm I chose the threshold value = 0.9984

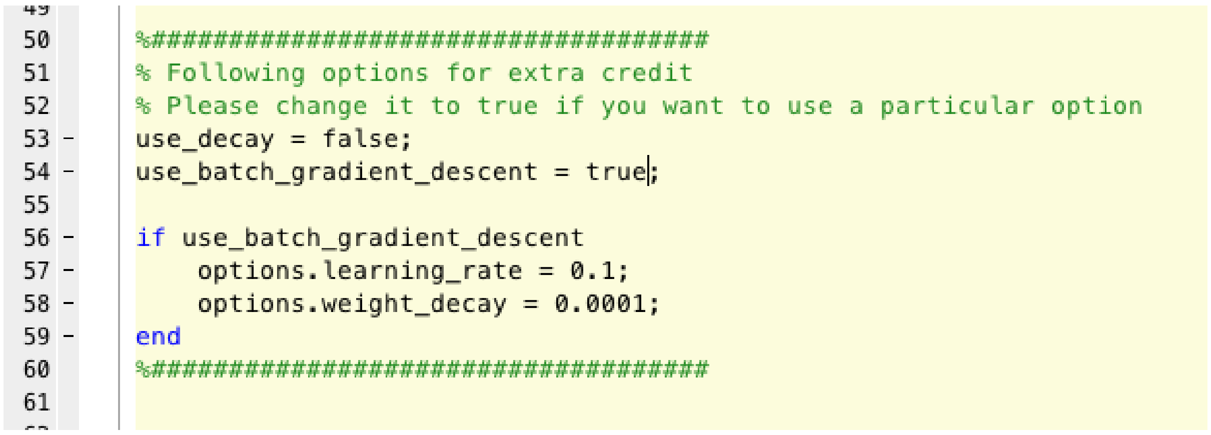


Star_trek2 result

2 Bonus Credits

a. Batch gradient descent

For using batch gradient descent, you need to set use_batch_gradient_descent to true in trainNN.m line 54.

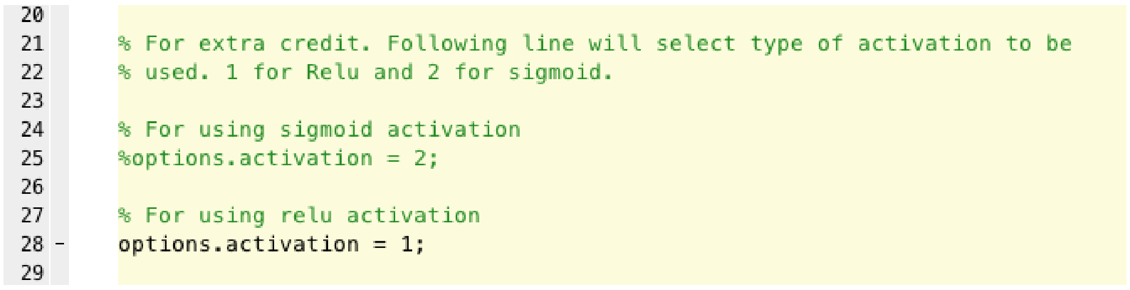


```
50      %########################################
51      % Following options for extra credit
52      % Please change it to true if you want to use a particular option
53  -   use_decay = false;
54  -   use_batch_gradient_descent = true;
55
56  -   if use_batch_gradient_descent
57  -       options.learning_rate = 0.1;
58  -       options.weight_decay = 0.0001;
59  -   end
60      %########################################
61
```

I chose the **batch size = 128**
Test error using batch gradient descent = 4.92%

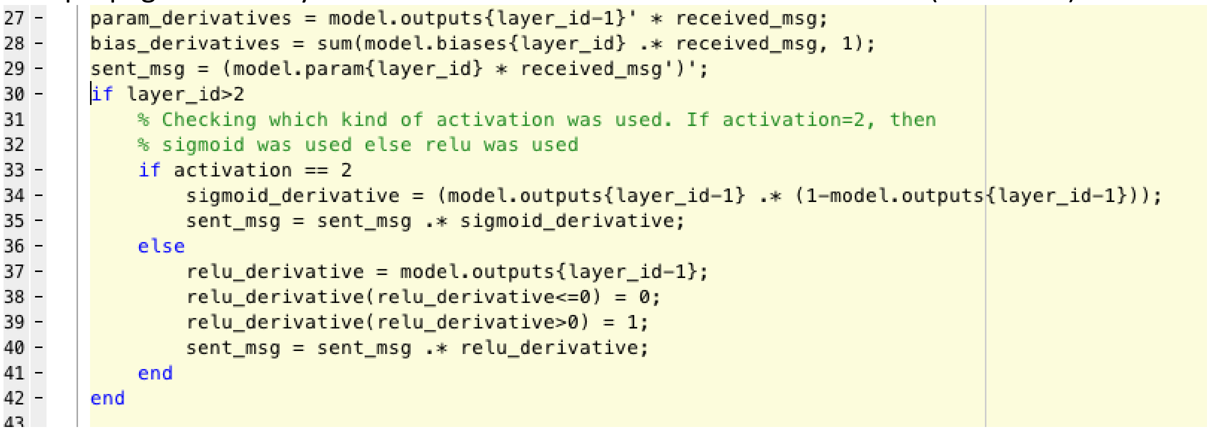
b. Relu activation

For using Relu activation instead of sigmoid activation, set options.activation = 1 in run.m line 28



```
20
21     % For extra credit. Following line will select type of activation to be
22     % used. 1 for Relu and 2 for sigmoid.
23
24     % For using sigmoid activation
25     %options.activation = 2;
26
27     % For using relu activation
28 -   options.activation = 1;
29
```

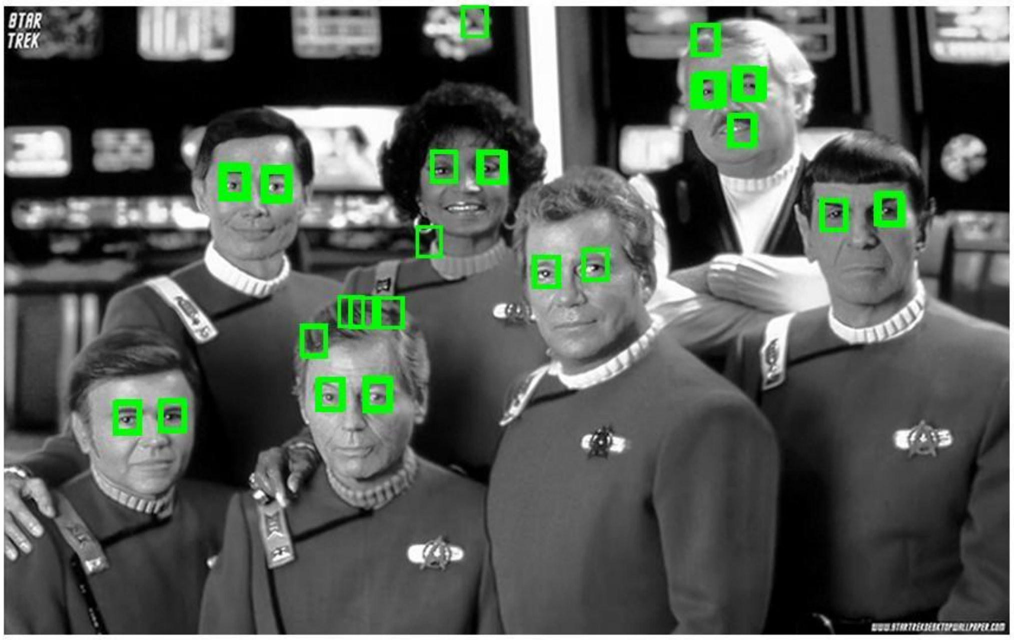
Backpropagation for layers with Relu activation is calculated as follows (line 37-40)



```
27 -  param_derivatives = model.outputs{layer_id-1}' * received_msg;
28 -  bias_derivatives = sum(model.biases{layer_id} .* received_msg, 1);
29 -  sent_msg = (model.param{layer_id} * received_msg')';
30 -  if layer_id>2
31         % Checking which kind of activation was used. If activation=2, then
32         % sigmoid was used else relu was used
33 -      if activation == 2
34 -          sigmoid_derivative = (model.outputs{layer_id-1} .* (1-model.outputs{layer_id-1}));
35 -          sent_msg = sent_msg .* sigmoid_derivative;
36 -      else
37 -          relu_derivative = model.outputs{layer_id-1};
38 -          relu_derivative(relu_derivative<=0) = 0;
39 -          relu_derivative(relu_derivative>0) = 1;
40 -          sent_msg = sent_msg .* relu_derivative;
41 -      end
42 -  end
43
```

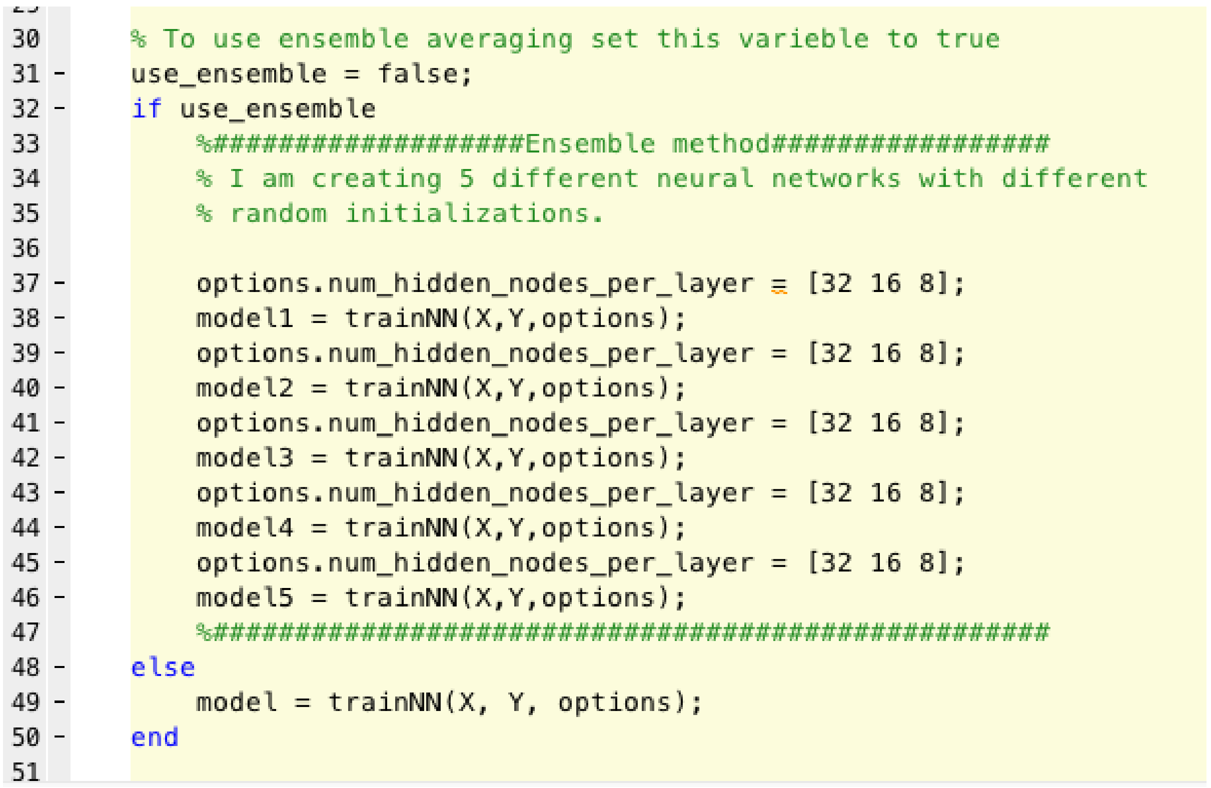
Test error using Relu is 3.69%

For star_trek1.pgm, I used threshold = 0.9999 and got the following result



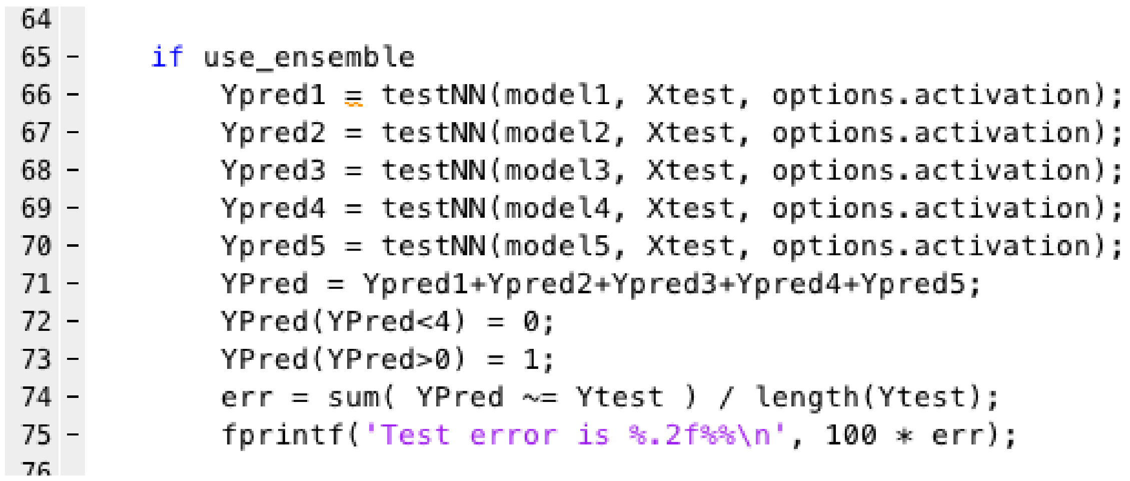
c. Ensemble Averaging

For ensemble averaging, I am training 5 neural networks with different weight initializations.
To use ensemble averaging, you need to set use_ensemble = true in run.m line 31.



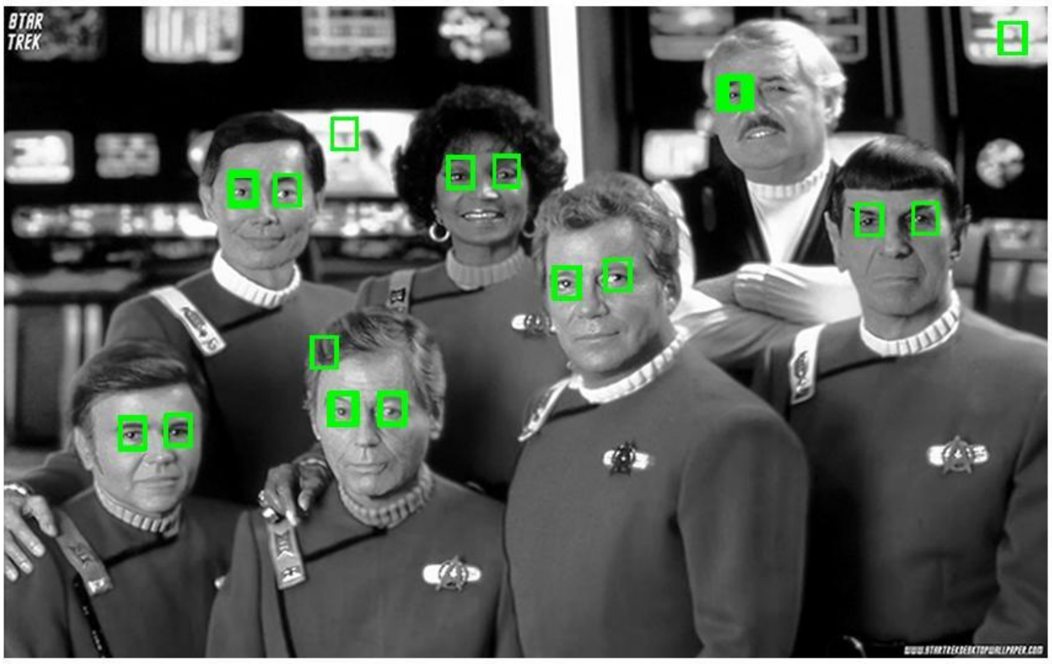
```
30      % To use ensemble averaging set this varieble to true
31 -    use_ensemble = false;
32 -    if use_ensemble
33          %##################Ensemble method#################
34          % I am creating 5 different neural networks with different
35          % random initializations.
36
37 -        options.num_hidden_nodes_per_layer = [32 16 8];
38 -        model1 = trainNN(X,Y,options);
39 -        options.num_hidden_nodes_per_layer = [32 16 8];
40 -        model2 = trainNN(X,Y,options);
41 -        options.num_hidden_nodes_per_layer = [32 16 8];
42 -        model3 = trainNN(X,Y,options);
43 -        options.num_hidden_nodes_per_layer = [32 16 8];
44 -        model4 = trainNN(X,Y,options);
45 -        options.num_hidden_nodes_per_layer = [32 16 8];
46 -        model5 = trainNN(X,Y,options);
47          %##################################################
48 -    else
49 -        model = trainNN(X, Y, options);
50 -    end
51
```

While testing with ensemble averaging, I am using majority voting technique. Test example is labeled positive only when more than 3 neural networks out of 5 predict it as positive example.



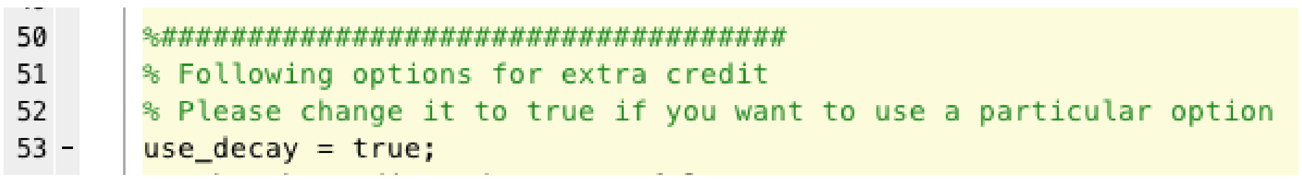
```
    if use_ensemble
        Ypred1 = testNN(model1, Xtest, options.activation);
        Ypred2 = testNN(model2, Xtest, options.activation);
        Ypred3 = testNN(model3, Xtest, options.activation);
        Ypred4 = testNN(model4, Xtest, options.activation);
        Ypred5 = testNN(model5, Xtest, options.activation);
        YPred = Ypred1+Ypred2+Ypred3+Ypred4+Ypred5;
        YPred(YPred<4) = 0;
        YPred(YPred>0) = 1;
        err = sum( YPred ~= Ytest ) / length(Ytest);
        fprintf('Test error is %.2f%%\n', 100 * err);
```

With this approach, I get test error = 5.26%.
But the number of false positives drops down. Take a look at the results using ensemble averaging. Activation used was sigmoid for this experiment. For this experiment, I have created tryEyeDetectorWithEnsemble.m



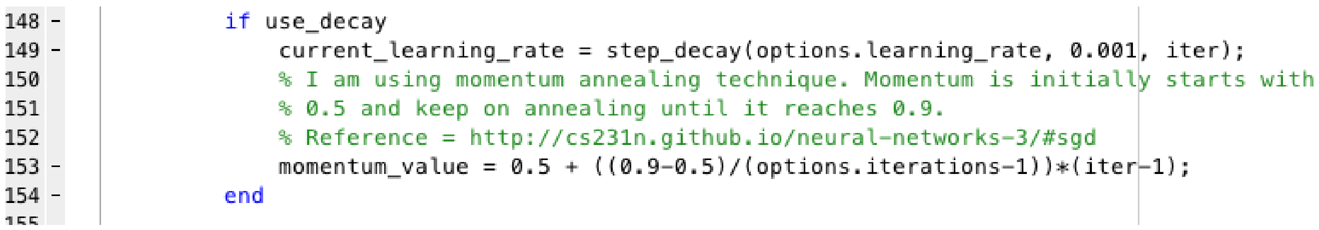
These results are much better than using just a single neural network.

d. Learning rate decay and Momentum update.



```
%#######################################
% Following options for extra credit
% Please change it to true if you want to use a particular option
use_decay = true;
```

When use_decay is set to true, stepwise decay is used for the learning_rate and the momentum value is slowly moved to 0.9 from 0.5. step_decay function can be found in step_decay.m



```
        if use_decay
            current_learning_rate = step_decay(options.learning_rate, 0.001, iter);
            % I am using momentum annealing technique. Momentum is initially starts with
            % 0.5 and keep on annealing until it reaches 0.9.
            % Reference = http://cs231n.github.io/neural-networks-3/#sgd
            momentum_value = 0.5 + ((0.9-0.5)/(options.iterations-1))*(iter-1);
        end
```