Assignment 4

AdaBoost:

The version of Ada Boost in "*Artificial Intelligence: A Modern Approach"* page 751 is suitable for binary classification. The problem at hand requires classifying the given set of test data into four classes.

Hence, a modified version of AdaBoost .i.e. AdaBoost SAMME is used.

Characteristics of AdaBoost SAMME:

1. Suitable for multiclass classification
2. Accuracy of the classifiers used in AdaBoost SAMME can be less than 50%.

Details for the AdaBoost can be found in the following paper titled Multi-Class AdaBoost: https://web.stanford.edu/~hastie/Papers/samme.pdf

As mentioned in the paper AdaBoost SAMME is very similar to traditional AdaBoost; only major difference is in calculation of alpha (or z as per algorithm in the book) where AdaBoost SAMME has an additional factor of log(K-1) where K is number of classes which is 4 in our case

Design Decisions:

1. Classifiers with accuracy lower than 50% are selected.
2. A total of 20 weak classifiers are used. Certain weak classifiers used are: First & Last row of pixels, First & Last column of pixels etc.
3. The RGB component for each pixel is added and considered as one single value. And for each classifier a set of 8 pixels are considered.

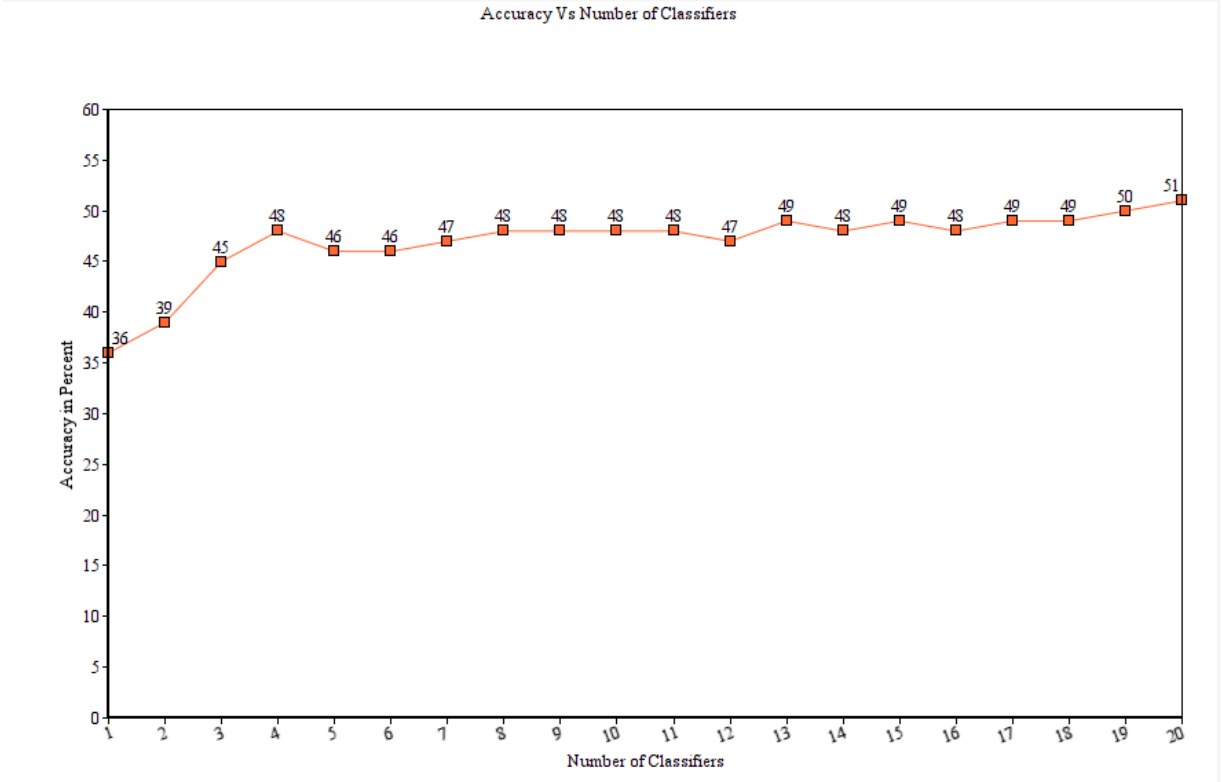
Out of the 20 classifiers used some the relatively stronger classifiers use following eight pixels:

1. Eight pixels in first column.
2. Eight pixels in the last row.
3. Combination of 8 pixels from the last two rows.

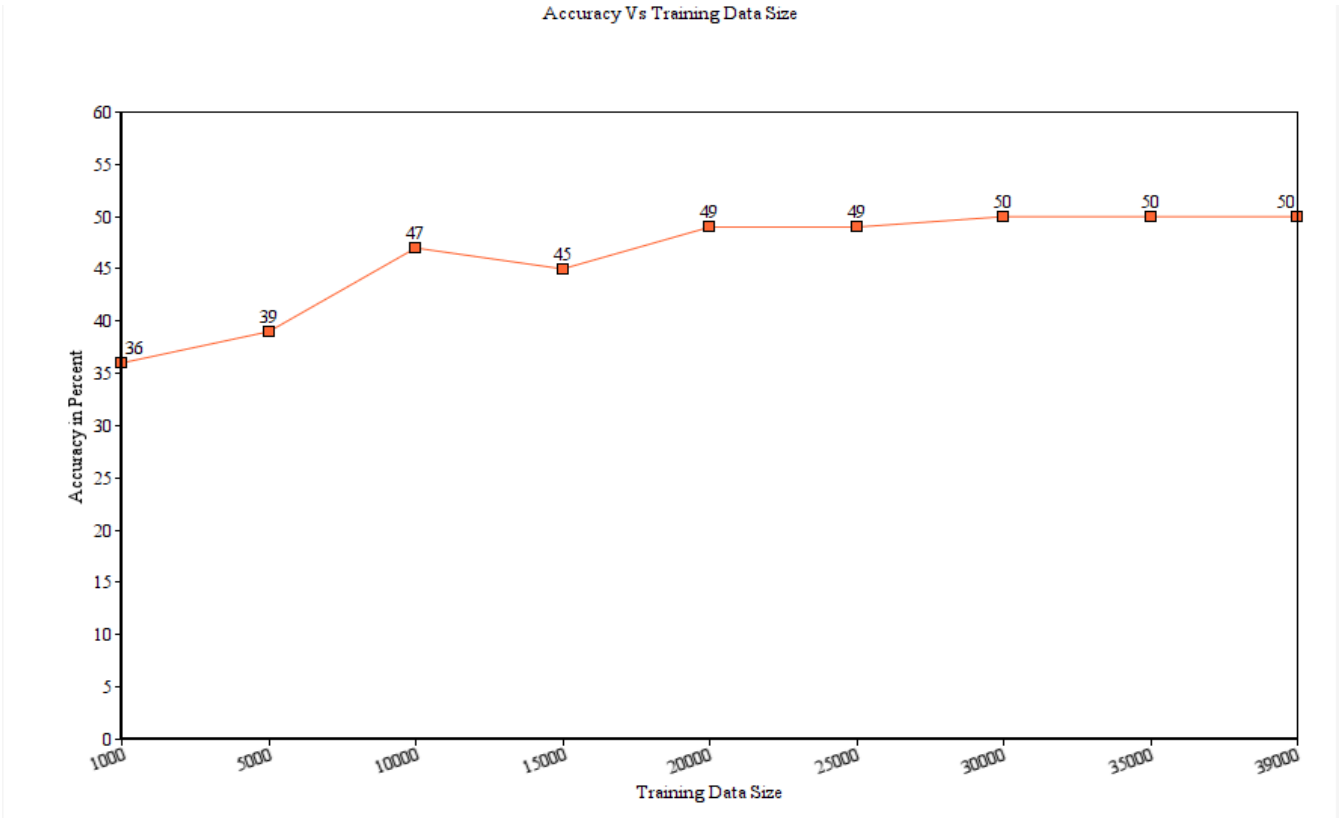
The above-mentioned features are better representative of orientation of the images. Accordingly, pixels in the center of the image have relatively less information about the orientation of the image.

The Final Accuracy obtained is 51%

The following graph shows accuracy as a function of number of classifiers:



The following graph shows varying accuracy for AdaBoost SAMME based on changing training dataset size:



Observations:

1. Accuracy for the above-mentioned method can be further increased by starting with relatively stronger classifiers. The classifiers used have accuracy in 32-36%.
2. Accuracy can also be increased further by adding more classifiers.

K Nearest Neighbors:

Design Decisions:

1. To find the nearest neighbor we use Euclidean Distance.
2. To reduce the run runtime of KNN, we added the R G and B component of a pixel and the total is considered as a one single intensity value for that pixel.

Varying Accuracy based on different values of K:

K	Accuracy
5	62.56
7	63.09
9	63.41
11	64.05
13	64.68
15	65.11
17	65.64
19	65.32

A value 17 is used for final classification as it offers maximum accuracy.

Neural Net Implementation:

We have implemented the backpropagation algorithm with traditional gradient descent using the following parameters.

No of inputs in the input layer = 192

No of layers =3 (1 input, 1hidden and 1 output)

No of neurons in the hidden layer =20

No of neurons in the output layer = 4

Activation function = Sigmoid function

Learning rate(Alpha) = 0.01

We initialized the weights for each layer using a random function generating a value between -1 and 1.

Results:

We trained our feed forward network by randomly shuffling the order of the images being passed to the input layer in every iteration. Initially we trained for around 10 iterations and gradually increased it to 1000 iterations.

Size of Training Data(No of images)	Size of Test Data(No of images)	No of Iterations	Time taken (training and testing combined)	Accuracy
500	1K	3	3.56s	37.84%
20K	1K	3	15.682s	47.72%
40K	1K	3	22.77 s	46.76%
40K	1K	10	1min 19s	57.05%
30K	1K	100	7min 9s	66.34 %
40K	1K	100	15min 3.3 s	60.55%
40K	1K	500	73min 54 s	65.55%
40K	1K	1000	155min 4s	67.12%

We observed that as we increase the number of iterations and the size of the training data set, the network got trained better and gave improved accuracies.

We also tried with 16, 25 and 100 neurons in the hidden layer. Although increasing neurons improved accuracy, but we were apprehensive of overfitting the data and hence stuck to our best result of 67.12 % accuracy with 20 neurons in the hidden layer, training for 1000 iterations on 40K data and testing on 1K data. Thus, we would recommend these parameters along with the set of parameters mentioned earlier to any potential client.

Best Algorithm

Out of the three algorithms, we are getting the best accuracy with the neural network implementation. So our recommended best algorithm is neural network.