AI-Assignment -5

Final Report – Trend Analysis.

Part 4: ANALYSIS and IMPROVEMENT:

1) **K-Nearest Neighbors:**
   Best Parameter tuning analysis for K – Nearest Neighbors:
   Parameter Tuned: k (Number of Nearest Neighbors)

Algorithm	Parameter Tuned	Accuracy	Time Taken
K – Nearest Neighbors	1	67.232	446.486
K – Nearest Neighbors	2	65.217	467.999
K – Nearest Neighbors	10	70.307	442.273
K – Nearest Neighbors	25	71.049	475.733
K – Nearest Neighbors	40	71.0498	420.371
K – Nearest Neighbors	50	71.261	418.451
K – Nearest Neighbors	75	70.519	428.441
K – Nearest Neighbors	100	70.307	425.845

Accuracy Table: The highest accuracy has been highlighted

As seen in the table above, the best accuracy of the model was obtained with 50 nearest neighbors. Hence building the model with 50 nearest neighbors

Trend Analysis of Top 25 test images – using K – nearest neighbors

Image	Predicted Class Label	Actual Class label
test/20506775873.jpg	270	270
test/7980366417.jpg	0	0
test/22390308255.jpg	180	180
test/10795283303.jpg	180	180
test/5596775850.jpg	90	180
test/5544033474.jpg	180	180
test/352506989.jpg	90	90
test/2689816071.jpg	0	180
test/6257414280.jpg	90	90
test/18060875084.jpg	90	270
test/216401349.jpg	180	180
test/9253749416.jpg	270	270
test/34893126.jpg	0	180
test/15157966369.jpg	270	270

test/3529395659.jpg	270	270
test/14516765865.jpg	0	0
test/10484444553.jpg	0	180
test/16271718816.jpg	270	90
test/218481531.jpg	180	180
test/335176365.jpg	90	90
test/27719307.jpg	180	180
test/4687466195.jpg	270	270
test/6432902573.jpg	0	0
test/13503513433.jpg	0	0
test/15474149268.jpg	90	90

On analyzing the results of the prediction model in top 25 test images, using the K – nearest neighbor model, the trend shows some similarity to that of Adaboost.
Even for kNN, the model has the most misclassifications in predicting the 180 degree rotated image, although it has none for straight oriented (0 degree) images.
The model has performed well for 90 and 270 degree rotated images as well.

But this analysis is just based on the above 25 images. The trend observed could be biased due to the less number of samples.
Let us observe on the entire test set images.

Confusion Matrix:
K = 50
Accuracy = 71.261%

Prediction Model Results

Actual Model	**0**	**90**	**180**	**270**	**Actual Count**
0	170	18	32	19	239
90	20	166	13	25	224
180	36	23	166	11	236
270	23	39	12	170	244

On considering the entire test set, we could observe a similar progression as compared to the AdaBoost model. The most correctly classified classes are 0 – degree and 270- degree rotated images. We have obtained a good improvement in the accuracy, owing to the increase in the number of correctly classified images in both 90 and 180- degree rotated images.

NOTE: For the "best" configuration, our code will run the knn algorithm, since its giving the highest accuracy.

2) **AdaBoost:**
Best Parameter tuning analysis for Adaboost:
Parameter Tuned: Stump Count (Number of Iterations)

Algorithm	**Stump Size**	**Accuracy**	**Time Taken**
Adaboost	1	32.449	1.0948
Adaboost	2	42.417	1.1216
Adaboost	3	33.828	1.215
Adaboost	4	43.584	1.540
Adaboost	5	43.478	1.380
Adaboost	6	49.098	1.495
Adaboost	7	51.113	1.571
Adaboost	8	51.476	1.726
Adaboost	9	51.643	1.766
Adaboost	10	50.901	1.878
Adaboost	20	56.203	2.631
Adaboost	25	59.597	3.059
Adaboost	30	60.339	3.451
Adaboost	35	62.142	3.929
Adaboost	50	64.262	5.0354
Adaboost	100	65.11	12.689
Adaboost	200	66.38	23.49
Adaboost	300	65.11	35.35

Accuracy Table: The highest accuracies have been highlighted

Note: The accuracy is low for low values of stump (< 10) since we are taking random points in each stump and not trying to find the best ones. Also, finding the best combination pairs would approximately give the same accuracy (difference of ~2-3% in accuracy), but takes atleast ~50 times more time to execute.

The accuracy rises quickly at first, but then slows down once the stump size becomes greater than 25. It gets consistent and stops improving much after stump = 100.
Also, since we are taking random points, the accuracies may vary (by ~2-3%) even for the same stump value.

As seen in the table above, the accuracy stops improving much after stump = 100 and hence we will consider this stump value to be giving us the best accuracy.

Below are the randomly selected 25 test outputs images.
Trend Analysis of Top 25 test images – using Adaboost

Image	Predicted Class Label	Actual Class Label
test/8848438121.jpg	180	180
test/3483554595.jpg	180	180
test/4720595075.jpg	90	90
test/12178947064.jpg	90	90
test/3916462478.jpg	90	0
test/15169136829.jpg	90	90
test/6822359662.jpg	90	90
test/16264419.jpg	270	270
test/21305225519.jpg	0	90
test/416997815.jpg	0	0
test/11273674.jpg	180	180
test/15468193222.jpg	270	180
test/11418669873.jpg	180	0
test/3856487590.jpg	0	0
test/4313726879.jpg	270	180
test/8130077012.jpg	180	0
test/16196776241.jpg	270	90
test/15610971496.jpg	90	90
test/15670709176.jpg	90	180
test/14514200941.jpg	0	270
test/5105562637.jpg	90	180
test/15473159366.jpg	0	0
test/8362818631.jpg	0	0
test/8230858567.jpg	90	0
test/14348917526.jpg	270	270

On analyzing the results of the prediction model in top 25 test images, the model has performed the best in the predicting the 90 degree orientation of the image. The next best performance of the model is on predicting 270-degree rotated images. The model has the most misclassifications in predicting the 0 and 180 degree rotated image.

But this analysis is just based on the above 25 images. We will now see the complete confusion matrix for AdaBoost at stump_count = 100

Confusion Matrix:
Stump Count: 100
Accuracy: 65.111%

Prediction Model Results

Actual Model	**0**	**90**	**180**	**270**	**Actual Count**
0	161	31	29	18	239
90	30	157	12	25	224
180	39	43	133	21	236
270	34	32	14	164	244

On analyzing the trend in the entire set of test images, the best performance of the model is in predicting the 270-degree rotated images, but the minimum number of misclassifications is for the 0-degree rotated images. Considering the fact that, the 270 - degree rotated images are 5 more than the 0 –degree rotated images in the test set it is a close call between the 0 and 270 – degree rotated images.
Also, it has failed the most for 180 degree images.

3) **Neural Network:**
Design Decision:
1) Firstly, to improve performance, we tried not using all the 192 features for Neural Network. Instead, we were taking an average of 2 consecutive neighbors and storing as one value. This made our input feature vector for each train/test file to be of 96 instead of 196.
This was done with an intuition that not much changes are captured in consecutive pixels and hence the information loss is minimal.
This trick gives us a little performance improvement, but sacrificed ~3-5% of the accuracy. Hence, reverted the change back and used full set of features.
The accuracy table contains the test of both the configurations
2) Also to avoid huge calculations, we are normalizing both the train and test input vectors.

Best Parameter tuning analysis for Neural Network:
After various tests, we came to a conclusion that below are the best parameter settings for our Neural Network implementation:
Best Fixed Parameter:
No of hidden layers = 1

Best Variable Parameters:
Hidden Count = 25
Iterations = 150
Learning Rate (α) = 0.9 *(I am increasing its power by 1 after every 10 iterations eg. After 20 iterations- $\alpha = (0.9)^2$; after 30 iterations- $\alpha = (0.9)^3$ and so on…)*
Activation Function = Stochastic Gradient Descent

Hidden Count	Learning Rate	Iterations	Accuracy	Time Taken
Half/Avg Features:				
50	0.1	100	65.111	465.92
10	0.1	100	64.68	154.95
10	0.1	1000	69.88	1430.86
50	0.2	100	66.80	200.34
20	0.2	500	69.88	1083.20
10	0.15	500	69.35	1115.82
100	0.1	1000	69.88	3435.91
10	0.05	200	64.68	478.59
Full/All Features:				
10	0.2	100	67.97	254.95
20	0.1	100	68.29	455.95
20	0.1	150	70.09	620.32
50	0.1	200	71.68	1135.11
50	0.1	125	69.77	660.40
25	0.1	150	70. 71.36	517.84
25	0.7	150	71.3	656.06
10	0.6	100	71.47	348.52
25	0.9	100	71.68	287.01

Accuracy Table: The highest accuracy has been highlighted

Below are the randomly selected 25 test outputs images.

Trend Analysis of Top 25 test images – using Neural Nets

Image	Predicted Class Label	Actual Class Label
test/20506775873.jpg	270	270
test/7980366417.jpg	0	0
test/22390308255.jpg	180	180
test/10795283303.jpg	180	180
test/5596775850.jpg	90	180
test/5544033474.jpg	180	180
test/352506989.jpg	90	90
test/2689816071.jpg	0	180
test/6257414280.jpg	90	90
test/18060875084.jpg	0	270
test/216401349.jpg	180	180
test/9253749416.jpg	270	270
test/34893126.jpg	0	180
test/15157966369.jpg	270	270

test/3529395659.jpg	270	270
test/14516765865.jpg	0	0
test/10484444553.jpg	0	180
test/16271718816.jpg	270	90
test/218481531.jpg	180	180
test/335176365.jpg	180	90
test/27719307.jpg	180	180
test/4687466195.jpg	270	270
test/6432902573.jpg	0	0
test/13503513433.jpg	0	0
test/15474149268.jpg	90	90

On analyzing the results of the prediction model in top 25 test images, the model has performed the best in the predicting the 0 and 270degree orientation of the image. The next best performance of the model is on predicting 90-degree rotated images. The model has the most misclassifications in predicting the 180 degree rotated image.
But this analysis is just based on the above 25 images.

Below is the complete confusion matrix for Neural Network:

Confusion Matrix:
Hidden Count = 25
Iterations = 100
Learning Rate (α) = 0.9 (Initial value. Then updating it as explained above.)
Accuracy = 71.686%

Prediction Model Results

Actual Model	**0**	**90**	**180**	**270**	**Actual Count**
0	172	18	31	18	239
90	15	163	18	28	224
180	26	27	163	20	236
270	23	26	17	178	244

Again, Neural Network performs well on 0 and 270 degree
On analyzing the trend in the entire set of test images, the best performance of the model is in predicting the 270-degree and then 0-degree rotated images, but the minimum number of misclassifications is for the 0-degree rotated images.
Also, it has failed the most for 180 degree images, but still performed better than Adaboost for such images.

Client Recommendation:

My recommendation to the client would really depend on his requirement, since there is a trade-off between speed and accuracy.
kNN and Neural Net provide high accuracy, but are kind of slow.
I would recommend the following, based on the requirements:

1) **kNN:** If the client has a small data set and needs a high accuracy, then KNN would be a good fit.
   But if the data set is too large, then kNN would take forever to produce an output.

2) **AdaBoost:** If the client has a large data set and is looking for almost a real time output (like may be for websites/apps) and can hence compromise a bit on accuracy, then AdaBoost would be a good fit.

   But if the accuracy is of utmost importance, then I would recommend AdaBoost, since it is a random approach and the accuracy fluctuates even on the same data set.
   Note: I believe the accuracy of AdaBoost can also be improved if the best/or better combination points are taken instead of random ones

3) **Neural Net:** This is again a good accuracy, but time consuming algorithm.
   So, the client will have to compromise on the time factor at least.

   Note: I believe that Neural Net can give the best accuracy if more hidden layers (not nodes) are added to it and also if more iterations are done.

But all in all, assuming my client is a part of the real world, with a not so small dataset and a requirement of a relatively quick output, I would recommend **AdaBoost** algorithm.

Note: Although, when running for the “best” configuration, we are executing kNN, since that’s the one giving the highest accuracy for now.