Xre-Lab – Homework-4

Problem-1

- For the Problem1 we, are using LBP and HOG as feature maps for training a model using SoftMax classifiers and Feedforward Neural Networks, as well as CNNs to do Face recognition
- We use the Orl dataset, the dataset is small and may be data augmentation might be helpful
- For first case we use Softmax+LBP to do the classification
- We have 40 subjects here, hence 40 classes
- Each subject has 10 images in total
- We have to use 60-40 split
- We need to one-hot encode the labels for both training and testing purposes

Results and discussions

LBP+Softmax
Step 0: training accuracy 0
Step 100: training accuracy 0.4
Step 200: training accuracy 0.9
Step 300: training accuracy 1
Step 400: training accuracy 1
Step 500: training accuracy 1
Step 600: training accuracy 1
Step 700: training accuracy 1
Step 800: training accuracy 1
Step 900: training accuracy 1
Test accuracy 0.5375

- For this case, the accuracy come out to be nearly 54%, which is not bad considering we use softmax classifiers with using LBP as the feature map
- LBP has the same dimensions as the image

HOG+Softmax
Step 14600: training accuracy 1
Step 14700: training accuracy 0.8
Step 14800: training accuracy 1
Step 14900: training accuracy 0.9
Test accuracy 0.7875

- For this case, the # of epochs is increased to 15000, and we reach very good test set accuracy of 80%, which is really good through the use of Softmax classifiers
- HOG is a 1D list, so there has to be some changes from the tensor flow code in respect to input dimensions

- The reason I increase the Epoch as the training is slow, as with HOG there is less overfitting and I can train for a longer time

LBP+FNN/CNN
Step 0: training accuracy 0
Step 100: training accuracy 0
Step 200: training accuracy 0.2
Step 300: training accuracy 0.3
Step 400: training accuracy 1
Step 500: training accuracy 1
Step 600: training accuracy 1
Step 700: training accuracy 1
Step 800: training accuracy 1
Step 900: training accuracy 1
Test accuracy 0.8375

- For this case, the test set accuracy is better and goes up to nearly 84%, I have changed the input size to 28x28 for the ease of using tensorflow, the filter size used here is 3x3 rather than 5x5, filter_size of 3 is better
- I changed the input size to 92x92 , which makes everything go for the worse, to a test-set accuracy of 14%
- Confusion matrix shows that The number of False positives are low

HOG+FNN/CNN
Step 0: training accuracy 0
Step 100: training accuracy 0.1
Step 200: training accuracy 0.1
Step 300: training accuracy 0.1
Step 400: training accuracy 0.4
Step 500: training accuracy 0.2
Step 600: training accuracy 0.7
Step 700: training accuracy 0.9
Step 800: training accuracy 0.9
Step 900: training accuracy 1
Step 1000: training accuracy 1
Step 1100: training accuracy 1
Step 1200: training accuracy 1
Step 1300: training accuracy 1
Step 1400: training accuracy 1
Step 1500: training accuracy 1
Step 1600: training accuracy 1
Step 1700: training accuracy 1
Step 1800: training accuracy 1
Step 1900: training accuracy 1
Test accuracy 0.925

- For this case, the test set accuracy is better and goes up to nearly **93%**, I have changed the input size to 16x16 for the ease of using tensorflow, the filter size used here is 5x5, number of epochs = 2000
- My HOG output length 280, so I reduced and used only 256 out of 280 for each image, that reduces some of the information, but that was necessary to using 16x16 as my input to the FNN
- Confusion matrix shows that the number of False positives is low, only one wrong prediction

Problem-2

- For the Problem2 we are making a end to end Face recognition system using CNNs
- The difference and complications arise with the use of demo.py , which is provided here to do the data-augmentation

Issues while using demo.py

- Demo.py only works with Linux environment
- We need to use py-3
- I used Paperspace to run the demo.py code
- The Path given to demo.py has to be exactly in the form : "input/s1im1.jpg", we cannot give any other form, not even a variable that stores the path
- We have to install **dlib**, package using conda forge, to make things work
- I also installed face_recognition
- The program does not recognize all the images so do not expect 6*50=300 images for all subjects , some will have less than that, so we need to pick 156 which is the lowest, I need to think how to tackle this
- Link to the Jupyter notebook - https://github.com/sananand007/Xrelab/blob/master/hw4/Homework4-P2_fin%20(1).ipynb
- I use a two later conv layer with max-pooling, and dropout with a big dense layer to get the output

- We use the Orl dataset, the dataset is small and may be data augmentation might be helpful
- We have 40 subjects here, hence 40 classes
- Each subject has 10 images in total
- We have to use 60-40 split
- We need to one-hot encode the labels for both training and testing purposes

Results and discussions

Step 4500: training accuracy 1
Step 4600: training accuracy 1
Step 4700: training accuracy 1
Step 4800: training accuracy 1
Step 4900: training accuracy 1
Test accuracy 0.2375

- **I get the worst performance for this case, looks like data augmentation is not helping but going in the wrong direction**
- **Test accuracy reaches almost ~30%,**