Problem 1

(1.1)

(a) As a warm-up, use the raw pixels as features to train a linear svm. Plot the error rate vs the number of training examples.

Here we plot the accuracy vs. number of training examples for raw pixel features.

(1.2)

(a) Explain briefly why adding these features should help over raw pixel values.

The local sums provide more information about larger neighborhoods in the image, which equates to more relational information. The overlapping windows provide some translational invariance.

(b) Implement these features and use them to train a linear svm. Plot the error rate vs the number of training examples. Do you get a significant boost over Q1?

Problem 1 [(1.2)] continued on next page...

Here we plot the error rate vs. number of training examples for both raw pixel features alone (in Blue) and raw pixel features plus local cxc neighborhood sums with overlapping windows of size c/2 for c=4,7 (in Green). For a small number of training examples our performance is better with the additional features than for the raw pixel features alone. However, with more training examples the raw pixels surpasses the raw pixels plus the pyramid features.

(1.3)

(a) Explain briefly why gradient orientations should help over pixel intensities : what is the gradient orientation trying to capture?

ANSWER GOES HERE

(b) Implement these features and use them to train a linear SVM. For computing the gradient, use the tap filter:[?1 0 1]. Use 9 orientation bins. Use the same cell sizes (4 and 7) as in Q2. Plot the error rate vs the number of training examples. Do you get a significant boost over Q2?

ANSWER GOES HERE

(c) Does the performance drop if you dont normalize the histograms before concatenating them? Why or why not? Plot error rate vs the number of training examples.

ANSWER GOES HERE

Problem 1 [(1.3)] continued on next page...

(d) How does performance change if you replace the tap filter with a Gaussian derivative filter?

ANSWER GOES HERE

(e) Explain how you choose the hyper-parameter in linear SVM and why.

ANSWER GOES HERE

(1.4)

(a) Visualize the images on which you go wrong. Do the errors your classifier makes seem reasonable to you? How many of these images do you as a human have a hard time recognizing?

ANSWER GOES HERE

Problem 2

(2.1)

(1) Draw a schematic of the neural network architecture described above. Clearly label all the layers of the network (including the non-linearity).

ANSWER GOES HERE (+FIGURE)

(2) Train this network for digit classification. Report the accuracy on the test set. How many parameters does this network have?

The accuracy for the two layer fully connected network is ???. The network has 20*784 parameters for ip1 and 500*20 parameters for ip2, for a total of 25680 parameters.

(3) Use three hidden layers of 20, 50 and 500 units respectively. Report the accuracy on the test set. How many parameters does this network have?

The accuracy for the three layer fully connected network is ???. The network has 20*784 parameters for ip1, plus 50*20 parameters for ip2, and 500*20 parameters for ip3, for a total of 25680000 parameters.

(4) Compare the performance of both these networks. Why is the performance greater/lower or the same?

ANSWER GOES HERE

(5) Change the non-linearity to Sigmoid and re-train the network. Do you observe any differences? If the depth of the network is increased - what effect would the Sigmoid non-linearity have on the magnitude of the gradients? Is this desirable? Is the behavior of ReLu units different?

ANSWER GOES HERE

(2.2)

(1) Report the performance of the LeNet architecture. How many parameters does this network have? Train the LeNet using only 10K examples and compare the performance with the SVM classifier you trained in the first part of the homework.

The LeNet Architecture has 430500 parameters. The performance for 10K examples is 97.36%, compared to 97.28% correct for the normalized Tap filter SVM classifier.

(2) Both LeNet and FC network have 20, 50, 500 units in the three hidden layers. Considering this, compare the performance of LeNet with the Fully connected(FC) network with three hidden layers.

The LeNet network has a performance of 97.36% compared to ??? for the three-layer FC network.

(3) Now, reduce the number of units in the third hidden layer of the LeNet to 32. Lets call this LeNet2. Compare the performance with the fully connected network. Why is the performance greater/lower or the same?

The performance of LeNet2 is 98.58% compared to ??? for the three-layer FC network. EXPLANATION GOES HERE.

(4) Experiment with the kernel sizes in LeNet2. Try kernel sizes of 3 and 7. What do you find?

The performance of LeNet2 is 98.38% and 98.53% for kernel sizes of 3 and 7, respectively.

(5) Visualize the first layer filters of the original LeNet network. What do you find? Based on this visualization - can you comment on what computation first layer is performing?

Here is a visualization of the first layer filters (note: image size is very small).
FURTHER EXPLANATION GOES HERE

(6) Based on your results, list the pros and cons of using a FC network in comparison to the ConvNet. Is the ConvNet more scalable to larger images? If yes, why and if no, why not?

ANSWER GOES HERE