CSC321: Project 3

Due on Monday, March 21, 2016

Mohdhar Noor, Maxwell Huang-Hobbs
(g3theuma), (g4rbage)

March 20, 2016

Part 1

Classification using single hidden layer neural network

The images were resized to 32×32 and converted to greyscale. The weights and biases in both layers were initialized to random values generated using `tf.random_normal` a Guassian distribution with a mean of 0 and a standard deviation of 0.01. The activation function used in the hidden layer was *tanh*. We used a fully connected network with a flattened 784 element vector denoting an image, a 300 unit hidden layer and 6 element one-hot enconding for a output, as was done in project 2. The learning rate is shown in Figure 2. The final learning rates for the network were 0.87 for the training set, 0.80 for the validation set and 0.77 for the test set.

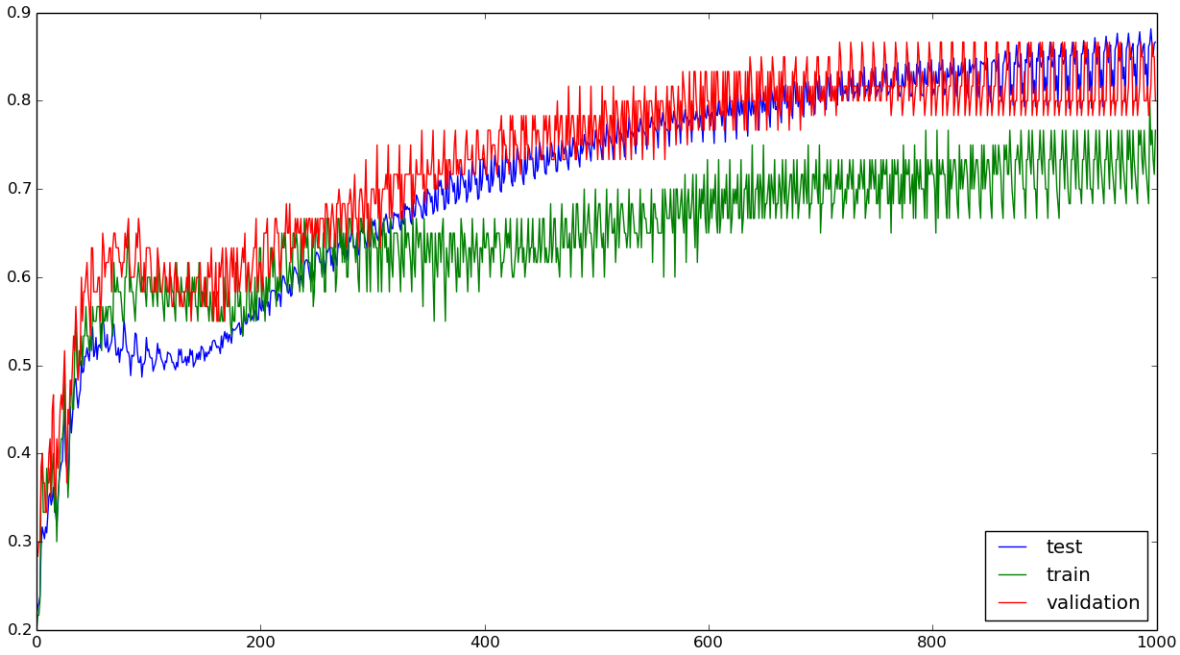

Figure 1: learning rate of a single hidden layer network

Part 2

Classification using the Colvolutional layers of Alexnet

In order to use the provided alexnet implementation, a new dataset of colored 227*227 RGB images was scraped. The provided alexnet weights were used as constants in the network, and fed into a fully connected neural network.

This is the same as running alexnet up to conv4 on the input data, recording the output, and feeding the information into a fully connected network.

The fully connected layer was initialized with random weights and biases, distributed using a Gaussian distribution with $\sigma = 0.001$. Gradient descent was performed with a learning rate of 0.005.

In order to shorten computation time, the network's performance was evaluated once every 100 generations. (x scale in 100s of generations)

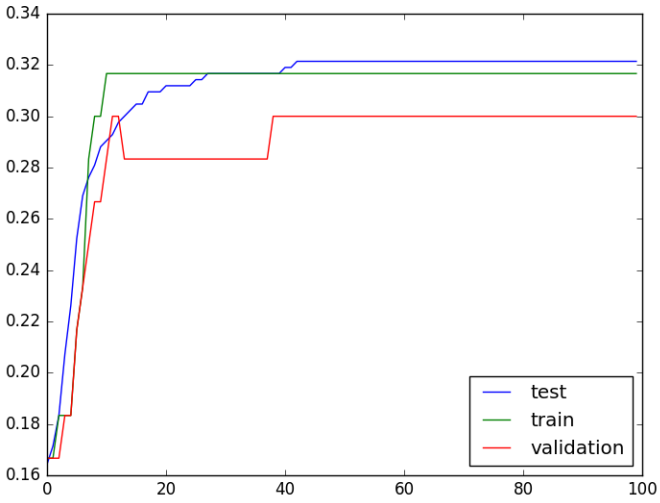

Figure 2: learning rate of the neural network.

The neural network seems to level out at about 30% accuracy at classification. This suggests that the network was poorly initialized and the network becae trapped in a local optima.

Likely this could have been fixed by increasing the batch size that the network learned on. The provided alexnet implementation could not batch process inputs, and due to time constraints we did not implement it ourselves, and so were forced to train with a batch size of 1.

Part 3

Visualizing Hidden Weights.
The weights used was the weight matrix connecting the features to the hidden layer ('hiddden weights'). The hidden weights for the 300 unit hidden network display discernable components of an actor's face such as the brows, nose, mouth, hairline and jaw while the 800 unit weights do not exhibit that level of detail.

(a) Example feature from the 300-hidden network.

(b) Example feature from the 300-hidden network.

(c) Example feature from the 800-hidden network.

(d) Example feature from the 800-hidden network.

Figure 3

Figure 7 Shows examples of features in the hidden layer of the 300 and 800 layer neural networks. They seem to be selecting strongly for certain facial features and against specific placements of other facial features. For example, figure 3b appears to be selecting towards a particluar placement of the eyes and against a specific relative positioning of the nose. This might suggest that some actors / actresses in the dataset can be distinguished by the relative position of their eyes and nose.

Part 4

Classification using AlexNet
Our implementation of part 2 covers the requirements for part 4 (the AlexNet input layers up to conv4 are implemented as constants, and the fully connected layer is trained on the output of part conv4).
AlexNet's layers from conv1-conv4 were fixed as constants using the provided weights, and a fully connected network was added from the conv4 output to the output of the network.

Network Output: 1
(Gerard Butler)

Figure 4: Example of the neural network classifying a face correctly

Network Output: 1
(Lorraine Braco)

Figure 5: Example of the neural network classifying a face incorrectly

Part 5

Gradients of the input layer
The gradient of the softmaxed output of the network with respect to any input was consistently 0, i.e. all the neurons on the input layer are dead.

Figure 6: Gradient of output likelihoods wrt input image

When the gradient of the non-softmaxed output is visualized, It gives images of noise with smooth areas around edges of the image, with what appear to be diamond patterns over most of the image
Its possible this is from the network memorizing features of the training set and using these as the criteria to decide between classes.

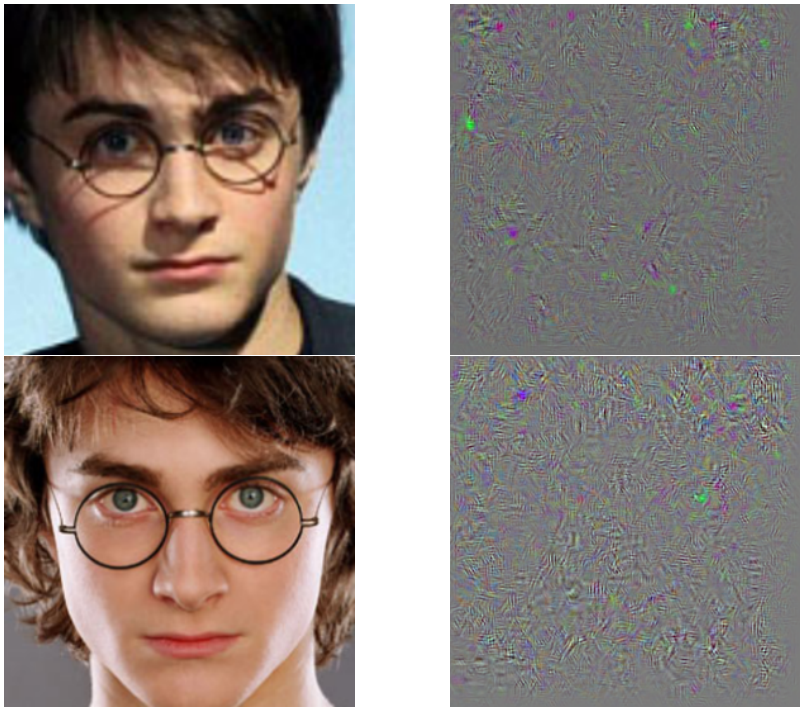

Figure 7: Gradient of output wrt input image