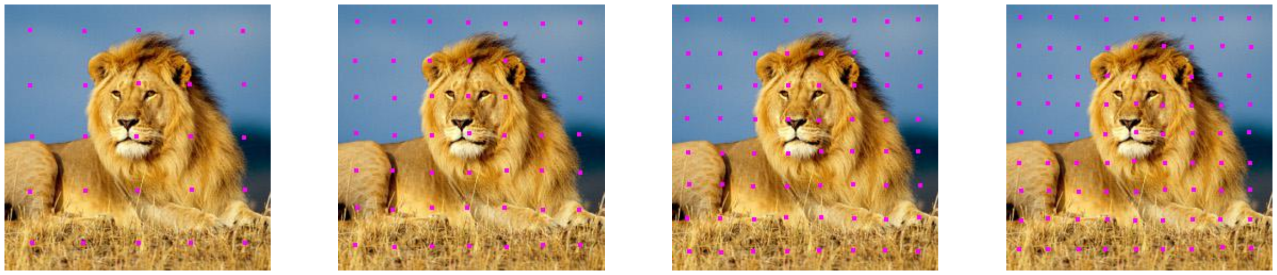
Solution 1

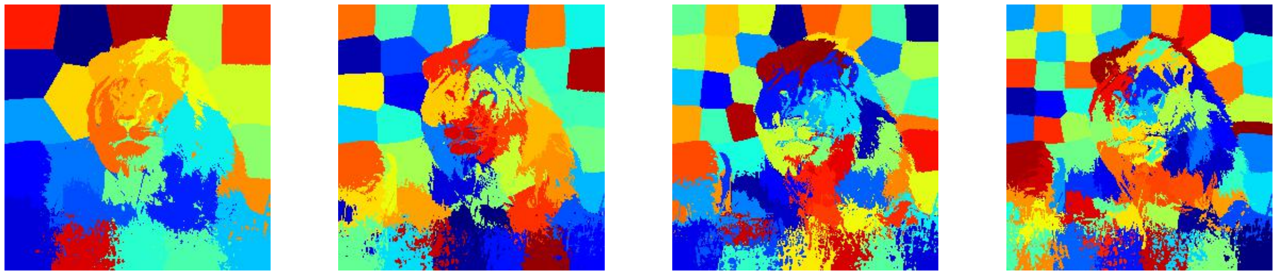
(a) Implementing the seeding algorithm to place the cluster centers on a rough grid resulted in the following centers.



(a) 25 Centers (b) 49 Centers (c) 64 Centers (d) 81 Centers

Figure 1: Placing cluster centers for various cluster amounts.

(b) The SLIC algorithm was then implemented, the result of which is shown below for various numbers of clusters.



(a) 25 Centers (b) 49 Centers (c) 64 Centers (d) 81 Centers

Figure 2: SLIC algorithm superpixels using various numbers of clusters.

Solution 2

(a) Changing decreasing the batch size to 25 results in a better final accuracy compared to a batch size of 25.
Increasing the learning rate allows the loss to decrease faster, thus increasing accuracy faster. By 50 epochs, the validation accuracy reaches 95%. This however, is prone to overlearning.
Decreasing the number of hidden units to 512 causes the model to become "simpler", in that there are less parameters to learn. This led to a faster runtime and a final validation accuracy of 92.10%
The limits on the uniform distribution are chosen such that the variance of the weights is $1/n$.

(b) Adding momentum to the gradient seems to improve validation accuracy. This is likely due to the learning occuring faster. With a batch size of 50 and learning rate of 0.01, a validation accuracy of 97% was achieved. This is an improvement over the plain SGD, which achieved 95% validation accuracy. Furthermore, adding momentum increased the learning speed. With a learning rate of 0.001, the experiment with momentum took 10 epochs to achieve 90% validation accuracy whereas the plain SGD took 17 epochs.

Solution 3

The convolutional layer object created implements the forward and backwards directions. When running the MNIST training program that was given, 90% validation accuracy was achieved. After changing the learning rate to 0.01, 95% validation accuracy was then achieved.

Information

This problem set took approximately 20 hours of effort.

I discussed this problem set with:

- Myself

I also got hints from the following sources:

- Wikipedia article on matrix calculus at https://en.wikipedia.org/wiki/Matrix_calculus
- Read numpy tutorial from https://docs.scipy.org/doc/numpy-1.13.0/user/basics.broadcasting.html
- numpy.moveaxis documentation https://numpy.org/doc/stable/reference/generated/numpy.moveaxis.html