Solution 1

(a) Outputs for the `get_centers` function:

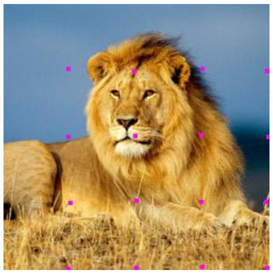

Figure 1: K=25

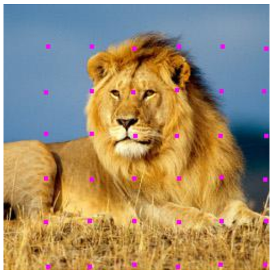

Figure 2: K=49

Figure 3: K=64

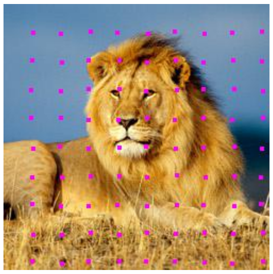

Figure 4: K=100

(b) Here are the outputs from my `slic` function. After experimenting, I set the spatial weight value to 5 (leaving the intensity weight at 1). I evaluated images based on the criteria that superpixels should be a) more or less spatially contiguous, and b) capture visually informative regions of the image. I found that a value of 4 or 5 gave the best results for this image. (I also tried normalizing the spatial and intensity distances to a common scale, but I found that this did not improve results).

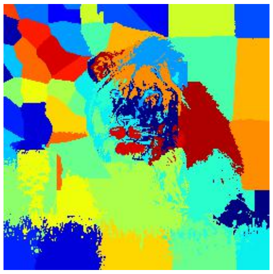

Figure 5: K=25

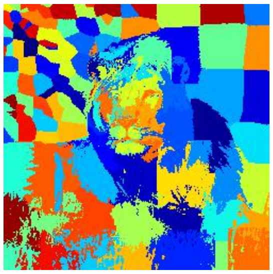

Figure 6: K=49

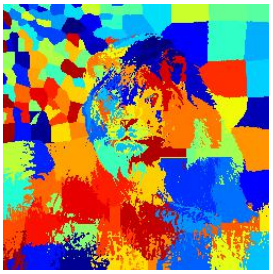

Figure 7: K=64

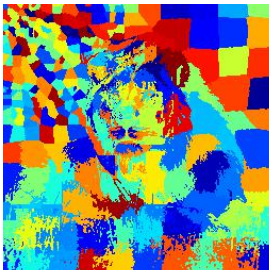

Figure 8: K=100

Solution 2

(a) Experimental results after 50 epochs:

Batch size	Learning rate	Hidden units	Training		Validation	
			Soft-max loss	Accuracy	Soft-max loss	Accuracy
50	**.001**	1024	.5390	85.70%	.4732	88.80%
50	**.005**	1024	.5623	83.12%	.4915	85.10%
50	**.010**	1024	.8127	72.78%	.7115	75.7%
10	.001	1024	.5287	83.40%	.4916	85.2%
25	.001	1024	.4582	86.52%	.4069	89.40%
100	.001	1024	.7639	80.45%	.6991	83.90%
50	.001	**128**	.4844	87.26%	.4394	89.6%
50	.001	**256**	.5067	87.72%	.4498	90.40%
50	.001	**512**	.5119	86.78%	.4619	89.30%
50	.001	**2048**	.6143	81.90%	.5632	84.70%
25	.001	256	.4566	87.24%	.3818	90.60%

The final trial combines the optimal values of each variable in the previous trials. Note that the accuracy statistics are only slightly improved from the (hidden units = 256) trial, suggesting that the number of hidden units may be the dominant variable.

Xavier initialization chooses the initial weights from a distribution with mean 0 and variance equal to $\frac{1}{n_{\text{in}}}$ where n_{in} is the number of inputs to the neuron. In brief, this is because we want the variance of input and output to be the same, so that the variance of the signal does not diminish (or increase) as it penetrates deeper into the network. It can be shown that the variance of the output is proportional to n x the variance of the weights themselves, so setting the input distribution's variance to $\frac{1}{n}$ accomplishes this.

(b) Results with momentum:

Batch size	Learning rate	Hidden units	Training		Validation	
			Soft-max loss	Accuracy	Soft-max loss	Accuracy
25	.001	1024	.005347	98.8%	.1027	97.2%
50	.001	1024	.01153	96.96%	.1392	96.1%
100	.001	1024	.01929	94.5%	.1925	95.1%
25	.001	128	.006691	98.12%	.1171	96.6%
25	.001	256	.006241	98.4%	.1007	97.2%
25	.001	512	.006112	98.84%	.1072	96.60%
25	.001	2048	.004417	99.00%	.09782	97.00%
25	.005	1024	.003071	100.0%	.08190	97.50%
25	.010	512	.001047	100.0%	.08555	97.70%

Based on these results, momentum greatly increased both the accuracy and the robustness of the gradient descent. It was able to tolerate a much higher learning rate and converged to 100% accuracy on the training set after only 20 epochs with the right learning parameters.

Solution 3

I implemented convolution in the source files as well as convolution + downsampling under the class name `conv2down`.

Information

This problem set took approximately 16 hours of effort.

I discussed this problem set with:

- Hao Sun on Piazza
- Ayan Chakrabarti

I also got hints from the following sources:

- Blogger's (very readable!) explanation of Xavier initialization: http://andyljones.tumblr.com/post/110998971763/an-explanation-of-xavier-initialization
- Article about convolutional layers: http://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/