CS 5335 HW8
Eric Grimaldi

V1A

Given:
W:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

X:

3	1	6	8	0
2	3	2	7	-6
-8	0	6	-5	-8
1	2	5	-2	-4
8	7	-2	-1	7

With 1 pixel of zero padding, X becomes:

0	0	0	0	0	0	0
0	3	1	6	8	0	0
0	2	3	2	7	-6	0
0	-8	0	6	-5	-8	0
0	1	2	5	-2	-4	0
0	8	7	-2	-1	7	0
0	0	0	0	0	0	0

With a stride of 2, Z=X*W is calculated to be:

1	3	1
0	2	-2
2	1	0

V1B

The convolutional layer computed above produces the same output as an average pooling layer with the same parameters would. I have a suspicion that an average pooling layer leaves division until the final step of execution, which may help avoid some floating point error, but other than that there does not seem to be any major difference.

V2B

In 5 runs of the network, we achieved accuracies of 0.7197, 0.7095, 0.7125, and 0.7180.

This value varies slightly due to sources of randomness in the network:

- Random initialization for weights
- Stochastic gradient descent
- Random selection of batch training data

V2C

Removing just the last convolutional layer from the network dropped the accuracy to 0.6750.

If you also remove the associated batch normalization, averaging pooling, and ReLU layers, we instead see a slightly improved accuracy of 0.6877.

V2D

Changing average pooling to max pooling resulted in accuracy of 0.7175. Virtually no change.

Average pooling has an effect similar to “smoothing” an image. However this can obliterate some very sharp features. Meanwhile, max pooling emphasizes the brightest pixels in a region. This is typically good if you have a bright foreground and dark background, or some variety of high contrast between regions, but may be susceptible to outliers in a very noisy image.

V3A

Some possible reasons this network achieves higher accuracy:

- This network is much larger, which allows for a more detailed approximation of the truth.
- This network is pretrained for the problem of classifying images of animals
- The final layers are fine tuned to our specific problem setting
- Also, in this case we consider 4 classes instead of 6, which simply has less opportunity for borderline cases

V3B

Softmax needs to have four outputs because we are trying to classify 4 classes of images. Softmax takes the outputs of the last FC layer (of which there are 4), and then scales them so that they all add up to 1, while still preserving the ratios between them. In other words: Softmax turns the output of the FC layer into a probability distribution for the 4 classes.

V3C

5 runs of AlexNet with the 5 layer fine tuning resulted in accuracies of 0.8357, 0.8267, 0.8355, 0.8303, and 0.8392.

5 runs of AlexNet with the 3 layer fine tuning (where 64 node FC and ReLU layers are removed) resulted in accuracies of 0.8568, 0.8735, 0.8640, 0.8625, and 0.8580.

It is worth considering that the original structure is similar to the 3 layer fine tuning case - the only difference is that the original structure outputs 1000 classes, whereas our fine tuning narrows down the entire net to only the 4 classes we care about. It seems that adding new nonlinearities to a pre-trained net during fine tuning may degrade performance. Perhaps it is worth considering the pre-trained net as having already approximated the function at hand, with the final FC layer before softmax and output as a “decision making” layer based on the properties considered by pre-trained layers.

V4

The label assigned to the webcam image flickering between captions could be the product of many phenomena.

If the net determines that 2 or more possible classes are extremely close to each other in terms of probability, small variations in image updates from the webcam can result in these nearly equally likely classes flickering past each other repeatedly.

This is likely exacerbated by the fact that the net is trained to classify a very high number of classes, which may or may not rely on relatively small differences that are hard to distinguish given the rather low resolution images used. For example, my cat was classified as a Persian cat, even though she is in fact just a regular tabby cat. Not to mention, some specific objects might not even be included in the list of classes the net was trained to find.

Also, it is worth considering that this net considers the whole image. If we give the net a busy image, it may have difficulty deciding which object deserves the dominant label. At the same time, certain very busy images with a consistent theme may actually have their own classification. For example, the net was able to correctly identify the entertainment center in my living room despite the large number of visually distinct objects.