Devansh Gupta
EECS 484 PS3

- **Evidence that your synapse matrix performs equivalent to 2-D Matlab convolution (conv2() function)**

```
output =

result of conv2d, SOH:


ans =

    1.2608    1.7690    1.3605    1.6100


output =

result of W1*(SOH(image1)):


ans =

    1.2608    1.7690    1.3605    1.6100
```

- **Evidence that derivatives match:**

Estimated bias sensitivity vectors
=

```
ans =

    0.0232
    0.0053
    0.0959
   -0.1747


ans =

    0.0116
    0.0034
    0.0289
   -0.0790
```

Computed bias sensitivity vectors
=

```
ans =

    0.0232
    0.0053
    0.0959
   -0.1747


ans =

    0.0116
    0.0034
    0.0289
   -0.0790
```

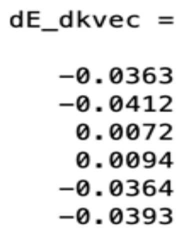
Computed dW_matrices =

```
ans =

  Columns 1 through 11

    0.0685    0.0909    0.0885    0.0335    0.1192    0.1222    0.0100    0.0589    0.0072    0.1123    0.1222
    0.0185    0.0107    0.0130    0.0073    0.0197    0.0242    0.0150    0.0048    0.0108    0.0092    0.0242
   -0.0503   -0.0970   -0.0889   -0.0284   -0.1150   -0.1091    0.0196   -0.0675    0.0142   -0.1287   -0.1091
   -0.1095   -0.0180   -0.0410   -0.0373   -0.0754   -0.1140   -0.1287    0.0077   -0.0933    0.0146   -0.1140

  Column 12

    0.1192
    0.0239
   -0.1056
   -0.1144


ans =

    0.0198    0.0131    0.0136    0.0141
    0.0038    0.0033    0.0034    0.0033
   -0.0240   -0.0141   -0.0146   -0.0157
   -0.0140   -0.0164   -0.0173   -0.0153
```

Estimated dW_matrices =

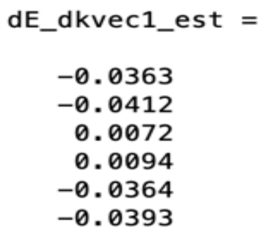
```
ans =

  Columns 1 through 11

    0.0685    0.0909    0.0885    0.0335    0.1192    0.1222    0.0100    0.0589    0.0072    0.1123    0.1222
    0.0185    0.0107    0.0130    0.0073    0.0197    0.0242    0.0150    0.0048    0.0108    0.0092    0.0242
   -0.0503   -0.0970   -0.0889   -0.0284   -0.1150   -0.1091    0.0196   -0.0675    0.0142   -0.1287   -0.1091
   -0.1095   -0.0180   -0.0410   -0.0373   -0.0754   -0.1140   -0.1287    0.0077   -0.0933    0.0146   -0.1140

  Column 12

    0.1192
    0.0239
   -0.1056
   -0.1144


ans =

    0.0198    0.0131    0.0136    0.0141
    0.0038    0.0033    0.0034    0.0033
   -0.0240   -0.0141   -0.0146   -0.0157
   -0.0140   -0.0164   -0.0173   -0.0153
```

Computed dE/dK vector
=



```
dE_dkvec =

   -0.0363
   -0.0412
    0.0072
    0.0094
   -0.0364
   -0.0393
```

Estimated dE/dK vector
=



```
dE_dkvec1_est =

   -0.0363
   -0.0412
    0.0072
    0.0094
   -0.0364
   -0.0393
```

The above tables show that the derivatives computed are correct.

Comments on training results:
As expected, training is very quick in the default problem. Within only a few hundred iterations, the error is as low as .0013. With 15000 iterations, the error converged to as low as 1.8931e-05.

Experiments:
First, I tried increasing the number of generated images from 2 to 20 while keeping the image size and feature the same. After 1000 iterations, the error was .0034. After 8000 iterations, the error was 0.0019. This makes sense because with many more training images it would take much longer to converge.

Next, I tried increasing the image size from 3x4 images to 5x6. I kept the number of generated images constant at 20 and I kept the feature the same. After 1000 iterations, the error was 0.0107. After 8000 iterations, the error was 0.0012. This again makes sense because when the number of generated images is held constant, but the image size increases, we expect it to take more iterations.
Finally, I kept the image size 5x6 and the number of generated images constant at 20, but I changed the feature to a 3x3:

After 1000 iterations, the error was 0.0028. After 8000 iterations, the error was 0.0003. With a larger kernel it seems that we are able to converge faster. This makes sense because a larger part of the image is the feature.