CS 615 - Deep Learning

Assignment 5 - Convolutional Neural Networks
Winter 2024
Alec Peterson (ap3842@drexel.edu)

Introduction

In this assignment we'll implement a simple convolutional neural network on both synthetic data and a real dataset.

Programming Language/Environment

As per usual, we are working in Python 3.x with the libraries/packages previously listed.

Allowable Libraries/Functions

In addition, you **cannot** use any libraries to do the training or evaluation for you. Using basic statistical and linear algebra function like *mean*, *std*, *cov* etc.. is fine, but using ones like *train*, *confusion*, etc.. is not. Using any ML-related functions, may result in a **zero** for the programming component. In general, use the "spirit of the assignment" (where we're implementing things from scratch) as your guide, but if you want clarification on if can use a particular function, DM the professor on slack.

Grading

Part 1 (Theory Questions)	10pts
Part 2 (Creating New Layers)	30pts
Part 3 (CNN for Binary Classification)	30pts
Part 6 (CNN for Yale Faces)	30pts
TOTAL	100pts

1 Theory

1. (2pts) Apply kernel K to data X. In other words, what is $X * K$?:

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \; K = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

ANSWER: $X * K = [(1 * 1 + 2 * 2 + 3 * 3 + ...9 * 9)] = [285]$

2. Given the feature map filter output, $F = \begin{bmatrix} 1 & 2 & 3 & 4 & 1 & 3 \\ 4 & 5 & 6 & 0 & 0 & 12 \\ 7 & 8 & 9 & 1 & 0 & 4 \\ -100 & -100 & -100 & -100 & -100 & -100 \end{bmatrix}$, what is the output from a pooling layer with width of 2 and stride of 2 if we are using:

   - (3pts) Max-Pooling?
     ANSWER:
     $$\begin{bmatrix} 5 & 6 & 12 \\ 8 & 9 & 4 \end{bmatrix}$$
   - (3pts) Mean-Pooling?
     ANSWER:
     $$\begin{bmatrix} 3 & 3.25 & 4 \\ -46.25 & -47.5 & -49 \end{bmatrix}$$

3. (2pts) Given an image X, what would the kernel K be that can reproduce the image when convoluted with it? That is, what is K such that $X * K = X$?

   For image X of height H and width W, output dimensions when convoluted with a square kernel K of dimension M x M are: $(H - M + 1)$ **x** $(W - M + 1)$

   If $H - M + 1 = H$ **and** $W - M + 1 = W$**, then** $M = 1$**, giving a kernel of** $K = [1]$

2 CNN Layers

(See submitted code.)

3 CNN for Classification of Synthetic Data

See submitted code.

a)

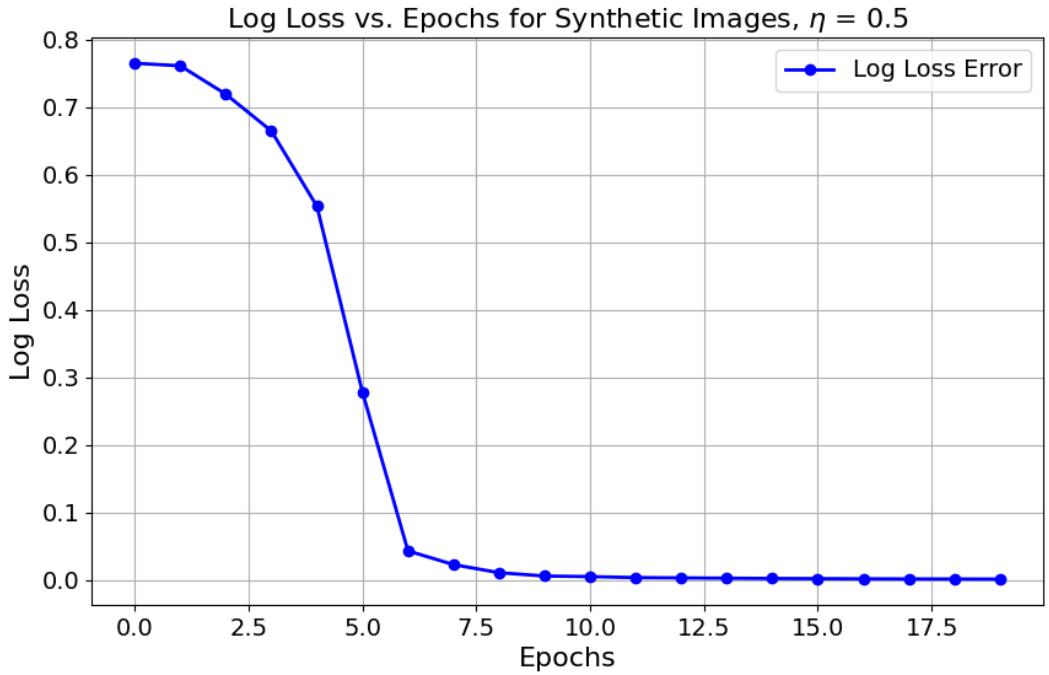
Initial kernel:

$$\begin{bmatrix} 9.76e-05 & 4.30e-04 & 2.06e-04 & 8.98e-05 & -1.53e-04 & 2.92e-04 & -1.25e-04 & 7.84e-04 & 9.27e-04 \\ -2.33e-04 & 5.83e-04 & 5.78e-05 & 1.36e-04 & 8.51e-04 & -8.58e-04 & -8.26e-04 & -9.60e-04 & 6.65e-04 \\ 5.56e-04 & 7.40e-04 & 9.57e-04 & 5.98e-04 & -7.70e-05 & 5.61e-04 & -7.63e-04 & 2.80e-04 & -7.13e-04 \\ 8.89e-04 & 4.37e-05 & -1.71e-04 & -4.71e-04 & 5.48e-04 & -8.77e-05 & 1.37e-04 & -9.62e-04 & 2.35e-04 \\ 2.24e-04 & 2.34e-04 & 8.87e-04 & 3.64e-04 & -2.81e-04 & -1.26e-04 & 3.95e-04 & -8.80e-04 & 3.34e-04 \\ 3.41e-04 & -5.79e-04 & -7.42e-04 & -3.69e-04 & -2.73e-04 & 1.40e-04 & -1.23e-04 & 9.77e-04 & -7.96e-04 \\ -5.82e-04 & -6.77e-04 & 3.06e-04 & -4.93e-04 & -6.74e-05 & -5.11e-04 & -6.82e-04 & -7.79e-04 & 3.13e-04 \\ -7.24e-04 & -6.07e-04 & -2.63e-04 & 6.42e-04 & -8.06e-04 & 6.76e-04 & -8.08e-04 & 9.53e-04 & -6.27e-05 \\ 9.54e-04 & 2.10e-04 & 4.79e-04 & -9.22e-04 & -4.34e-04 & -7.60e-04 & -4.08e-04 & -7.63e-04 & -3.64e-04 \end{bmatrix}$$

Final kernel:

$$\begin{bmatrix} -1.36e-01 & -1.57e-01 & -7.03e-02 & -6.98e-02 & -3.42e-01 & -3.37e-01 & -2.87e-01 & -2.29e-01 & -3.02e-02 \\ -9.49e-02 & -1.16e-01 & -2.91e-02 & -2.84e-02 & -2.99e-01 & -2.97e-01 & -2.46e-01 & -1.89e-01 & 1.09e-02 \\ -1.06e-01 & -1.27e-01 & -3.97e-02 & -3.95e-02 & -3.12e-01 & -3.07e-01 & -2.58e-01 & -2.00e-01 & -1.96e-03 \\ -1.72e-01 & -1.94e-01 & -1.07e-01 & -1.07e-01 & -3.77e-01 & -3.74e-01 & -3.23e-01 & -2.67e-01 & -6.74e-02 \\ -1.72e-01 & -1.94e-01 & -1.06e-01 & -1.06e-01 & -3.78e-01 & -3.74e-01 & -3.23e-01 & -2.67e-01 & -6.74e-02 \\ -2.02e-01 & -2.25e-01 & -1.38e-01 & -1.37e-01 & -4.08e-01 & -4.03e-01 & -3.53e-01 & -2.95e-01 & -9.82e-02 \\ -1.74e-01 & -1.96e-01 & -1.08e-01 & -1.08e-01 & -3.79e-01 & -3.76e-01 & -3.25e-01 & -2.69e-01 & -6.87e-02 \\ -2.27e-01 & -2.49e-01 & -1.61e-01 & -1.60e-01 & -4.33e-01 & -4.27e-01 & -3.78e-01 & -3.19e-01 & -1.21e-01 \\ -1.97e-01 & -2.20e-01 & -1.32e-01 & -1.33e-01 & -4.04e-01 & -4.00e-01 & -3.49e-01 & -2.93e-01 & -9.35e-02 \end{bmatrix}$$

b)



c) Hyperparameter and learning decisions:

- Used mini-batches of 1, i.e. propagated a single image at a time (simplified math for tensors compared to trying to process all images at once, especially for updating weights)
- Learning rate of 0.5, as indicated in figure
- Uniform initialization used for kernel in convolutional layer, Xavier initialization used for FCL
- Standard gradient descent used i.e. no Adam or other optimizer used

4 CNN For Image Classification

See submitted code.

a)

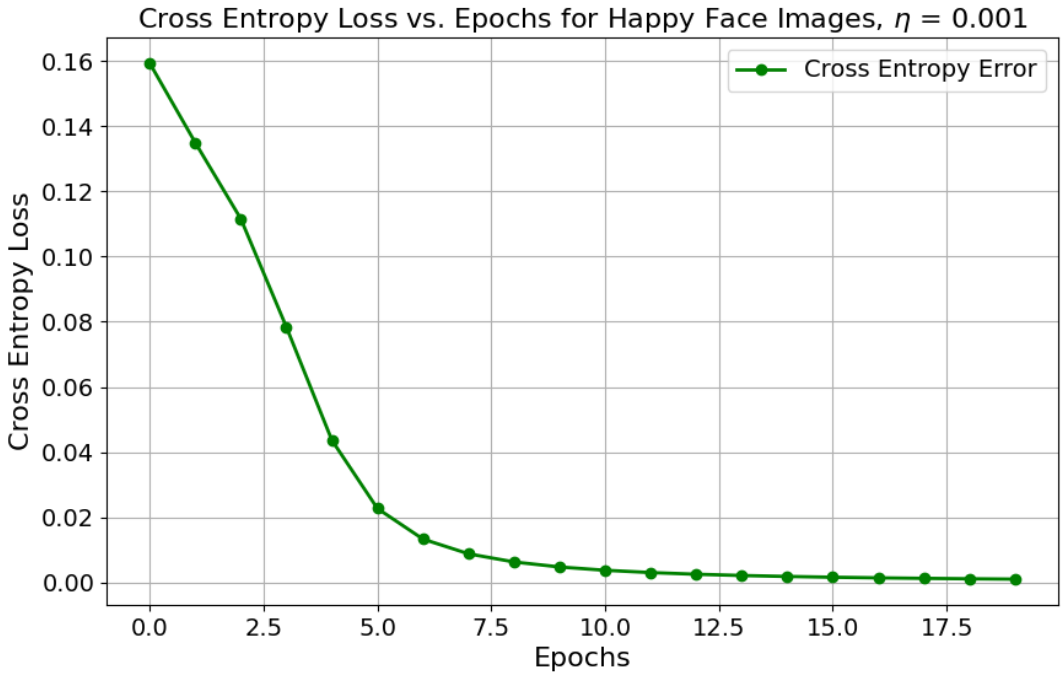
Initial kernel:

$$\begin{bmatrix}
9.76e-05 & 4.30e-04 & 2.06e-04 & 8.98e-05 & -1.53e-04 & 2.92e-04 & -1.25e-04 & 7.84e-04 & 9.27e-04 \\
-2.33e-04 & 5.83e-04 & 5.78e-05 & 1.36e-04 & 8.51e-04 & -8.58e-04 & -8.26e-04 & -9.60e-04 & 6.65e-04 \\
5.56e-04 & 7.40e-04 & 9.57e-04 & 5.98e-04 & -7.70e-05 & 5.61e-04 & -7.63e-04 & 2.80e-04 & -7.13e-04 \\
8.89e-04 & 4.37e-05 & -1.71e-04 & -4.71e-04 & 5.48e-04 & -8.77e-05 & 1.37e-04 & -9.62e-04 & 2.35e-04 \\
2.24e-04 & 2.34e-04 & 8.87e-04 & 3.64e-04 & -2.81e-04 & -1.26e-04 & 3.95e-04 & -8.80e-04 & 3.34e-04 \\
3.41e-04 & -5.79e-04 & -7.42e-04 & -3.69e-04 & -2.73e-04 & 1.40e-04 & -1.23e-04 & 9.77e-04 & -7.96e-04 \\
-5.82e-04 & -6.77e-04 & 3.06e-04 & -4.93e-04 & -6.74e-05 & -5.11e-04 & -6.82e-04 & -7.79e-04 & 3.13e-04 \\
-7.24e-04 & -6.07e-04 & -2.63e-04 & 6.42e-04 & -8.06e-04 & 6.76e-04 & -8.08e-04 & 9.53e-04 & -6.27e-05 \\
9.54e-04 & 2.10e-04 & 4.79e-04 & -9.22e-04 & -4.34e-04 & -7.60e-04 & -4.08e-04 & -7.63e-04 & -3.64e-04
\end{bmatrix}$$

Final kernel:

$$\begin{bmatrix}
-6.45e-02 & -5.52e-02 & -5.31e-02 & -4.68e-02 & -5.34e-02 & -5.73e-02 & -5.13e-02 & -4.77e-02 & -3.75e-02 \\
-6.87e-02 & -5.20e-02 & -4.98e-02 & -5.21e-02 & -5.43e-02 & -6.23e-02 & -5.84e-02 & -4.98e-02 & -3.60e-02 \\
-6.15e-02 & -4.78e-02 & -4.69e-02 & -5.01e-02 & -5.63e-02 & -5.89e-02 & -5.60e-02 & -4.79e-02 & -3.41e-02 \\
-6.17e-02 & -5.35e-02 & -5.15e-02 & -4.69e-02 & -5.00e-02 & -5.63e-02 & -5.44e-02 & -4.64e-02 & -3.21e-02 \\
-6.38e-02 & -5.85e-02 & -5.11e-02 & -4.35e-02 & -5.01e-02 & -5.81e-02 & -5.46e-02 & -4.42e-02 & -3.23e-02 \\
-6.12e-02 & -5.54e-02 & -4.81e-02 & -4.26e-02 & -4.82e-02 & -5.56e-02 & -5.23e-02 & -4.32e-02 & -3.50e-02 \\
-6.00e-02 & -5.05e-02 & -4.49e-02 & -4.35e-02 & -4.67e-02 & -5.24e-02 & -4.87e-02 & -4.38e-02 & -3.57e-02 \\
-5.99e-02 & -5.19e-02 & -4.82e-02 & -4.54e-02 & -4.70e-02 & -4.95e-02 & -4.65e-02 & -3.98e-02 & -3.73e-02 \\
-5.88e-02 & -5.22e-02 & -4.93e-02 & -4.73e-02 & -4.50e-02 & -4.77e-02 & -4.26e-02 & -4.17e-02 & -3.83e-02
\end{bmatrix}$$

b)



c) Hyperparameter and learning decisions:

- Used mini-batches of 1, i.e. propagated a single image at a time (simplified math for tensors compared to trying to process all images at once, especially for updating weights)
- Learning rate of 0.001, as indicated in figure
- Uniform initialization used for kernel in convolutional layer, Xavier initialization used for FCL
- Standard gradient descent used i.e. no Adam or other optimizer used