HWK7 [Due Tues, Nov 13]

1. Consider an input image with 2 channels:

$$\text{channel 1} = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 2 & 1 & 2 \\ 2 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix} \qquad \text{channel 2} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 1 & 3 & 2 & 1 \end{bmatrix}$$

and a convolutional module with 2 filters:

$$\text{filter 1:} \qquad \text{channel 1} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \text{channel 2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\text{filter 2:} \qquad \text{channel 1} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad \text{channel 2} = \begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix}$$

Compute with pen and paper the resulting activation maps (do NOT use any zero padding).

2. Redo problem 1 but this time use Pytorch. You need to create a convolutional module with the desired weights. Set the bias to False. Then create the input image, feed it to the convolutional module and check that you obtain the same result than the one you computed by hand in the previous problem. Use the jupyter notebook Problem2.ipynb.

   Hint 1: Use the command torch.stack() in order to build the input image and the filters.
   Hint 2: You will need the command nn.Parameter() in order to convert tensors to parameters.
   Hint 3: Don't forget that Pytorch only work with batch. So here you are going to do a batch of 1 image. Use the command view() in order to deal with this.

3. Redo problem 1 but this time express everything in term of matrix-matrix multiplication (that is, think like a GPU). To be more precise, the computation you did in problem 1 can be written as

$$WX = Y$$

   where W is the matrix containing the filters, X is the matrix containing the patches of the input image, and Y is the matrix containing the activation maps. Write down these three matrices, do the matrix-matrix multiplication with matlab or pytorch, and check that everything work properly. (So be clear: you need to explicitly write down the matrices W, X and Y.)

4. Let consider an image with 2 channels:

$$\text{channel 1} = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 2 & 1 & 2 \\ 2 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix} \qquad \text{channel 2} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 1 & 3 & 2 & 1 \end{bmatrix}$$

and a convolutional module with one filter:

$$\text{filter 1:} \qquad \text{channel 1} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad \text{channel 2} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Compute with pen and paper the resulting activation map. Use zero padding so that the activation map and the input image have same dimension.

5. Consider the following one layer convnet with a cross entropy criterion:

$$\text{convolution with 3-by-3 filters} \longrightarrow \text{“global” max pooling} \longrightarrow \text{Cross Entropy Criterion}$$

This network processes gray scale images (only one channel). There are two categories: the first category consists of images containing a "+", and the second category consist of images containing a "x".

- Category 0: "+"
- Category 1: "x"

Suppose that after training, the filters in the convolutional layer are:

$$\text{filter 0: } \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \text{filter 1: } \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

In other words, one filter is here to detect "+", and one to detect "x". Consider the following labelled images:

$$\text{image 0} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{label } 0 = 0 \qquad \text{(this image is a “+”)}$$

$$\text{image 1} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{label } 1 = 1 \qquad \text{(this image is a “x”)}$$

Note that these are 5-by-10 gray scale images.

(a) Compute with pen and paper the activation maps that you obtain when processing these images through the convolutional layer (do not use any padding). It is highly recommended that you check your answer with pytorch.

(b) Compute the loss associated with each image. The max pooling layer operate on the entire activation map (it is a "global pooling"). You do not need pytorch for this. The softmax is easy to compute with a calculator.

6. Make a convnet and try to obtain the best possible result on the SVHN dataset. Do this on the jupyter notebook Problem6.ipynb

7. Make a convnet and try to obtain the best possible result on the fashion-MNIST dataset. Do this on the jupyter notebook Problem7.ipynb