Homework 3

1. **(2 points)** *Exercises in convolution.* Suppose that the input is a 1D array (or signal; call it x). Find (i.e., design by hand) a convolutional filter that produces the following output (call it y). All you need to do is to specify the filter weights.

   a. (Approximate) derivatives:
   $$y[n] = x[n+1] - x[n-1].$$

   b. (Approximate) second derivatives:
   $$y[n] = x[n+2] - 2x[n] + x[n-2].$$

   c. (Approximate) integrals:
   $$y[n] = x[n-\Delta] + x[n-\Delta+1] + \ldots + x[n+\Delta-1] + x[n+\Delta].$$

   d. cross-correlations:
   $$y[n] = \sum_i x[i]x[i+n].$$

   Solution

   a. $w[1] = 1, w[0] = 0, w[-1] = -1$. OK to flip $w[1]$ and $w[-1]$.

   b. $w[2] = 1, w[1] = 0, w[0] = -2, w[-1] = 0, w[-2] = 1$.

   c. $w[i] = 1$ for $-\Delta \leq i \leq \Delta$.

   d. $w[i] = x[i]$.

2. **(2 points)** 1×1 *convolution.* In the definition of a convolutional layer, if we have I input channels, J output channels, and the filter size is chosen to be $\Delta = 0$, show that the operation is equivalent to applying a regular dense layer in the *channel* domain. What is the number of trainable parameters in this layer?

   Solution

   Assume the input is 1D; the answer remains the same for 2D as well. Assume also no biases. As defined in class, for multichannel convolution, we have a convolution filter w_{ij} for each pair i, j, and the output is defined as:
   $$z_i = \phi\left(\sum_j w_{ij} \star x_j\right)$$
   where x_j is the feature map at the j^{th} channel. By definition of the convolution operation,
   $$(w_{ij} \star x_j)[l] = \sum_{m=-\Delta}^{\Delta} w_{ij}[m]x_j[l-m]$$

but since $\Delta = 0$ this summation simply reduces to a scalar multiplication with scalar $w_{ij}[0]$. Let's just call this w_{ij}. Therefore, the output simplifies to:

$$z_i[l] = \phi\left(\sum_j w_{ij} x_j[l]\right)$$

which is the same as taking each spatial location and applying a dense weight matrix W of size $I \times J$ in the channel domain.

The number of trainable parameters, therefore, is $I \times J$.

(Some people used biases, which changes the answer to $(I + 1) \times J$. That's also ok).

3. **(2 points)** *The IoU metric*. Recall the definition of the IoU metric (or the Jaccard similarity index) for comparing bounding boxes.

   a. Using elementary properties of sets, prove that the IoU metric between any two pair of bounding boxes is always a non-negative real number in $[0, 1]$.

   b. If we represent each bounding box as a function of the top-left and bottom-right coordinates (assume all coordinates are real numbers) then argue that the IoU metric is *non-differentiable* and hence cannot be directly optimized by gradient descent.

Solution

a. The definition of IOU for any two bounding boxes A and B is given by:

$$IOU(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Since the right hand side is non-negative, this number has to be bigger than (or equal to) 0. Moreover, $A \cap B \subseteq A \cup B$, and hence the numerator has to be no bigger than the denominator. Therefore the IOU is bounded between 0 and 1 (inclusive).

b. Here is a simple counter-example. Let's take two identical size boxes A and B (say, square), both aligned at the same horizontal level. Fix B and then imagine "sliding" A from left to right. As A moves, the IOU will start from zero (no overlap), increase (until there is perfect overlap), and then decrease (until there is no overlap again). So, if we plot IOU as a function of horizontal displacement, we should get a curve like this:

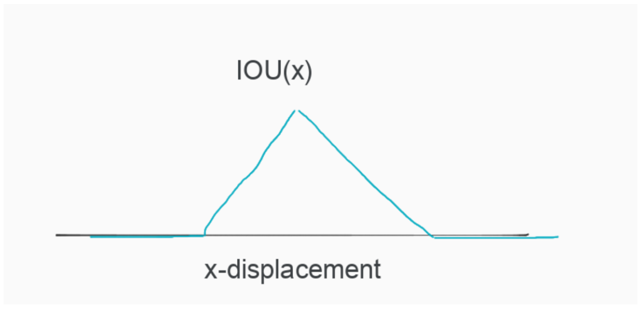

Figure 1: IOU as a function of displacement

which has 3 kinks and hence is non-differentiable.

4. **(4 points)** In this programming exercise, we will explore the performance of three different object detection networks. We will be using Detectron2, Facebook AI's object detector library; here is the repository. It will be helpful to go through the excellent tutorial here.

   a. Download the following test image (a picture of pedestrians in Central Park). We will run two different detectors on this image.

   b. First, consider the COCO Keypoint Person Detector model with a ResNet50-FPN base network, which is trained to detect human silhouettes. This can be found in the Detectron2 Model Zoo in the "COCO Keypoint" table. Use this model to detect as many silhouttes of people in the test image as you can. You may have to play around with the thresholds to optimize performance.

   c. Second, repeat the above procedure, but with the Mask R-CNN model with ResNet50-FPN backbone, available in the Model Zoo in the "COCO Instance Segmentation" table. This time, you should be able to detect both people as well as other objects in the scene. Comment on your findings.

   d. It appears that the balloons in the test image are not being properly detected in either model. This is because the COCO dataset used to train the above models does not contain balloons! Following the tutorial code above, start with the above pre-trained Mask R-CNN model and train a balloon detector using the (fine-tuning) balloon image dataset provided here. Test it on the original test image and show that you are now able to identify all the balloons.

   Solution

   A solution notebook has been posted on NYUClasses.