CS-GY 9223G / Fall 2020

Homework 3

---

1. **(2 points)** *Convolutional layers*. Suppose that a convolutional layer of a neural network has an input 3D tensor $X[i, j, k]$ and computes an output as follows:

$$Z[i, j, m] = \sum_{k_1} \sum_{k_2} \sum_{n} W[k_1, k_2, n, m] X[i + k_1, j + k_2, n] + b[m]$$
$$Y[i, j, m] = \text{ReLU}(Z[i, j, m])$$

for some convolutional kernel W and bias b. Suppose X and W have shapes $(48, 64, 10)$ and $(3, 3, 10, 20)$ respectively. Assume no padding (i.e., neither zero- or circular-padding is applied), and stride 1.

   a. What are the shapes of Z and Y?

   b. What are the number of input and output channels?

   c. What are the total number of trainable parameters in this layer?

   d. How many arithmetic multiplications are needed to perform the convolution operation for this layer?

   Solution

   a. Both Z and Y are tensors of size $(48 - 3 + 1, 64 - 3 + 1, 20) = (46, 62, 20)$.

   b. The number of input channels is 10, and number of output channels is 20.

   c. The total number of trainable parameters = $3 \cdot 3 \cdot 10 \cdot 20 + 20 = 1820$.

   d. Each entry of Z requires summing over the multiplication of the input with a 3×3 kernel and over 10 channels, making for approximately $3 \times 3 \times 10 = 90$ multiplications/additions. Multiplying this with the size of Z, we get $46 \times 62 \times 20 \times 90 \approx 5 \times 10^6$ multiplications/additions. (One could speed this up via FFTs for larger inputs but for small sizes, direct multiplication is reasonable).

2. **(2 points)** *Overfitting in neural nets*. This is a three-part question.

   a. Consider neural network learning, and sketch *typical* curves of loss versus iteration count for the training and validation sets using stochastic gradient descent. (Assume that overfitting to the training set happens at some point.)

   b. We discussed data augmentation as a possible approach for combat overfitting in neural net training. List any two other strategies.

   c. For image classification, explain why using data augmentation may work for cat-vs-dog classifiers, but is **not** the right thing to do for distinguishing between English handwritten characters (a,b,c,d,...).

   Solution

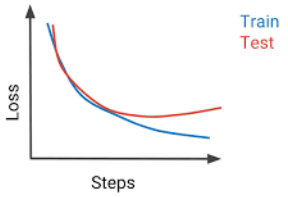

Figure 1: Typical loss curves

a. The following figure is typical:

b. Any combination of: early stopping; dropout; adding a regularization penalty; weight decay; bottleneck layers; transfer learning; etc.

c. It depends on how you do the data augmentation. Given a training image sample, a common approach is to rotate/flip/crop the sample to create several (artificial) new samples, but *keep the class label the same*. This is fine for cats-vs-dogs, but if we rotate/flip/crop handwritten characters, they might accidentally resemble images from other classes. For example, flipping a b gives a d, rotating a d by 180 degrees gives a p, etc. This may lead to erroneous classification results.

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:

   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;

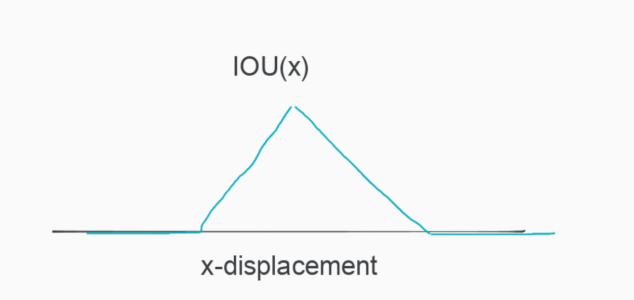

Figure 2: IOU as a function of displacement

here is the repository. It will be helpful to go through the excellent tutorial (with example code) 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 backbone, 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.