Problem Set 2
Deep Learning E1394

Alvaro Guijarro May, Santiago Sordo

Out on Oct 4, 2022
Due on Oct 18, 2022, at 23:59

Submit your written answers as a pdf typed in LaTeXtogether with your code. Submit one answer per group (as assigned on Moodle) and include names of all group members in the document. Round answers to two decimal places as needed. Include references to any external sources you have consulted (points are deducted if those were used but not cited). See "Submission" at the bottom of the problem set for more details on how to submit using Github classroom.

1 Convolutional neural networks

1.1 Kernels (7 pts)

(a) Design a 3×3 kernel that leaves the input image unchanged. Describe also how you may need to modify the input image before applying the kernel so that the output stays the same.

Answer: A 3 x 3 kernel that would leave the input image unchanged would be an identity kernel, which would be the following:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

To ensure that the input image remains unchanged when convolved with a 3x3 identity kernel, it's essential that the input image has the same dimensions as the kernel, i.e., a 3x3 size.

If the input image is **larger**, the convolution operation will produce a smaller output that represents a smoothed or averaged version of the original image. If the input image is **smaller**, some portions of the input will not be affected by the convolution operation. Therefore, the dimensions of the input image should match the dimensions of the 3x3 identity kernel to maintain the original image.

(b) How does the following kernel modify an input image: $K = \frac{1}{16}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$?

This kernel performs a simple averaging operation. When applied to an input image through convolution, it calculates the average of the pixel values in a 4x4 neighborhood and assigns this average value to the central pixel. This process results in a blurred version of the input image since it reduces high-frequency components.

(c) Design a custom kernel of any size and describe how it transforms an input image or what it highlights in an image.

The costume edge detection kernel: $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 10 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ is designed to enhance edges by placing strong emphasis on the central pixel and its immediate neighbors, while suppressing contributions from surrounding pixels. This results in a pronounced enhancement of well-defined edges in the image. The kernel's behavior remains consistent across different image sizes, effectively enhancing edges in proportion to the image's dimensions. The primary impact is the pronounced enhancement of edges, making them more prominent in larger images.

1.2 Convolution and pooling (15 pts)

1.2.1 Convolutional layer (10/15 pts)

Given an input image

$$X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0.5 & 1 & 0.5 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \tag{1}$$

and a kernel $K = \begin{bmatrix} 0 & 2 & 0 \\ 2 & 1 & 1 \\ 0 & 2 & 0 \end{bmatrix}$, compute the feature map (output after the convolution and applying a sigmoid activation function). Modify the input such that the feature map has the same dimension as the original input image. Show the intermediate result after the convolution and before applying the activation function as well.

Answer: Knowing that the original input image is $X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0.5 & 1 & 0.5 \\ 0 & 1 & 0 & 1 \end{bmatrix}$, since we are aplying a 3x3 kernel to a 4x4 image, the result would be a 2x2 feature map, in order to get a 4x4 feature map, we have to make our input a 6x6 dimension image, we will do this by adding 0's in the "outer ring" of the input image.

$$X' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 2 & 0 \\ 2 & 1 & 1 \\ 0 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 0 & 2 \\ 3 & 2 & 5 & 2 \\ 1.5 & 7.5 & 2.5 & 6.5 \\ 3 & 2 & 5 & 2 \end{bmatrix}$$

This is the intermediate result after the convolution, now we will apply the sigmoid activation function, which is the following:

$$\sigma(z) = \frac{1}{\left(1 + e^{(-z)}\right)} \tag{2}$$

After applying the sigmoid activation funcion, our feature map will be the following:

$$\begin{bmatrix} 0.5 & 0.8808 & 0.5 & 0.8808 \\ 0.9526 & 0.8808 & 0.9933 & 0.8808 \\ 0.8176 & 0.9995 & 0.9241 & 0.9985 \\ 0.9526 & 0.8808 & 0.9933 & 0.8808 \end{bmatrix} \tag{3}$$

1.2.2 Pooling layer (2/15 pts)

Apply 2×2 max pooling to the feature map (no overlap/stride 2).

From the feature map, we will proceed to apply a max pooling with no overlap / stride 2, and for visualization purposes, the chosen values will be in bold:

$$\begin{bmatrix} 0.5 & 0.8808 & 0.5 & 0.8808 \\ \mathbf{0.9526} & 0.8808 & \mathbf{0.9933} & 0.8808 \\ 0.8176 & \mathbf{0.9995} & 0.9241 & \mathbf{0.9985} \\ 0.9526 & 0.8808 & 0.9933 & 0.8808 \end{bmatrix} = \begin{bmatrix} \mathbf{0.9526} & \mathbf{0.9933} \\ \mathbf{0.9995} & \mathbf{0.9985} \end{bmatrix} \tag{4}$$

1.2.3 Discussion (3/15 pts)

Describe any problem(s) that you may see in training the model if a feature map like this one was typical for your CNN.

Answer: In the context of CNNs, one potential issue with feature maps obtained after max pooling is the substantial loss of spatial information. Max pooling, while effective for reducing computational complexity and controlling overfitting, downsizes the spatial resolution of the data. This type of reduction can hurt a model's ability to determine complex patterns and spatial relationships, specially in tasks where precise localization or fine details are necessary.

1.3 Pooling transformed into convolution (8 pts)

How do you represent 2×2 average pooling as a convolution? You may show this by the example of 4×4 input data. Provide the kernel size, kernel values, stride, etc. as appropriate.

Answer: In order to represent a 2x2 average pooling as a convolution, we have to set a filter (kernel) that have equal weights of (1/4) for each element, in the case of an 4x4 input image.

$$\text{Input Data (4x4)} = \begin{bmatrix} 2 & 4 & 6 & 8 \\ 10 & 12 & 14 & 16 \\ 18 & 20 & 22 & 24 \\ 26 & 28 & 30 & 32 \end{bmatrix}$$

$$\text{Kernel (2x2)} = \begin{bmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix}$$

Striding = 2

The result of a 2x2 average pooling as a convolution would look like this:

$$\begin{bmatrix} (0.25*2)+(0.25*4)+(0.25*10)+(0.25*12) & (0.25*6)+(0.25*8)+(0.25*14)+(0.25*16) \\ (0.25*18)+(0.25*20)+(0.25*26)+(0.25*28) & (0.25*22)+(0.25*24)+(0.25*30)+(0.25*32) \end{bmatrix}$$

$$= \begin{bmatrix} 7 & 11 \\ 23 & 27 \end{bmatrix} \tag{5}$$

1.4 Dimensions of CNN layers (10 pts)

For the CNN shown in Figure 1, write down the dimensions of each each layer and how you computed them. The input image is first increased to $3 \times 227 \times 227$ with padding.

Tip: A pooling layer with 'stride 2' means that after each pooling operation the next pooling area is 2 pixels apart. A 2×2 pooling operation with stride 2 would result in our example from class with no overlap. A 3×3 pooling operation with stride 2 has overlap. 'Pad 2' refers to padding with 2 pixels on each side.

1. Input Image $= 3 * 224 * 224$
2. 11 * 11 Conv (96), stride 4 $= 96 * 54 * 54$
   - $output\_height = (224 - 11)/4 + 1 = 54$

  - $output\_width = (224 - 11)/4 + 1 = 54$

3. 3 * 3 MaxPool, stride 2 $= 96 * 26 * 26$
    - $output\_height = (54 - 3)/2 + 1 = 26$
    - $output\_width = (54 - 3)/2 + 1 = 26$
4. 5 * 5 Conv (256), pad 2 $= 384 * 12 * 12$
    - $output\_height = (26 + 2 * 2 - 5) + 1 = 26$
    - $output\_width = (26 + 2 * 2 - 5) + 1 = 26$
5. 3 * 3 MaxPool, stride 2 $= 256 * 12 * 12$
    - $output\_height = (26 - 3)/2 + 1 = 12$
    - $output\_width = (26 - 3)/2 + 1 = 12$
6. 3 * 3 Conv (384), pad 1 $= 384 * 12 * 12$
    - $output\_height = (12 + 2 * 1 - 3) + 1 = 12$
    - $output\_width = (12 + 2 * 1 - 3) + 1 = 12$
7. 3 * 3 Conv (384), pad 1 $= 384 * 12 * 12$
    - $output\_height = (12 + 2 * 1 - 3) + 1 = 12$
    - $output\_width = (12 + 2 * 1 - 3) + 1 = 12$
8. 3 * 3 Conv (256), pad 1 $= 256 * 12 * 12$
    - $output\_height = (12 + 2 * 1 - 3) + 1 = 12$
    - $output\_width = (12 + 2 * 1 - 3) + 1 = 12$
9. 3 * 3 MaxPool, stride 2 $= 256 * 5 * 5$
    - $output\_height = (12 - 3)/2 + 1 = 5$
    - $output\_width = (12 - 3)/2 + 1 = 5$
10. FC (4096)
11. FC (4096)
12. FC (1000)

`../img/PS2_network.png`

Figure 1: Convolutional neural network in simplified form. Note that the input image is first increased to $3 \times 227 \times 227$ with padding.

2 Monitoring power plant operations with CNNs

You will create an image classifier using PyTorch to estimate power plant operation. Specifically, you will learn how to apply CNNs to binary cassify LAWS' Sentinel 2 L2A satellite pictures (0: power plant is off, 1: power plant is on). All tasks are marked in their specific section and described in more detail in the Jupyter notebook. For some tasks, you may need to elaborate on your implementation. You can add the answers to these questions directly below the respective implementation task that you then upload to your GitHub repository. Remember to always show the code output in the final upload and use the "test your code" sections to test your implementation.

1. Implement your Convolutional Neural Network (20 pt)
2. Implement a hyperparameter tuner and fine-tune your network (20 pt)
3. Load and train a pre-trained model (10 pt)
4. Compare and discuss your results (10 pt)

Submission

The submission of the whole problem set is done via GitHub classroom. You have to register for the assignment with your GitHub account at `https://classroom.github.com/a/Ej3Bnsnb`. Please make sure that your GitHub account profile includes your real name. When starting the assignment, please create or join a team according to the teams assigned in Moodle (use the exact team name from Moodle). Please upload / push all solutions to the GitHub repository which was created for your team. Please upload your solutions for the theoretical part (1) as a PDF to GitHub. If you upload multiple PDF files, please indicate in the file name to which subtask they are corresponding (we much prefer one single pdf). For the practical part (2), just push your code changes to the existing files. Anybody in your team can push as often as they want. Once the deadline has passed, you are not able to push to your repository anymore and all changes on your **main** branch will be considered for grading. See the README.md in the repository for more details on how to push your changes to GitHub.