Problem Set 2
Deep Learning E1394

Gresa Smolica (224725)
Jiayu Yang (213444)

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.

Solution: The kernel that leaves the input image unchanged, also known as identity kernel is:

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

To ensure the output image remains the same size as the input, we will need to pad the input image with a one-pixel border. Zero-padding would be sufficient.

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

Solution: The given kernel is an averaging kernel. When convolved with an image, it will replace each pixel value with the average of itself and its 15 neighbors, resulting in a blurring effect. As it reduces high frequency details in the image, it results in a smoother appearance of the image.

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

Solution: A kernel for detecting edges is the Sobel operator:

$$K = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

This kernel will detect the vertical edges in an image. It is widely used in the Sobel algorithm to calculate estimates of the derivatives in the vertical direction of an image.

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.

Solution: To ensure the feature map has the same dimension as the original input image, we need to use 'same' padding - padding the input image with a one-pixel border, eg. zero-padding is applied below:

Padded X:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0.5 & 1 & 0.5 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

After convolution:

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 2.5 & 3.5 & 3 \\ 2 & 4 & 3 & 5 \\ 0 & 3 & 2 & 3 \end{bmatrix}$$

We apply the sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$ to each component. The feature map after the sigmoid activation function results in:

$$\begin{bmatrix} 0.5 & 0.731 & 0.881 & 0.953 \\ 0.881 & 0.924 & 0.971 & 0.953 \\ 0.881 & 0.982 & 0.953 & 0.993 \\ 0.5 & 0.953 & 0.881 & 0.953 \end{bmatrix}$$

1.2.2 Pooling layer (2/15 pts)

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

Solution: After applying a 2×2 max pooling with stride 2, our feature map looks as below:

$$\begin{bmatrix} 0.924 & 0.971 \\ 0.982 & 0.993 \end{bmatrix}$$

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.

Solution:

- Loss of Spatial Information: The pooling operation reduces the spatial dimensions and might disregard details and patterns in the data.
- Vanishing/Exploding Gradients: High or low activations can lead to issues during training and can hinder the whole process of weight updates.
- Overfitting: The network might be overfitting to the training data. Too much pooling makes the CNN lose the ability to learn from the limited spatial context available and resultantly not generalize well.

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.

Solution: To represent 2×2 average pooling as a convolution, we use the following kernel:

$$K = \begin{bmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix}$$

with a stride $s = 2$.
Given a 4×4 input data matrix X:

$$X = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

Convolving X with K using a stride of 2, we get:

$$\text{Pooled} = \begin{bmatrix} \frac{a+b+e+f}{4} & \frac{c+d+g+h}{4} \\ \frac{i+j+m+n}{4} & \frac{k+l+o+p}{4} \end{bmatrix}$$

This is equivalent to 2×2 average pooling on X.

1.4 Dimensions of CNN layers (10 pts)

For the CNN shown in Figure **??**, write down the dimensions of 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.

`../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.

Solution:
The formula to compute the layers:

$$\text{Output Width} = \frac{(\text{Input Width} - \text{Filter Width})}{\text{Stride}} + 1$$

$$\text{Output Height} = \frac{(\text{Input Height} - \text{Filter Height})}{\text{Stride}} + 1$$

1. Input Image:

- Dimensions: $3 \times 227 \times 227$

2. 11×11 Convolution (96 filters, stride 4):

- Output dimensions: $96 \times 55 \times 55$

$$\text{Output Width} = \frac{227 - 11}{4} + 1 = 55$$

$$\text{Output Height} = \frac{227 - 11}{4} + 1 = 55$$

3. 3×3 Max Pooling (stride 2):

- Output dimensions: $96 \times 27 \times 27$

$$\text{Output Width} = \frac{55}{2} = 27$$

$$\text{Output Height} = \frac{55}{2} = 27$$

4. 5×5 Convolution (256 filters, padding 2):

- Output dimensions: $256 \times 27 \times 27$ (No change in spatial dimensions due to padding)

$$\text{Output Width} = \frac{27 + 2 \times 2 - 5}{1} + 1 = 27$$

$$\text{Output Height} = \frac{27 + 2 \times 2 - 5}{1} + 1 = 27$$

5. 3×3 Max Pooling (stride 2):

- Output dimensions: $256 \times 13 \times 13$

$$\text{Output Width} = \frac{27}{2} = 13$$

$$\text{Output Height} = \frac{27}{2} = 13$$

6. 3×3 Convolution (384 filters, padding 1):

- Output dimensions: $384 \times 13 \times 13$ (No change in spatial dimensions due to padding)

$$\text{Output Width} = \frac{13 + 2 \times 1 - 3}{1} + 1 = 13$$

$$\text{Output Height} = \frac{13 + 2 \times 1 - 3}{1} + 1 = 13$$

7. 3×3 Convolution (384 filters, padding 1):

- Output dimensions: $384 \times 13 \times 13$ (No change in spatial dimensions due to padding)

$$\text{Output Width} = \frac{13 + 2 \times 1 - 3}{1} + 1 = 13$$

$$\text{Output Height} = \frac{13 + 2 \times 1 - 3}{1} + 1 = 13$$

8. 3×3 Convolution (256 filters, padding 1):

- Output dimensions: $256 \times 13 \times 13$ (No change in spatial dimensions due to padding)

$$\text{Output Width} = \frac{13 + 2 \times 1 - 3}{1} + 1 = 13$$

$$\text{Output Height} = \frac{13 + 2 \times 1 - 3}{1} + 1 = 13$$

9. 3×3 Max Pooling (stride 2):

- Output dimensions: $256 \times 6 \times 6$

$$\text{Output Width} = \frac{13}{2} = 6$$

$$\text{Output Height} = \frac{13}{2} = 6$$

10. Fully Connected (FC) layer with 4096 neurons.

- No spatial dimensions, only depth (neurons) is relevant.
- Output dimensions: 4096

11. Fully Connected (FC) layer with 4096 neurons.

- No spatial dimensions, only depth (neurons) is relevant.
- Output dimensions: 4096

12. Fully Connected (FC) layer with 1000 neurons.

- No spatial dimensions, only depth (neurons) is relevant.
- Output dimensions: 1000

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.