Problem Set 2
Deep Learning E1394

Giulia Maria Petrilli (236888)
Benjamin Adoba Ayida (235194)
Xiaohan Wu (237867)

Out on Oct 3, 2025
Due on Oct 17, 2025, at 23:59

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:

Since convolution is a weighted sum, each output pixel must equal the same input pixel to keep the input image unchanged. We can achieve this by padding with 0's and setting setting the center weight = 1.
This implies the kernel must be the **identity kernel**.

$$I = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

When this kernel is centered over a pixel, it multiplies the center pixel's value by 1 and all its surrounding neighbors by 0, effectively isolating and returning only the original center pixel's value.

For the output image to be exactly the same, two things are necessary:

- Padding: apply 1 pixel of zero padding around the input image. This prevents the output from shrinking, because it ensures that the 3×3 kernel can be centered over every original pixel, including those on the edges.

- Stride = 1. The stride determines how far the kernel moves across the image at each step, and stride of 1 means the filter slides one pixel at a time, covering every position, while larger strides skip pixels and reduce the output side. A stride of 1 means that the kernel processes every pixel, and prevents skipping information and the shrinking of the output.

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

Answer:

Given a 4×4 matrix of ones, this is implies sum of all weights = 16. Therefore each output pixel is given as:

$$y[i,j] = \sum_{u-0}^{3}\sum_{v-0}^{3}\frac{1}{16}x[i-u, j-v]$$

$$\frac{1}{16} \times 16 = 1$$

This is a 4×4 mean filter. The kernel smooths/blurs the input image by replacing each pixel's value with the average of its 4×4 neighborhood. Since the normalized sum of all kernel elements equals 1 ($\frac{1}{16} \times 16 = 1$), and no zero padding is applied, the overall image brightness is preserved. The averaging operation reduces sharp intensity changes between neighboring pixels, and attenuates high-frequency components such as noise and fine edges, which results in a blurred 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: We designed the following 3×3 kernel:

$$K = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

This kernel highlights areas in the image where there are dark spots surrounded by a light area, or the other way around. This kernel finds it easier to detect strong intensity difference along the lines delineating a segment, rather than low contrast areas, since the gap between 1 and -8 is quite stark. The center value -8 represents the pixel being processed. The surrounding 1s represent neighboring pixels that are subtracted from the central pixel's intensity. When the central pixel is much darker than its surroundings, the weighted sum, so the output of the convolution, will be negative with large magnitude. When the central pixel is much brighter than its surroundings, the result will be positive with large magnitude. When all pixels have similar intensity, so in flat regions, the output will be close to zero.

1.2 Kernels applied to an image (10 pts)

You discover that your convolutional neural network kernel has learned the following weights (the operation is implemented as a cross-correlation)

$$K = \begin{bmatrix} -0.89 & -0.92 & -0.9 \\ 0.01 & 0.02 & 0.005 \\ 0.9 & 0.92 & 0.89 \end{bmatrix}. \tag{1}$$

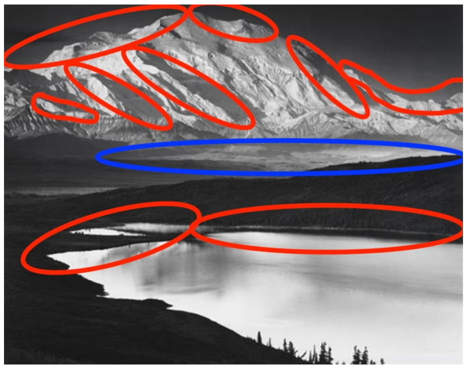
(a) Describe what pattern the kernel is filtering for in one or two sentences.

Answer:

The kernel is filtering for a dark-to-bright intensity transition along the vertical direction, meaning it detects horizontal edges. It strongly penalizes dark pixels (low values) at the top and heavily weights bright pixels (high values) at the bottom, indicating it activates when it encounters a pattern that is dark on top and bright on the bottom.

(b) In the image in Figure 1, precisely circle all of the larger area(s) that the output feature map of this convolutional layer activates with high positive values on, and explain in one additional sentence why you circled these area(s).

Answer:



Explanation: The convolution operation yields a high positive value precisely when the negative weights of the kernel's top row align with low pixel intensity (dark) areas, and the positive weights of the bottom row align with high pixel intensity (bright) areas. The strongest activation occurs at the shoreline because it represents the sharpest and most sustained transition from the dark shadow cast by the foreground ridge onto the bright, reflective water. The blue circle represents an area where activation still verifies, but to a lower extent, since there is less contrast between black and white areas.

1.3 Convolution and pooling (15 pts)

1.3.1 Convolutional layer (10/15 pts)

Given an input image

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

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 as the final feature map.

Solution: To ensure that the output image after convolution has the same size as the input image, we apply zero padding such that

$$P = \frac{K-1}{2},$$

assuming the stride $S = 1$ (where the input is a 4x4 matrix and the kernel is 3x3, the stride can only be one, as a larger one would result in no positions where the kernel can be applied). The general relationship between input and output sizes is:

$$F = \frac{N - K + 2P}{S} + 1$$

where F is output size, N is input size. P is padding, K is kernel size, and S is stride. Substituting the given values:

$$4 = \frac{4 - 3 + 2P}{1} + 1$$

$$4 = 2P + 2$$

$$P = 1$$

Thus, one layer of zero-padding on each side of the 4×4 input image increases its dimensions to the 6x6 matrix:

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

Since this is not a cross correlation, all the multiplications between the different masks in the input matrix and the kernel will be performed with its flipped version:

$$K_{\text{flipped}} = \begin{bmatrix} 0 & 2 & 0 \\ 1 & 1 & 2 \\ 0 & 2 & 0 \end{bmatrix}$$

After multiplying the 3×3 patch of X_P elementwise with kernel Kflipped and summing the results, we have

$$Z = \begin{bmatrix} 0 & 2 & 0 & 2 \\ 4 & 2 & 5 & 2 \\ 2 & 7.5 & 2.5 & 5.5 \\ 4 & 2 & 5 & 2 \end{bmatrix},$$

Let‘s now apply the sigmoid activation to get the feature map A rounded to 2 dp:

$$A = \sigma(Z) = \frac{1}{1 + e^{-Z}}$$

The following line is an example of how the application of the sigmoid function happens for one unit:

$$\sigma(0) = \frac{1}{1 + e^{-0}} = \frac{1}{1 + 1} = 0.5$$

This is the final feature map. Once having understood the procedure (as shows in the step above), we used an online sigmoid function calculator to compute the sigmoid of every other entry in the matrix Z:

$$A = \begin{bmatrix} 0.50 & 0.88 & 0.50 & 0.88 \\ 0.98 & 0.88 & 0.99 & 0.88 \\ 0.88 & 1.00 & 0.92 & 1.00 \\ 0.98 & 0.88 & 0.99 & 0.88 \end{bmatrix}$$

1.3.2 Pooling layer (2/15 pts)

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

Solution: The pooling window slides 2 pixels at a time, taking the maximum in each 2×2 block. We divide A into four non-overlapping 2×2 regions:

$$\text{Region 1} = \begin{bmatrix} 0.50 & 0.88 \\ 0.98 & 0.88 \end{bmatrix}$$

$$\text{Region 2} = \begin{bmatrix} 0.50 & 0.88 \\ 0.99 & 0.88 \end{bmatrix}$$

$$\text{Region 3} = \begin{bmatrix} 0.88 & 1.00 \\ 0.98 & 0.88 \end{bmatrix}$$

$$\text{Region 4} = \begin{bmatrix} 0.92 & 1.00 \\ 0.99 & 0.88 \end{bmatrix}$$

This is a list of the max value from each region:

$$\max(R1) = 0.99, \max(R2) = 0.99, \max(R3) = 1.00, \max(R4) = 1.00$$

Arrange into output feature map

$$A_{pooled} = \begin{bmatrix} 0.99 & 0.99 \\ 1.00 & 1.00 \end{bmatrix}$$

1.3.3 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
If a feature map like this were typical for the CNN, where almost all activations are very high values close to 1, this would mean that the network's neurons are saturated after the sigmoid function.

When the sigmoid outputs is between 1 or 0, and when there are many layers, the gradient of the activation function becomes very small, which is the vanishing gradient problem. During backpropagation, these small gradients propagate backward through the layers, leading to extremely slow or stalled learning, especially in earlier layers. As a result, the CNN would struggle to update its weights effectively, converge slowly, or get stuck without learning meaningful representations.

1.4 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:

Following the general rule: For a $k \times k$ average pooling, uses $k \times k$ kernel for all $\frac{1}{k^2}$ with k strides and 0 padding.

A 2×2 average pooling operation can be implemented as a convolution with the following properties: kernel size: 2×2, stride: 2, padding: 0.

To make convolution exactly equal to average pooling the sum of weights (elements) must be 1. Therefore, for a 2×2 average pooling, the convolutional kernel (filter) used to perform average pooling is:

$$K = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

This kernel computes the mean of each 2×2 region of the input, just as average pooling does. For example, consider a 4×4 input matrix (ChatGPT generated):

$$X = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Applying the convolution with kernel K and stride $S = 2$ gives the output:

$$Y = \begin{bmatrix} \frac{1+2+5+6}{4} & \frac{3+4+7+8}{4} \\ \frac{9+10+13+14}{4} & \frac{11+12+15+16}{4} \end{bmatrix} = \begin{bmatrix} 3.5 & 5.5 \\ 11.5 & 13.5 \end{bmatrix}$$

Thus, the 2×2 average pooling operation can be represented equivalently as a convolution using a 2×2 kernel with all entries equal to $\frac{1}{4}$, stride 2, and no padding.

1.5 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.*

Answer:
Given an input $3 \times 227 \times 227$ with padding, we apply the formula below:

$$F = \frac{N - K + 2P}{S} + 1$$

From the formula we first calculate the convolution output size. Layer 1 - Conv1: 11×11 kernel, stride 4, 96 filters

$$F = \frac{N - K + 2P}{S} + 1 = \frac{227 - 11 + 0}{4} + 1 = 55$$

Output: $96 \times 55 \times 55$
Layer 2 - MaxPool: 3×3 kernel, stride 2

$$F = \frac{55 - 3}{2} + 1 = 27$$

Output: $96 \times 27 \times 27$
Layer 3 - Conv2: 5×5 kernel, pad 2, stride 1, 256 filters

$$F = \frac{27 - 5 + 4}{1} + 1 = 27$$

Output: $256 \times 27 \times 27$
Layer 4 - MaxPool: 3×3 kernel, stride 2

$$F = \frac{27 - 3}{2} + 1 = 13$$

Output: $256 \times 13 \times 13$
Layer 5 - Conv3: 3×3 kernel, pad 1, 384 filters

$$F = \frac{13 - 3 + 2}{1} = 13$$

Output: $384 \times 13 \times 13$
Layer 6 - Conv4: 3×3 kernel, pad 1, 384 filters Output: $384 \times 13 \times 13$
Layer 7 - Conv5: 3×3 kernel, pad 1, 256 filters Output: $256 \times 13 \times 13$
Layer 8 - Maxpool: 3×3 kernel, stride 2

$$F = \frac{13 - 3}{2} + 1 = 6$$

Output: $256 \times 6 \times 6$
Flatten:

- Flattened vector: $256 \times 6 \times 6 = 9216$
- FC1: 4096
- FC2: 4096
- FC: 1000

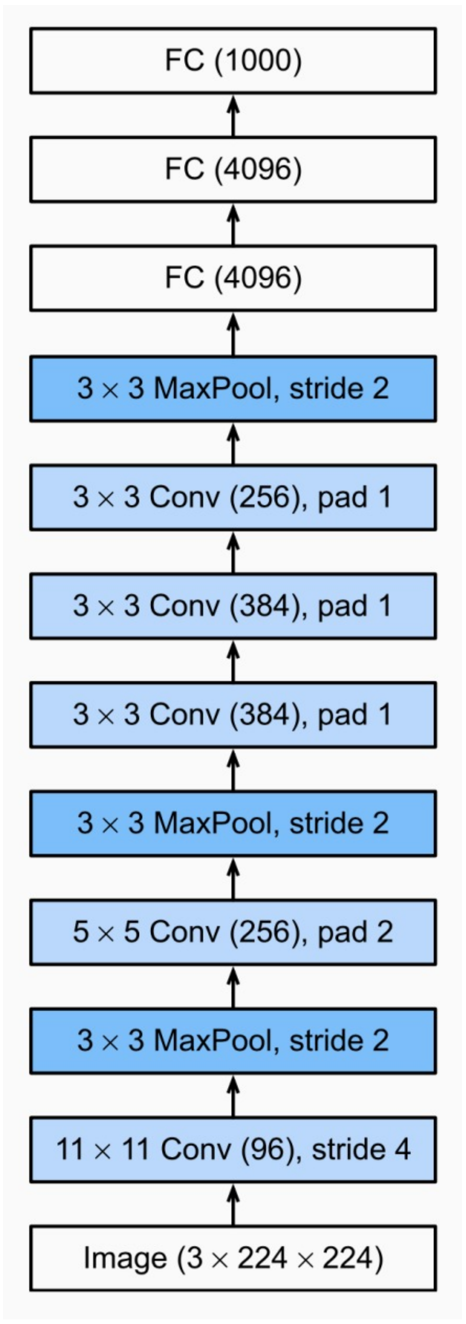

FC (1000)
FC (4096)
FC (4096)
3 × 3 MaxPool, stride 2
3 × 3 Conv (256), pad 1
3 × 3 Conv (384), pad 1
3 × 3 Conv (384), pad 1
3 × 3 MaxPool, stride 2
5 × 5 Conv (256), pad 2
3 × 3 MaxPool, stride 2
11 × 11 Conv (96), stride 4
Image (3 × 224 × 224)

Figure 1:

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/Pj6Hy7cJ`. 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.

AI and external sources disclaimer

GPT-5 via ChatGPT was used to help understand concepts and guide the process of answering the questions (e.g., using prompts such as: "What are the general steps when flipping a kernel and multiplying it with the input matrix?"). It was also used to verify the correctness of computations for some exercises, such as Exercise 1.5. In addition, it assisted in the typesetting of LaTeX.

The resources available at `https://mystaticsite.com/kernelconvolution/` and `https://demonstrations.wolfram.com/ImageKernelsAndConvolutionLinearFiltering/` were used to explore the effects of different kernels on images and to validate the answer in Exercise 1.2. (Never before a thorough and active usage of our 'artisanal' intelligence :D)