Deep Learning E1394
Problem Set 2

Nadine Daum (245963), Nicolas Reichardt (245611), Elena Murray (239793)

Team PS 2 B

Due on Oct 18, 2025 (EOD)

GitHub repo: assignment-2-ps-2-b

AI use statement: Generative AI tools (e.g., ChatGPT-5, GitHub Copilot, Claude, and DeepL for translation) were consulted for proofreading, LaTeX formatting, and general optimization support. All final answers were developed + verified by the authors.

1 Convolutional neural networks

1.1 Kernels

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.

A convolution kernel that leaves the input image unchanged must act as an identity operation. This occurs when the kernel transmits the central pixel value without modification and ignores its neighboring values. For a 3×3 kernel, this can be represented as:

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

During convolution, each output pixel is computed as the weighted sum of the corresponding local neighborhood in the input image. In this case, only the central pixel contributes to the output, as it is multiplied by 1 while all other neighboring pixels are multiplied by 0. Consequently, the output image is identical to the input. Since convolution without padding reduces spatial dimensions, the input image must be zero-padded with one pixel on each side to preserve the original output dimensions.

For instance, if the input image is:

$$\mathrm{I} = \begin{bmatrix} 7 & 2 & 5 \\ 1 & 6 & 3 \\ 4 & 8 & 9 \end{bmatrix}$$

Then zero-padding with size 1 will result in:

$$I_{\text{padded}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 2 & 5 & 0 \\ 0 & 1 & 6 & 3 & 0 \\ 0 & 4 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This ensures that the convolution performs an identity mapping across the entire image.

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

The given kernel is defined as:

$$K = \frac{1}{16}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Each element of the kernel contributes equally to the output, meaning that the convolution computes the mean value of every 4×4 neighborhood in the image. This operation performs an averaging or box blur, where each pixel in the output is replaced by the average intensity of its surrounding pixels. As a result, the image becomes smoother, high-frequency details such as edges and noise are attenuated, and only low-frequency components remain visible.

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

A possible custom kernel can be designed to detect horizontal edges in an image. Such a filter highlights regions where there is a strong change in pixel intensity along the vertical direction. One common choice is the following 3×3 kernel:

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

When this kernel is applied to an image, it produces high positive activations in areas where the intensity changes from dark to bright (from top to bottom), and strong negative activations in the opposite direction. Flat regions with uniform intensity yield values close to 0. As a result, the filter effectively emphasizes horizontal edges and removes regions of low contrast.

1.2 Kernels applied to an image

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

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

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

The kernel shows only negative weights in the top row, near-zero values in the middle, and positive weights in the bottom row. Similarly to the custom kernel designed in the previous question, this structure corresponds to a horizontal edge detector (from dark at the top to bright at the bottom).

1.2.2) 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).

The kernel produces high positive activations along horizontal transitions from dark to bright regions in the image. These occur primarily along the mountain ridges (where darker rock faces meet the bright snow) Also along the shoreline of the lake (where the dark land contrasts with the bright water surface).

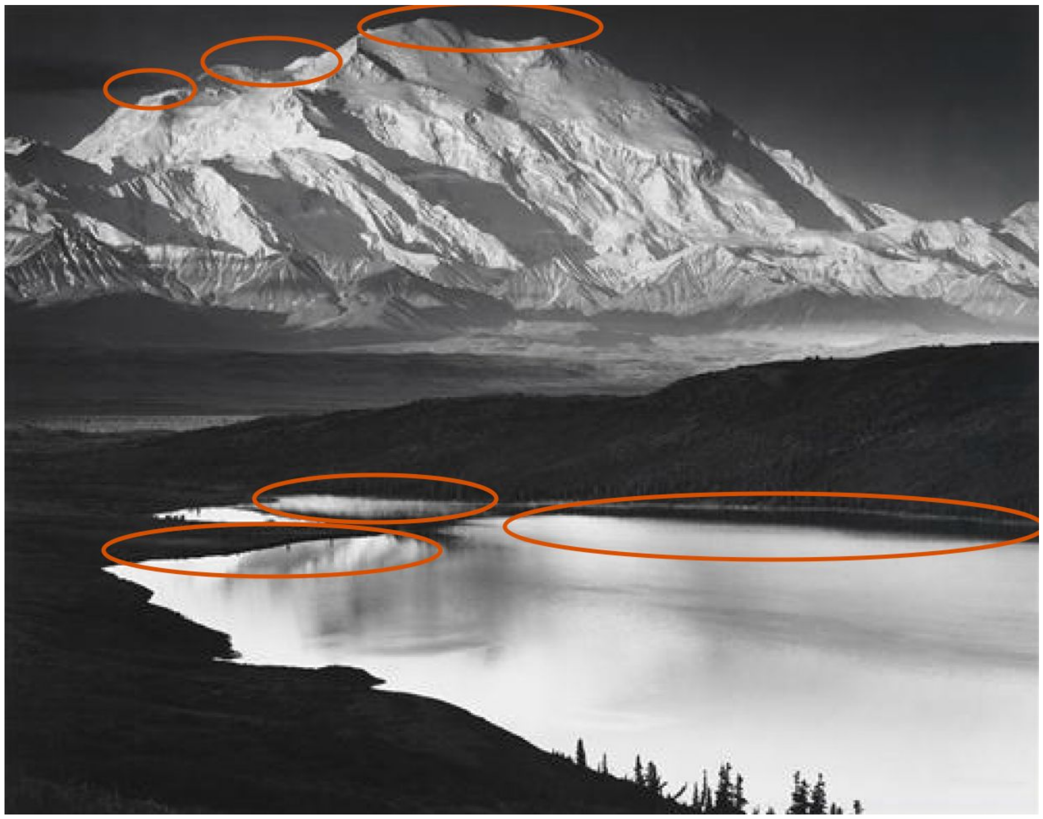

Figure 1: Circled by students: “Mount McKinley and Wonder Lake” Denali National Park, Alaska, 1947, by Ansel Adams.

1.3 Convolution and pooling

1.3.1 Convolution layer

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.**

Before applying the modifications, let us compute the feature map without modifications. We are given the following input image X and kernel.

$$\textbf{Input:} \quad X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0.5 & 1 & 0.5 \\ 0 & 1 & 0 & 1 \end{bmatrix} \qquad \textbf{Kernel:} \quad K = \begin{bmatrix} 0 & 2 & 0 \\ 2 & 1 & 1 \\ 0 & 2 & 0 \end{bmatrix}$$

Since the exercise doesn't mention that we're dealing with a cross-correlation, we first have to flip the kernel by 180°. K' denotes our flipped kernel. Further, we assume a stride of 1 and no padding.

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

We can now perform the 3×3 convolution (which results in an output of 2×2).

First, we apply the kernel to the top-left patch (position (0,0)):

$$X_{00} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0.5 & 1 \end{bmatrix}$$

$$X_{00} = (0 \cdot 0 + 0 \cdot 2 + 0 \cdot 0) + (0 \cdot 1 + 1 \cdot 1 + 0 \cdot 2) + (1 \cdot 0 + 0.5 \cdot 2 + 1 \cdot 0) = 2.0$$

Then, to the top-right patch (position (0,1)):

$$X_{01} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0.5 & 1 & 0.5 \end{bmatrix}$$

$$X_{01} = (0 \cdot 0 + 0 \cdot 2 + 0 \cdot 0) + (1 \cdot 1 + 0 \cdot 1 + 1 \cdot 2) + (0.5 \cdot 0 + 1 \cdot 2 + 0.5 \cdot 0) = 5.0$$

Bottom-left patch (position (1,0)):

$$X_{10} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0.5 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$X_{10} = (0 \cdot 0 + 1 \cdot 2 + 0 \cdot 0) + (1 \cdot 1 + 0.5 \cdot 1 + 1 \cdot 2) + (0 \cdot 0 + 1 \cdot 2 + 0 \cdot 0) = 7.5$$

Bottom-right patch (position (1,1)):

$$X_{11} = \begin{bmatrix} 1 & 0 & 1 \\ 0.5 & 1 & 0.5 \\ 1 & 0 & 1 \end{bmatrix}$$

$$X_{11} = (1 \cdot 0 + 0 \cdot 2 + 1 \cdot 0) + (0.5 \cdot 1 + 1 \cdot 1 + 0.5 \cdot 2) + (1 \cdot 0 + 0 \cdot 2 + 1 \cdot 0) = 2.5$$

This results in the following pre-activation feature map:

$$S = \begin{bmatrix} 2.0 & 5.0 \\ 7.5 & 2.5 \end{bmatrix}$$

Now, we can apply the sigmoid activation function and round the results to two decimals as stated in the header:

$$\text{Apply the sigmoid } \sigma(z) = \frac{1}{1+e^{-z}} \text{ elementwise:}$$

$$\sigma(2.0) = \frac{1}{1+e^{-2.0}} \approx 0.88$$
$$\sigma(5.0) = \frac{1}{1+e^{-5.0}} \approx 0.99$$
$$\sigma(7.5) = \frac{1}{1+e^{-7.5}} \approx 1$$
$$\sigma(2.5) = \frac{1}{1+e^{-2.5}} \approx 0.92$$

This leads to the following final feature map:

$$Y = \sigma(S) \approx \begin{bmatrix} 0.88 & 0.99 \\ 1 & 0.92 \end{bmatrix}.$$

Now, let us apply the modifications in order to achieve a feature map with the same dimensions as the original input image, we simply need to pad the input image by one on each side. Indeed, by still assuming a stride of 1, the output size is given by and equals to the input size when p = 1:

$$n_{\text{out}} = \frac{n - k + 2p}{s} + 1,$$

where

- n = input dimension,
- k = kernel size,
- p = padding size,
- s = stride.

We can now calculate the same as we did before with the new input:

$$X' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad K' = \begin{bmatrix} 0 & 2 & 0 \\ 1 & 1 & 2 \\ 0 & 2 & 0 \end{bmatrix}$$

Position (0,0):

$$\begin{aligned} X_{00} &= 0(0) + 0(2) + 0(0) + 0(1) + 0(1) + 0(2) + 0(0) + 0(2) + 1(0) \\ &= 0 \end{aligned}$$

Position (0,1):

$$\begin{aligned} X_{01} &= 0(0) + 0(2) + 0(0) + 0(1) + 0(1) + 0(2) + 0(0) + 1(2) + 0(0) \\ &= 2 \end{aligned}$$

Position (0,2):

$$\begin{aligned} X_{02} &= 0(0) + 0(2) + 0(0) + 0(1) + 0(1) + 0(2) + 1(0) + 0(2) + 1(0) \\ &= 0 \end{aligned}$$

Position (0,3):

$$\begin{aligned}X_{03} &= 0(0) + 0(2) + 0(0) + 0(1) + 0(1) + 0(2) + 0(0) + 1(2) + 0(0) \\ &= 2\end{aligned}$$

Position (1,0):

$$\begin{aligned}X_{10} &= 0(0) + 0(2) + 0(0) + 0(1) + 0(1) + 1(2) + 0(0) + 1(2) + 0.5(0) \\ &= 4\end{aligned}$$

Position (1,1):

$$\begin{aligned}X_{11} &= 0(0) + 0(2) + 0(0) + 0(1) + 1(1) + 0(2) + 1(0) + 0.5(2) + 1(0) \\ &= 2\end{aligned}$$

Position (1,2):

$$\begin{aligned}X_{12} &= 0(0) + 0(2) + 0(0) + 1(1) + 0(1) + 1(2) + 0.5(0) + 1(2) + 0.5(0) \\ &= 5\end{aligned}$$

Position (1,3):

$$\begin{aligned}X_{13} &= 0(0) + 0(2) + 0(0) + 0(1) + 1(1) + 0(2) + 1(0) + 0.5(2) + 0(0) \\ &= 2\end{aligned}$$

Position (2,0):

$$\begin{aligned}X_{20} &= 0(0) + 0(2) + 1(0) + 0(1) + 1(1) + 0.5(2) + 0(0) + 0(2) + 1(0) \\ &= 2\end{aligned}$$

Position (2,1):

$$\begin{aligned}X_{21} &= 0(0) + 1(2) + 0(0) + 1(1) + 0.5(1) + 1(2) + 0(0) + 1(2) + 0(0) \\ &= 7.5\end{aligned}$$

Position (2,2):

$$\begin{aligned}X_{22} &= 1(0) + 0(2) + 1(0) + 0.5(1) + 1(1) + 0.5(2) + 1(0) + 0(2) + 1(0) \\ &= 2.5\end{aligned}$$

Position (2,3):

$$\begin{aligned}X_{23} &= 0(0) + 1(2) + 0(0) + 1(1) + 0.5(1) + 0(2) + 0(0) + 1(2) + 0(0) \\ &= 5.5\end{aligned}$$

Position (3,0):

$$\begin{aligned}X_{30} &= 0(0) + 1(2) + 0.5(0) + 0(1) + 0(1) + 1(2) + 0(0) + 0(2) + 0(0) \\ &= 4\end{aligned}$$

Position (3,1):

$$\begin{aligned} X_{31} &= 1(0) + 0.5(2) + 1(0) + 0(1) + 1(1) + 0(2) + 0(0) + 0(2) + 0(0) \\ &= 2 \end{aligned}$$

Position (3,2):

$$\begin{aligned} X_{32} &= 0.5(0) + 1(2) + 0.5(0) + 1(1) + 0(1) + 1(2) + 0(0) + 0(2) + 0(0) \\ &= 5 \end{aligned}$$

Position (3,3):

$$\begin{aligned} X_{33} &= 1(0) + 0.5(2) + 0(0) + 0(1) + 1(1) + 0(2) + 0(0) + 0(2) + 0(0) \\ &= 2 \end{aligned}$$

Feature map before activation:

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

Now, we apply $\sigma(z) = \frac{1}{1+e^{-z}}$ to each element, still rounding to two decimals as requested:

$$\sigma(0) = \frac{1}{1+e^{0}} = \frac{1}{2} = 0.50$$

$$\sigma(2) = \frac{1}{1+e^{-2}} \approx 0.88$$

$$\sigma(2.5) = \frac{1}{1+e^{-2.5}} \approx 0.92$$

$$\sigma(4) = \frac{1}{1+e^{-4}} \approx 0.98$$

$$\sigma(5) = \frac{1}{1+e^{-5}} \approx 0.99$$

$$\sigma(5.5) = \frac{1}{1+e^{-5.5}} \approx 1.00$$

$$\sigma(7.5) = \frac{1}{1+e^{-7.5}} \approx 1.00$$

Final feature map after sigmoid:

$$\sigma(S') = \begin{bmatrix} 0.50 & 0.88 & 0.50 & 0.88 \\ 0.98 & 0.88 & 0.99 & 0.88 \\ 0.88 & 1.00 & 0.92 & 1.00 \\ 0.98 & 0.88 & 0.99 & 0.88 \end{bmatrix}$$

We can see that the final feature map has the same dimensions as the original input image.

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

We simply have to take the highest value of the four non-overlapping regions (since stride = 2):

Top-left region:

$$\begin{bmatrix} 0.50 & 0.88 \\ 0.98 & 0.88 \end{bmatrix} \rightarrow \max = 0.98$$

Top-right region:

$$\begin{bmatrix} 0.50 & 0.88 \\ 0.99 & 0.88 \end{bmatrix} \rightarrow \max = 0.99$$

Bottom-left region:

$$\begin{bmatrix} 0.88 & 1.00 \\ 0.98 & 0.88 \end{bmatrix} \rightarrow \max = 1.00$$

Bottom-right region:

$$\begin{bmatrix} 0.92 & 1.00 \\ 0.99 & 0.88 \end{bmatrix} \rightarrow \max = 1.00$$

This results in the following output:

$$\begin{bmatrix} 0.98 & 0.99 \\ 1.00 & 1.00 \end{bmatrix}$$

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

Having a feature map like this in my CNN would be problematic for mainly two reasons. Both resulting from the sigmoid activation function. Indeed, because of the sigmoid activation function all values are transformed into values between 0 and 1. In our case, most values are at the higher end. By taking a max pooling approach, we end up with an output with values heavily concentrated near 1.0. This means that the pooled features are nearly identical, which provides little useful information to distinguish different patterns. More importantly however is the fact that we'll face some issues during backpropagation. Since the derivative of the sigmoid near 1 is extremely small, we might end up in a situation where the gradients vanish, causing the weights to update very slowly or not at all (a phenomenon known as vanishing gradient problem).

1.4 Pooling transformed into convolution

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.

Let us take the example of the following simple 4x4 matrix and let us apply the stated pooling layer to see what it does. According to Zhang et al. (2023), we can simply calculate the dimensions using the following formula. As we only deal with square inputs and kernels, it applies for the width and the height.

$$O = \left\lfloor \frac{I - K + 2P}{S} \right\rfloor + 1$$

O: output size. I: input size. K: kernel size. P: pooling size. S: stride.

$$X = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

2×2 average pooling with stride 2:

$$\text{Top-left: } \frac{1+2+1+2}{4} = 1.5$$

$$\text{Top-right: } \frac{3+4+3+4}{4} = 3.5$$

$$\text{Bottom-left: } \frac{1+2+1+2}{4} = 1.5$$

$$\text{Bottom-right: } \frac{3+4+3+4}{4} = 3.5$$

$$Y = \begin{bmatrix} 1.5 & 3.5 \\ 1.5 & 3.5 \end{bmatrix}$$

Per definition, the average pooling layer takes the average of the values. In this case, we have a 2x2 pooling layer, which takes the average of 4 values each time. Therefore, we can achieve the same result by taking a 2x2 kernel with values 1/4. The output height and width are determined by the input size, kernel size, stride, and padding. For a 2×2 kernel with stride 2 and no padding, the output dimensions are exactly half those of the input in both spatial directions (resulting in our desired output feature dimension).

Let us try applying this to the same matrix:

Equivalent convolution:

$$K = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix}, \quad \text{stride} = 2, \quad \text{padding} = 0$$

$$\text{Top-left: } (1/4)(1 + 2 + 1 + 2) = 1.5,$$

$$\text{Top-right: } (1/4)(3 + 4 + 3 + 4) = 3.5$$

$$\text{Bottom-left: } (1/4)(1 + 2 + 1 + 2) = 1.5,$$

$$\text{Bottom-right: } (1/4)(3 + 4 + 3 + 4) = 3.5$$

$$Y = \begin{bmatrix} 1.5 & 3.5 \\ 1.5 & 3.5 \end{bmatrix}$$

We can see that the proposed convolution yields the same result as the 2x2 average pooling.

1.5 Dimensions of CNN layers

For the CNN shown in Figure 2, write down the dimensions of each each layer and how you computed them. The input image is first increased to 3×227×227 with padding.

As stated in the exercise, we start with an input image of size 3 × 227 × 227. We use the same formula as in the previous exercise.

$$O = \left\lfloor \frac{I - K + 2P}{S} \right\rfloor + 1$$

We will simply apply this to the following layers to get the dimensions. The amount of output channels comes directly from the convolution specs from Figure 2. Pooling layers do not change the amount of channels.

Conv Layer: 11 × 11 Conv (96), stride=4, pad=0

$$\text{Output size: } 96 \times 55 \times 55, \qquad \left\lfloor \frac{227 - 11}{4} \right\rfloor + 1 = 55$$

MaxPool Layer: 3 × 3 MaxPool, stride=2

$$\text{Output size: } 96 \times 27 \times 27, \qquad \left\lfloor \frac{55 - 3}{2} \right\rfloor + 1 = 27$$

Conv Layer: 5 × 5 Conv (256), stride=1, pad=2

$$\text{Output size: } 256 \times 27 \times 27, \quad \left\lfloor \frac{27 + 2 \cdot 2 - 5}{1} \right\rfloor + 1 = 27$$

MaxPool Layer: 3 × 3 MaxPool, stride=2

$$\text{Output size: } 256 \times 13 \times 13, \quad \left\lfloor \frac{27 - 3}{2} \right\rfloor + 1 = 13$$

Conv Layer: 3 × 3 Conv (384), stride=1, pad=1

$$\text{Output size: } 384 \times 13 \times 13, \quad \left\lfloor \frac{13 + 2 \cdot 1 - 3}{1} \right\rfloor + 1 = 13$$

Conv Layer: 3 × 3 Conv (384), stride=1, pad=1

$$\text{Output size: } 384 \times 13 \times 13, \quad \text{same as above}$$

Conv Layer: 3 × 3 Conv (256), stride=1, pad=1

$$\text{Output size: } 256 \times 13 \times 13, \quad \text{same as above}$$

MaxPool Layer: 3 × 3 MaxPool, stride=2

$$\text{Output size: } 256 \times 6 \times 6, \quad \left\lfloor \frac{13 - 3}{2} \right\rfloor + 1 = 6$$

Flattening Layer:

Before the input is fed to the FC layers, we need to flatten it (by multiplying).

$$256 \times 6 \times 6 = 9{,}216$$

10. Fully Connected Layer: 4096 neurons as stated in Figure 2.
11. Fully Connected Layer: 4096 neurons as stated in Figure 2.
12. Fully Connected Layer: 1000 neurons as stated in Figure 2.

References

- Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola, *Dive into Deep Learning*, MIT Press, 2023. Available at: `https://d2l.ai`.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, Chapter 9, 2016. Available at: `https://www.deeplearningbook.org/contents/convnets.html`
- Felizia Quetscher, *A Comprehensible Explanation of the Dimensions in CNNs*, Towards Data Science, 2021. Available at: `https://towardsdatascience.com/a-comprehensible-explanat`