Problem Set 2
Deep Learning E1394

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

Team name: PS 2 G

Team members: Minho Kang (239742) Padma Bareddy (236167) Saurav Kumar Jha (249354)

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, which includes generative AI tools for which you additionally need to describe what content was generated. 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 identity kernel is a 3×3 matrix that leaves the input image unchanged. It is represented as:

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

Before applying this kernel, padding should be apply to the input image. For example, let's say the input image is,

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

The image can be changed like the following with zero-padding

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

Now if we apply the kernel K, then the input image stays as same. □

(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. As a result, this kernel produces a blurred, smoothed image with reduced detail through local average, but preserved overall brightness.

For example, The input image contains this 4x4 region of pixel values M:

$$M = \begin{bmatrix} 8 & 16 & 24 & 32 \\ 40 & 48 & 56 & 64 \\ 72 & 80 & 88 & 96 \\ 104 & 112 & 120 & 128 \end{bmatrix}$$

Then, if we apply the kernel over this region, then we can get the below. So through this process, we can get blur (which means gradually change), but preserved overall brightness.

$$M * I = \frac{1}{16}(1088) = 68$$

which is the average of all 16 pixel values in M □

(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 can think of diagonal edge detector kernel like the following

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

This gives large positive or negative results where intensity changes along diagonals, and near zero elsewhere.

For example, we can think of the below image

$$P = \begin{bmatrix} 255 & 0 \\ 0 & 255 \end{bmatrix}$$

The result after applying the kernel is $255 + 255 = 510$
Another example is,

$$N = \begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix}$$

The result is $5 - 10 - 15 + 20 = 0$.
So this kernel makes more brighter for diagonal and more darker for opposite diagonal. And if pixel values are similar, then it gives nearly 0.

□

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)

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

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

   Solution. It detects negative value on the top and positive value on the bottom. That is, it will result high positive value for this part. □

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).

   Solution. I choose the part where the top is dark, and the bottom is bright. This part will be converted to the high positive value with the above kernel. □

Figure 1: "Mount McKinley and Wonder Lake" Denali National Park, Alaska, 1947, by Ansel Adams.

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. Input X is 4*4 matrix Kernel K is 3*3 matrix Padding 1 pixel on all sides to keep the orginal image same size i.e 4 x 4

Pad the input as below:

$$\mathbf{K} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Compute convolution output, slide input over Xpad:
example for position (0,0)

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

$$z_{0,0} = 0 \cdot 0 + 2 \cdot 0 + 0 \cdot 0 + 2 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 2 \cdot 0 + 0 \cdot 1 = 0$$

For position (0,1)

$$\text{patch} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$z_{0,1} = 0 \cdot 0 + 2 \cdot 0 + 0 \cdot 0 + 2 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 2 \cdot 1 + 0 \cdot 0 = 2$$

For position (1,0)

$$\text{patch} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0.5 \end{bmatrix}$$

Compute elementwise:

$$0 + 0 + 0 + 2 \cdot 0 + 1 \cdot 1 + 1 \cdot 0 + 0 \cdot 0 + 2 \cdot 1 + 0 \cdot 0.5 = 0 + 0 + 1 + 2 = 3$$

So $z_{1,0} = 3$.
Full preactivation featuer map after computing for all positions

$$Z = \begin{bmatrix} 0.0 & 2.0 & 0.0 & 2.0 \\ 3.0 & 2.0 & 5.0 & 2.0 \\ 1.5 & 7.5 & 2.5 & 6.5 \\ 3.0 & 2.0 & 5.0 & 2.0 \end{bmatrix}$$

Each number is derived by summing nine element wise products of the kernel with the 3×3 patch at that location.

sigmoid activation

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

z	$\sigma(z)$	rounded
0.0	0.500	0.50
2.0	0.881	0.88
3.0	0.953	0.95
5.0	0.993	0.99
6.5	0.9985	1.00
7.5	0.99945	1.00
1.5	0.818	0.82
2.5	0.924	0.92

Post-activation feature map:

$$A = \begin{bmatrix} 0.50 & 0.88 & 0.50 & 0.88 \\ 0.95 & 0.88 & 0.99 & 0.88 \\ 0.82 & 1.00 & 0.92 & 1.00 \\ 0.95 & 0.88 & 0.99 & 0.88 \end{bmatrix}$$

1.3.2 Pooling layer (2/15 pts)

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

Starting post-activation feature map (4×4)

$$A = \begin{bmatrix} 0.50 & 0.88 & 0.50 & 0.88 \\ 0.95 & 0.88 & 0.99 & 0.88 \\ 0.82 & 1.00 & 0.92 & 1.00 \\ 0.95 & 0.88 & 0.99 & 0.88 \end{bmatrix}$$

Pooling parameters: window $= 2 \times 2$, stride $= 2$, no overlap.

Pooling windows and choosing maximum

1. **(rows 0–1, cols 0–1):**

$$\begin{bmatrix} 0.50 & 0.88 \\ 0.95 & 0.88 \end{bmatrix} \quad \Rightarrow \quad \max = 0.95$$

2. **(rows 0–1, columns 2–3):**

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

3. **(rows 2–3, columns 0–1):**

$$\begin{bmatrix} 0.82 & 1.00 \\ 0.95 & 0.88 \end{bmatrix} \quad \Rightarrow \quad \max = 1.00$$

4. **(rows 2–3, columns 2–3):**

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

Final pooled output (2×2)

$$\begin{bmatrix} 0.95 & 0.99 \\ 1.00 & 1.00 \end{bmatrix}$$

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:

The disappearance gradient is one of the most critical problems that can occur as most values are either fully saturated (1.0) or close to saturation (0.88) (close to sigmoid's asymptotes) Backpropagation will produce extremely small gradient signals and learning effectively stops for saturated neurons. Earlier layers receive virtually no gradient (gradients vanish exponentially through layers)

Other major problems are a) loss of gradient diversity as most activations are in the range 0.88-1.00 range. This implies that the network has less ability to distinguish fine differences between inputs b) potential for overfitting;Strong, repeated high activations across many spatial locations can indicate redundant filters (all firing similarly) or that the network is memorizing rather than learning generalizable features.

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.

Solution:

A 2×2 average pooling operation can be exactly represented as a convolution operation with the following parameters:

$$Y_{i,j} = \frac{1}{k^2} \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} X_{i+m,j+n}, \quad \text{for a } k \times k \text{ pooling window}$$

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix}$$

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

$$\begin{aligned}
y_{11} &= \frac{1}{4}(x_{11} + x_{12} + x_{21} + x_{22}) \\
y_{12} &= \frac{1}{4}(x_{12} + x_{13} + x_{22} + x_{23}) \\
y_{13} &= \frac{1}{4}(x_{13} + x_{14} + x_{23} + x_{24}) \\
y_{21} &= \frac{1}{4}(x_{21} + x_{22} + x_{31} + x_{32}) \\
y_{22} &= \frac{1}{4}(x_{22} + x_{23} + x_{32} + x_{33}) \\
y_{23} &= \frac{1}{4}(x_{23} + x_{24} + x_{33} + x_{34}) \\
y_{31} &= \frac{1}{4}(x_{31} + x_{32} + x_{41} + x_{42}) \\
y_{32} &= \frac{1}{4}(x_{32} + x_{33} + x_{42} + x_{43}) \\
y_{33} &= \frac{1}{4}(x_{33} + x_{34} + x_{43} + x_{44})
\end{aligned}$$

Numerical example for 2x2 average pooling with stride 1.

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

Convolution outputs

$$\begin{aligned}
y_{11} &= \frac{1}{4}(1+2+5+6) = 3.5, & y_{12} &= \frac{1}{4}(2+3+6+7) = 4.5, & y_{13} &= \frac{1}{4}(3+4+7+8) = 5.5, \\
y_{21} &= \frac{1}{4}(5+6+9+10) = 7.5, & y_{22} &= \frac{1}{4}(6+7+10+11) = 8.5, & y_{23} &= \frac{1}{4}(7+8+11+12) = 9.5, \\
y_{31} &= \frac{1}{4}(9+10+13+14) = 11.5, & y_{32} &= \frac{1}{4}(10+11+14+15) = 12.5, & y_{33} &= \frac{1}{4}(11+12+15+16) = 13.5
\end{aligned}$$

Final 3x3 output

$$Y = \begin{bmatrix} 3.5 & 4.5 & 5.5 \\ 7.5 & 8.5 & 9.5 \\ 11.5 & 12.5 & 13.5 \end{bmatrix}$$

1.5 Dimensions of CNN layers (10 pts)

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 $3 \times 227 \times 227$ 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.

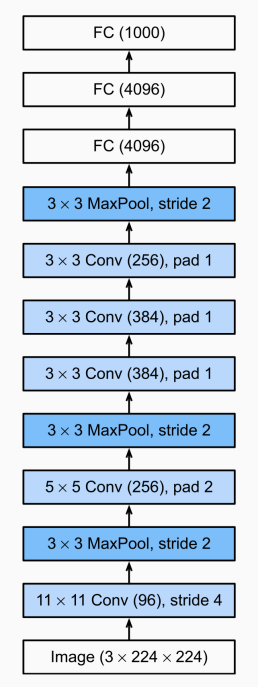

Figure 2: Convolutional neural network in simplified form. Note that the input image is first increased to $3 \times 227 \times 227$ with padding.

Solution:
For convolutional layers, the output spatial dimensions are calculated as:

$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} - K + 2P}{S} \right\rfloor + 1, \quad W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} - K + 2P}{S} \right\rfloor + 1 \tag{2}$$

For pooling layers, the output spatial dimensions are:

$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} - K_{\text{pool}}}{S} \right\rfloor + 1, \quad W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} - K_{\text{pool}}}{S} \right\rfloor + 1 \tag{3}$$

Where $H_{\text{in}}, W_{\text{in}}$ are input height and width; $H_{\text{out}}, W_{\text{out}}$ are output height and width; K is kernel size; P is padding; S is stride; and C_{out} is the number of output channels. For pooling layers, the

number of channels remains unchanged.
Layer-by-Layer Calculations:

Layer	Operation	Input	Output	Calculation
0	Input (after padding)	–	$3 \times 227 \times 227$	Given
1	11×11 Conv(96), $S = 4$	$3 \times 227 \times 227$	$96 \times 55 \times 55$	$\lfloor(227-11+0)/4\rfloor+1 = \lfloor 216/4\rfloor+1 = 55$
2	3×3 MaxPool, $S = 2$	$96 \times 55 \times 55$	$96 \times 27 \times 27$	$\lfloor(55-3)/2\rfloor + 1 = \lfloor 52/2\rfloor + 1 = 27$
3	5×5 Conv(256), $P = 2$	$96 \times 27 \times 27$	$256 \times 27 \times 27$	$\lfloor(27-5+4)/1\rfloor + 1 = \lfloor 26/1\rfloor + 1 = 27$
4	3×3 MaxPool, $S = 2$	$256 \times 27 \times 27$	$256 \times 13 \times 13$	$\lfloor(27-3)/2\rfloor + 1 = \lfloor 24/2\rfloor + 1 = 13$
5	3×3 Conv(384), $P = 1$	$256 \times 13 \times 13$	$384 \times 13 \times 13$	$\lfloor(13-3+2)/1\rfloor + 1 = \lfloor 12/1\rfloor + 1 = 13$
6	3×3 Conv(384), $P = 1$	$384 \times 13 \times 13$	$384 \times 13 \times 13$	$\lfloor(13-3+2)/1\rfloor + 1 = \lfloor 12/1\rfloor + 1 = 13$
7	3×3 Conv(256), $P = 1$	$384 \times 13 \times 13$	$256 \times 13 \times 13$	$\lfloor(13-3+2)/1\rfloor + 1 = \lfloor 12/1\rfloor + 1 = 13$
8	3×3 MaxPool, $S = 2$	$256 \times 13 \times 13$	$256 \times 6 \times 6$	$\lfloor(13-3)/2\rfloor + 1 = \lfloor 10/2\rfloor + 1 = 6$
9	Flatten	$256 \times 6 \times 6$	9216	$256 \times 6 \times 6 = 9216$
10	FC	9216	4096	Fully connected layer
11	FC	4096	4096	Fully connected layer
12	FC (Output)	4096	1000	Fully connected layer

Table 1: Dimensions of each layer in the CNN architecture.

2 Identifying land use with CNN

You will create an image classifier using PyTorch to identify land use and land cover for the EuroSAT database (link). All tasks are described in their specific sections and marked by `#TODO`. You may need to import additional libraries, but do not change the existing trainer and input of the functions. Sometimes, a brief comment on your implementation is required: add the answers to these questions directly below the code. Remember to always show the code output in the final upload.

1. Transform the data (10 pt)
2. Implement your Convolutional Neural Network (15 pt)

3. Tune the model hyperparameters (optional, max 10 pt)
4. Load and fine-tune a pre-trained model (10 pt)
5. Compare models and discuss results (5 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/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.