Assignment 2

Visual Computing Fundamentals: Image Processing

Pablo Díaz Viñambres : *pablodi@ntnu.no*

November 7, 2022

Contents

1 Convolutional Neural Networks - Theory

(a) Given a single convolutional layer with a stride of 1, kernel size of 5 × 5, and 6 filters. If I want the output shape (Height × Width) of the convolutional layer to be equal to the input image, how much padding should I use on each side?

We can use the formulas

$$W_2 = \frac{W_1 - F_W + 2P_W}{S_W} + 1$$
$$H_2 = \frac{H_1 - F_H + 2P_H}{S_H} + 1$$

Plugging in $F_W = F_H = 5$ for the kernel size, $S = 1$ for the stride and $W_2 = W_1, H_2 = H_1$ for size conservation, we get $P_W = P_H = 2$. That is, we should pad the image with 2 pixels on each size.

Consider a CNN whose inputs are RGB color images of size 512×512. The network has two convolutional layers. Using this information, answer the following:

(b) You are told that the spatial dimensions of the feature maps in the first layer are 504×504, and that there are 12 feature maps in the first layer. Assuming that no padding is used, the stride is 1, and the kernel used are square, and of an odd size, what are the spatial dimensions of these kernels? Give the answer as (Height) × (Width).

Now we plug into the same formula the values $W_1 = H_1 = 512$ for the original image dimensions, $W_2 = H_2 = 504$ for the output (feature maps) dimensions, $S = 1$ for the stride and $P_W = P_H = 0$ since we don't use any padding. Solving for F_W and F_H we get kernel dimensions of 9x9.

(c) If subsampling is done using neighborhoods of size 2 × 2, with a stride of 2, what are the spatial dimensions of the pooled feature maps in the first layer? (assume the input has a shape of 504 × 504). Give the answer as (Height) × (Width).

Since the stride is equal to the subsampling field height and width, the subsampling process will process disjoint and adjacent squares of pixels. Therefore, for every 2x2 pixels it will compute a single value and the final size will be half the width and height of the original feature map, 252x252 pixels.

(d) The spatial dimensions of the convolution kernels in the second layer are 3 × 3. Assuming no padding and a stride of 1, what are the sizes of the feature maps in the second layer? (assume the input shape is the answer from the last task). Give the answer as (Height) × (Width).

We now have an input size of $W_1 = H_1 = 252$ pixels, also $F_W = F_H = 3$ because of the kernel size, $S = 1$ and $P = 0$. Solving for W_2 and H_2 we get a size of 250x250 for the feature maps of the second layer.

(e) Table 1 shows a simple CNN. How many parameters are there in the network? In this network, the number of parameters is the number of weights + the number of biases. Assume the network takes in an 32×32 image.

We computed the number of weights and biases for each layer following the formulas $W = F_W \cdot F_H \cdot C_1 \cdot C_2, B = C_2$ for the weights and biases of the convolutional layers. For the fully connected layers, we also used $W = N_{in} \cdot N_{out}, B = N_{out}$ where N_{in} and N_{out} are the number of initial and output neurons on the layer. We also assumed the image to be monochrome, so $C_1 = 1$. The results are summarised on the following table:

Layer	Layer Type	Hidden Units/Filters	Weights	Biases	Output size
1	Conv2D (kernel size=5, stride=1, padding=2)	32	800	32	32x32
1	MaxPool2D (kernel size=2, stride=2)	-	-	-	16x16
2	Conv2D (kernel size=3, stride=1, padding=1)	64	576	64	16x16
2	MaxPool2D (kernel size=2, stride=2)	-	-	-	8x8
3	Conv2D (kernel size=3, stride=1, padding=1)	128	1152	128	8x8
3	MaxPool2D (kernel size=2, stride=2)	-	-	-	4x4
	Flatten	-	-	-	16
4	Fully-Connected	64	1024	64	64
5	Fully-Connected	10	640	10	10

In total, we have 4192 weights and 330 biases, that is, 4522 parameters.

2 Convolutional Neural Networks - Programming

(a) Implement the network in Table 1. Report the final accuracy on the validation set for the trained network. Include a plot of the training and validation loss during training.

(We will answer the first two subtasks together)

(b) Compare SGD with Adam optimizer.

After implementing the network in `task2.ipynb` we get the following train and test loss graphs for tasks a) and b):

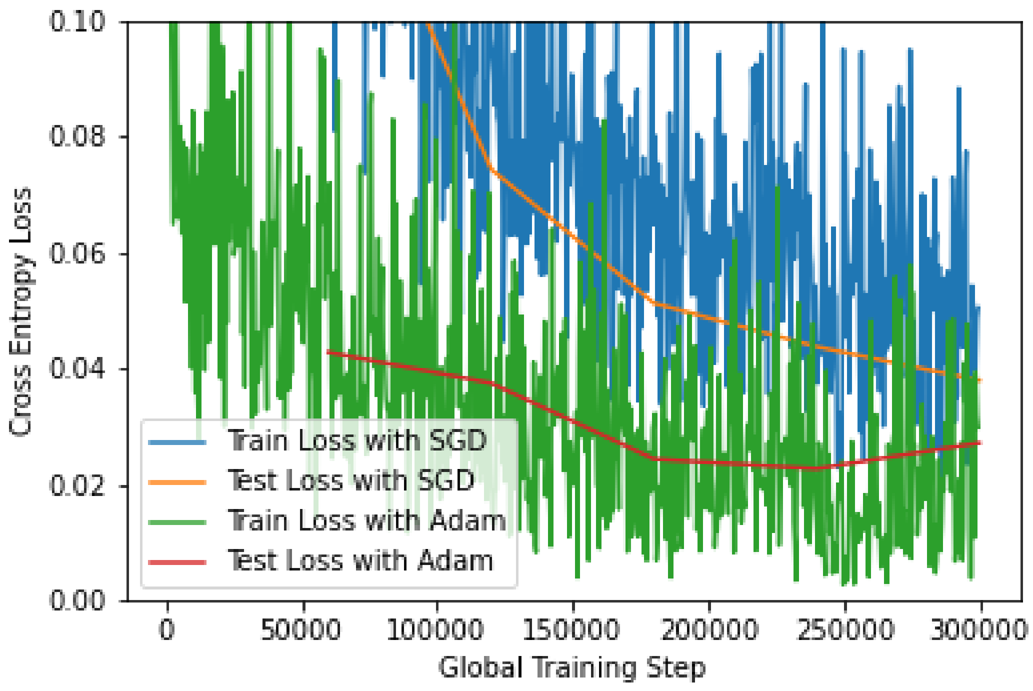


Figure 1: Loss graphs for train and tests sets with SGD and Adam optimizers

When using the SGD optimizer, the final test loss is 0.0373 and the final test accuracy is 0.988. But when using the Adam optimizer, the final test loss is 0.0270 and the final test accuracy is 0.9901.

We get better results from the Adam optimizer overall. However, we start to see some overfitting with this optimizer. Around the training step #250000, the test loss starts to increase. This happens because we are extracting too much information from the training data, making the model too dependent on it. Thus, our test data predictions get worse. The SGD optimizer displays less overfitting but we also see some stagnation on the test loss in the last steps.

(c) Run the image `zebra.jpg` through the first layer of the ResNet50 network. Visualize the filter, and the grayscale activation of a the filter, by plotting them side by side. Use the pre-trained network ResNet50 and visualize the convolution filters with indices [5, 8, 19, 22, 34].

We get the following visualization images from the weights and activations on those indices:

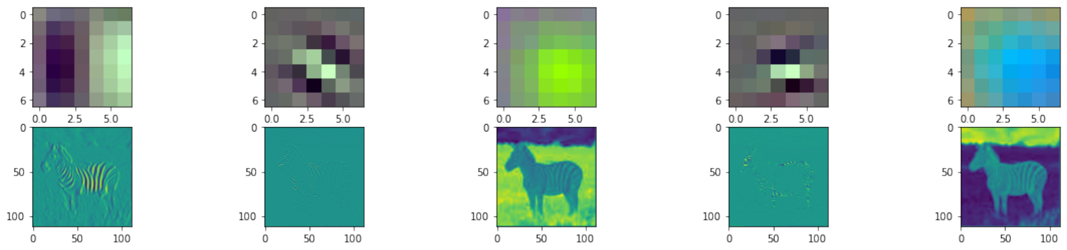


Figure 2: Loss graphs for train and tests sets with SGD and Adam optimizers

(d) Looking at the visualized filter, and its corresponding activation on the zebra image, describe what kind of feature each filter extracts. Explain your reasoning.

- **Filter 5:** this filter is doing vertical edge detection. We can see a dark left area and a bright right area, resembling the structure of an horizontal Sobel kernel.
- **Filter 8:** this filter is also detecting edges, but in a tilted angle. On the activation image, we can see some parts of the silhouette of the zebra.
- **Filter 19:** this filter is trying to find the green areas of the image. We can see that the RGB weights are mostly green and the grass is activated in the output.
- **Filter 22:** this filter is also trying to detect some tilted edges. In a similar way to filter 8, we can see some parts of the zebra silhouette activated.
- **Filter 34:** this filter is trying to find the blue areas of the image. We can see that the RGB weights are mostly blue and the sky is activated in the output.

3 Filtering in the Frequency Domain - Theory

(a) Given the images in the spatial and frequency domain in Figure 3, pair each image in the spatial domain (first row) with a single image in the frequency domain (second row). Explain your reasoning.

We first include an example for our matching and then enumerate the rest:

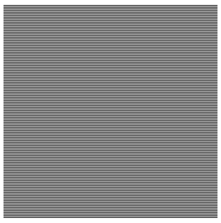


Figure 1a

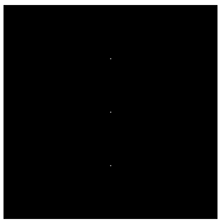


Figure 2e

Figure 1a goes with Figure 2e because 1a is composed of the thinnest horizontal stripes, while figure 2e has the most separated vertical points.

In the other images made of stripes, the general rule is that the Fourier transform flips the orientation and the distance between the points is proportional to the frequency of the stripes.

The rest of the matching is shown in the next page.

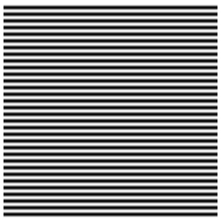


Figure 1b

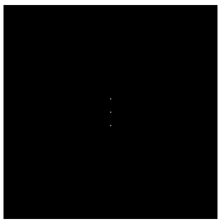


Figure 2c

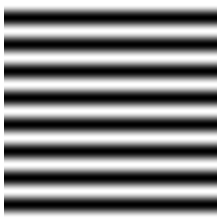


Figure 1c

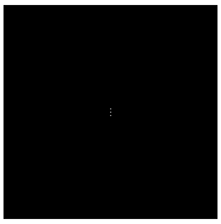


Figure 2f

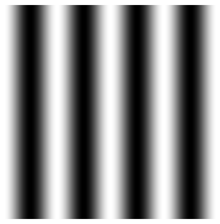


Figure 1d

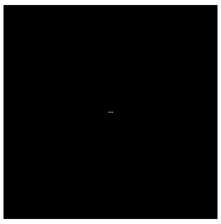


Figure 2b

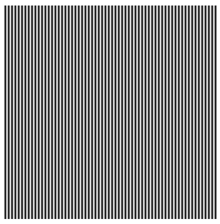


Figure 1e

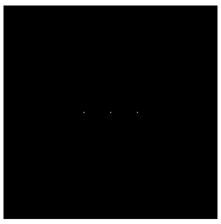


Figure 2d

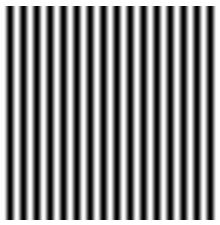


Figure 1f

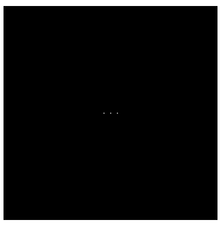
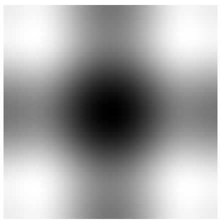


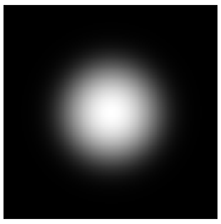
Figure 2a

(b) What are high-pass and low-pass filters?

High-pass filters remove low frequencies from an wave in its frequency domain, conversely, low-pass filters remove high frequencies. In our image filtering domain, this filters are applied to the Fourier transform of the image via multiplication with a mask, usually circle-shaped.

(c) The amplitude $|\mathcal{F}\{g\}|$ of two commonly used convolution kernels can be seen in Figure 4. For each kernel (a, and b), figure out what kind of kernel it is (high- or low-pass). Shortly explain your reasoning.


(a)


(b)

Kernel (a) is a high-pass filter, since the low frequencies located at the center of the Fourier transform are dark. Therefore, they will be multiplied by 0 and removed.

Kernel (b) is a low-pass filter, since the high frequencies located at the edges of the Fourier transform are dark. Therefore, they will be multiplied by 0 and removed.

4 Filtering in the Frequency Domain - Programming

(a) Implement a function that takes an grayscale image, and a kernel in the frequency domain, and applies the convolution theorem. Try it out on a low-pass filter and a high-pass filter on the grayscale image "camera man".

The visualizations given include:

- The original image (in spatial domain).
- The Fourier Transform of the image, taking its amplitude and applying the log transform to it
- The filter mask
- The multiplied (convoluted) Fourier Transform of the image
- The result image

We see that the low pass filter doesn't alter the image that much, since most of the important frequencies are intact, but the high pass filter destroys most of the information of the image and only leaves some barely visible edges.

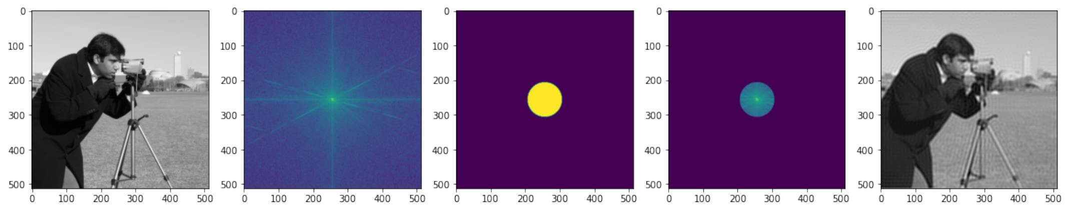


Figure 10: Applying a low-pass filter to the camera man image.

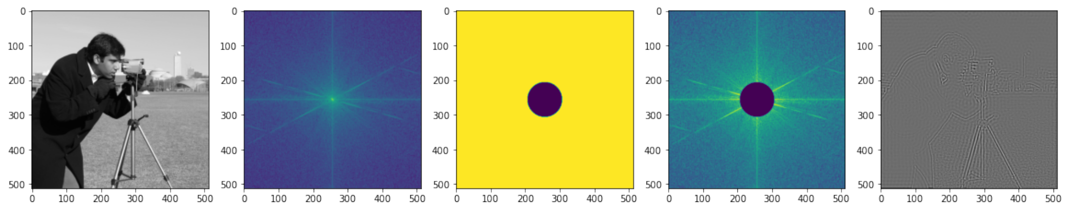


Figure 11: Applying a high-pass filter to the camera man image.

(b) Implement a function that takes an grayscale image, and a kernel in the spatial domain, and applies the convolution theorem. Try it out on the gaussian kernel given in assignment 1, and a horizontal sobel filter (G_x).

For this task, we took the original filter, padded it with zeros to fit the image size, and computed its Fourier Transform. We then used most of the code from the previous subtask for performing the convolution and obtaining the final visualization.

We get the following results, mimicking those gotten in the spatial filtering done in the previous assignment.

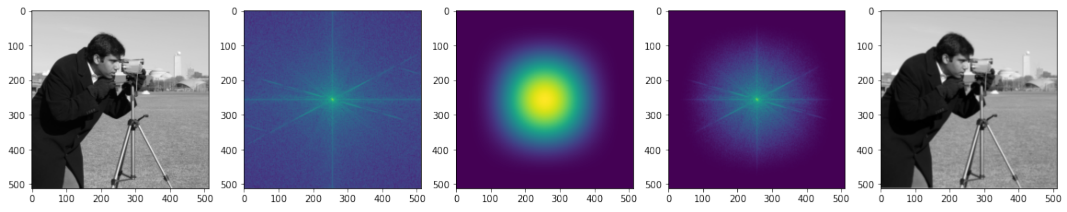


Figure 12: Applying a Gaussian filter to the camera man image.

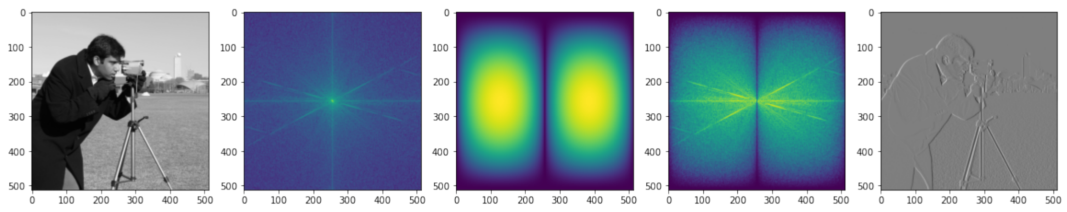


Figure 13: Applying an horizontal Sobel filter to the camera man image.

(c) Use what you've learned from the lectures and the recommended resources to remove the noise in the image seen in the moon picture. Note that the noise is a periodic signal.

For the noise removal, we noted that the Fourier transform of the moon image has abnormal bumps along the horizontal axis. This makes sense when we compare it with the results of Task 2, where we saw that vertical stripes on an image produce horizontal dots on the frequency domain.

Thus, we created a filter that consists of 6 circles that cover those dots and multiply them by 0. The filter and final result can be seen in the following figure:

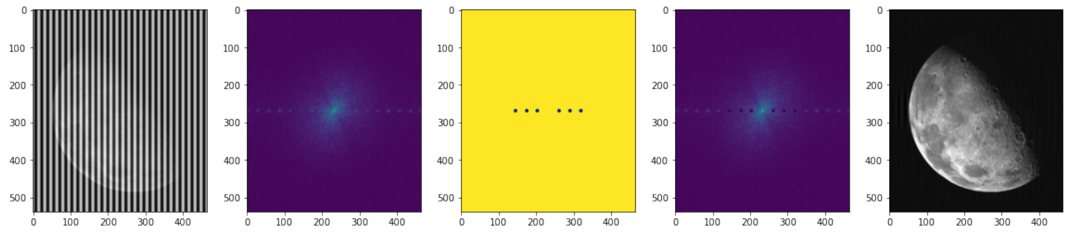


Figure 14: Removing unwanted frequencies from an image Fourier transform.

And we get the following final image:

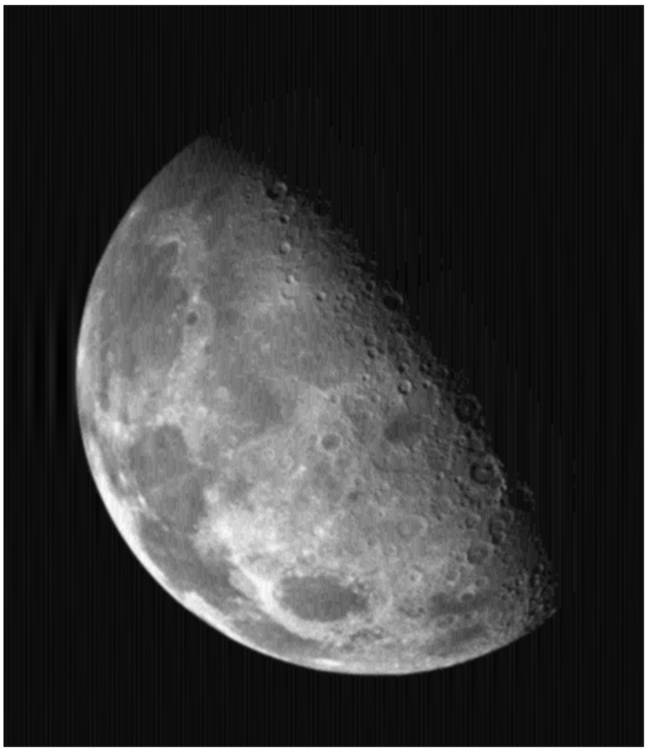


Figure 15: The moon picture after removing the noise.

(d) Create a function to automatically find the rotation of scanned documents, such that we can align the text along the horizontal axis. Use the frequency domain to extract a binary image which draws a rough line describing the rotation of each document.

To find the binary image, we computed the Fourier transform to the original image, took its absolute value, log-transformed it, and normalized it to the range [0, 255]. Then, we opened the frequency spectrum in GIMP, and using the threeshold tool, we checked for the lowest value that produces a sharp straight line.

We found this value to be around 120 for the sample images. The binary image was then composed of zeros on the pixels where the normalized frequency spectrum was below 120 and of ones in the pixels that were above.

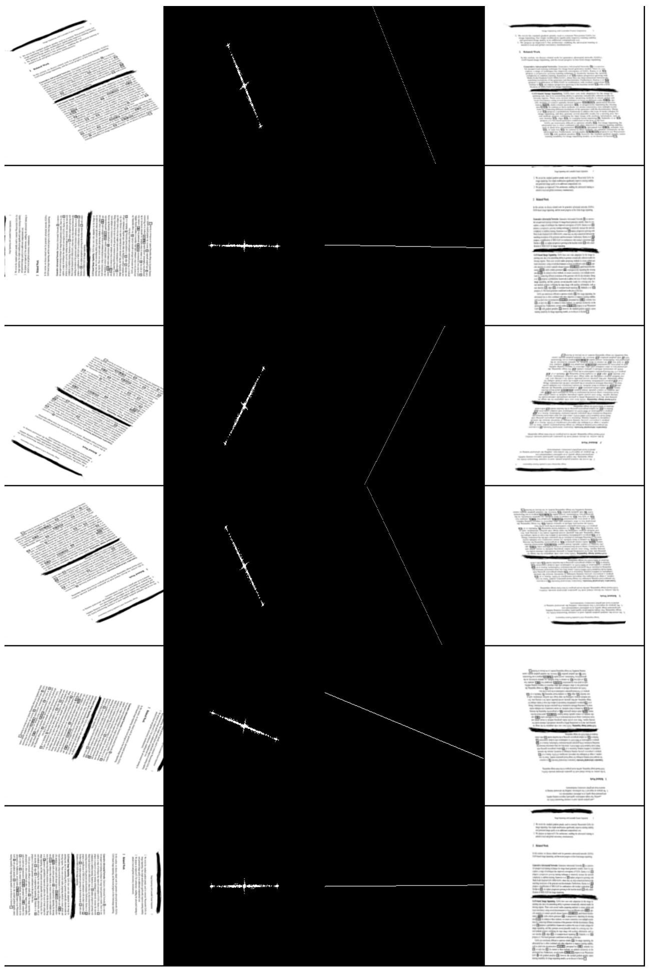


Figure 16: Finding orientation of scanned documents using the Fourier Transform.