Image Processing - Assignment 2

Convolutional Neural Networks [2.5 points]

Task 1: Theory [0.9 points]

Table 1: A simple CNN. Number of hidden units specifies the number of hidden units in a fully-connected layer. The number of filters specifies the number of filters/kernels in a convolutional layer. The activation function specifies the activation function that should be applied after the fully-connected/convolutional layer. The flatten layer takes an image with shape (Height) × (Width) × (Number of Feature Maps), and flattens it to a single vector with size (Height) · (Width) · (Number of Feature Maps).

a) **[0.1pt] 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?**

$W_1 = W_2 = W,\ H_1 = H_2 = H,\ C_1 = 3,\ F_W = 5,\ F_H = 5,\ S_W = 1, S_H = 1,\ P_W = ?,\ P_H = ?$

$$W_2 = \frac{(W_1 - F_W + P_W)}{S_W} + 1 \Rightarrow P_W = \frac{S_w(W_2 - 1) + F_W - W_1}{2} = \frac{W(S_W - 1) + F_W - S_W}{2} = \frac{0 + 5 - 1}{2} = 2$$

$$H_2 = \frac{(H_1 - F_H + P_H)}{S_H} + 1 \Rightarrow P_H = \frac{S_H(H_2 - 1) + F_H - H_1}{2} = \frac{H(S_H - 1) + F_H - S_H}{2} = \frac{0 + 5 - 1}{2} = 2$$

Input: 512 x 512 x 3 -> Output: 512 x 512 x 6

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) **[0.2pt] 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 kernels used are square, and of an odd size, what are the spatial dimensions of these kernels? Give the answer as (Height) × (Width).**

$W_1 = 512,\ H_1 = 512,\ C_1 = 3,\ F_W = 2n + 1,\ F_H = 2n + 1,\ S_W = 1, S_H = 1,\ P_W = 0,\ P_H = 0$

$W_2 = 504,\ H_2 = 504,\ C_2 = 12$

$$W_2 = \frac{(W_1 - F_W + P_W)}{S_W} + 1 \Rightarrow F_W = W_1 + P_W - S_W(W_2 - 1)$$

$$\Rightarrow 2n + 1 = 512 + 0 - 1(504 - 1) \Rightarrow n = 4$$

$F_W = 2n + 1 = 9,\ F_H = 2n + 1 = 9$

$=> 9 \times 9$

c) [0.1pt] 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).

Let's treat the feature map as a spatial input, and the subsampling window as the kernel.

$W_2 = 504,\ H_2 = 504,\ C_2 = 12,\ N_W = 2,\ N_H = 2,\ S_W = 2, S_H = 2,\ P_W = 0,\ P_H = 0$

$W_p = ?,\ H_p = ?$

$$W_p = \frac{(W_2 - N_W + P_W)}{S_W} + 1 = \frac{(504-2)}{2} + 1 = 252$$

$$H_p = \frac{(H_2 - N_H + P_H)}{S_H} + 1 = \frac{(504-2)}{2} + 1 = 252$$

$=> 252 \times 252$

d) [0.2pt] 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).

$W_p = 252,\ H_p = 252,\ C_2 = 12,\ F_W = 3,\ F_H = 3,\ S_W = 1, S_H = 1,\ P_W = 0,\ P_H = 0$

$$W_2 = \frac{(W_p - F_W + P_W)}{S_W} + 1 = \frac{(252 - 3 + 0)}{1} + 1 = 250$$

$$H_2 = \frac{(H_p - F_H + P_H)}{S_H} + 1 = \frac{(252 - 3 + 0)}{1} + 1 = 250$$

$=> 250 \times 250$

e) [0.3pt] 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 a 32 × 32 image.

Layer	Layer Type	Number of Hidden Units/Filters	Activation
1	Conv2D (kernel size=5, stride=1, padding=2)	32	ReLU
1	MaxPool2D (kernel size=2, stride=2)	–	–
2	Conv2D (kernel size=3, stride=1, padding=1)	64	ReLU
2	MaxPool2D (kernel size=2, stride=2)	–	–
3	Conv2D (kernel size=3, stride=1, padding=1)	128	ReLU
3	MaxPool2D (kernel size=2, stride=2)	–	–
	Flatten	–	–
4	Fully-Connected	64	ReLU
5	Fully-Connected	10	Softmax

ID	Layer Type	Activation Shape	Activation Size	# Parameters
1	Input Layer:	(32, 32, 3)	3072	-
2	Conv2D (kernel size=5, stride=1, padding=2)	(32, 32, 32)	32768	2432
3	MaxPool2D (kernel size=2, stride=2)	(16, 16, 32)	8192	-
4	Conv2D (kernel size=3, stride=1, padding=1)	(16, 16, 64)	16384	18496
5	MaxPool2D (kernel size=2, stride=2)	(8, 8, 64)	4096	-
6	Conv2D (kernel size=3, stride=1, padding=1)	(8, 8, 128)	8192	73856
7	MaxPool2D (kernel size=2, stride=2)	(4, 4, 128)	2048	-
8	Flatten	(2048, 1)	-	-
9	Fully-Connected	(64, 1)	64	133120
10	Fully-Connected	(10, 1)	10	650

11	sum()	-	-	228554

(ID = 2): Activation Shape is calculated by using the formula from a) it gives us:
(32 × 32) and the number of hidden units/filters = 3.
Number of parameters in a CONV layer: ((m * n * d)+1)* k), where
((shape of width of filter * shape of height filter * number of filters in the previous layer+1)*number of filters).
Parameters = ((5 * 5 * 3)+1)* 32) = 2432

(ID = 3): Activation Shape is calculated by using the formula from a) it gives us:
(16 × 16) and the number of hidden units/filters stay the same = 3. There are no weights in a pooling layer.

(ID = 4): Activation Shape is calculated by using the formula from a) it gives us:
(16 × 16) and the number of hidden units/filters = 64.
Parameters = ((3 * 3 * 32)+1)* 64) = 25632

(ID = 5): Activation Shape is calculated by using the formula from a) it gives us:
(8 × 8) and the number of hidden units/filters stay the same = 64. There are no weights in a pooling layer.

(ID = 6): Activation Shape is calculated by using the formula from a) it gives us:
(8 × 8) and the number of hidden units/filters = 128.
Parameters = ((3 * 3 * 64)+1)* 128) = 25632

(ID = 7): Activation Shape is calculated by using the formula from a) it gives us:
(4 × 4) and the number of hidden units/filters stay the same = 128. There are no weights in a pooling layer.

(ID = 8): The flatten layer takes an image with shape (Height) × (Width) × (Number of Feature Maps), and flattens it to a single vector with size (Height) · (Width) · (Number of Feature Maps) => 4 · 4 · 128 = 2048.

(ID = 9): Parameters in the FC layer is the number of neurons in the current layer c, multiplied with the number of neurons on the previous layer p + the a bias to every neuron in the current layer;
((current layer c*previous layer p)+1*c) = ((64 * 2048)+1*2048) = 133120

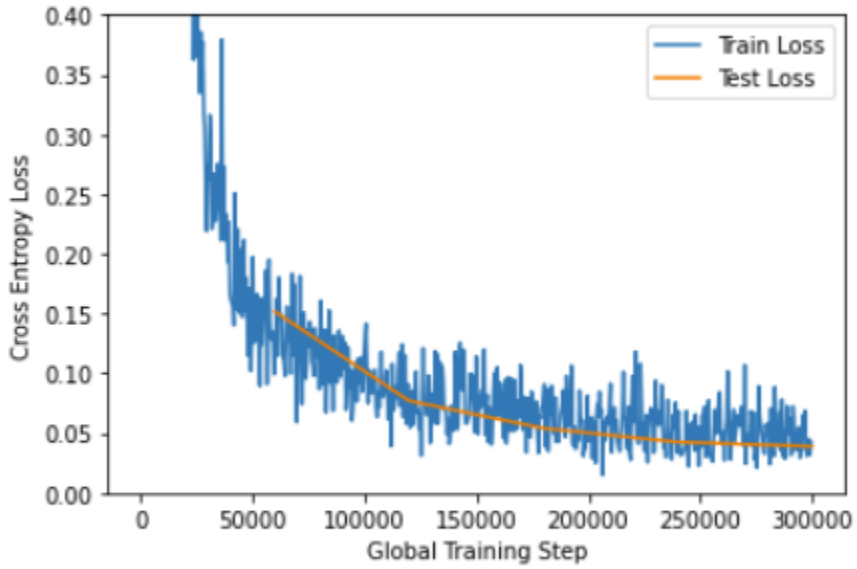
(ID = 10): Parameters in the FC layer;

((current layer c*previous layer p)+1*c) = ((10 * 64)+1*10) = 650

(ID = 11): The sum of all the parameters in the whole CNN is:
2432 + 18496 + 73856 + 133120 + 650 = 228554.

Task 2: Programming [1.6 points]

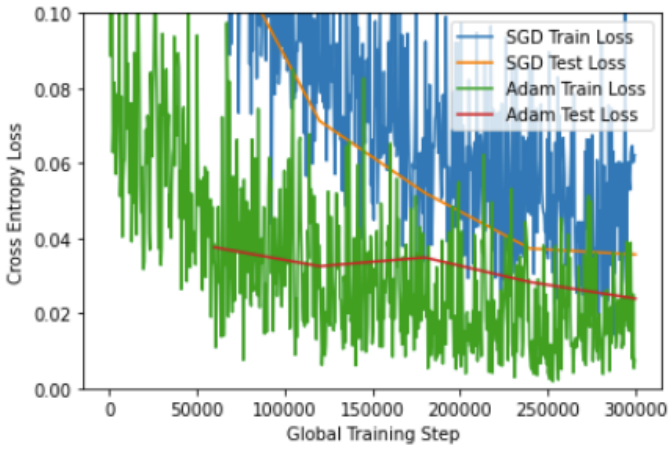
a) **By looking at the final train/validation loss/accuracy, do you see any evidence of overfitting? Shortly summarize your reasoning.**




```
Final Test loss: 0.03889294118163692. Final Test accuracy: 0.988
```

Overfitting is when the network over-interprets the train set, thereby achieving a better score on the train set, but also a worse score on the test set. By comparing the train to test loss, it is obvious that this is not the case since they have about the same loss.

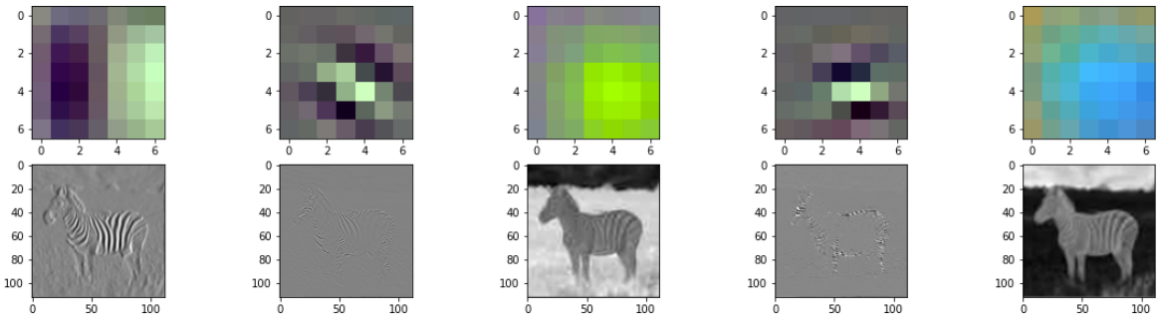
b) **Plot the training/validation loss from both models (the model with Adam and the one with SGD) in the same graph and include this in your report.**




```
SGD Final Test loss: 0.03565605740811065. SGD Final Test accuracy: 0.9881
Adam Final Test loss: 0.023904940627714096. Adam Final Test accuracy: 0.9933
```

We can see that the CNN with an Adam optimizer performs better, but it looks more prone to overfitting.

c) Run the image zebra.jpg through the first layer of the ResNet50 network. Visualize the filter, and the grayscale activation of 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]. Include the visualized filters and activations in your report.



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: The filter looks like it could be horizontal edge detection, and the corresponding activation image confirms this as it detects edge features as white on the right-hand side and black on the left hand-side.
Filter 8: The filter looks like some kind of sharpening with a high value in the middle, and surrounded by negative values, or maybe a diagonal edge detection. The activation image looks more like edge detection though.

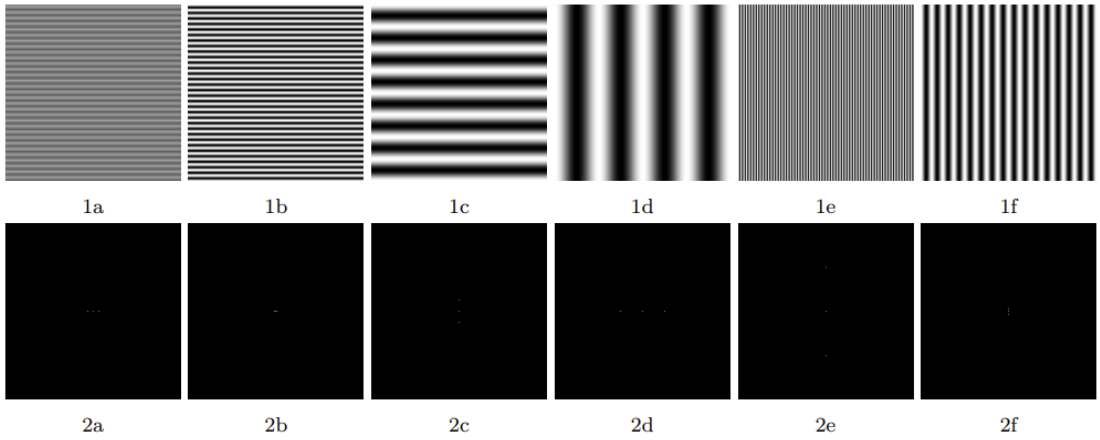
Filter 19 and Filter 34: The two kernels look very alike, and their corresponding activation image too. They seem to detect the object, and detect different levels of backgrounds. They are also the inverse of each other.
Filter 22: The kernel looks like some other kind of edge detection, maybe one of the two below. The corresponding activation image agrees with this.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

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

Task 3: Theory [0.5 points]

a) [0.3pt] 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.

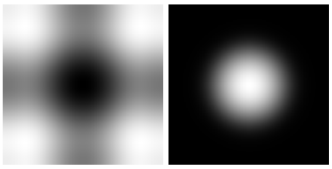


Since the “wave” is vertically oriented in image 1a, 1b and 1c, they correspond to 2c, 2e and 2f. 1c has the lowest frequency (slowest change), and therefore the corresponding fourier domain image is 2f (most centered). 1a has the highest frequency, and therefore corresponds to 2e (high frequency manifests in the outer domain), which leaves 1b too 2c.If we apply the same logic for the horizontal waves we get (1d, 2b), (1e, 2d) and (1f, 2a).

b) [0.05pt] What are high-pass and low-pass filters?

A high-pass filter allows higher frequencies to pass through the filter, and attenuates frequencies below a certain value. A low-pass filter allows lower frequencies to pass through the filter, and attenuates frequencies above a certain value.

c) **[0.15pt] The amplitude |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.**

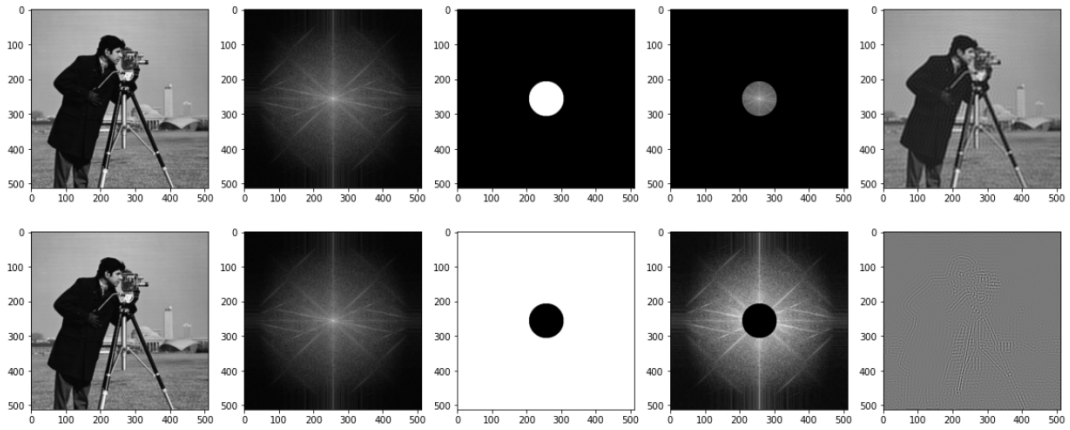


(a) (b)

The gaussian kernel b) has a “smoothing” effect. In other words, the change of intensity over the spatial domain is less, and the filter therefore acts as a low-pass filter. This also becomes obvious if we remember that high frequencies manifest themselves as white points at the outer edges of the fourier domain. Therefore a) is a sharpening filter only passing high frequencies (high-pass), as it is the opposite of b), and is white in the outer edges

Task 4: Programming [2 points]

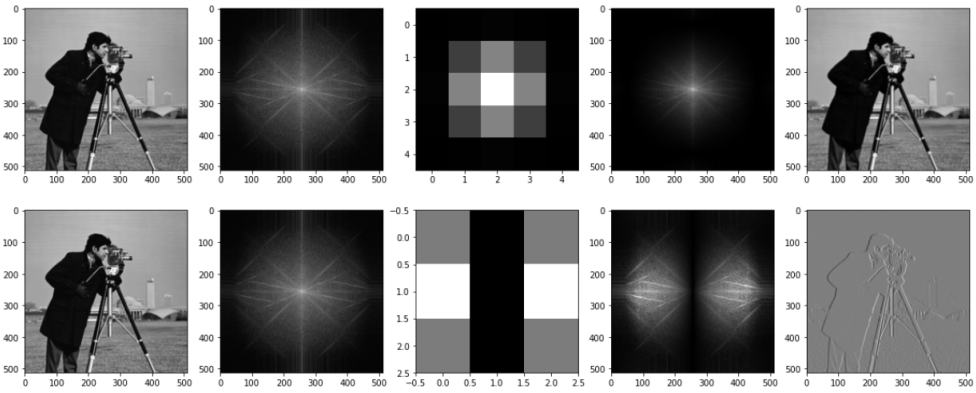
a) **[0.5pt] Implement a function that takes a grayscale image, and a kernel in the frequency domain, and applies the convolution theorem (seen in Equation 4). Try it out on a low-pass filter and a high-pass filter on the grayscale image ”camera man”(im = skimage.data.camera()). Include in your report the filtered images and the before/after amplitude |F{f}| of the transform. Make sure to shift the zero-frequency component to the center before displaying the amplitude. 8 Implement this in the function convolve_im in task4a.py/task4a.ipynb. The high-pass and low-pass filter is already defined in the starter code. You will observe a ”ringing” effect in the filtered image. What is the cause of this?**



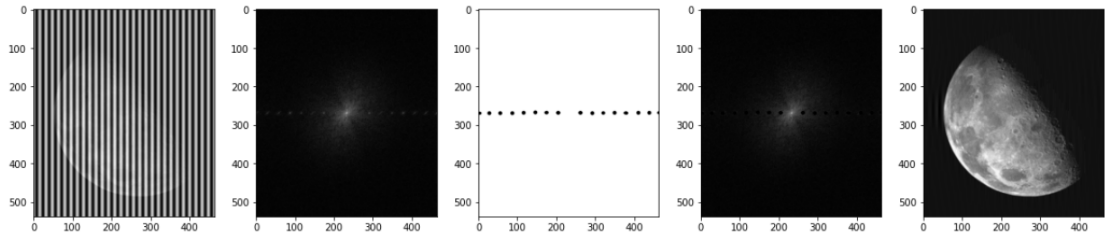
The ringing, also known as Gibbs phenomenon, happens when high frequency information is lost (low-pass), and the approximation of a discontinuous function

(like pictures which are made up of discrete values) overshoots the target. If we were to approximate a continuous function with infinitely many sin/cosine functions, this effect would go away.

b) **[0.2pt] 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 (Gx). Include in your report the filtered images and the before/after amplitude |F{f}| of the transform. Make sure to shift the zero-frequency component to the center before displaying the amplitude. Implement this in the function convolve_im in task4b.py/task4b.ipynb. The gaussian and sobel filter are already defined in the starter code.**



c) **[0.7pt] Use what you've learned from the lectures and the recommended resources to remove the noise in the image seen in Figure 5a. Note that the noise is a periodic signal. Also, the result you should expect can be seen in Figure 5b Include the filtered result in your report. Implement this in the file task4c.py/task4c.ipynb. Hint: Try to inspect the image in the frequency domain and see if you see any abnormal spikes that might be the noise.**



d) **[0.6pt] Now we will create a function to automatically find the rotation of scanned documents, such that we can align the text along the horizontal axis. You will use the frequency domain to extract a binary image which draws a rough line describing the rotation of each document. From this, we can use a hough transform to find a straight line intersecting most of the points in the binary image. When we have this line, we can easily find the rotation of the line and the document. Your task is to generate the binary image by using the frequency spectrum. See Figure 6 which shows you what to expect. We've implemented most of the code for you in this task; you only need to alter the function create_binary_image in task4d.py/task4.ipynb. Include the generated image in your report (similar to Figure 6). Hint: You can use a thresholding function to threshold the**

magnitude of the frequency domain to find your binary image.