TDT4195 - IP Assignment 2

Sigurd Totland — MTTK

November 10, 2019

Task 1 CNN Theory

a)

From equation (1) in the assignment,

$$W_2 = \frac{W_1 - F_W + 2P_W}{S_W} + 1, \tag{1}$$

we get that the padding P_W should be

$$P_W = \frac{1}{2}(S_W W_2 - W_1 + F_W - S_W). \tag{2}$$

Since we want the output image to have equal dimensions to the input, we require $W_2 = W_1$. Furthermore, we are given stride $S_W = 1$ and filter width $F_W = 5$, which yields

$$P_W = \frac{1}{2}(W_1 - W_1 + 5 - 1) = 2. \tag{3}$$

Since the filter is square, the horisontal and vertical padding is identical.

b)

We rewrite (1) to

$$F_W = W_1 + S_W + P_W - S_W W_2 \tag{4}$$

Inserting the numbers we get

$$F_W = 512 + 1 + 0 - 1 \cdot 504 = 9. \tag{5}$$

Again, since the input image is square, kernel height and width are the same so the dimensions are 9-by-9.

c)

Since the feature maps are of dimension 504x504 and subsampling is done with a 2x2 neighborhood and stride 2, this resulution is effectively halved, so the resulting pooled feature maps will be of size 252x252.

d)

The number of pooled feature maps must match the corresponding number of feature maps, so 12.

e)

This time, plugging in $F_W = 3$ and $W_1 = 252$ into (1), we get

$$W_2 = 252 - 3 + 1 = 250, \tag{6}$$

i.e. 250x250.

f)

The convolutional layers will have a total of

$$\begin{aligned} n_{1,2,3} &= 3 \cdot 5 \cdot 5 \cdot 32 + 32 \\ &\quad + 32 \cdot 5 \cdot 5 \cdot 64 + 64 \\ &\quad + 64 \cdot 5 \cdot 5 \cdot 128 + 128 \\ &= 258624 \end{aligned} \tag{7}$$

parameters to them. After these three layers, the image will have been max-pooled 3 times, bringing the resolution down to $32/2^3 = 4$. This means the flattening operation will create a vector of size $4 \cdot 4 \cdot 128 = 2048$. Consequently, the final two fully-connected layers will have a total number of

$$\begin{aligned} n_{4,5} &= 2048 \cdot 64 + 1 \\ &\quad + 64 \cdot 10 + 1 \\ &= 131714 \end{aligned} \tag{8}$$

parameters. In total then, the CNN will have $n = n_{1,2,3} + n_{4,5} = 390338$ parameters.

Task 2 CNN Programming

a) Maxpooling

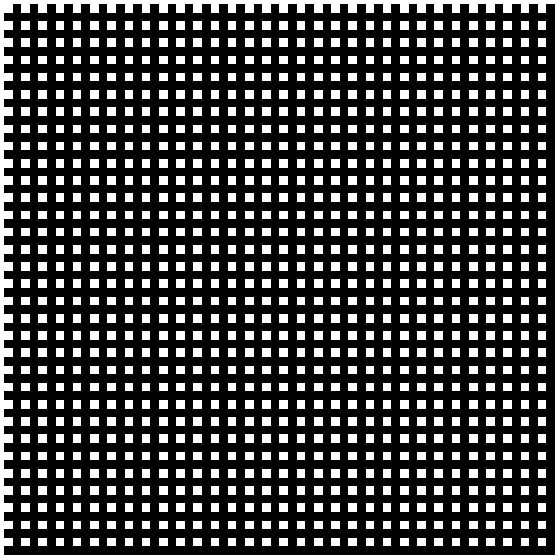
When maxpooling is applied to Chelsea, we get the result in figure 1.

(a) Original Chelsea

(b) Post-maxpool Chelsea

Figure 1: Chelsea before and after maxpooling

The checkerboard has 2x2 checkers, meaning that the maxpooled image is simply a white square, as shown in figure 2.



(a) Original board

(b) Post-maxpool board

Figure 2: Chelsea before and after maxpooling

b) CNN MNIST Classifier

Implementing the CNN form Table 1 in the assignment and training it on the MNIST dataset yields final training loss of 0.044 and 99% accuracy(!). The plot of loss is shown in figure 3 below.

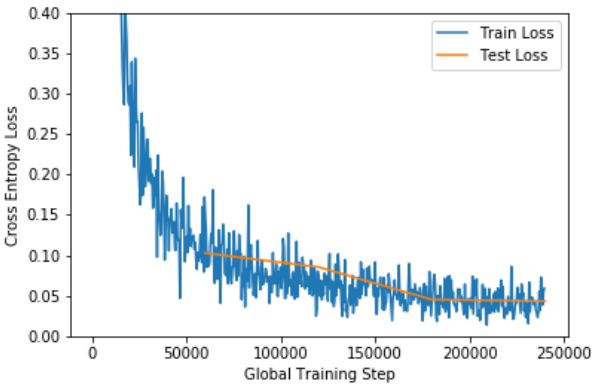


Figure 3: Training loss of 4 epochs

From figure 3, we do not see evidence of overfitting with these hyperparameters, as the test loss and final test accuracy corresponds well with the training loss. Had the test loss been significantly worse than the training loss however, one would suspect the network to be overfitted to the training data.

c)

When trained for 10 epochs instead of two, the training loss develops like shown in figure 4 below.

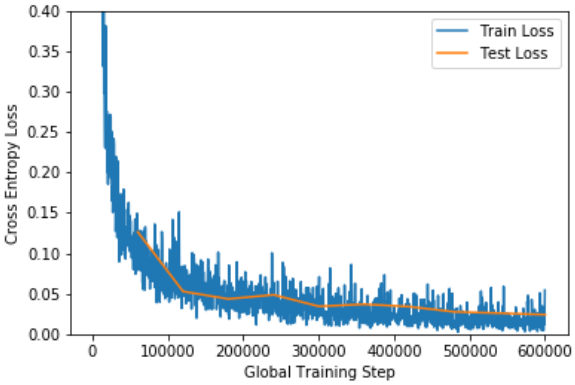


Figure 4: Training loss of 10 epochs

This time, the final validation accuracy is 99.3% and the final validation loss is 0.022 and there is seemingly no evidence of overfitting form the loss plot. However, by calculating the final training loss and accuracy in addition to the

validation, we find that it is 0.010 and 99.7%, respectively. This means we do see slightly better performance on the training set than on the test set, meaning the network has indeed been slightly overfittet to the training data!

d)

We plot the weights numbered 5, 8, 19, 22, and 34 in the first convolutional layer of ResNet50 along with the corresponding activation on the input zebra image. These results are shown in 5 below.

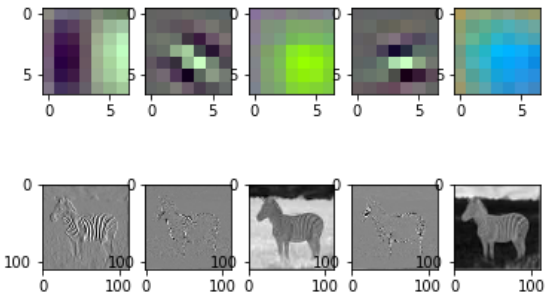


Figure 5: ResNet50 weights and corresponding activation on zebra image

e)

We see, as expected, how the filters more strongly activate those parts of the image that they are supposed to. The first filter for instance, resembles a simple edge detection filter although widened a bit, and indeed it brings out the stripes of the zebra the most, in addition to applying a light edge detection on the entire zebra. The second filter looks like a sobel as well, but it is rotated. Note how it does not bring out the zebra stripes by a lot, but it does apply some edge detection, especially on the edges of the zebra with similar orientation to it. The third filter is green and smooth, and seems to bring out the smooth grass background. For the fourth filter we have yet another, this time horisontal, edge detector. Notice how it does not activate on the vertical stripes of the zebra at all, and activates almost exclusively in the areas where the stripes are horisontal. Lastly, we have an almost entirely smooth, blue filter, which unsurprisingly brings out the smooth, blue sky.

Task 3 Frequency Domain Filtering Theory

a)

To perform the convolution $f * g$ using the convolution theorem, we

1. take the Fourier transform of f and g,
2. multiply the transformed functions, obtaining $\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$,
3. inverse transform the result, obtaining $f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f * g\}\}$.

b)

High pass filters damp the lowest frequencies of a function and preserves high-frequency components. Low pass filters do the opposite and damp the highest frequencies, such as noise, while preserving slow dynamics.

c)

The filter in a) corresponds to a high pass filter, but only in the vertical direction. In the horisontal, it damps all frequencies, also high ones. The filter in b) is a high pass filter – the center (low frequencies) are damped and the outer region is preserved. The final filter in c) is a low pass filter as only center frequencies (low frequencies) are preserved.

Task 4 Frequency Domain Programming

a)

We implement kernel convolution via the frequency domain using the convolution theorem. Applying a high pass filter on the `skimage.data.camera()` camera man image, we get the results shown in figure 6.

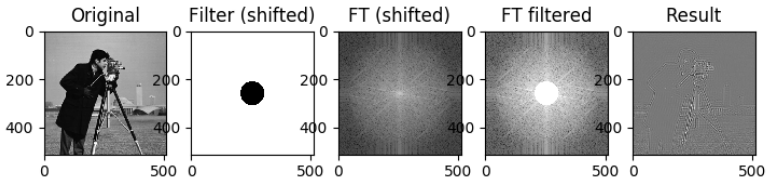


Figure 6: Camera man high pass filtered

Applying a low pass filter on the same image results in the results in figure 7 below.

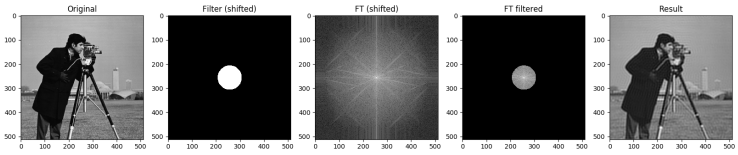
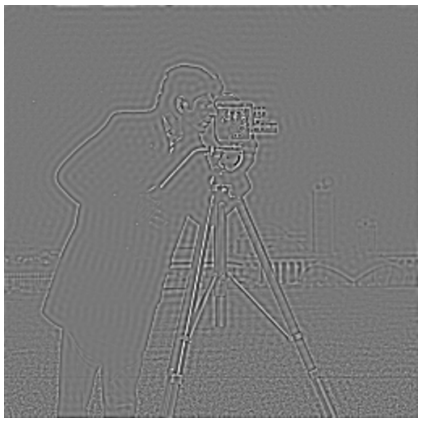


Figure 7: Camera man low pass filtered

Higher resolution versions of the final outputs are shown in figure 8 below.

(a) Low pass filtered camera man



(b) High pass filtered camera man

Figure 8: High pass and low pass filtered camera man

b)

In this task we implement kernel convolution through the frequency plane. We must then pad the input kernel in the bottom left corner to match the dimensions of the input image before taking the Fourier transform. With a gaussian kernel, we get the results shown in figure 9 below.

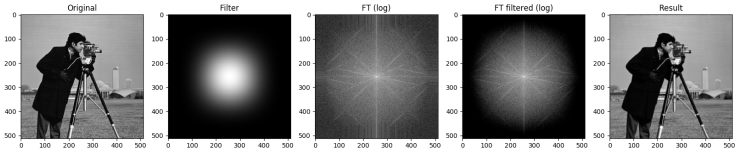


Figure 9: Camera man gaussian blurred in frequency domain

Applying a 3-by-3 sobel in the x-direction with the same methods yields the results in figure 10.

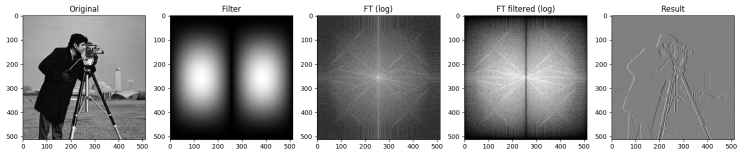


Figure 10: Camera man with sobel filter applied in the frequency domain

Larger scale versions of the two processed images are shown in figure 11.

(a) Sobel result (b) Gaussian blur result

Figure 11: Images processed in the frequency domain

c)

We apply the notch filter to the clown image. The results of this can be seen in figure 12.

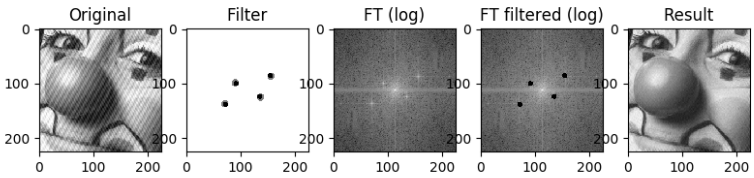


Figure 12: Clown filter results

Full size before and after images for the clown image, which clearly shows the periodic pattern present in the original image, and that it is removed almost

entirely (except around the edges of the image) in the filtered image are shown in figure 13.

(a) Original (b) Notch filtered

Figure 13: Clown image before and after notch filtering

The filter used here is an example of a notch filter. A notch filter damps a specific frequency range in the signal, which makes it the ideal tool to remove unwanted periodic patterns in an image, such as the pattern overlayed on the clown image. Looking at the Fourier transform of the input image, we can see four bright spots which correspond to the unwanted pattern. A notch filter can then be specifically crafted to eliminate the pattern from the image.

d)

We apply the sharpening technique as given in the assignment and obtain the result shown in figure 14.

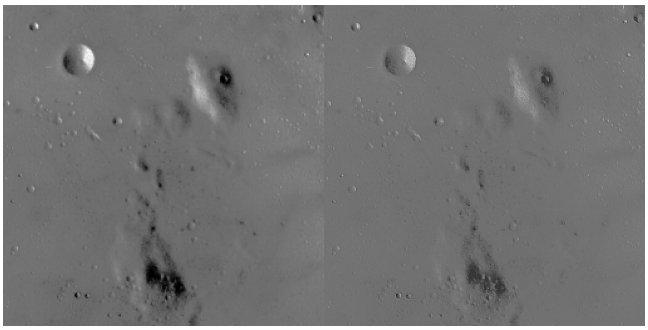


Figure 14: Moon (original to the left, sharpened to the right)

As we see, the contrast is worsened by adding the high pass filtered version

of the image on top of the original. To better see the sharpening, we can apply histogram equalization before and after the sharpening operator, producing the results shown in figure 15. This more clearly showns that the image has been sharpened.

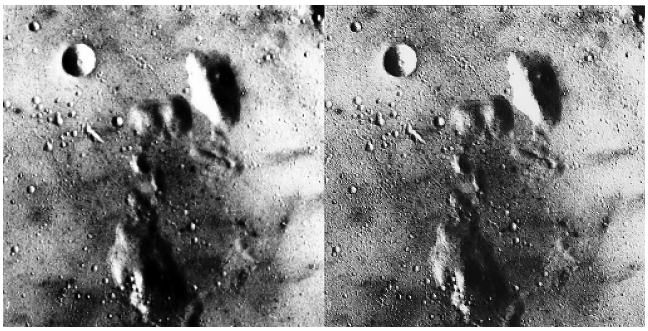


Figure 15: Moon sharpened and equalized