TDT4195 - Visual Computing Fundamentals

Image Processing - Assignment 2

Author:
Martin Skatvedt

8.11.2022

1 Convolutional Neural Networks

Task 1: Theory

a)

Formulas for W_2 and H_2

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

Putting in parameters

$$W_2 = W_1 - 5 + 2P_W + 1$$
$$H_2 = H_1 - 5 + 2P_H + 1$$

By looking at the equations over we need

$$2P_W - 4 = 0$$
$$2P_H - 4 = 0$$

To have $W_1 = W_2$ and $H_1 = H_2$ padding should be equal to

$$P_W = 2 \text{ and } P_H = 2$$

b)

$$W_1 = 512, H_1 = 512, W_2 = 504, H_2 = 504, P_W = 0, P_H = 0, S_W = 1, S_H = 1$$

Changing formulas to solve for F_W and F_H

$$F_W = W_1 + 2P_W + S_W - W_2 S_W$$
$$F_H = H_1 + 2P_H + S_H - H_2 S_H$$

$$F_W = 512 + 2 * 0 + 1 - 504 * 1 = 9$$
$$F_H = 512 + 2 * 0 + 1 - 504 * 1 = 9$$

$$F_H x F_W = 9x9$$

c)

$$W_1 = 504, H_1 = 504, F_W = 2, F_H = 2, P_W = 0, P_H = 0, S_W = 2, S_H = 2$$

Using equation 1 and 2

$$W_2 = \frac{W_1 - F_W + 2P_w}{S_W} + 1 = \frac{504 - 2 + 2 * 0}{2} + 1 = 252$$

$$H_2 = \frac{H_1 - F_H + 2P_H}{S_H} + 1 = \frac{504 - 2 + 2 * 0}{2} + 1 = 252$$

$$H_2 x W_2 = 252x252$$

d)

$$W_1 = 252, H_1 = 252, F_W = 3, F_H = 3, P_W = 0, P_H = 0, S_W = 1, S_H = 1$$

$$W_2 = \frac{W_1 - F_W + 2P_w}{S_W} + 1 = \frac{252 - 3 + 2 * 0}{1} + 1 = 250$$
$$H_2 = \frac{H_1 - F_H + 2P_H}{S_H} + 1 = \frac{252 - 3 + 2 * 0}{1} + 1 = 250$$

$$H_2 x W_2 = 250x250$$

e)

Layer 1:

Weights: $F_H x F_W x C_1 x C_2 = 5x5x3x32 = 2400$

Bias: $C_2 = 32$

Layer 2:

Weights: $F_H x F_W x C_1 x C_2 = 3x3x32x64 = 18432$

Bias: $C_2 = 64$

Layer 3:

Weights: $F_H x F_W x C_1 x C_2 = 3x3x64x128 = 73728$

Bias: $C_2 = 128$

Layer 4:

Weights: 64

Bias: 64

Layer 5:

Weights: 10

Bias: 10

Total weights = 94634

Total bias = 298

Total parameters: $94634 + 298 = 94932$

Task 2: Programming

a)

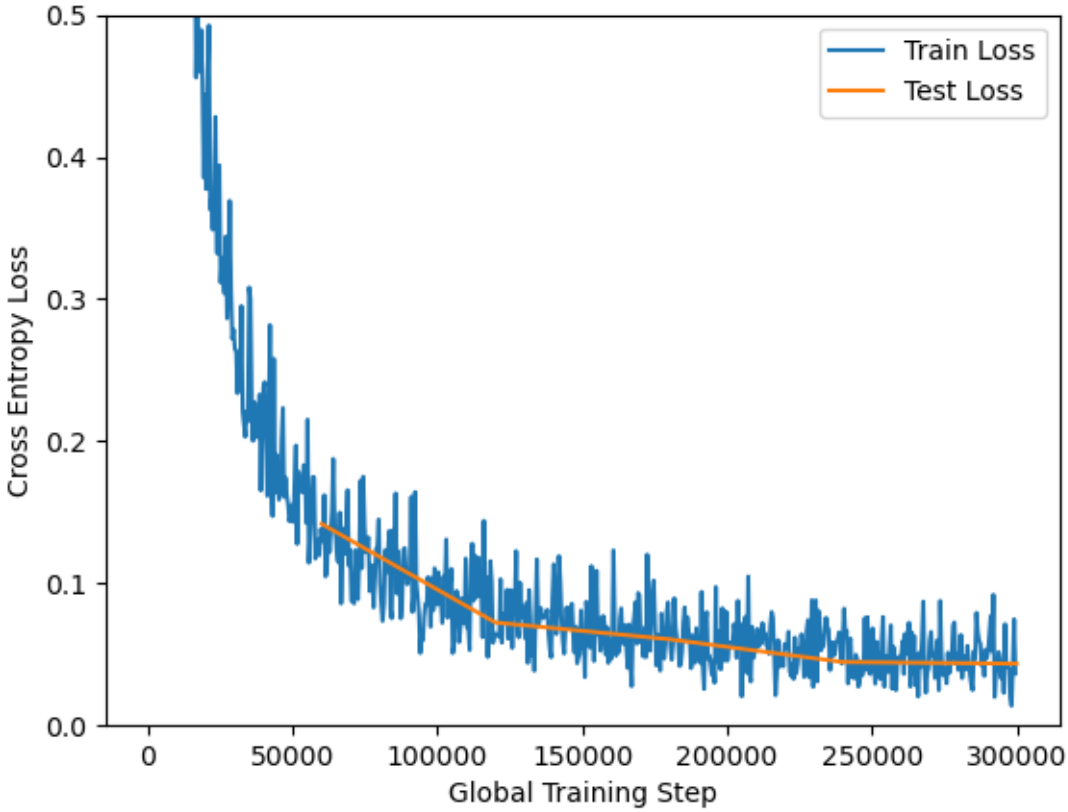

Figure 1: Train and test loss of network from *table 1*

Final Test loss: 0.04348754433779325. Final Test accuracy: 0.9858

I do not see any signs of overfitting. The test loss follows the *center* of the train loss curve evenly. It does not jump up or down to follow the train loss. The only thing that could make the model seem overfitted, is the high accuracy and low final loss.

b)

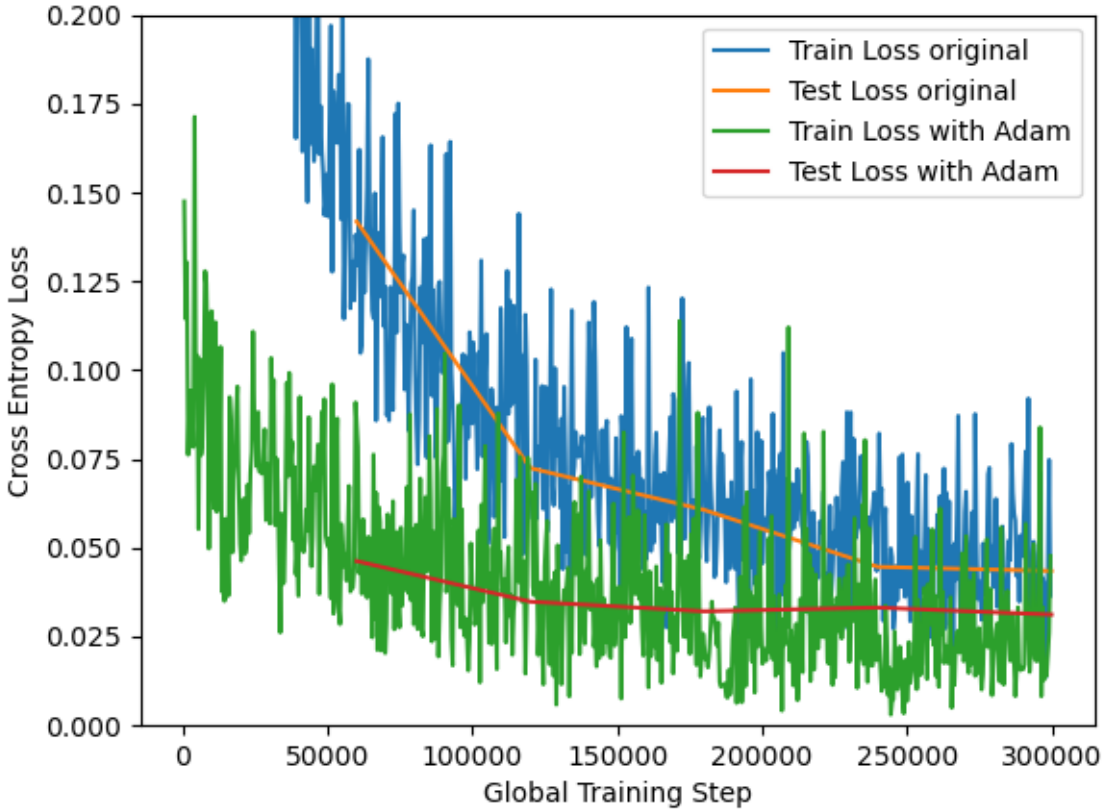

Figure 2: Train and test loss from both original model, and model which uses Adam optimizer

c)

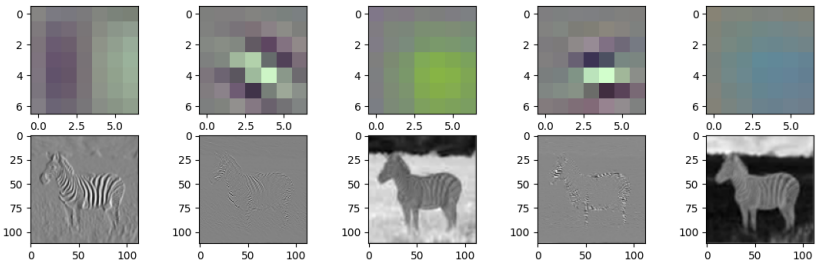

Figure 3: Figure of filters and its activation filter:[5,8,19,22,34]

d)

Filter 5: Vertical lines with dark on the left side and light on the right side

Filter 8: Diagonal light lines from top left to bottom right

Filter 19: Exctracts dark pixels pixels

Filter 22: Almost the same as filter 8, except it extracts shorter lines

Filter 34: Extracts light pixels

2 Filtering in the Frequency Domain

Task 3: Theory

a)

- 1a and 2f

- 1b and 2c

- 1c and 2e

- 1d and 2d

- 1e and 2b

- 1f and 2a

b)

A low pass filter is a blurring or smoothing filter, which will average out big changes in pixel intensity. This can reduce noise in images.

A high pass filter is a sharpering filter, which will boost the pixel intensity, if it is brighter than its neighbours. This will make the image sharper, but also amplify the noise.

c)

A = High pass filter: We can see that that the image has a big *dot* for its low frequencies, and also quite big *dots* for its high frequencies, it looks like there is a big variance in its frequencies. Which makes it looks like it has been sharpened

B = Low pass filter: - Since the low frequencies are in the middle, and high frequencies can be seen far from the center. The image has a small variance for the high frequencies, which makes it look like it has been blurred.

Task 4: Programming

a)

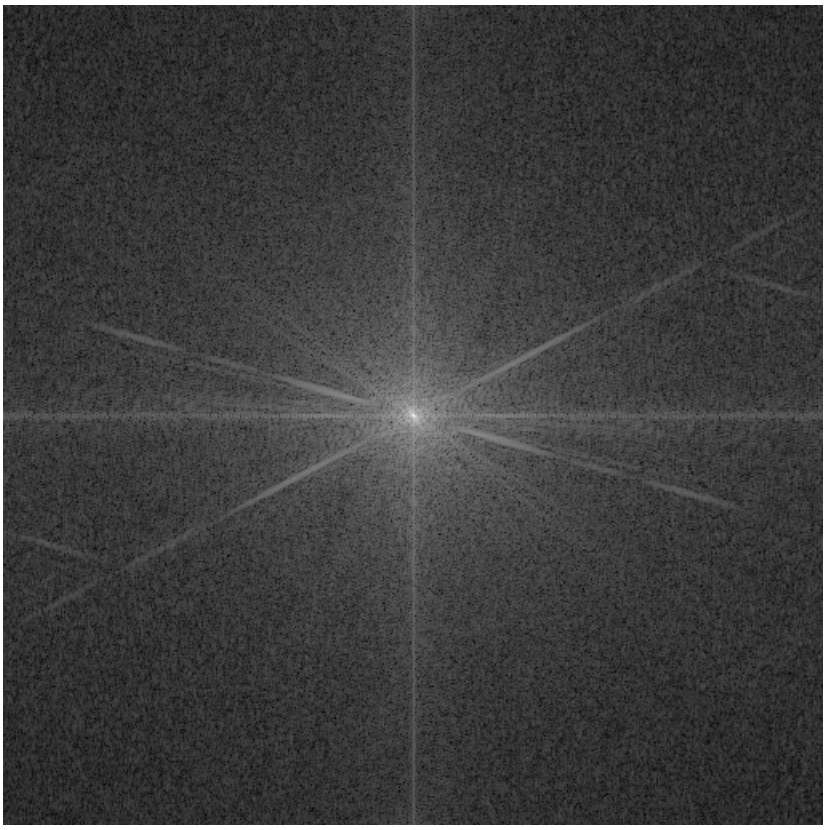

Figure 4: Amplitude before low pass

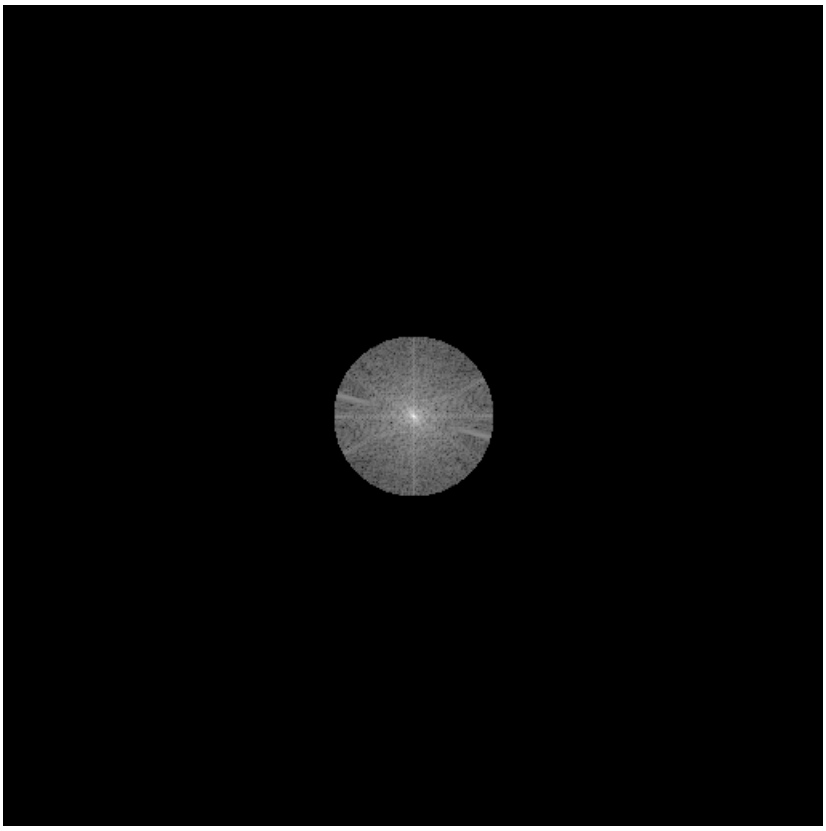

Figure 5: Amplitude after low pass

Figure 6: Result after low pass

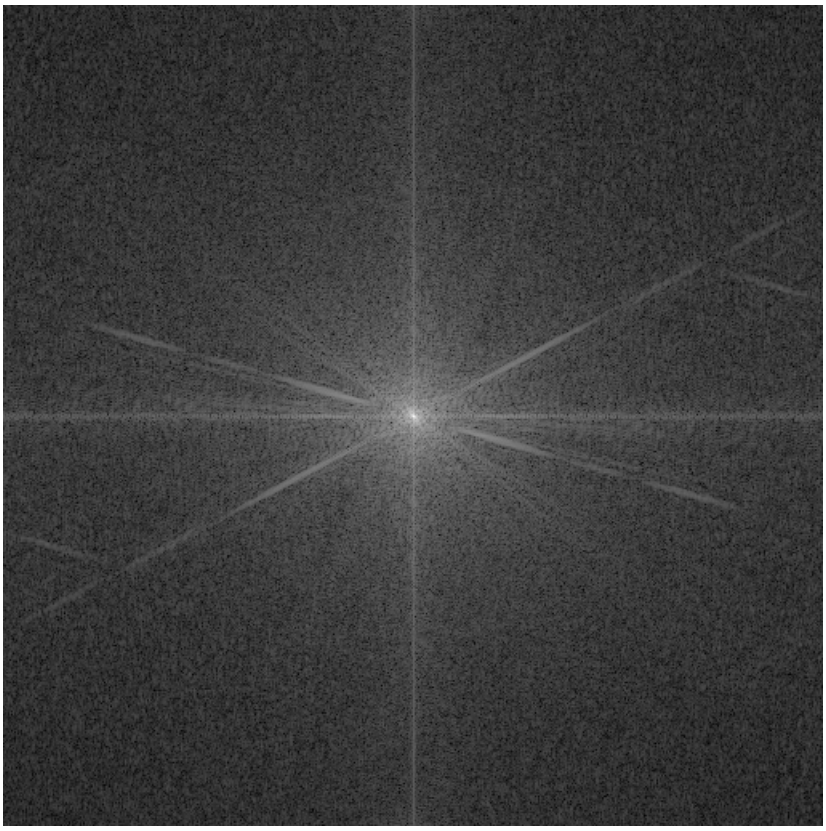

Figure 7: Amplitude before high pass

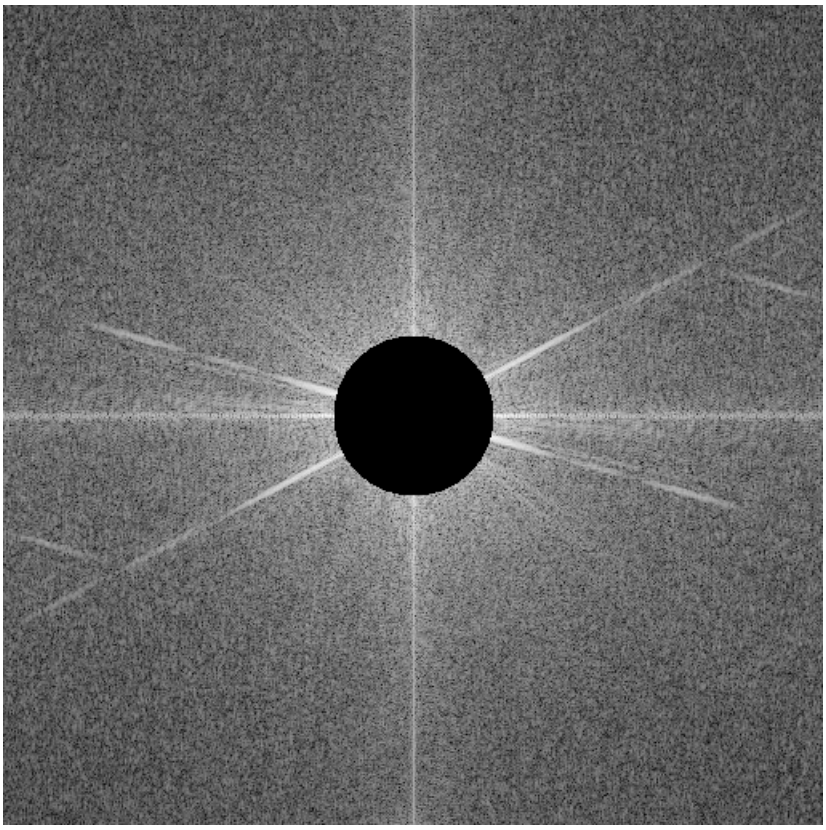

Figure 8: Amplitude after high pass

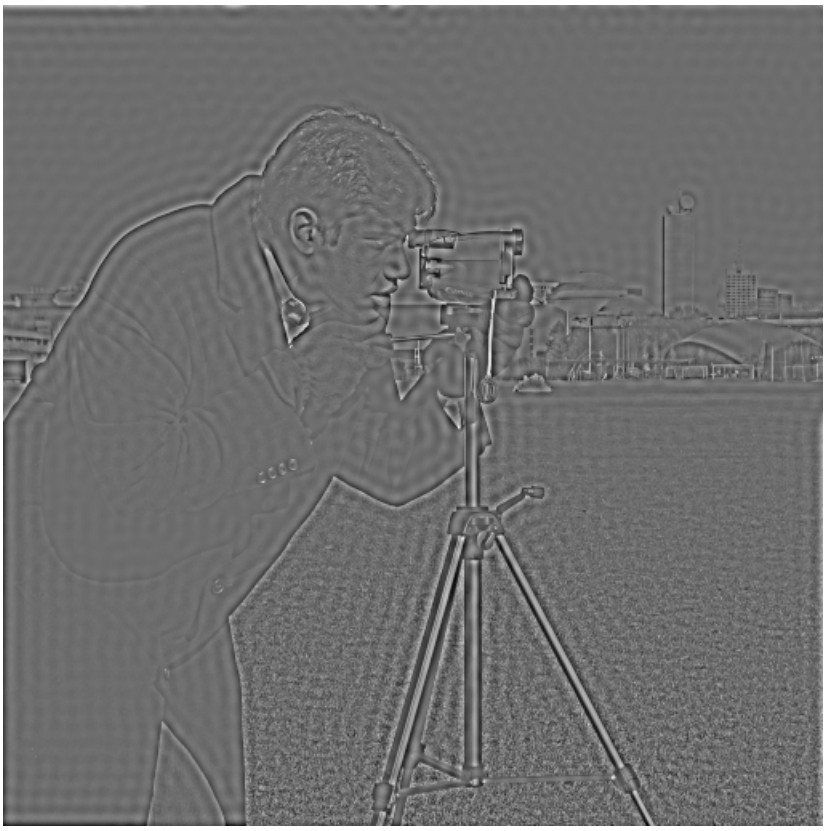

Figure 9: Result after high pass

b)

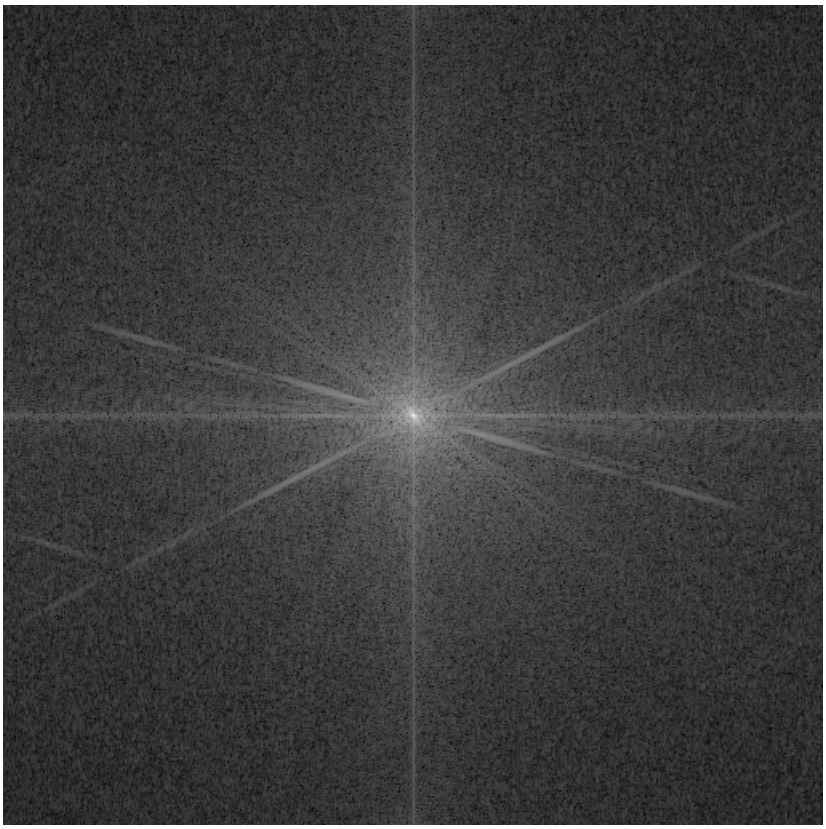

Figure 10: Amplitude before gaussian

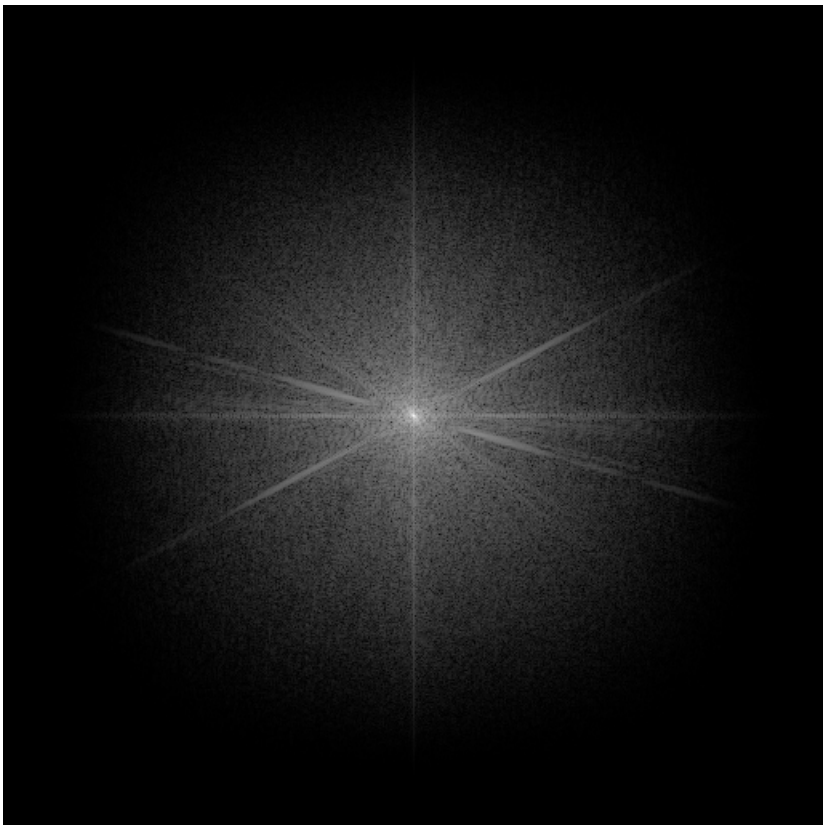

Figure 11: Amplitude after gaussian

Figure 12: Result after gaussian

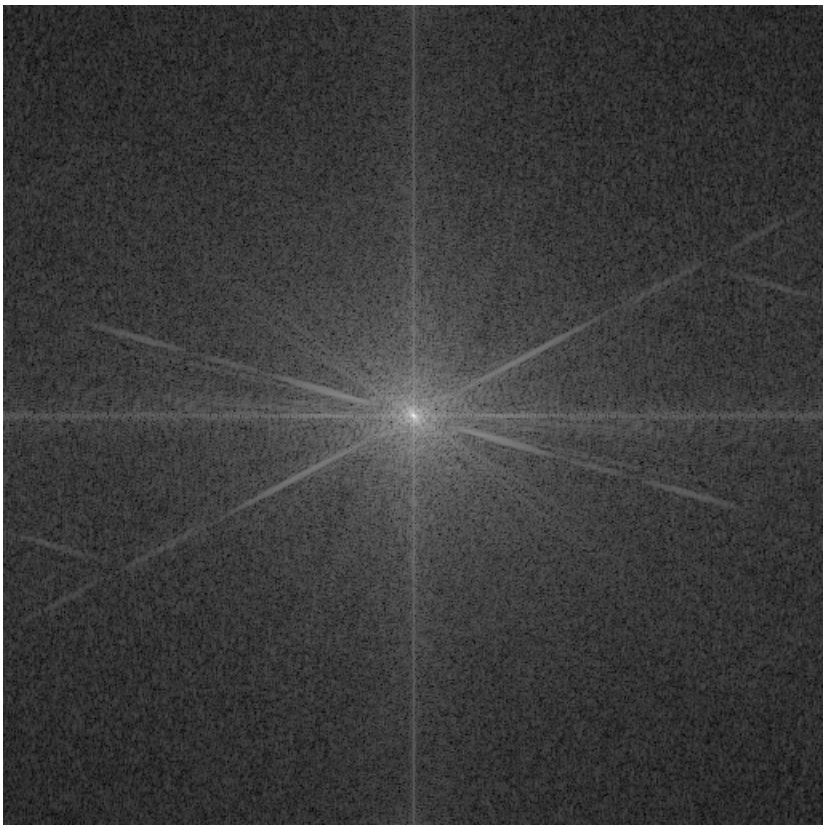

Figure 13: Amplitude before sobelx

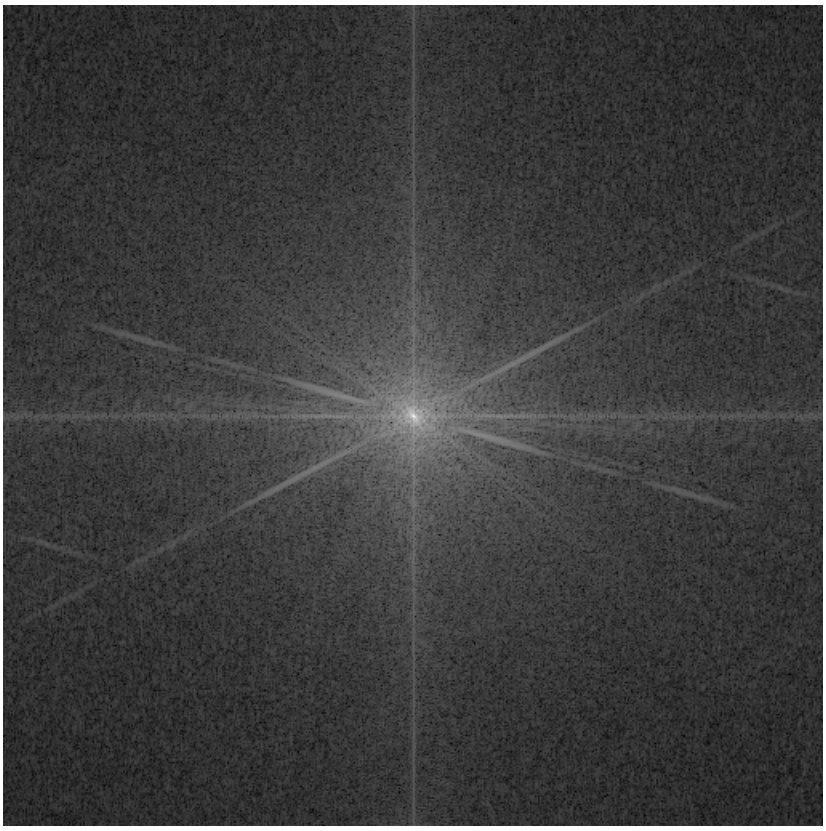

Figure 14: Amplitude after sobelx

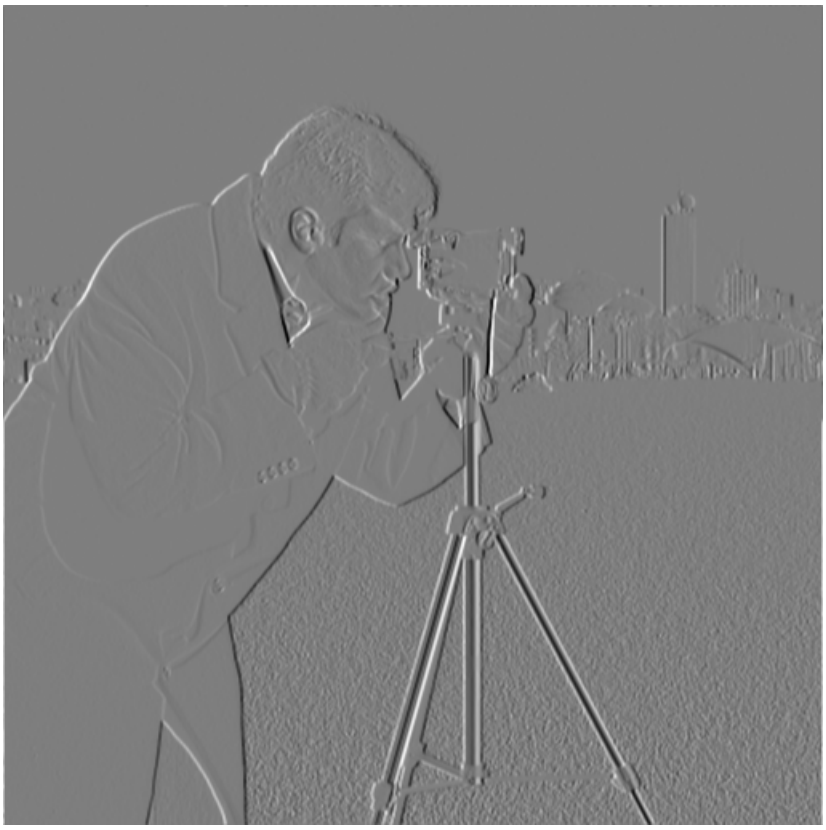

Figure 15: Result after sobelx

c)

Figure 16: Moon after filtering

d)

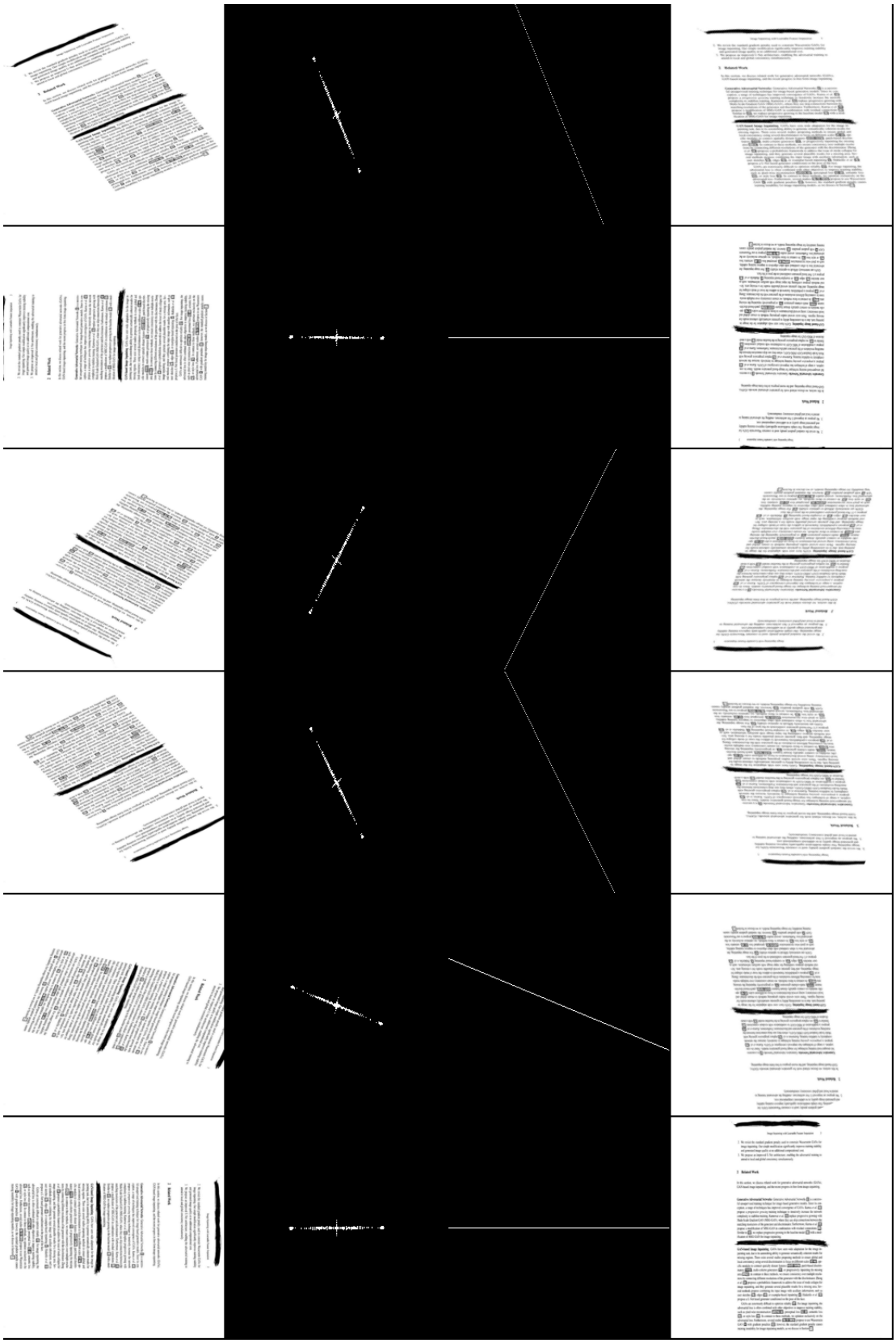

Figure 17: Produced image after finding binary images