TDT4195 Assignment 1

Olof Ljunggren

TDT4195 Image Processing Assignment 1

Task 1: Theory, Spatial Filtering [2.4pt]

Task 1.a

[0.2pt] Explain in one sentence what sampling is.

- In image processing sampling is the spatial discreteization. Samplerate corresponds to pixelsize.

Task 1.b

[0.2pt] Explain in one sentence what quantization is.

- Quantization is the discreteization of an image intensity. Corresponds to the specific intensity a pixel gets.

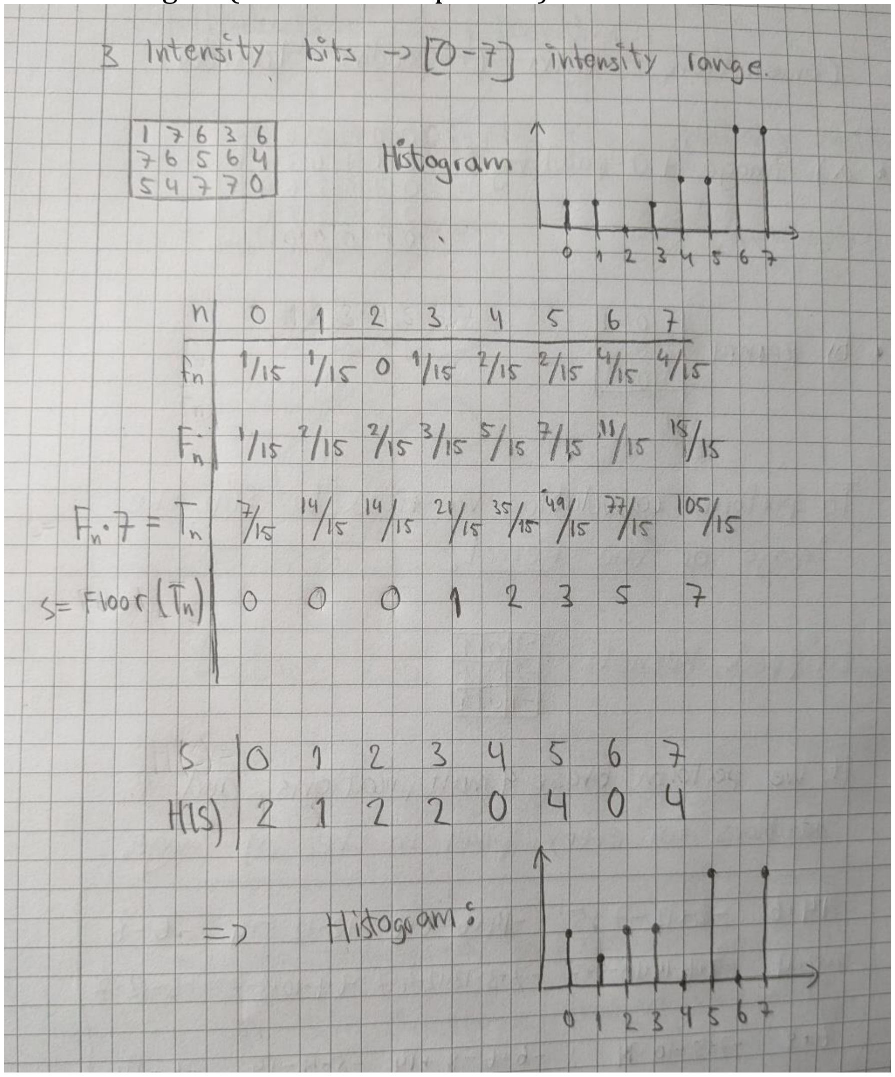
Task 1.c

[0.4pt] Looking at an image histogram, how can you see that the image has high contrast?

- If there is a large diffrence between high and mid and low intensities. Which for a histogram means there is a lot of high and low intensity pixels and not very many in the grey zone inbetween. Should have a little bit of a u-shape.

Task 1.d

[0.6pt] Perform histogram equalization by hand on the 3-bit (8 intensity levels) image in Figure 1a Your report must include all the steps you did to compute the histogram, the transformation, and the transformed image. Round down any resulting pixel intensities that are not integers (use the floor operator).



3 Intensity bits → [0-7] intensity range.

1	7	6	3	6
7	6	5	6	4
5	4	7	7	0

Histogram

0 1 2 3 4 5 6 7

n	0	1	2	3	4	5	6	7
f_n	$1/15$	$1/15$	0	$1/15$	$2/15$	$2/15$	$4/15$	$4/15$
F_n	$1/15$	$2/15$	$2/15$	$3/15$	$5/15$	$7/15$	$11/15$	$15/15$
$F_n \cdot 7 = T_n$	$7/15$	$14/15$	$14/15$	$21/15$	$35/15$	$49/15$	$77/15$	$105/15$
$s = \text{Floor}(T_n)$	0	0	0	1	2	3	5	7

S	0	1	2	3	4	5	6	7
$H(S)$	2	1	2	2	0	4	0	4

⇒ Histogram:

0 1 2 3 4 5 6 7

Histogram.

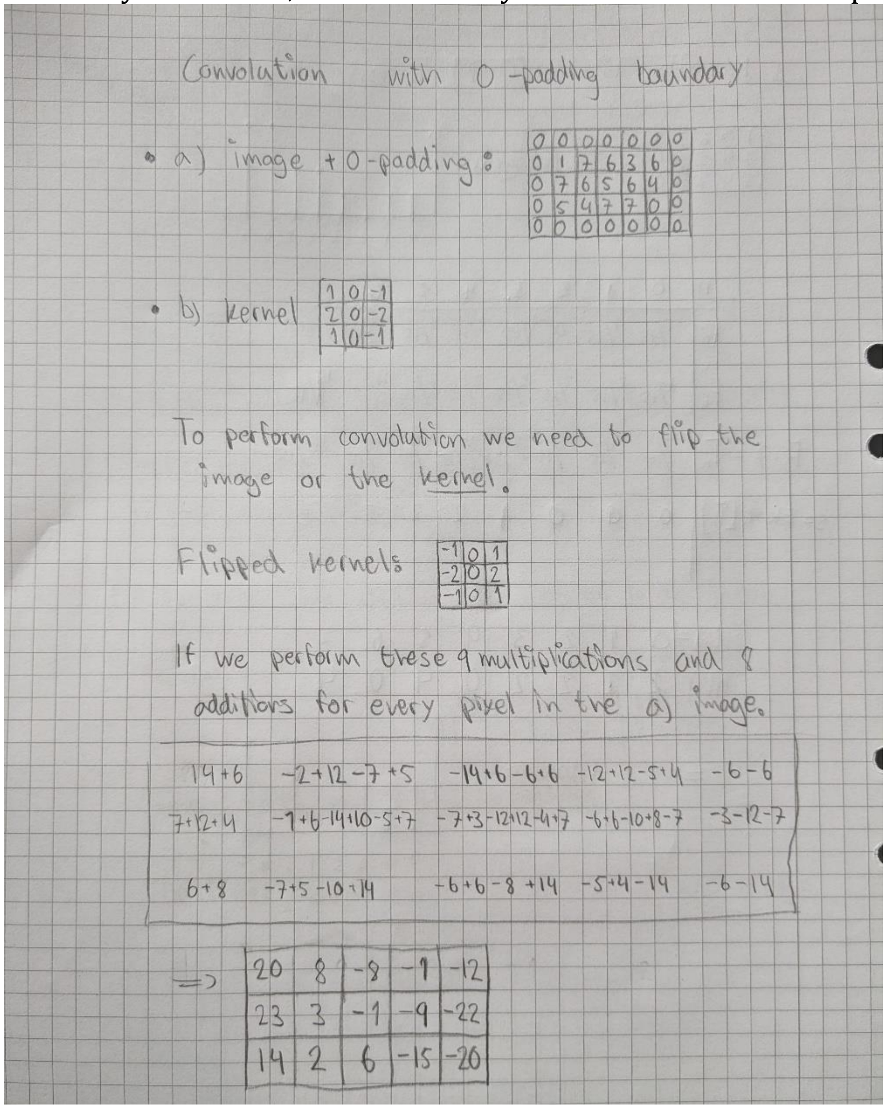
Task 1.e

[0.2pt] What happens to the dynamic range if we apply a log transform to an image with a large variance in pixel intensities?

- The log-transformation compresses the dynamic range. We get more detailed low intensities but less detailed high intensities.

Task 1.f

[0.8pt] Perform spatial convolution by hand on the image in Figure 1a using the kernel in Figure 1b. The convolved image should be 3 ×5. You are free to choose how you handle boundary conditions, and state how you handle them in the report.



Convolution with 0-padding boundary

- a) image + 0-padding:

0	0	0	0	0	0	0
0	1	7	6	3	6	0
0	7	6	5	6	4	0
0	5	4	7	7	0	0
0	0	0	0	0	0	0

- b) kernel

1	0	-1
2	0	-2
1	0	-1

To perform convolution we need to flip the image or the kernel.

Flipped kernel:

-1	0	1
-2	0	2
-1	0	1

If we perform these 9 multiplications and 8 additions for every pixel in the a) image.

14+6	-2+12-7+5	-14+6-6+6	-12+12-5+4	-6-6
7+12+4	-1+6-14+10-5+7	-7+3-12+12-4+7	-6+6-10+8-7	-3-12-7
6+8	-7+5-10+14	-6+6-8+14	-5+4-14	-6-14

$\Rightarrow$

20	8	-8	-1	-12
23	3	-1	-9	-22
14	2	6	-15	-20

Convolution.

The boundaries was handled by padding with zeros equal to the floor(kernel_size/2). This makes that the image has the same size. Though we might lose some information in the edges.

Task 2: Programming [1.6pt]

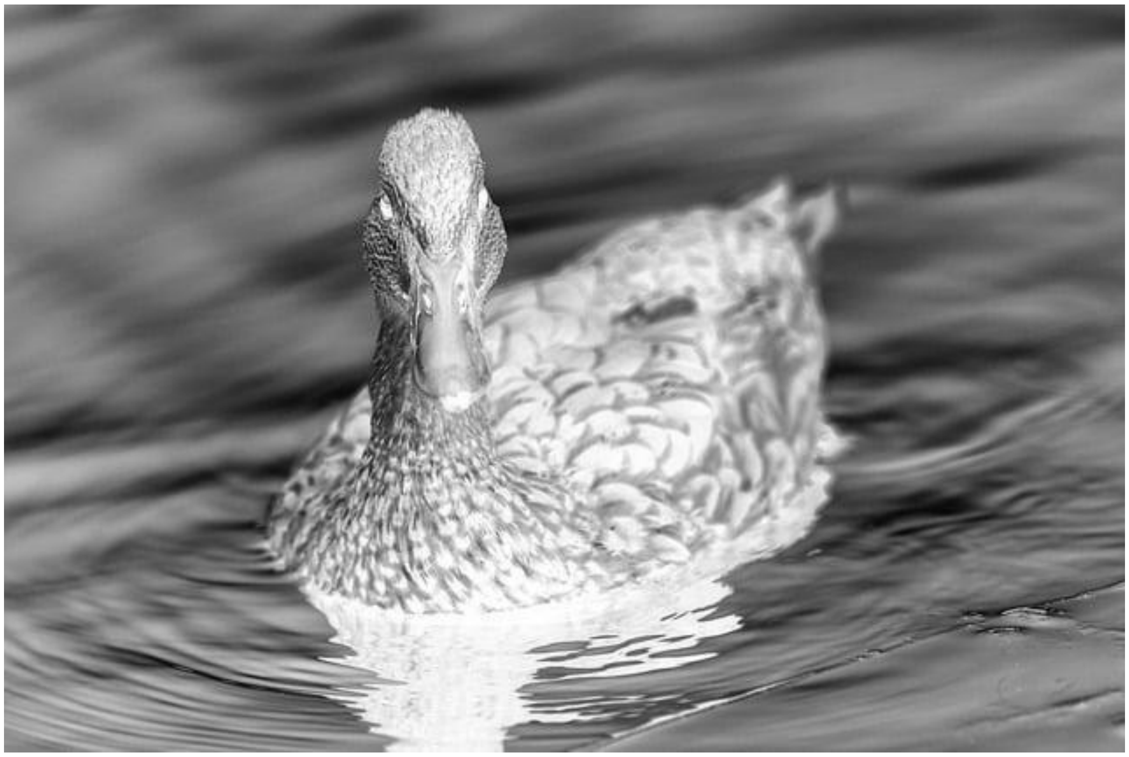
Task 2.a

[0.3pt] Implement a function that converts an RGB image to grayscale. Use Equation 1. Implement this in the function greyscale.

Greyscale duck.

Task 2.b

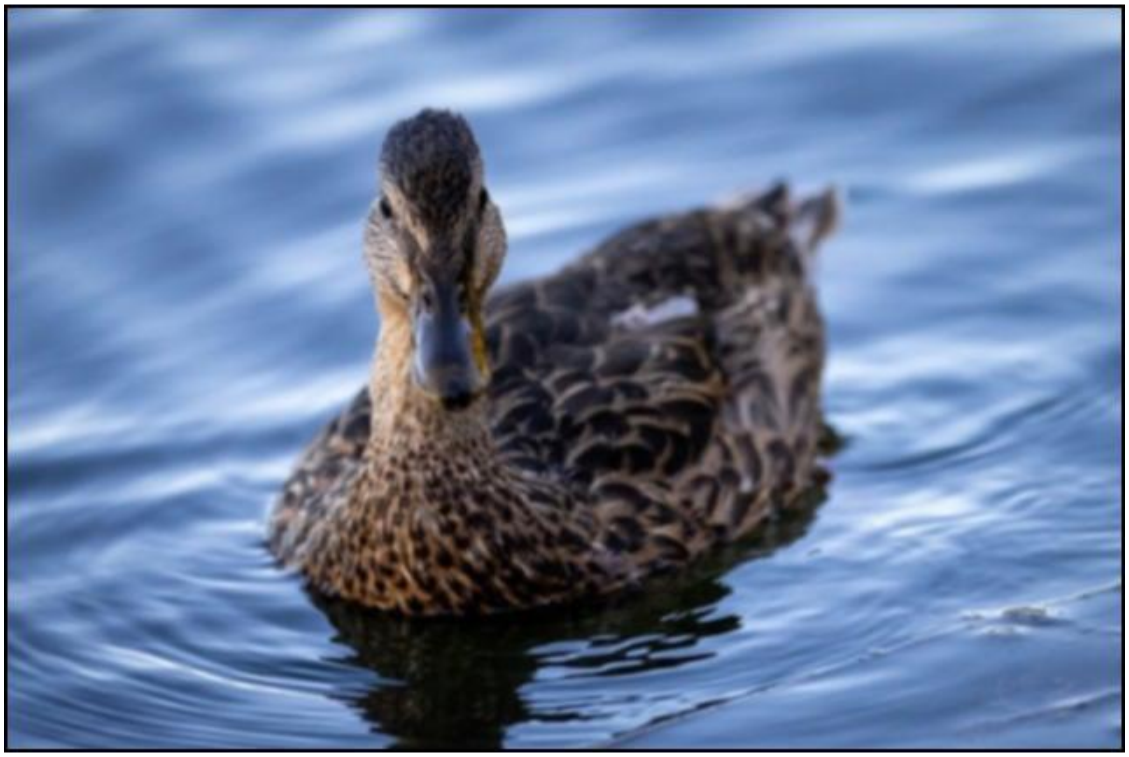
[0.4pt] Implement a function that takes a grayscale image and applies the following intensity transformation T(p) = 1 − p. Implement this in the function inverse.



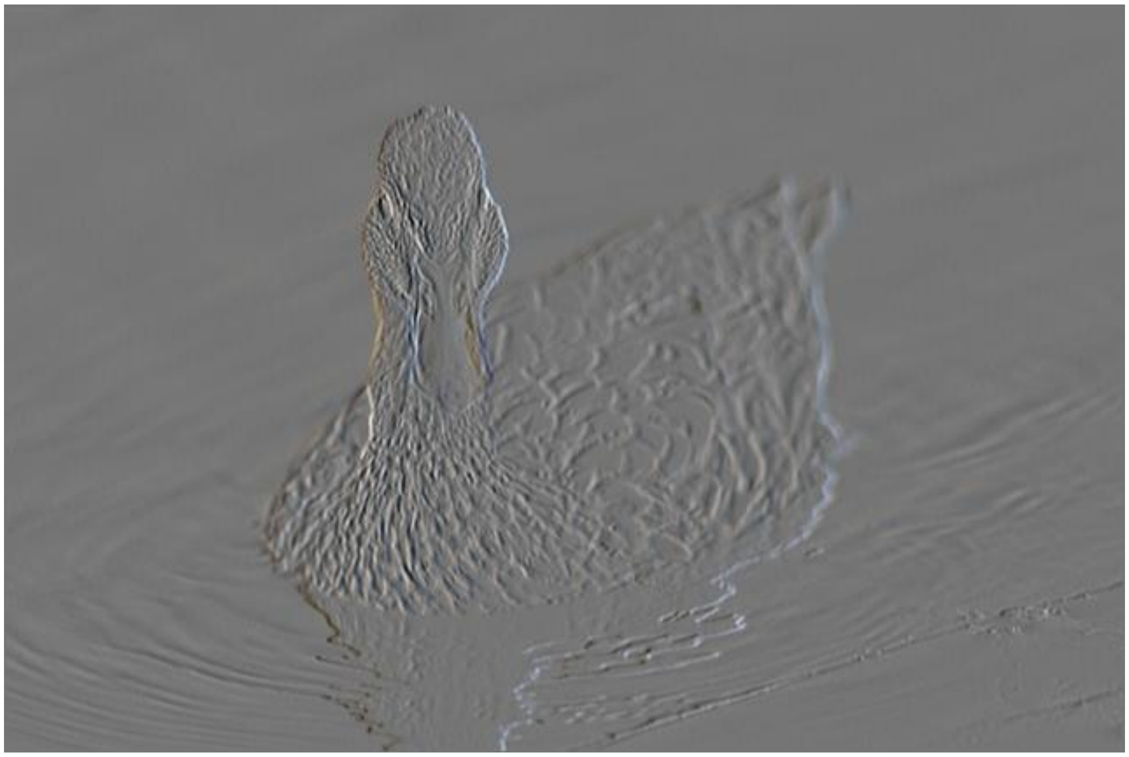
Inverted intensities duck.

Task 2.c

[0.9pt] Implement a function that takes an RGB image and a convolutional kernel as input, and performs 2D spatial convolution. Assume the size of the kernel is odd numbered, e.g. 3 × 3, 5 × 5, or 7 × 7. You must implement the convolution operation yourself from scratch. Implement the function in convolve_im.



Smoothened duck.



Sobel kernel duck.

Task 3: Theory, Neural Networks [1.6pt]

Task 3.a

Try to implement the logic gates AND, OR, NOT, NOR, NAND, or XOR in a single layert neural network. Given that all negative and zero outputs count as 0 and that all positive outputs counts as 1 we can construct every of these except XOR with a single layer network.

Useful soruce: https://medium.com/@stanleydukor neural-representation-of-and-or-not-xor-and-xnor-logic-gates-perceptron-algorithm-b0275375fea1

Task 3.b

A hyperparameter is values related to a nerual net which has to be initiated. For instance this can be the number of epochs and the batch size. One epoch corresponds for the model walking thorugh all the data one time. The batch size is the number of times the net skips to update the model parameters. So if batch equals two it will only train for "every second" value.

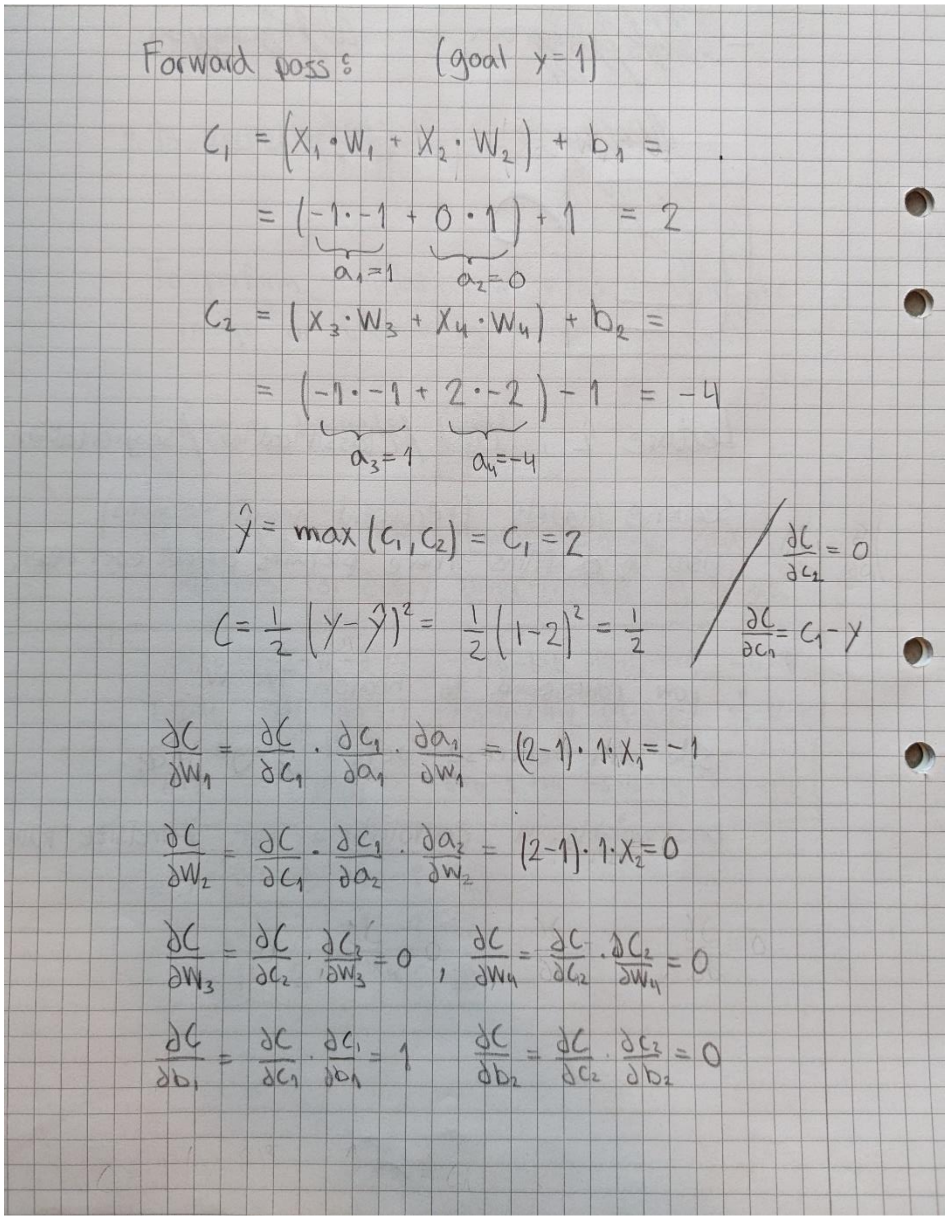
Task 3.c

[0.2pt] Why is the softmax activation functioned used in the last layer for neural networks trained to classify objects?

Softmax is the last layer of a neural net and has the same size as the output. This is used to create probabilities for a classifier. This makes it easier to make decisions from the data. Also it gives more information than just using max which will return zeros and a single one.

Task 3.d

[0.7pt] Figure 2 shows a simple neural network. Perform a forward pass and backward pass on this network with the given input values. Use Equation 3 as the cost function and let the target value be y = 1.



Forward pass: (goal $y=1$)

$$C_1 = (X_1 \cdot W_1 + X_2 \cdot W_2) + b_1 =$$

$$= (\underbrace{-1 \cdot -1}_{a_1=1} + \underbrace{0 \cdot 1}_{a_2=0}) + 1 = 2$$

$$C_2 = (X_3 \cdot W_3 + X_4 \cdot W_4) + b_2 =$$

$$= (\underbrace{-1 \cdot -1}_{a_3=1} + \underbrace{2 \cdot -2}_{a_4=-4}) - 1 = -4$$

$$\hat{y} = \max(C_1, C_2) = C_1 = 2$$

$$C = \frac{1}{2}(y-\hat{y})^2 = \frac{1}{2}(1-2)^2 = \frac{1}{2}$$

$$\frac{\partial C}{\partial C_2} = 0$$

$$\frac{\partial C}{\partial C_1} = C_1 - y$$

$$\frac{\partial C}{\partial W_1} = \frac{\partial C}{\partial C_1} \cdot \frac{\partial C_1}{\partial a_1} \cdot \frac{\partial a_1}{\partial W_1} = (2-1) \cdot 1 \cdot X_1 = -1$$

$$\frac{\partial C}{\partial W_2} = \frac{\partial C}{\partial C_1} \cdot \frac{\partial C_1}{\partial a_2} \cdot \frac{\partial a_2}{\partial W_2} = (2-1) \cdot 1 \cdot X_2 = 0$$

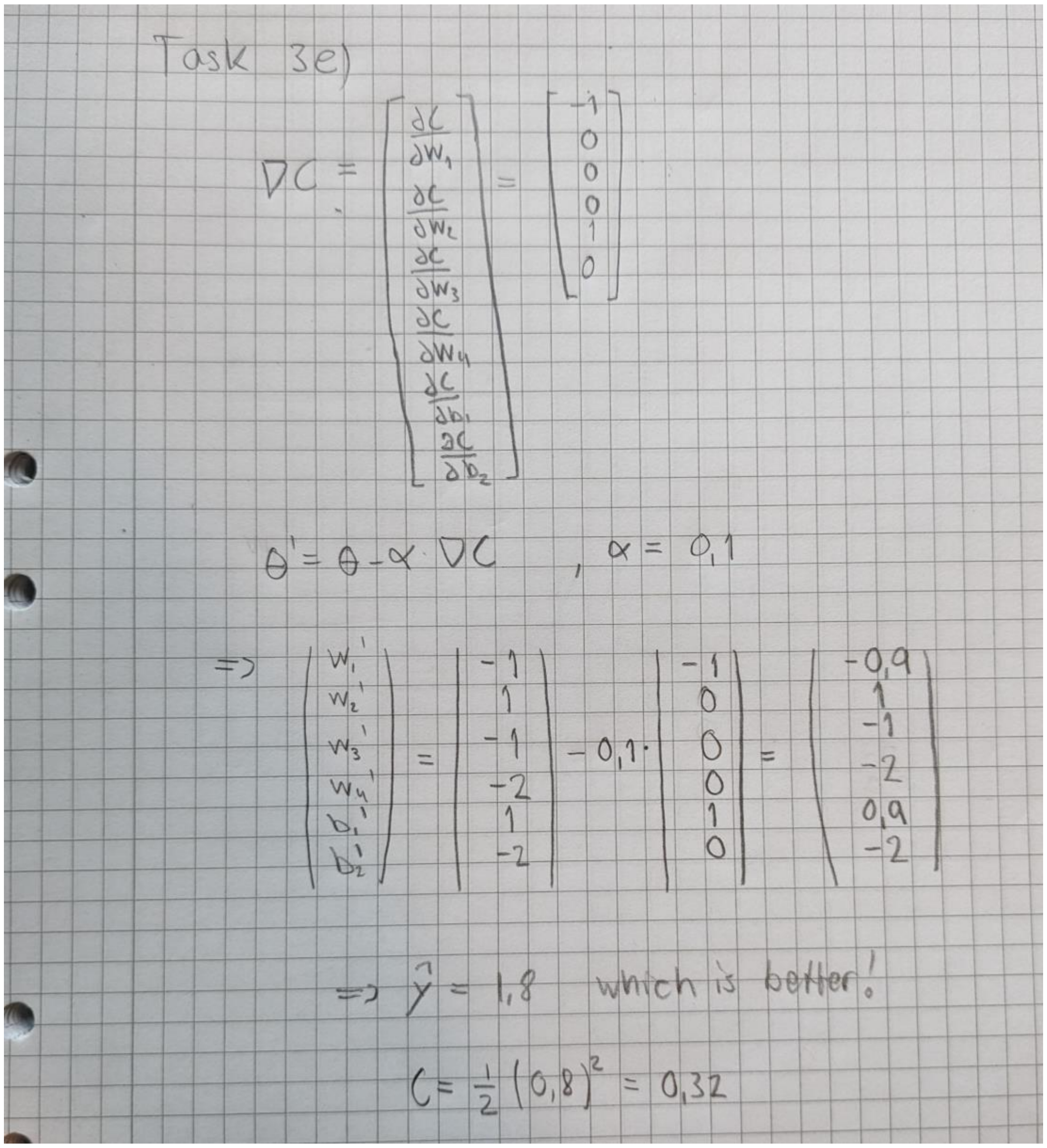
$$\frac{\partial C}{\partial W_3} = \frac{\partial C}{\partial C_2} \cdot \frac{\partial C_2}{\partial W_3} = 0 \quad , \quad \frac{\partial C}{\partial W_4} = \frac{\partial C}{\partial C_2} \cdot \frac{\partial C_2}{\partial W_4} = 0$$

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial C_1} \cdot \frac{\partial C_1}{\partial b_1} = 1 \qquad \frac{\partial C}{\partial b_2} = \frac{\partial C}{\partial C_2} \cdot \frac{\partial C_2}{\partial b_2} = 0$$

Forward and backward propagation.

Task 3.e

[0.3pt] Compute the updated weights w1, w3, and b1 by using gradient descent and the values you found in task d. Use α = 0.1



Task 3e)

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w_1} \\ \frac{\partial C}{\partial w_2} \\ \frac{\partial C}{\partial w_3} \\ \frac{\partial C}{\partial w_4} \\ \frac{\partial C}{\partial b_1} \\ \frac{\partial C}{\partial b_2} \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\theta' = \theta - \alpha \cdot \nabla C \quad , \quad \alpha = 0{,}1$$

$$\Rightarrow \begin{vmatrix} w_1' \\ w_2' \\ w_3' \\ w_4' \\ b_1' \\ b_2' \end{vmatrix} = \begin{vmatrix} -1 \\ 1 \\ -1 \\ -2 \\ 1 \\ -2 \end{vmatrix} - 0{,}1 \cdot \begin{vmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{vmatrix} = \begin{vmatrix} -0{,}9 \\ 1 \\ -1 \\ -2 \\ 0{,}9 \\ -2 \end{vmatrix}$$

$\Rightarrow \hat{y} = 1{,}8$ which is better!

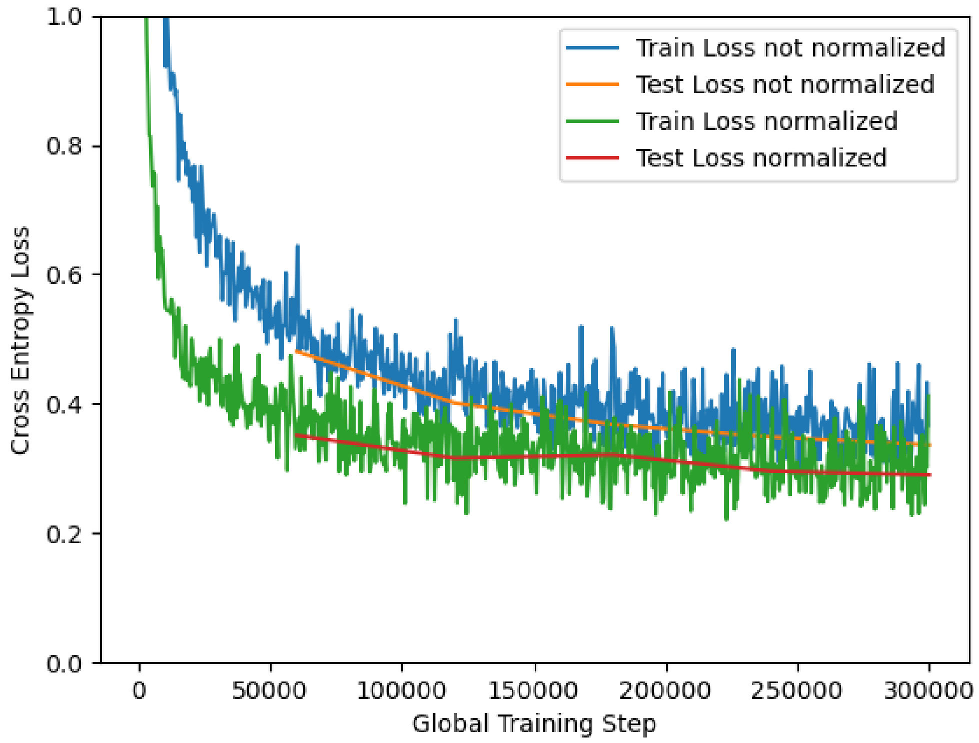
$$C = \frac{1}{2}(0{,}8)^2 = 0{,}32$$

Updating parameter with gradient descent.

Task 4: Programming [2.4pt]

Task 4.a

[0.6pt] Use the given starter code and train a single-layer neural network with batch size of 64. Then, normalize every image between a range of [-1. 1], and train the network again. Plot the training and validation loss from both of the networks in the same graph. Include the graph in your report. Do you notice any difference when training your network with/without normalization?



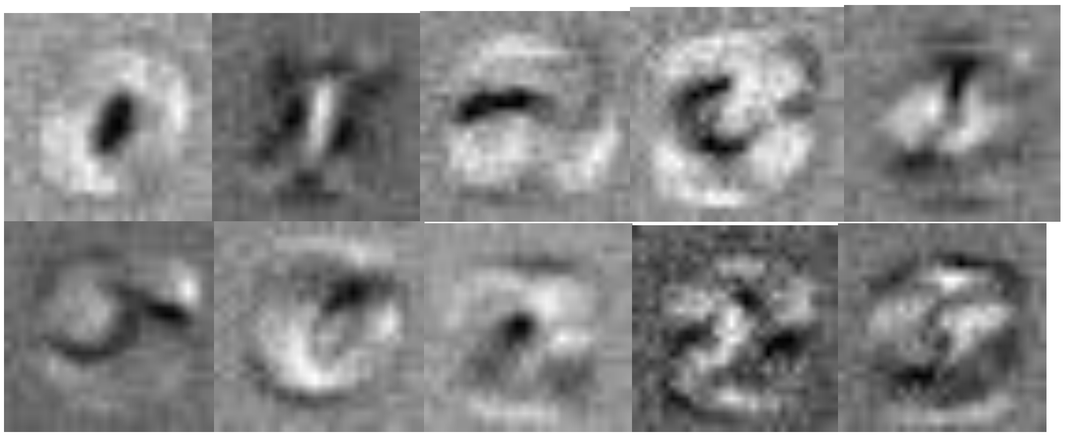
Normalized training data.

- It appears that the normalized network converges faster than the not normalized one. Although it seems that the training took a little bit more time when using normalization. Overall the biggest advantage is the faster and more consistent performance.

Task 4.b

[0.6pt] The trained neural network will have one weight with shape [num classes, 28 × 28]. To visualize the learned weight, we can plot the weight as a 28 × 28 grayscale image. For each digit (0-9), plot the learned weight as a 28 × 28 image. In your report, include the image for each weight,and describe what you observe (1-2 sentences).

- I began explaining each of the weights but then realized maybe you just wanted overall comments. So generally we have high weights for the written part / shape of the number. The other part is a low weight part which can help to distinguish characteristics.



Number weights 0-9.

- Number 0: The most important features here is the low value weights in middle corresponding to the hole in the zero. Also it seems like a round ring with higher weights.

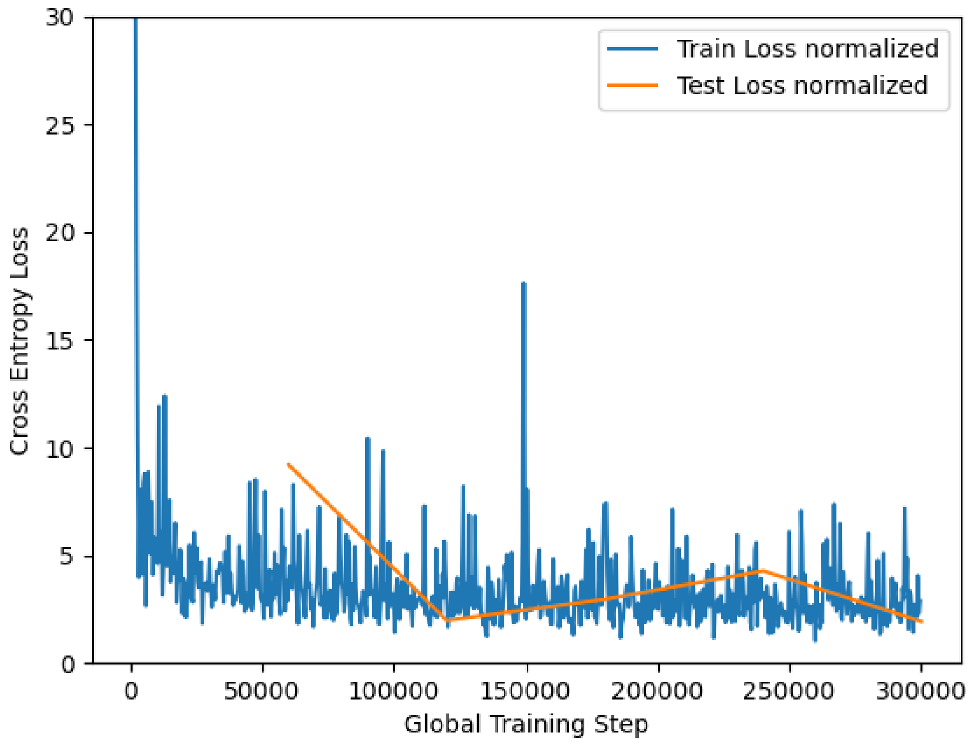
- Number 1: Here it seems like the most important part is the vertical line in the middle. There is also smaller weights around to really discretize the line.

- Number 2: For the two we can see a little bit of high values in the shape of a two. Though the most critical part looks to be the low weights for the pixels in the "upper hook".

- Number 3: Also here the shape of the high weights corresponds to the number three. Here we also have a low weight part which amplifies the shape of the number. In this case an arc.

Task 4.c

[0.6pt] Set the learning rate to lr = 1.0, and train the network from scratch. Report the accuracy and average cross entropy loss on the validation set. In 1-2 sentences, explain why the network achieves worse/better accuracy than previously.

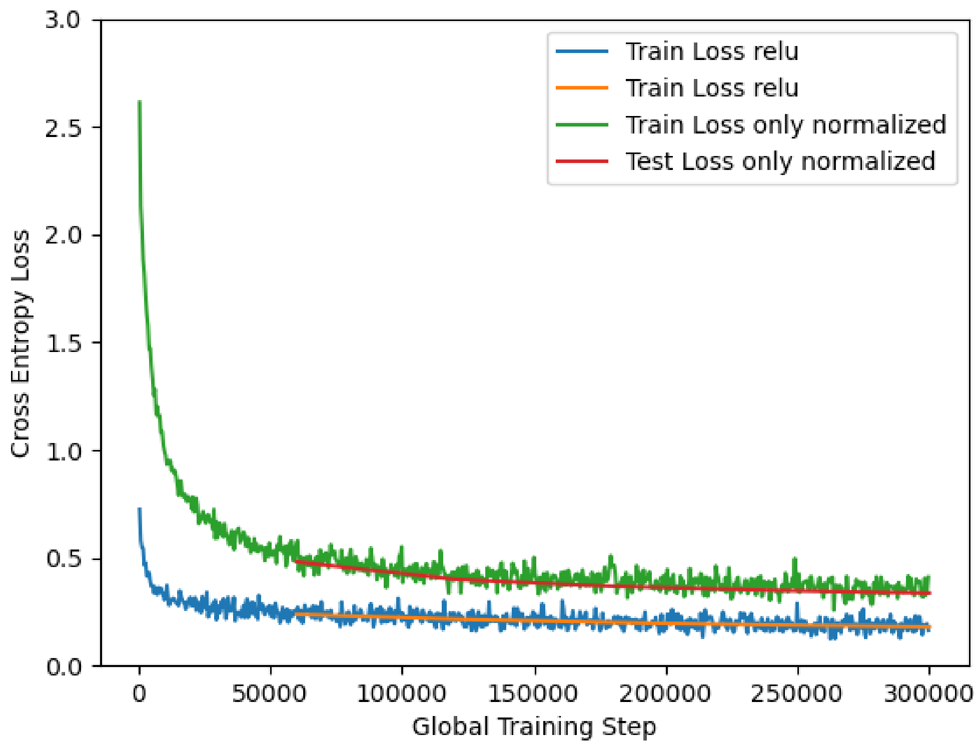


Increased learning rate.

- Final Test loss: 2.8491579969802894. Final Test accuracy: 0.8631
- The reason the network performed worse is the too high learning rate. Ideally increaseing learning rate might just improve the speed of the convergence. Though in this case it seems like the network did not really converge to anything at all. It just oscillated around some point.

Task 4.d

[0.6pt] Include an hidden layer with 64 nodes in the network, with ReLU as the activation function for the first layer. Train this network with the same hyperparameters as previously. Plot the training and validation loss from this network together with the loss from task (a). Include the plot in your report. What do you observe?



Initial compared to relu.

- In this case I observe a significantly better and fast convergence than the linear model. This makes some sense since the numbers can not really be describes linearly.