Assignment 4 - TDT4195

Nora Evensen Jansrud, Olaf Rosendahl

October 25, 2022

Task 1: Theory

a) Sampling is the process of digitizing discrete coordinate values in the image

b) Quantization is the process of transforming the sampling values into discrete amplitude values

c) An image with high contrast will have a histogram with quite low values in the middle and peaks in the start and end of the histogram.

d) Initial matrix:

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

$MN = 3 \times 5 = 15$

r	$H(r)$	$p(r)$	$F(r)$	$T(r)$	Sum of $T(r)$
0	1	$\frac{1}{15}$	$\frac{1}{15}$	$7 \times \frac{1}{15}$	0
1	1	$\frac{1}{15}$	$\frac{2}{15}$	$7 \times \frac{2}{15}$	0
2	0	$\frac{0}{15}$	$\frac{2}{15}$	$7 \times \frac{2}{15}$	0
3	1	$\frac{1}{15}$	$\frac{3}{15}$	$7 \times \frac{3}{15}$	1
4	2	$\frac{2}{15}$	$\frac{5}{15}$	$7 \times \frac{5}{15}$	2
5	2	$\frac{2}{15}$	$\frac{7}{15}$	$7 \times \frac{7}{15}$	3
6	4	$\frac{4}{15}$	$\frac{11}{15}$	$7 \times \frac{11}{15}$	5
7	4	$\frac{4}{15}$	$\frac{15}{15}$	$7 \times \frac{15}{15}$	7

Final matrix:

7	5	3	5	2
3	2	7	7	0
0	7	5	1	5

e) When applying a log transform to an image, the lower end of the histogram is squeezed (boosted) and the higher end is widened (flattened). As a result of this, the image's dynamic range is compressed. When applied to images with a large variance in pixel intensities this will lead to impact the dynamic range even more then for images with low variance since a larger compression can be done. If most of the pixels are close to the mean intensity, there isn't many opportunities to flatten the high end or boost the low end more.

f) We use zero-padding on the image to handle boundary conditions and flips the kernel for convolution calculation. We then calculate the spatial convolution by for each element in the image, multiply the 3×3 matrix around it with the flipped kernel and sum it to find the new value.

Zero-padded image:

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

Initial kernel:

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

Flipped kernel:

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

For each element in the result matrix:

$M_{11} = (-1) * 0 + 0 * 0 + 1 * 0 + (-2) * 0 + 0 * 7 + 2 * 6 + (-1) * 0 + 0 * 5 + 1 * 4 = 16$

$M_{21} = 1 * 6 + 2 * 4 + 1 * 7 = 21$

$M_{31} = 1 * 4 + 2 * 7 = 18$

$M_{12} = (-2) * 7 + (-1) * 5 + 2 * 5 + 1 * 7 = -2$

$M_{22} = (-1) * 7 + (-2) * 5 + (-1) * 1 + 1 * 5 + 2 * 7 + 1 * 6 = 7$

$M_{32} = (-1) * 5 + (-2) * 1 + 1 * 7 + 2 * 6 = 12$

$M_{13} = (-2) * 6 + (-1) * 4 + 2 * 6 + 1 * 7 = 3$

$M_{23} = (-1) * 6 + (-2) * 4 + (-1) * 7 + 1 * 6 + 2 * 7 + 1 * 3 = 2$

$M_{33} = (-1) * 4 + (-2) * 7 + 1 * 7 + 2 * 3 = -5$

$M_{14} = (-2) * 5 + (-1) * 7 + 2 * 4 = -9$

$M_{24} = (-1) * 5 + (-2) * 7 + (-1) * 6 + 1 * 4 + 1 * 6 = -15$

$M_{34} = (-1) * 7 + (-2) * 6 + 2 * 6 = -7$

$M_{15} = (-2) * 6 + (-1) * 7 = -19$

$M_{25} = (-1) * 6 + (-2) * 7 + (-1) * 3 = -23$

$M_{35} = (-1) * 7 + (-2) * 3 = -13$

Result matrix:

16	-2	3	-9	-19
21	7	2	-15	-23
18	12	-5	-7	-13

Task 2: Programming

a)

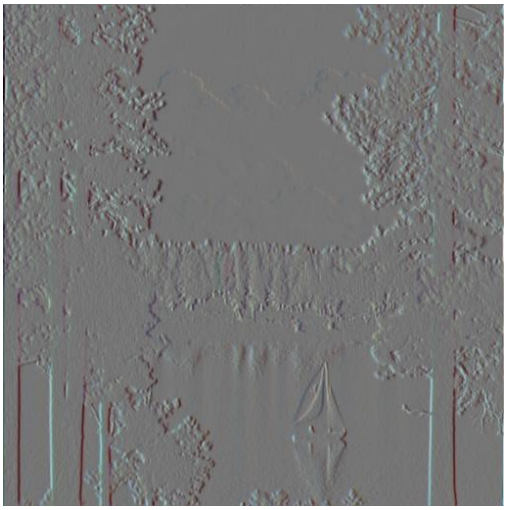
Figure 1: Greyscale

b)

Figure 2: Inverse

c)

(a) Smoothing kernel



(b) Sobel kernel

Task 3: Theory

a) Since single-layer neural networks is a linear function, it must be possible to separate the output points with a linear function. By looking at Figure 4, we can see that the OR output points can be separated with a linear function, but the XOR output points cannot. XOR can therefor not be represented by a single-layer neural network.

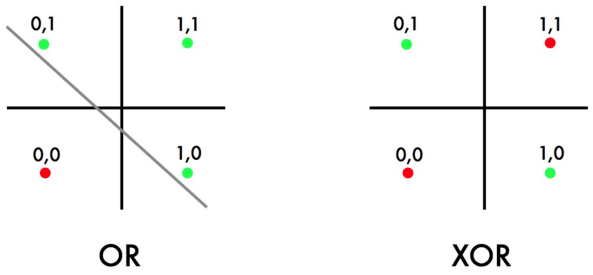


Figure 4: OR vs XOR output points

b) A hyperparameter is a constant parameter which is set once per learning process and used to control the process.

Some examples of a hyperparameter are batch size and learning rate α.

c) The softmax activation function is used in the last layer because it transforms a vector of number to a vector of probabilities. These probabilities are all between 0 and 1, and

also sums to 1. This gives the user a more understandable output. This also makes sure that the networks doesn't get extremely high values which may cause problems in back propagation.

d) Forward pass:

$c_1 = X_1 * w_1 + X_2 * w_2 + b_1 = (-1) * (-1) + 0 * 1 + 1 = 2$

$c_2 = X_3 * w_3 + X_4 * w_4 + b_2 = (-1) * (-1) + 2 * (-2) + (-1) = -4$

$\hat{y} = max(c_1, c_2) = 2$

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

Backward pass:

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

$\frac{\partial C}{\partial c_1} = \frac{\partial C}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial c_1} = \frac{\partial C}{\partial \hat{y}} * \frac{\partial}{\partial c_1} c_1 = \frac{\partial C}{\partial \hat{y}} * 1 = 1 * 1 = 1$ (because $C = max(c_1, c_2)$ and $c_1 > c_2$)

$\frac{\partial C}{\partial c_2} = \frac{\partial C}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial c_2} = \frac{\partial C}{\partial \hat{y}} * \frac{\partial}{\partial c_2} c_1 = \frac{\partial C}{\partial \hat{y}} * 0 = 0$

$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial c_1} * \frac{\partial c_1}{\partial b_1} = \frac{\partial C}{\partial c_1} * \frac{\partial}{\partial b_1}(w_1 * x_1 + w_2 * x_2 + b_1) = 1 * 1 = 1$

$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial c_1} * \frac{\partial c_1}{\partial w_1} = \frac{\partial C}{\partial c_1} * \frac{\partial}{\partial w_1}(w_1 * x_1 + w_2 * x_2 + b_1) = 1 * (-1) = -1$

$\frac{\partial C}{\partial w_2} = \frac{\partial C}{\partial c_1} * \frac{\partial c_1}{\partial w_2} = \frac{\partial C}{\partial c_1} * \frac{\partial}{\partial w_2}(w_1 * x_1 + w_2 * x_2 + b_1) = 1 * 0 = 0$

$\frac{\partial C}{\partial b_2} = \frac{\partial C}{\partial c_2} * \frac{\partial c_2}{\partial b_2} = 0 * \frac{\partial c_2}{\partial b_2} = 0$

$\frac{\partial C}{\partial w_3} = \frac{\partial C}{\partial c_2} * \frac{\partial c_2}{\partial w_3} = 0 * \frac{\partial c_2}{\partial w_3} = 0$

$\frac{\partial C}{\partial w_4} = \frac{\partial C}{\partial c_2} * \frac{\partial c_2}{\partial w_4} = 0 * \frac{\partial c_2}{\partial w_4} = 0$

e) General formula: $\theta^{n+1} = \theta^n - \alpha * \frac{\partial C}{\partial \theta^n}$

${w_1}^2 = {w_1}^1 - \alpha * \frac{\partial C}{\partial {w_1}^1} = (-1) - 0.1 * (-1) = -0.9$

${w_3}^2 = {w_3}^1 - \alpha * \frac{\partial C}{\partial {w_3}^1} = (-1) - 0.1 * 0 = -1$

${b_1}^2 = {b_1}^1 - \alpha * \frac{\partial C}{\partial {b_1}^1} = 1 - 0.1 * 1 = 0.9$

Task 4: Programming

a) When training with normalization the loss decreases faster because normalization makes it easier for the model to recognize patterns.

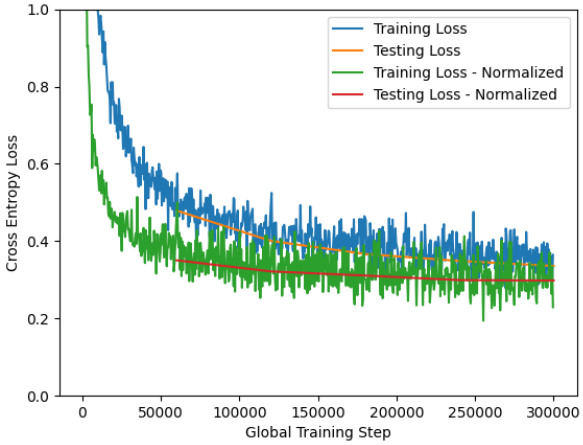
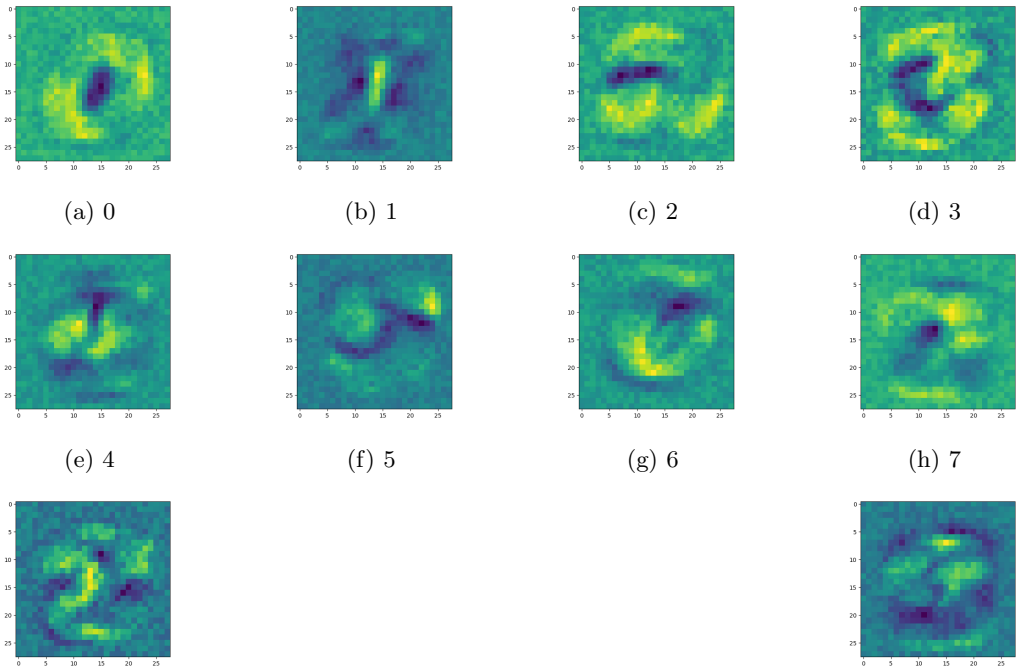


Figure 5: Training and validation loss from network with and without normalization



(a) 0 (b) 1 (c) 2 (d) 3

(e) 4 (f) 5 (g) 6 (h) 7

(i) 8 (j) 9

Figure 6: The learned weight plotted as a 28 $\times$ 28 image

b) We can in Figure 6 see the different numbers if we use out imagination a little bit. Each image represents a heatmap of where the digits are placed in the image. Looking at number 0 it's clear how it's a circle with nothing in the center, while for number 1 it's a line in the center and nothing around.

c) Figure 7 shows the result with learning rate = 1.0. This results in worse accuracy, due to the large learning rate. A large learning rate results in overfitting where the model isn't able to find an optimal value and the loss therefore move up and down. We did also have to change the Y-axis to a 0-20 scale in order to view the data, compared to the 0-1 scale used in previous tasks.

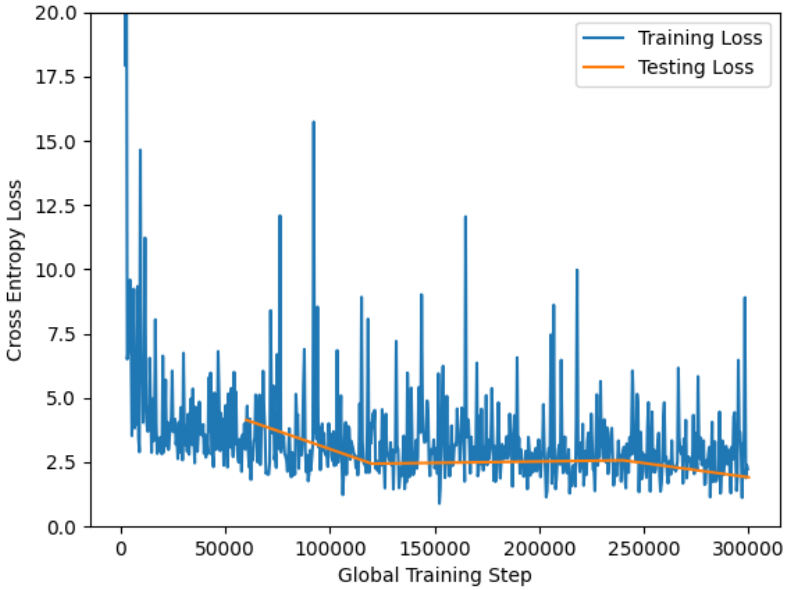


Figure 7: Training with learning rate of 1.0

d) As seen in Figure 8, the hidden layer model seems to be even better then the model from task A. The hidden layer is even more sensitive for small differences/changes, and can there for decrease then loss even more. The task A model seems in comparison to flat out where the hidden layer model continues to decrease the loss.

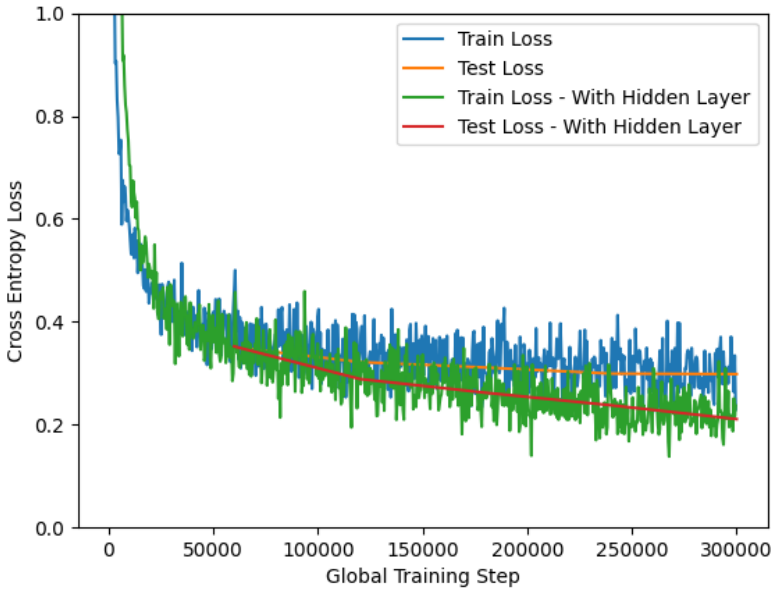


Figure 8: Networks with and without a hidden layer