Assignment 1

Visual Computing Fundamentals: Image Processing

Pablo Díaz Viñambres : *pablodi@ntnu.no*

October 22, 2022

Contents

1 Task 1: Spatial Filtering - Theory

a) Explain in one sentence what sampling is.

Sampling is the extraction process used to transform a continuous signal to a discrete one using a subset of point measurements.

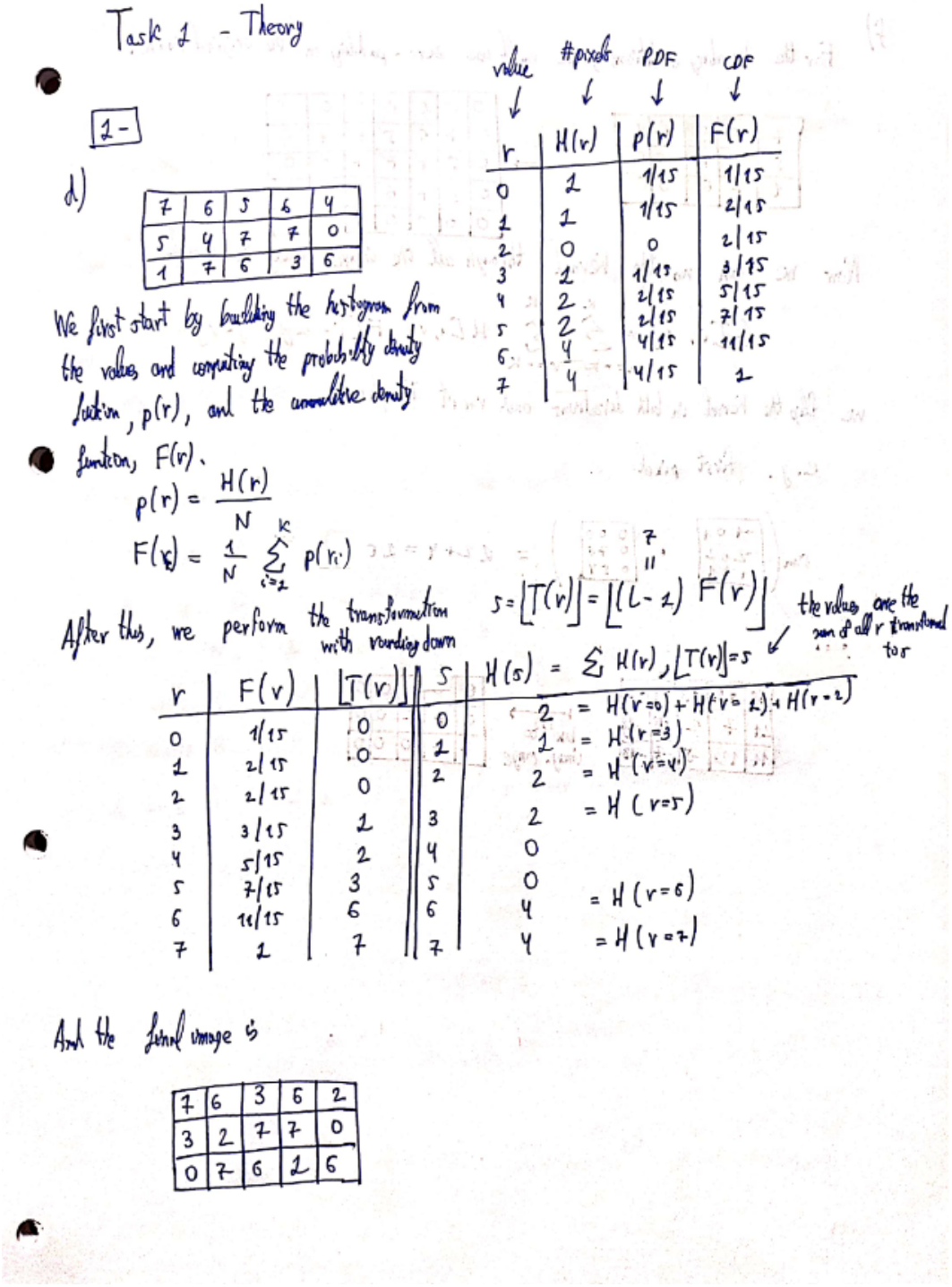
b) Explain in one sentence what quantization is.

Quantization is the discretization of of the values of a continuous function, in digital images, this refers to the discretization of the intensity of the color channels.

c) Looking at an image histogram, how can you see that the image has high contrast?

An image with high contrast has a high concentration of values in both extremes of the histogram, implying that the variance of the values in the distribution is high.

d) Perform histogram equalization by hand on the 3-bit (8 intensity levels) image in Figure 1a



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

r	H(r)	p(r)	F(r)
0	1	1/15	1/15
1	1	1/15	2/15
2	0	0	2/15
3	1	1/15	3/15
4	2	2/15	5/15
5	2	2/15	7/15
6	4	4/15	11/15
7	4	4/15	1

r	F(r)	⌊T(r)⌋	s
0	1/15	0	0
1	2/15	0	1
2	2/15	0	2
3	3/15	1	3
4	5/15	2	4
5	7/15	3	5
6	11/15	6	6
7	1	7	7

7	6	3	6	2
3	2	7	7	0
0	7	6	2	6

Figure 1: Histogram equalization by hand.

e) What happens to the dynamic range if we apply a log transform to an image with a large variance in pixel intensities?

The dynamic range is compressed, since low intensities are widened and high squeezed. That is, dark pixels become brighter and bright pixels become dimmer.

f) Perform spatial convolution by hand on the image in Figure 1a using the kernel in Figure 1b.

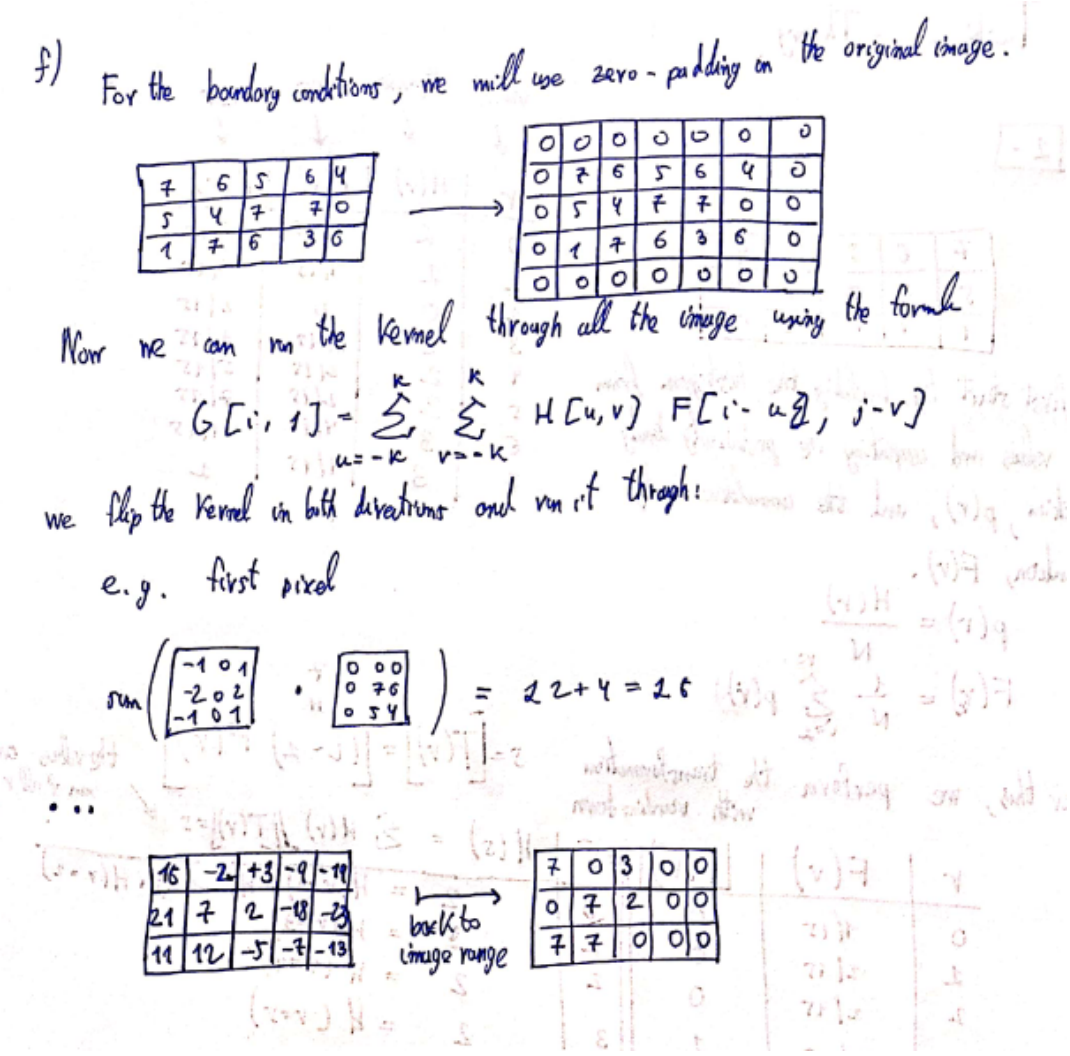


Figure 2: Spatial convolution by hand.

2 Task 2: Spatial Filtering - Practice

a) Implement a function that converts an RGB image to grayscale

After implementing the function `grayscale()` using the given formula, we get the following image:

Figure 3: The lake image in grayscale.

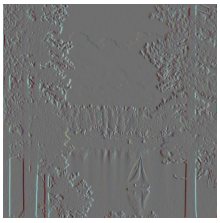
b) Implement a function that takes a grayscale image and inverts its color

After implementing the function `inverse()` using the given formula, we get the following image:

Figure 4: The lake image in an inverted grayscale.

c) Implement a function that takes an RGB image and a convolutional kernel as input, and performs 2D spatial convolution.

We implemented the function `convolve_im()` by doing a double loop on every image pixel and then using a *numpy* sum on the corresponding products of the kernel elements with the image pixels. After repeating the process for every color channel, we got the following results for the Sobel and smoothing kernels:



(a) The image convolved with the Sobel kernel

(b) The image convolved with the smoothing kernel

3 Task 3: Neural Networks - Theory

a) A single-layer neural network is a linear function. Which of these binary operation(s) can not be represented by a single-layer neural network: `AND`, `OR`, `NOT`, `NOR`, `NAND`, or `XOR`?

.

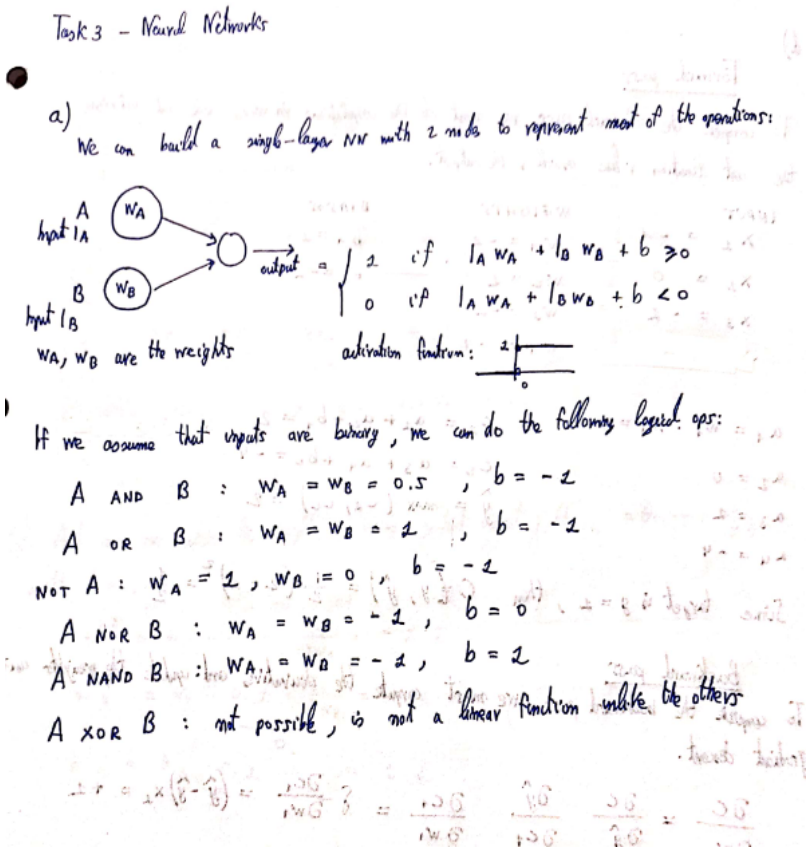


Figure 6: Single layer neural network for logical operators.

b) Explain in one sentence what a hyperparameter for a neural network is. Give two examples of a hyperparameter.

Hyperparameters are network parameters that can not be learned in the training process unlike neurons' weights and biases. Examples of them in FCNNs are the learning rate, the number of layers and the number of neurons in each one.

c) Why is the *Softmax* activation functioned used in the last layer for neural networks trained to classify objects?

Softmax is used because it can convert neural networks numerical output values to probabilities that add up to one. By doing this, object classification networks can output the highest(s) probability class and the confidence they have on it as a percentage.

d) Perform a forward pass and backward pass on Figure 2 network with the given input values.

d)

Forward pass

To complete the forward pass, we must do the computations in every node and retrieve the cost function value aswell as the output.

INPUT	WEIGHTS	BIASES
$x_1 = -1$	$w_1 = -1$	$b_1 = 1$
$x_2 = 0$	$w_2 = 1$	$b_2 = -1$
$x_3 = -1$	$w_3 = -1$	
$x_4 = 2$	$w_4 = -2$	

$a_1 = w_1 \cdot x_1 = 1$

$a_2 = 0$

$a_3 = 1$

$a_4 = -4$

$c_1 = a_1 + a_2 + b_1 = 2$

$c_2 = a_3 + a_4 + b_2 = -4$

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

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

Backward pass

To complete the backward pass, we must compute the derivatives and update the weights using gradient descent.

$$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial c_1} \frac{\partial c_1}{\partial w_1} = \delta \frac{\partial c_1}{\partial w_1} = (y - \hat{y}) x_1 = +1$$

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

$\hat{y} = \max(c_1, c_2) = \begin{cases} c_1 \text{ if } c_1 > c_2 \\ c_2 \text{ if } c_1 < c_2 \end{cases} \Rightarrow \frac{\partial \hat{y}}{\partial c_1} = \begin{cases} 1 \text{ if } c_1 > c_2 \\ 0 \text{ if } c_1 < c_2 \end{cases}$

$\delta = y - \hat{y} = -1$

$c_1 = w_1 x_1 + w_2 x_2 + b_1 \Rightarrow \frac{\partial c_1}{\partial w_1} = x_1 = -1$

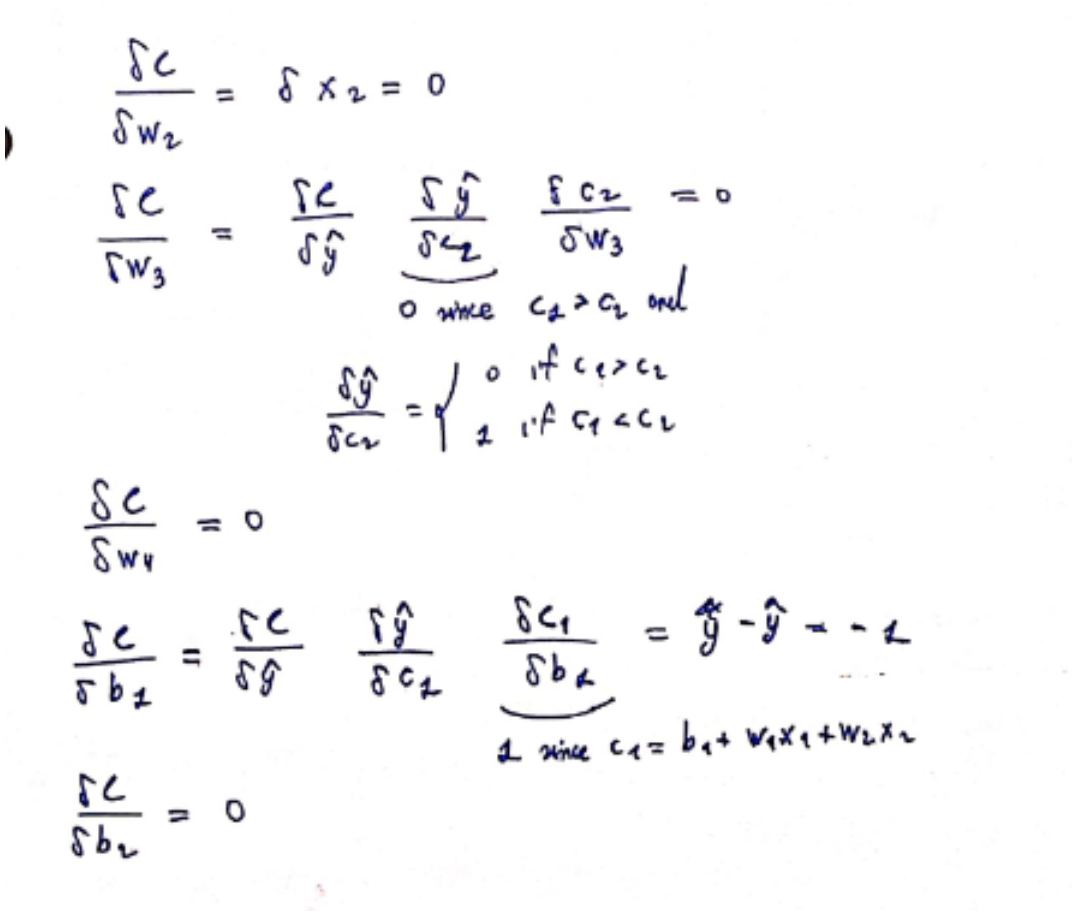


Figure 7: Forward pass and derivative computing in the neural network.

e) Compute the updated weights w_1, w_3, and b_1 by using gradient descent and the values you found in task d).

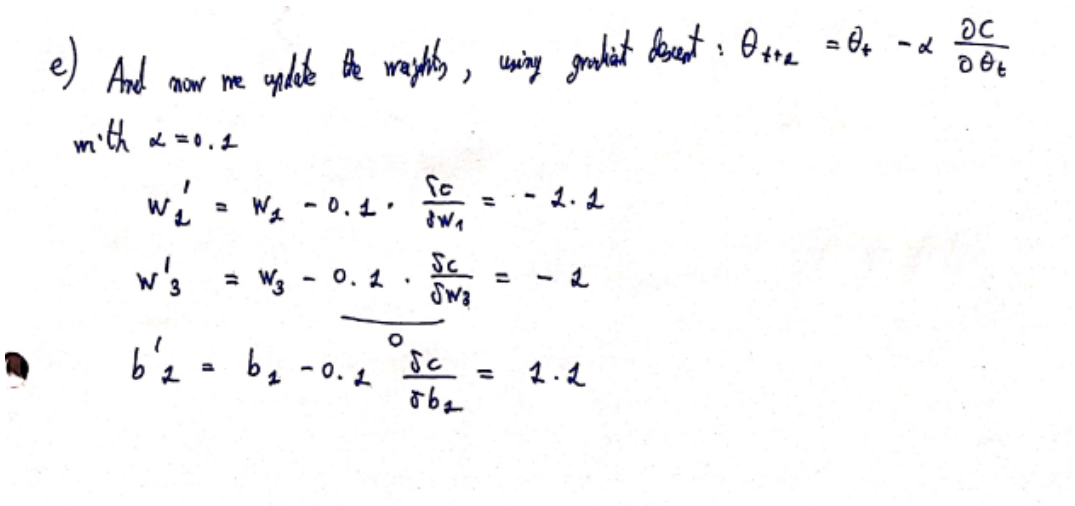


Figure 8: Gradient descent step on the neural network.

4 Task 4: Neural Networks - Practice

a) Training of neural networks and image normalization

We trained the given neural network with and without image normalization. As we see on the comparison plot, the normalization improves the performance of the network significantly, so we will be using them in the following tasks.

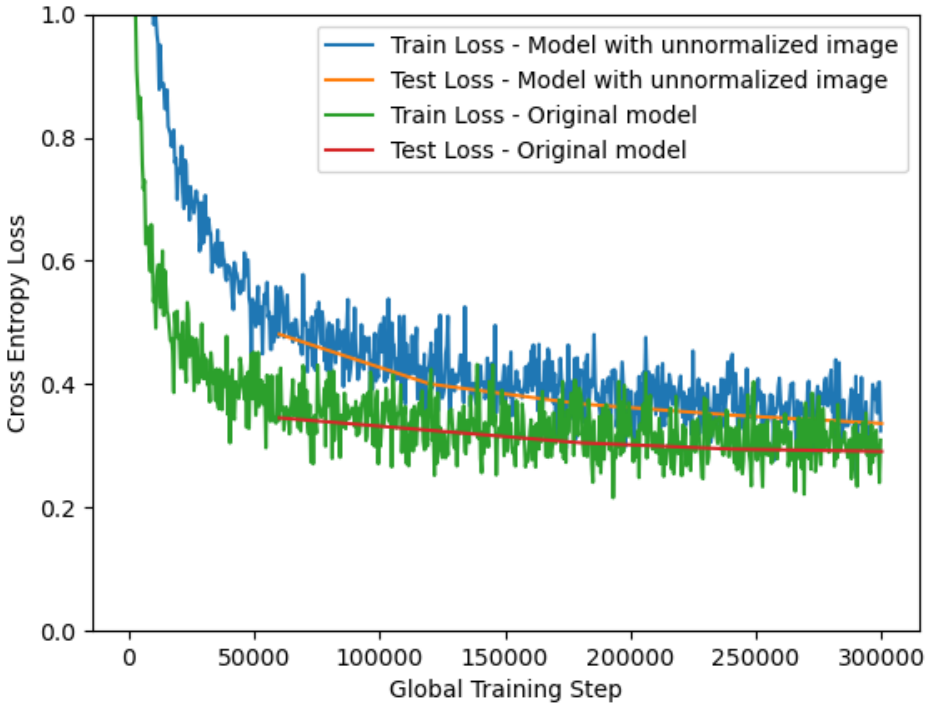


Figure 9: Comparison between the model trained with and without image normalization.

b) Plotting of the network weights as grayscale images

For this task, we obtained the networks weights, scaled them to the 0 to 255 grayscale image range and converted them to images using the *PIL* library. On the resulting images, we observe higher values (i.e. higher brightness) on the corresponding digits generic contour.

This result is expected as the neurons that correspond to pixels that are usually part of the number calligraphy should have a higher impact in the output. Per example, digit '0' has a clear white ring around a center that is completely dark, because all zeroes are round and they never always have a hole.

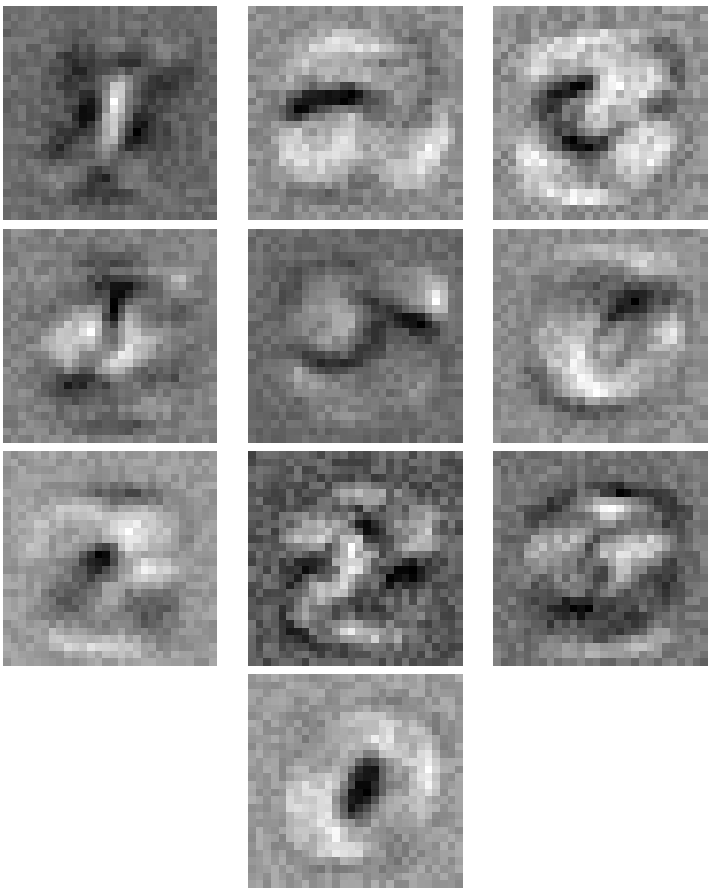


Table 1: The learned weights for each digit in a telephone layout.

c)Set the learning rate to $lr = 1.0$, and train the network from scratch.

When setting the learning rate so high, the accuracy of the model becomes very chaotic and it performs way worse than before. We get an accuracy of 0.8316 and a very high entropy rate as seen on the plot.

The reason for such poor performance is that a learning rate so high makes the parameters change too quickly. Therefore, we never get to descent on the gradient, we only jump from one 'side of the hill' to another on the direction given by the last gradient.

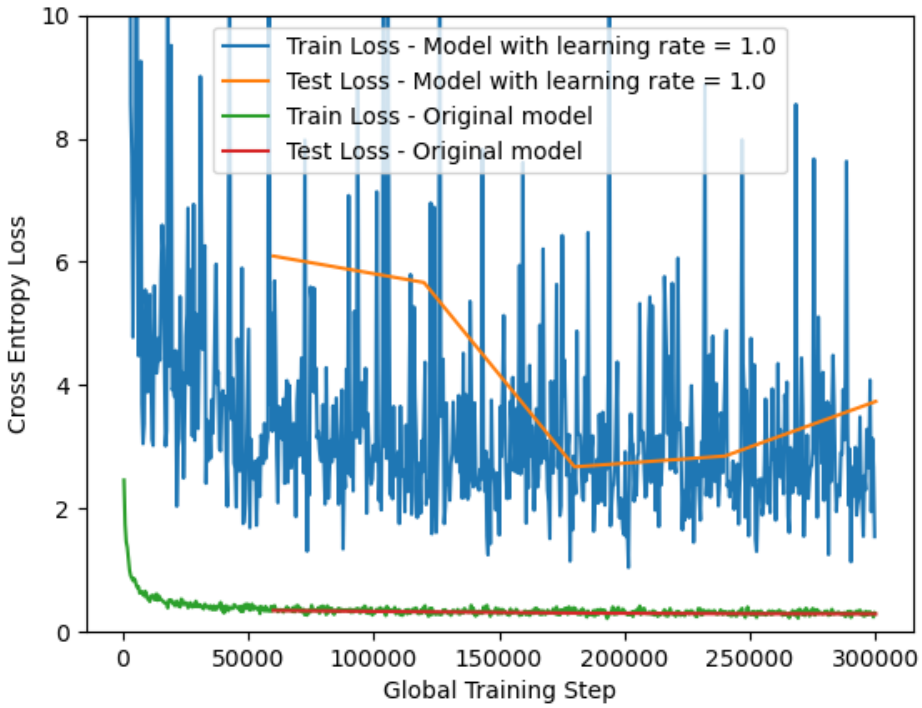


Figure 10: Comparison between the model when trained with $\alpha = 0.0192$ and $\alpha = 1.0$

d) Include a hidden layer with 64 nodes in the network, with *ReLU* as the activation function for the first layer.

After adding this extra layer, we observe a better overall network performance. This happens because recognizing digits is itself a complex problem, so adding some hidden layers gives us a more adequate model for completing the task.

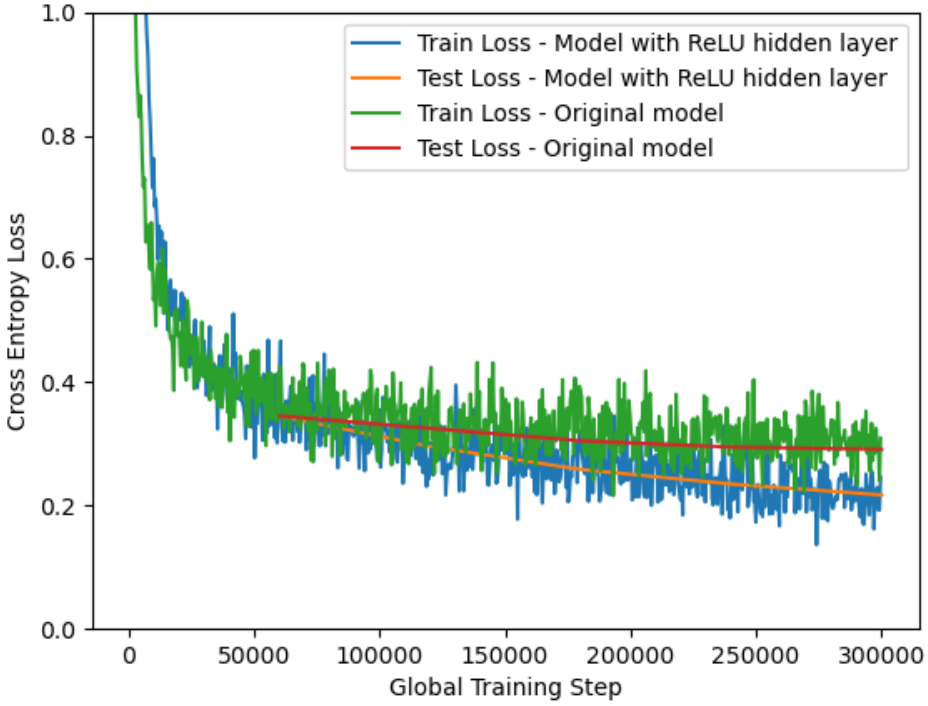


Figure 11: Comparison between the original model and the one with the *ReLU* hidden layer.