TDT4195 Visual Computing Fundamentals, Image Processing Assignment 1

Task 1

a)

Explain in one sentence what sampling is.
Sampling is the process of discretising the spatial position of a signal source into a pixel grid.

---

b)

Explain in one sentence what quantization is.
Quantisation is the process of discretising the intensity of a signal source.

---

c)

Looking at an image histogram, how can you see that the image has high contrast?
A histogram with a wide range of intensity values suggests a contrast image. A high contrast image will have large separation between peaks in the histogram.

---

d)

Perform histogram equalisation 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).

r_k	$H_r(r_k)$	$p_r(r_k)$	$F_r(r_k)$	$T(r_k)$
0	1	$\frac{1}{15}$	$\frac{1}{15}$	$\lfloor \frac{7}{15} \rfloor = 0$
1	1	$\frac{1}{15}$	$\frac{2}{15}$	$\lfloor \frac{14}{15} \rfloor = 0$
2	0	0	$\frac{2}{15}$	$\lfloor \frac{14}{15} \rfloor = 0$
3	1	$\frac{1}{15}$	$\frac{3}{15}$	$\lfloor \frac{21}{15} \rfloor = 1$
4	2	$\frac{2}{15}$	$\frac{5}{15}$	$\lfloor \frac{35}{15} \rfloor = 2$
5	2	$\frac{2}{15}$	$\frac{7}{15}$	$\lfloor \frac{49}{15} \rfloor = 3$
6	4	$\frac{4}{15}$	$\frac{11}{15}$	$\lfloor \frac{77}{15} \rfloor = 5$
7	4	$\frac{4}{15}$	$\frac{15}{15}$	$\lfloor \frac{105}{15} \rfloor = 7$

Table 1: Input image histogram

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

Table 2: Output image histogram

---

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 will be compressed. Because the function is logarithmic and the input to the logarithmic function is always above 1 it will always compress values.

$$f(c, r) = c \cdot log(1 + r) \tag{1a}$$
$$f(c, 10) = c \cdot log(1 + 10) \approx c \cdot 1.04 \tag{1b}$$
$$f(c, 100) = c \cdot log(1 + 100) \approx c \cdot 2.00 \tag{1c}$$
$$\tag{1d}$$

---

f)

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.

To apply cross-correlation, the kernel is flipped:

$$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

The image is padded s.t. the kernel doesn't extend outside of the image that is convolved.

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 7 & 6 & 3 & 6 & 0 \\ 0 & 7 & 6 & 5 & 6 & 4 & 0 \\ 0 & 5 & 4 & 7 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Applying the kernel to the first pixel in the image gives the following operations. The pixel where the kernel is applied is marked in green, the kernel entries are marked in red, and the image values are in black.

$$\begin{matrix} -1\cdot 0 & 0\cdot 0 & 1\cdot 0 & 0 & 0 & 0 & 0 \\ -2\cdot 0 & 0\cdot 1 & 2\cdot 7 & 6 & 3 & 6 & 0 \\ -1\cdot 0 & 0\cdot 7 & 1\cdot 6 & 5 & 6 & 4 & 0 \\ 0 & 5 & 4 & 7 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

This yields the following result for the first pixel, applied to the original image:

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20 & 7 & 6 & 3 & 6 & 0 \\ 0 & 7 & 6 & 5 & 6 & 4 & 0 \\ 0 & 5 & 4 & 7 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

The kernel is then applied to the second pixel in the image:

0	-1· 0	0· 0	1· 0	0	0	0
0	-2· 1	0· 7	2· 6	3	6	0
0	-1· 7	0· 6	1· 5	6	4	0
0	5	4	7	7	0	0
0	0	0	0	0	0	0

with the result:

0	0	0	0	0	0	0
0	20	8	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

The kernel applied to the rest of the image yields:

0	0	0	0	0	0	0
0	20	8	-8	-1	-12	0
0	23	3	-1	-13	-22	0
0	14	2	6	-15	-20	0
0	0	0	0	0	0	0

Such that the convolved image is

20	8	-8	-1	-12
23	3	-1	-13	-22
14	2	6	-15	-20

Task 2

a)

Implement a function that converts an RGB image to grayscale. Use Equation 1. Implement this in the function `greyscale`.

Figure 1: Image of duck in greyscale

Listing 1: Greyscale function

```
1       def greyscale(im):
2       """ Converts an RGB image to greyscale
3
4       Args:
5           im ([type]): [np.array of shape [H, W, 3]]
6
7       Returns:
8           im ([type]): [np.array of shape [H, W]]
9       """
10
11      W  = im.shape[0]
12      H  = im.shape[1]
13      CV = im.shape[2]
14
15      im_copy = np.copy(im)
16
17      for w in range(W):
18          R = np.reshape( 0.212*im_copy[w,:,0], (H, 1))
```

```
        G = np.reshape(0.7152*im_copy[w,:,1], (H, 1))
        B = np.reshape(0.0722*im_copy[w,:,2], (H, 1))

        grey = np.concatenate((R,G,B), axis = 1)
        grey = np.reshape(grey.sum(axis=1), (grey.shape[0], 1))
        grey = np.concatenate((grey,grey,grey), axis=1)
        im_copy[w,:,:] = grey
    return im_copy
```

---

b)

Implement a function that takes a grayscale image and applies the following intensity transformation $T(p) = 1 - p$. Implement this in the function `inverse`.

Listing 2: Inverse function

```
def inverse(im):
""" Finds the inverse of the greyscale image

Args:
    im ([type]): [np.array of shape [H, W]]

Returns:
    im ([type]): [np.array of shape [H, W]]
"""

W  = im.shape[0]
H  = im.shape[1]
CV = im.shape[2]

im_copy = np.copy(im)

for w in range(W):
    R = np.reshape(
        np.subtract(np.ones((H,)),
        im_copy[w,:,0]),
        (H, 1))
    G = np.reshape(
        np.subtract(
            np.ones((H,)),
            im_copy[w,:,1]),
        (H, 1))
    B = np.reshape(
        np.subtract(
            np.ones((H,)),
            im_copy[w,:,2]),
        (H, 1))
    rgb_inverse = np.concatenate((R,G,B), axis=1)
    im_copy[w,:,:] = rgb_inverse
return im_copy
```

---

c)

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`
Convolve the image duck.jpeg with the sobel kernel (ha) and the smoothing kernel (hb) in Equation 2.

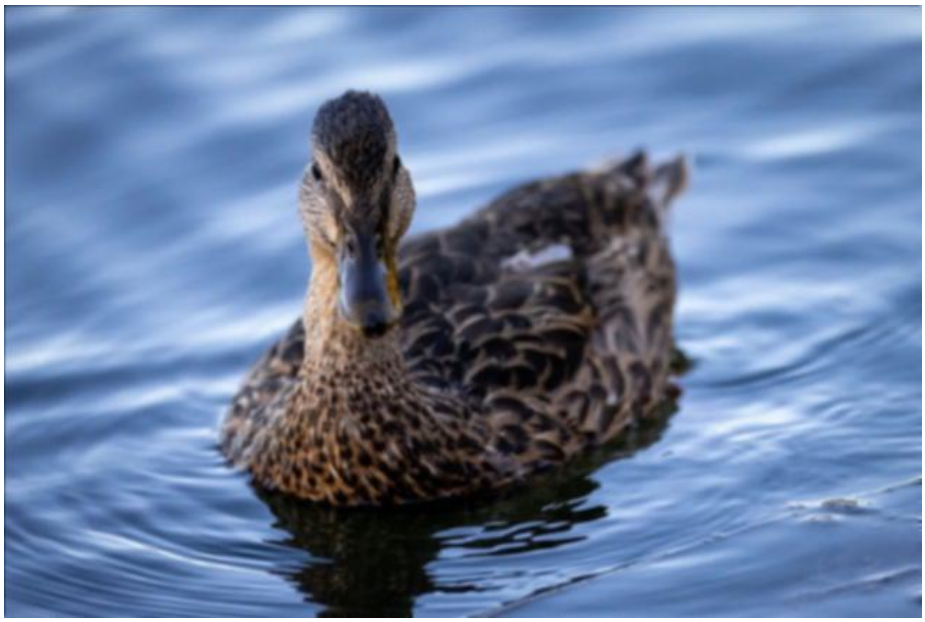

Figure 2: Quack

Figure 3: Image of duck convolved with sobel kernel

Listing 3: Convolve function

```
def convolve_im(im, kernel,
                ):
    """ A function that convolves im with kernel

    Args:
        im ([type]): [np.array of shape [H, W, 3]]
        kernel ([type]): [np.array of shape [K, K]]

    Returns:
        [type]: [np.array of shape [H, W, 3]. should be same as im]
    """
    assert len(im.shape) == 3
    assert kernel.shape[0]%2 == 1
    assert kernel.shape[1]%2 == 1

    kernel   = np.flipud(np.fliplr(kernel))
    k_center = (kernel.shape[0]-1)//2

    out_im = np.copy(im)
    padded_im = np.pad(
        np.copy(im),
        ((k_center,k_center),
        (k_center,k_center),
        (0,0)),
        'constant',
        constant_values=0)
```

```
27
28      W = im.shape[0]
29      H = im.shape[1]
30
31      start_w = k_center
32      stop_w  = W
33
34      start_h = k_center
35      stop_h  = H
36
37      ones = np.ones((kernel.shape[0],))
38      for w in range(start_w, stop_w):
39          for h in range(start_h, stop_h):
40              for c in range(0, 3):
41                  im_slice = padded_im[
42                      w-k_center:w+k_center+1,
43                      h-k_center:h+k_center+1,
44                      c]
45                  sum = np.dot(np.dot(im_slice*kernel,ones), ones)
46                  out_im[w-k_center,h-k_center,c] = sum
47      return out_im
```

Task 3

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).
If a perceptron is used, AND, OR, NOT, NOR and NAND can be represented. XOR cannot be represented.

---

b)

Explain in one sentence what a hyperparameter for a neural network is. Give two examples of a hyperparameter.
Hyperparameters are controlled parameters in the neural network that are defined in the setup of the neural network. These parameters are not updated during training.
Examples of hyperparameters are the learning rate of the neural network and data batch size.

---

c)

Why is the softmax activation functioned used in the last layer for neural networks trained to classify objects?
The output activations from a softmax function is both positive and sum up to one. As such, if one output activation increases, the others will correspondingly decrease. This means that they can be interpreted as a probability distribution. That means that the classification of objects can be thought of in terms of the probability of that object belonging to each class.

---

d)

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.
Find and report the final values for $\frac{\delta C}{\delta w_1}$, $\frac{\delta C}{\delta w_2}$, $\frac{\delta C}{\delta w_3}$, $\frac{\delta C}{\delta w_4}$, $\frac{\delta C}{\delta b_1}$, $\frac{\delta C}{\delta b_2}$.
Explain each step in the computation, such that it is clear how you compute the derivatives.

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

The neural network gives the following variables

$$x_1 = -1 \tag{3a}$$
$$x_2 = 0 \tag{3b}$$
$$x_3 = -1 \tag{3c}$$
$$x_4 = 2 \tag{3d}$$
$$w_1 = -1 \tag{3e}$$
$$w_2 = 1 \tag{3f}$$
$$w_3 = -1 \tag{3g}$$
$$w_4 = -2 \tag{3h}$$
$$b_1 = 1 \tag{3i}$$
$$b_2 = -1 \tag{3j}$$

and the following expressions for c_1, c_2:

$$c_1 = w_1 x_1 + w_2 x_2 + b_1 \tag{4a}$$
$$c_2 = w_3 x_3 + w_4 x_4 + b_2 \tag{4b}$$

Inserting values into Equation 4a and Equation 4b gives:

$$c_1 = -1 \cdot (-1) + 1 \cdot 0 + 1 \Rightarrow c_1 = 2 \tag{5a}$$
$$c_2 = -1\dot{c}(-1) + -2 \cdot 2 + -1 \Rightarrow c_2 = -4 \tag{5b}$$

Such that the cost function for this iteration is

$$C(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2 \tag{6a}$$
$$C(y, c_1, c_2) = \frac{1}{2}(y - max(c_1, c_2))^2 \tag{6b}$$
$$C(y, c_1) = \frac{1}{2}(y - c_1)^2 \tag{6c}$$
$$C(y, w_1, x_1, w_2, x_2, b_1) = \frac{1}{2}(y - w_1 x_1 + w_2 x_2 + b_1)^2 \tag{6d}$$

$$\frac{\delta C}{\delta w_1} = \frac{\delta C}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta w_1} \tag{7a}$$
$$= -x_1(y - w_1 x_1 - w_2 x_2 - b_1) \tag{7b}$$
$$\frac{\delta C}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta w_1}\Big|_{\substack{w_1=-1, x_1=-1 \\ w_2=1, x_2=0 \\ b_1=1}} = -1 \tag{7c}$$

$$\frac{\delta C}{\delta w_2} = \frac{\delta C}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta w_2} \tag{8a}$$

$$= -x_2(y - w_1x_1 - w_2x_2 - b_1) \tag{8b}$$

$$\frac{\delta C}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta w_2}\Big|_{\substack{w_1=-1, x_1=-1 \\ w_2=1, x_2=0 \\ b_1=1}} = 0 \tag{8c}$$

$$\frac{\delta C}{\delta w_3} = \frac{\delta C}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta w_3} \tag{9a}$$

$$= 0 \tag{9b}$$

$$\frac{\delta C}{\delta w_4} = \frac{\delta C}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta w_4} \tag{10a}$$

$$= 0 \tag{10b}$$

$$\frac{\delta C}{\delta b_1} = \frac{\delta C}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta b_1} \tag{11a}$$

$$= (y - w_1x_1 - w_2x_2 - b_1 \tag{11b}$$

$$\frac{\delta C}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta b_1}\Big|_{\substack{w_1=-1, x_1=-1 \\ w_2=1, x_2=0 \\ b_1=1}} = -1 \tag{11c}$$

$$\frac{\delta C}{\delta b_2} = \frac{\delta C}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta b_2} \tag{12a}$$

$$= 0 \tag{12b}$$

---

d)

Compute the updated weights w_1, w_3, and b_1 by using gradient descent and the values you found in task c. Use $\alpha = 0.1$

Letting the current iteration be 1 and using the notation θ_c^k where θ is a parameter, $(\cdot)_c$ refers to the number associated with each parameter, and $(\cdot)^k$ refers to the iteration.

Using the update law

$$\theta_c^{k+1} = \theta_c^k - \alpha \frac{\delta C}{\delta \theta_c^k} \tag{13}$$

This yields:

$$w_1^2 = w_1^1 - \alpha \frac{\delta C}{w_1^1} \Rightarrow w_1^2 = -1 - 0.1 \cdot (-1) = -1.1 \tag{14a}$$

$$w_3^2 = w_3^1 - \alpha \frac{\delta C}{w_3^1} \Rightarrow w_3^2 = -1 - 0.1 \cdot 0 = 0 \tag{14b}$$

$$b_1^2 = b_1^1 - \alpha \frac{\delta C}{b_1^1} \Rightarrow b_1^2 = 1 - 0.1 \cdot (-1) = 0.9 \tag{14c}$$

$$\tag{14d}$$

Task 4

a)

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?

The validation loss of the network with normalized image data initially has a steeper rate downwards. This means the network gets better results with less training.

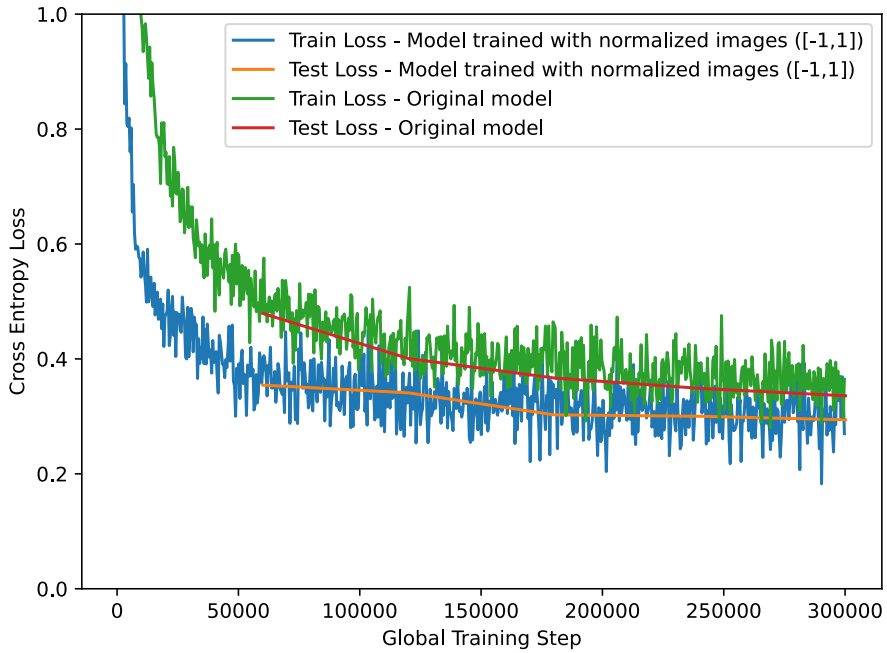

Figure 4: Plot of training and validation loss of normalized and non-normalized data set

---

b)

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).

The figures shows images that somewhat resemble the numbers 0-9.
Each weight is associated with a pixel in the input images. According to the trained network, the lighter the colour is in the output image is, the more likely it is that an input image with the same classification also will have pixels at that location.

(a) 0

(b) 1

(c) 2

(d) 3

(e) 4

(f) 5

(g) 6

(h) 7

(i) 8

(j) 9

Figure 5: Learned input weights to the last layer of the neural network visualised

c)

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.

The cross entropy loss varies widely with the learning rate set to 1. The test loss does not converge to one value for each epoch, but rather fluctuates. The results are worse because the step taken when updating the parameters of the neural network is too large and thus the neural network does not converge on the minimum of the loss function, but rather overshoots it. This can lead to undeterministic behaviour when training.

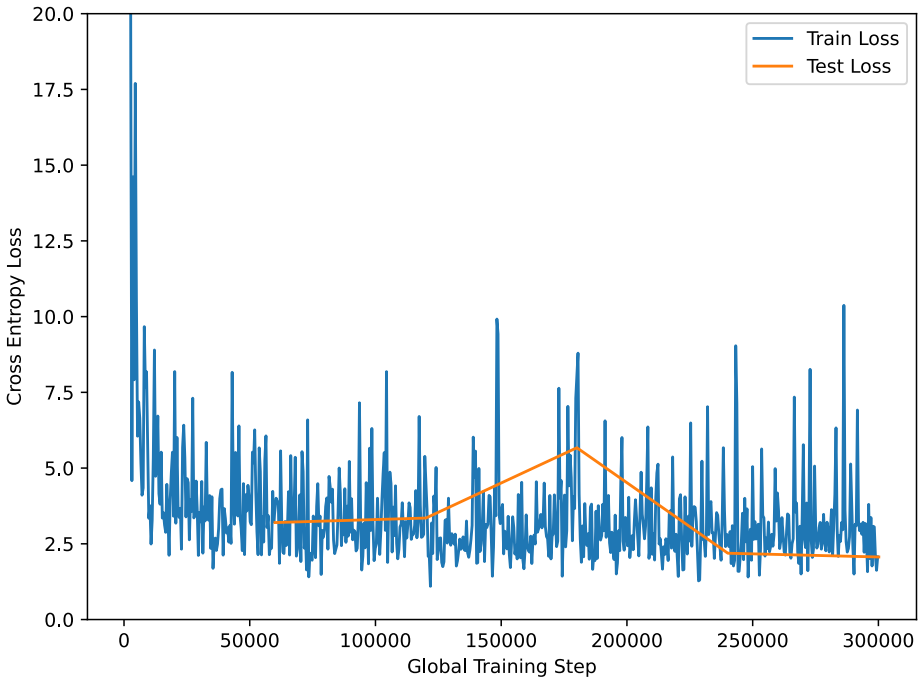

Figure 6: Plot of training and validation loss with a learning rate of 1.

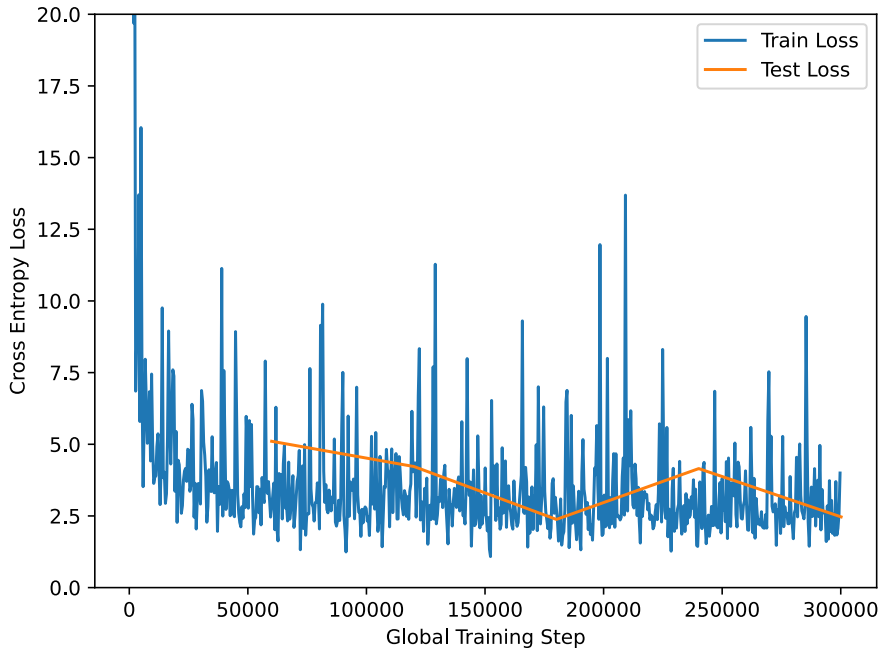

Figure 7: Plot of training and validation loss with a learning rate of 1 and a different randomisation seed.

---

d)

Include a hidden layer with 64 nodes in the network, with ReLU as the activation function for the first layer. Train this network with the same hyper-parameters 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?

The neural network with the ReLU activation function achieves better cross entropy loss than the neural network without and seems to still have a decreasing trend in the plot.

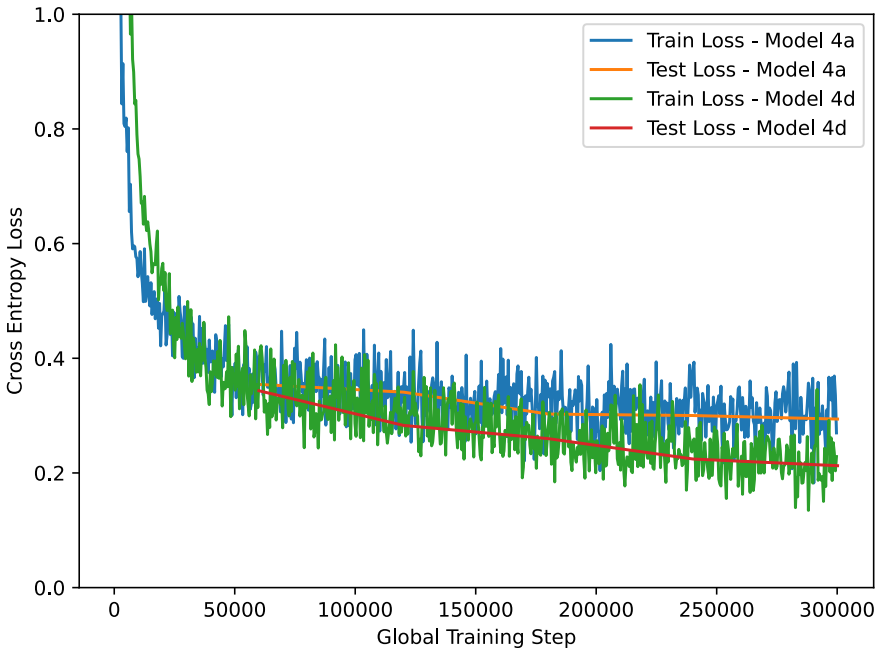

Figure 8: Plot of training and validation loss of neural network with and without layer with ReLU activation function