Task 1

a)

To *sample* is to digitize the coordinate values of a function given by the waveform of the continuous voltage output of a sensor. In practice, spacial sampling is accomplished by selecting the number of individual mechanical increments at which we activate the sensor to collect data.

b)

Quantization is digitizing the amplitude of a sampled value.

c)

One can tell that an image is high-contrast by its histogram by identifying that an even distribution across the entire domain of possible values. Meaning that the pixel Iamge uses all available intensity values.

d)

Hr(r)	**Pr(r)**	**Fr(r)**	**T(r)**
Hr(0) = 1	Pr(0) = 1/15	1/15	T(0) = 7/15
Hr(1) = 1	Pr(1) = 1/15	2/15	T(1) = 14/15
Hr(2) = 0	Pr(2) = 0	2/15	T(2) = 14/15
Hr(3) = 1	Pr(3) = 1/15	3/15	T(3) = 21/15
Hr(4) = 2	Pr(4) = 2/15	5/15	T(4) = 35/15
Hr(6) = 4	Pr(6) = 4/15	11/15	T(6) = 77/15
Hr(5) = 2	Pr(5) = 2/15	7/15	T(5) = 49/15
Hr(7) = 4	Pr(7) = 4/15	15/15	T(7) = 7

e)

Applying a log transform will widen (brighten) the low intensities and squeeze (darken) the high intensities. Effectivley lowering the dynamic range of the imgage

f)

To handle boundaries we chose reflecting the information across the edge, and on the corners we used the same pixel as on the sides of the corner.

Padded image

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

Kernel

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

Convolved image

-17	-13	12	1	-7
-3	-3	1	9	8
4	-4	-9	22	23

Task 2

a)

Figure 1: Grey Scaled Duck

b)

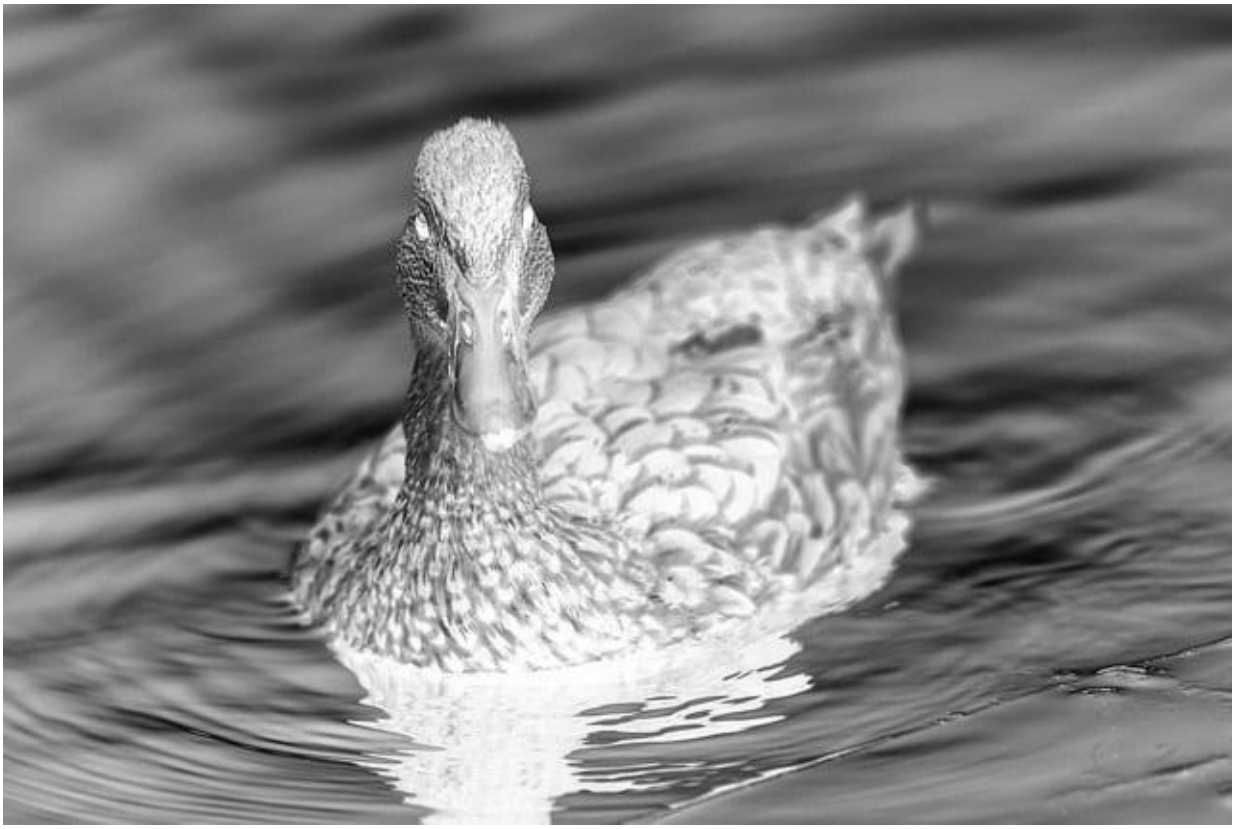

Figure 2: Inverse Grey Scaled Duck

c)

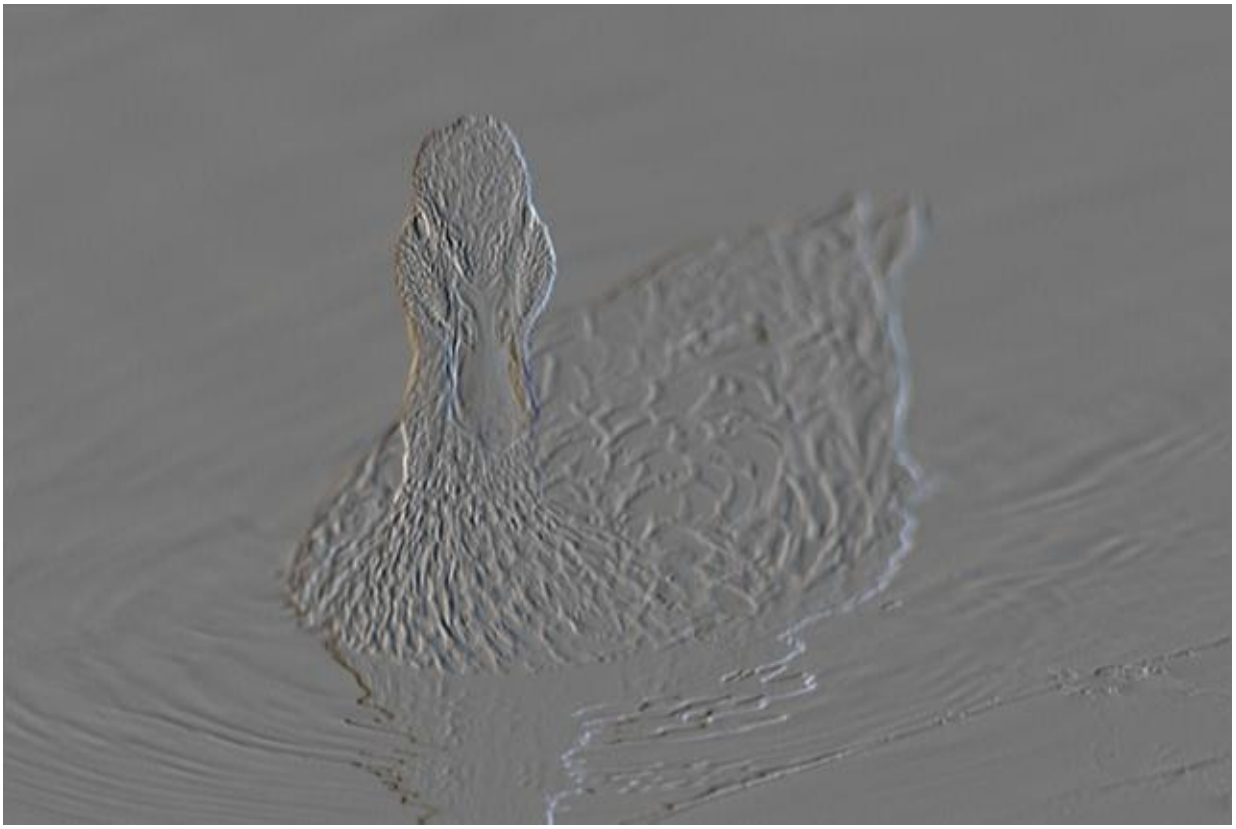

Figure 3: Sobel kernel applied to duck

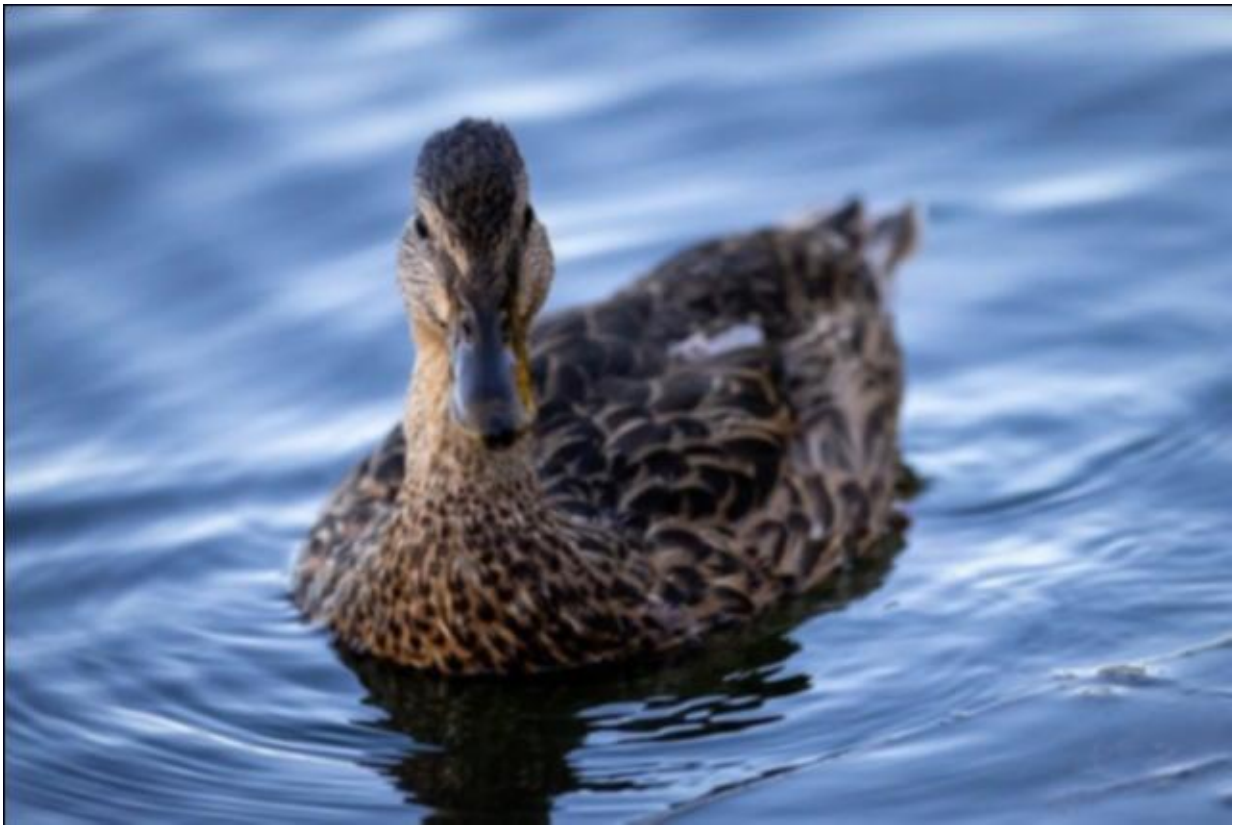

Figure 4: Smoothing kernel applied to duck

Task 3

a)

The XOR operation cannot be represented by a single-layer, as it's not linearly separable. The input space cannot be separated into two regions representing 1 and 0 using a single linear function, which by design is all that a single layer neural net can do because its only a linear combination of the inputs.

b)

Hyperparameters are essentially external configuration parameters for the model chosen before training the model. For example the number of layers, and hiden units per layer.

c)

The goal of multiclass clasification networks is to take an input and predict which one of the K classes it belongs to. To achieve this the network attempts to model a classification distribution over the k classes. The output value of each of the networks K output nodes will then be the probability that input x belongs to class {1,2,…,K}. For the probability distribution outputted by

the neural net to be valid the sum of all outputs from the k nodes must be 1 and all outputs must be in the range [0,1]. We cannot guarantee that the neural network respects these limitations, so we must use a function that maps the entire number line onto [0,1] ensuring that all the outputs of the function sum to 1. This is what softmax does

d)

Firse we list out the values asked for in the task, and below we've attached an image showing the working out by hand which explains how each of the derivatives were computed

$$\frac{\partial C}{\partial w_1} = -1 \quad , \quad \frac{\partial C}{\partial w_2} = 0 \quad , \quad \frac{\partial C}{\partial w_3} = 0 \quad , \quad \frac{\partial C}{\partial w_4} = 0 \quad , \quad \frac{\partial C}{\partial b_1} = 1 \quad , \quad \frac{\partial C}{\partial b_2} = 0 \quad ,$$

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

Forward pass

$a_1 = 1 \quad c_1 = 2 \qquad \hat{y} = 2 \qquad C = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}$

$a_2 = 0$

$a_3 = 1 \quad c_2 = -4$

$a_4 = -4$

Backward pass

$$\frac{\partial C}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}}\left(\frac{1}{2}(y-\hat{y})^2\right) = -(y-\hat{y}) = 1 \qquad \hat{y}' = 1$$

$$\frac{\partial C}{\partial c_1} = \frac{\partial \hat{y}}{\partial c_1}\frac{\partial C}{\partial \hat{y}} = \left(\frac{\partial \max(c_1, c_2) = 1}{\partial c_1}\right) \cdot 1 = 1$$

$$\frac{\partial C}{\partial c_2} = \frac{\partial \hat{y}}{\partial c_2}\frac{\partial C}{\partial \hat{y}} = \left(\frac{\partial \max(c_1, c_2) = 0}{\partial c_2}\right) \cdot 1 = 0$$

$$\frac{\partial C}{\partial b_1} = \frac{\partial c_1}{\partial b_1}\frac{\partial C}{\partial c_1} = \frac{\partial (a_1 + a_2 + b_1)}{\partial b_1} \cdot 1 = 1$$

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

$$\frac{\partial C}{\partial a_1} = \frac{\partial c_1}{\partial a_1}\frac{\partial C}{\partial c_1} = 1$$

$$\frac{\partial C}{\partial a_2} = 1$$

$$\frac{\partial C}{\partial a_3} = 0 \qquad \frac{\partial C}{\partial a_4} = 0$$

$$\frac{\partial C}{\partial w_1} = \frac{\partial a_1}{\partial w_1} \cdot \frac{\partial C}{\partial a_1} = 1 \cdot x_1 = x_1 = -1$$

Figure 5: Calculation

$$\bar{w}_1 = w_1 - \alpha \frac{\partial C}{\partial w_1} = -0.9$$

e)

$$\bar{w}_2 = w_2 - \alpha \frac{\partial C}{\partial w_2} = 0$$

$$\bar{b}_1 = b_1 - \alpha \frac{\partial C}{\partial b_1} = 0.9$$

Task 4

a)

As is clear in the plot, the model trained on normalized data outperforms the one trained on non normalized data.

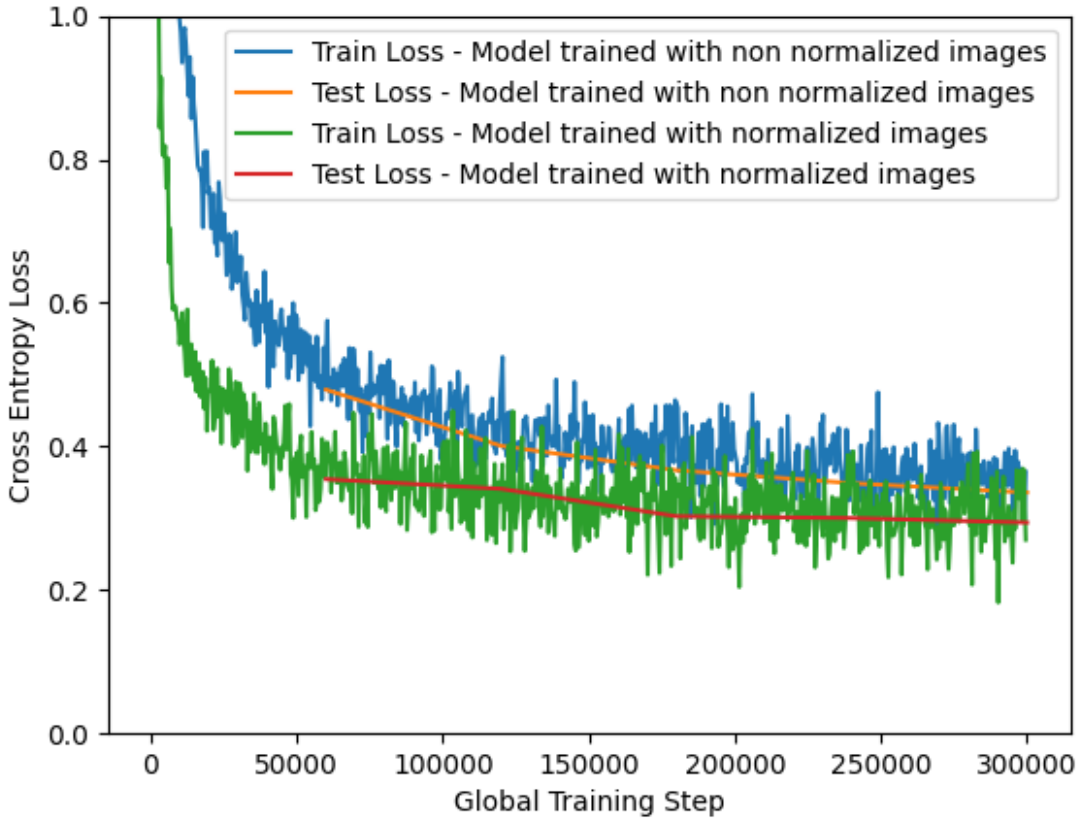

Figure 6: Normalized Data vs Unormalized Data

b)

In the Image we see that the highest weights drawing the shape of the numebr, with the lowest weights tracing the edges of the numbers. It seems that the network is learning to recognize the numbers by their shape and edges

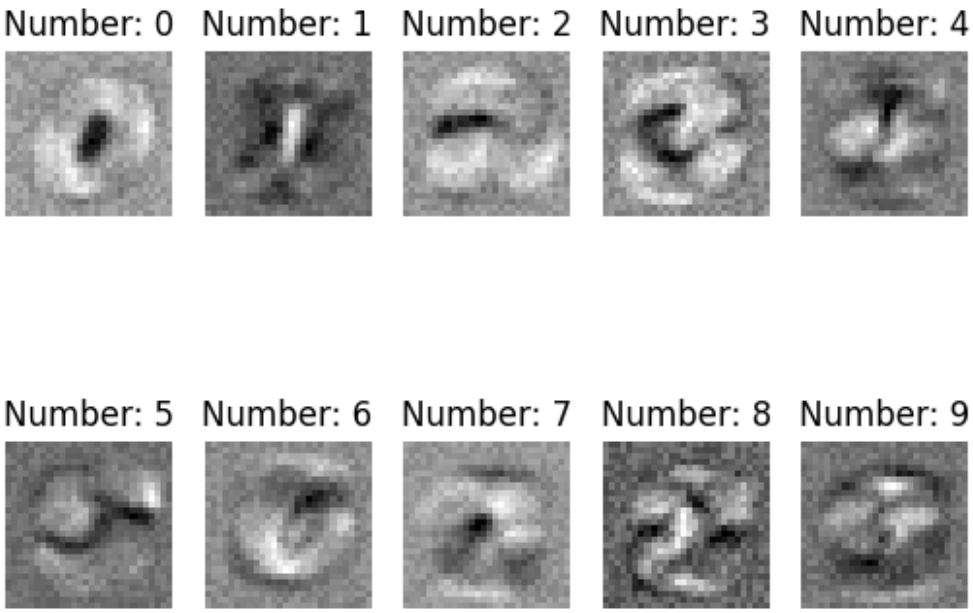

Figure 7: Weights visualized as images

c)

The network achieves worse performance than previously because with a learning rate of 1.0 the network is unable to converge on a local/global minimum. In the graph its clear from the test loss plot that the model takes wild jumps during gradient descent and is physically not able to converge to the minimum with that big step sizes.

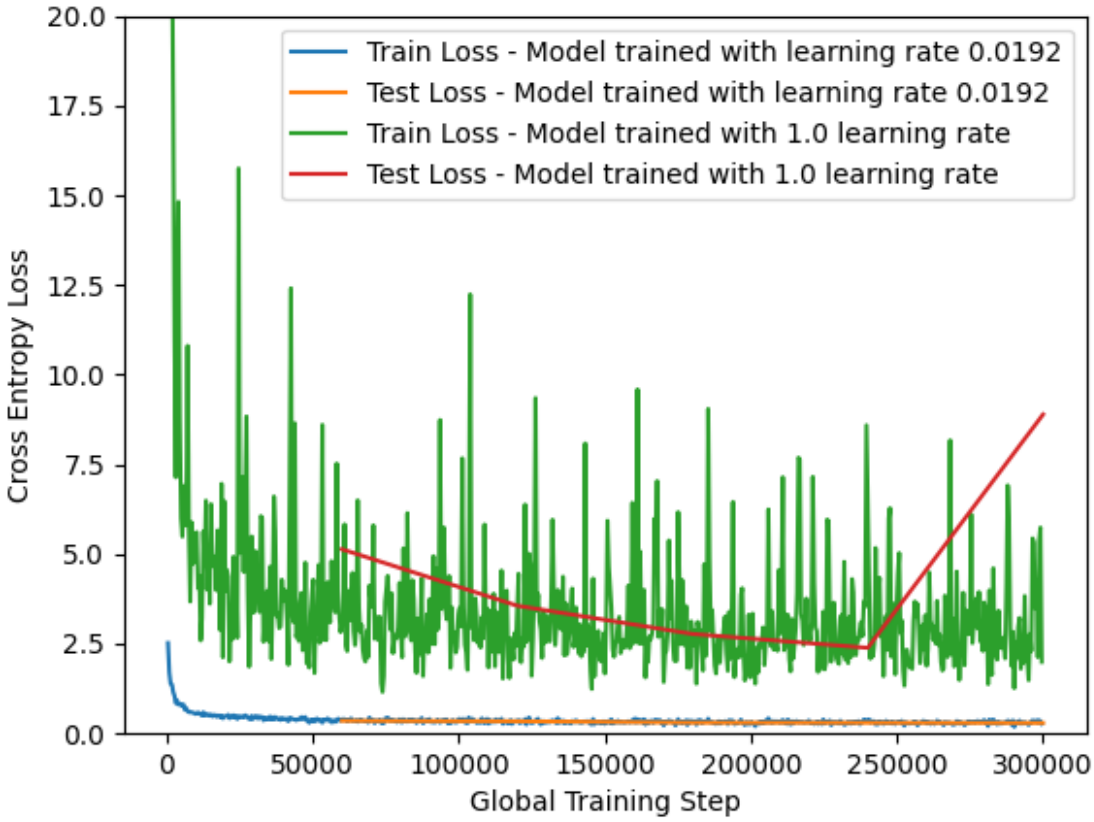

Figure 8: Model Trained with learning rate 1.0

d)

The network with the hidden layer achieves greater performance with a hidden layer bec1ause it is able to map the input to the output non linearly, Meaning it can model more complex relationships in the data. Adding the hidden layer allows the model to approximate more functions. Also it increases the capacity of the network.

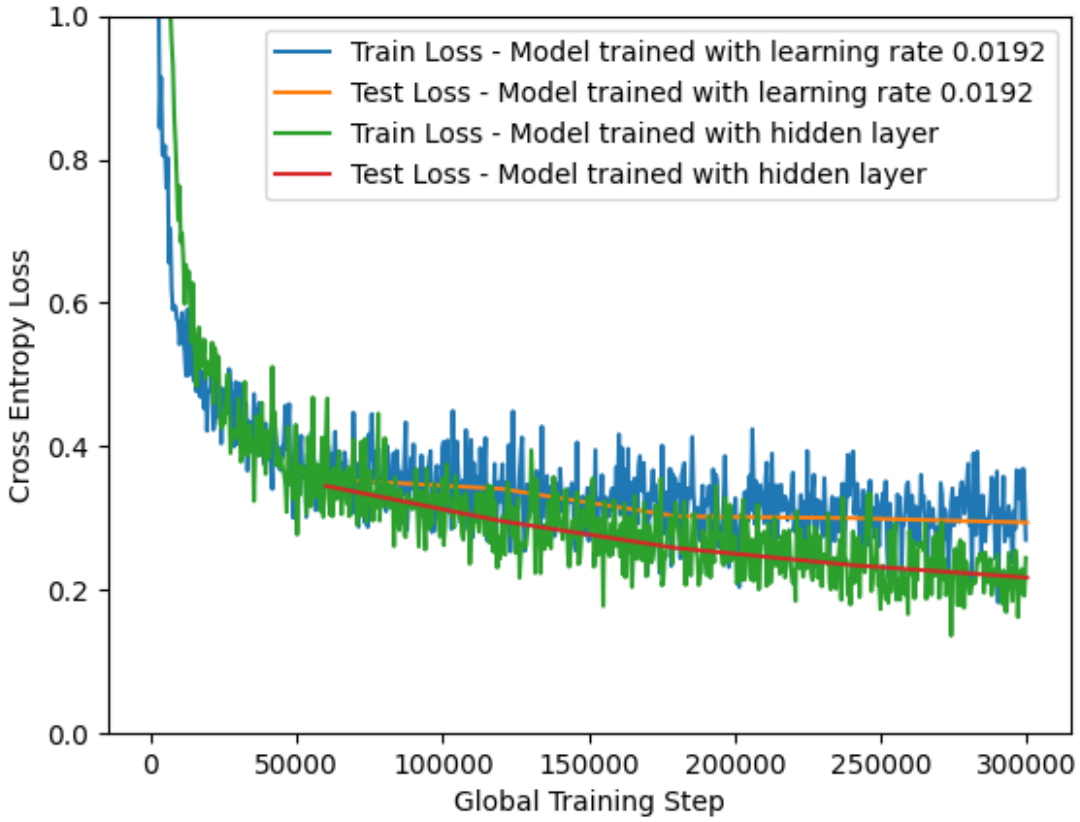

Figure 9: Model trained with hidden layer