ASTR4004 - Assignment 4

u7291368

October 2024

The GitHub repository containing all data, figures and the python notebook used to create and run the neural network is available here: https://github.com/Pat-OC/astr4004-assignment-4/tree/main.

Question 1

First define the regression problem you're solving. For example, this could involve fitting data points or using an astrophysical model to predict the CMB optical depth based on the neutral fraction of the intergalactic medium. Essentially, you'll need to specify some input data points - typically a set of x values and their corresponding outputs y. In the lecture example, $x = 1$ and $y = 1$, meaning the task was to map the input exactly to itself, i.e. $y = x$. This part accounts for 10% of the total score for Assignment 4.

Answer: Let's use data from a real astrophysical scenario. My research investigates the relationship between $\alpha \equiv M_{\rm Dyn}/M_{\rm SPS}$ where and σ_e where $M_{\rm Dyn}$ is the dynamical mass of a galaxy, $M_{\rm SPS}$ is the stellar mass from SED fitting and σ_e is the central velocity dispersion. The parameter α, is then the so-called 'IMF mismatch parameter', the ratio between the dynamical and stellar masses. For quiescent galaxies at high-z, we expect $\alpha \approx 1$ if the IMF is chosen correctly, yet observational data suggests this is not the case. Rather, a significant fraction of galaxies at $z \sim 2$ find $\alpha < 1$.

To obtain an accurate measure of the dynamical mass of a galaxy, one must be able to model the surface brightness profile as a means of inferring the stellar surface density. This is crucial to solve the Jeans equations by which one may obtain a dynamical mass estimate. A fundamental parameter in modelling the surface brightness is the half-light size or effective radius $r_{\rm eff}$. In my research I am fortunate enough to use measurements from the new JWST telescope and so an interesting point of discussion involves a comparison of the sizes of galaxies measured using JWST and HST. For the neural network described in this assignment, we will attempt to fit a linear relationship to this comparison from my Honours research.

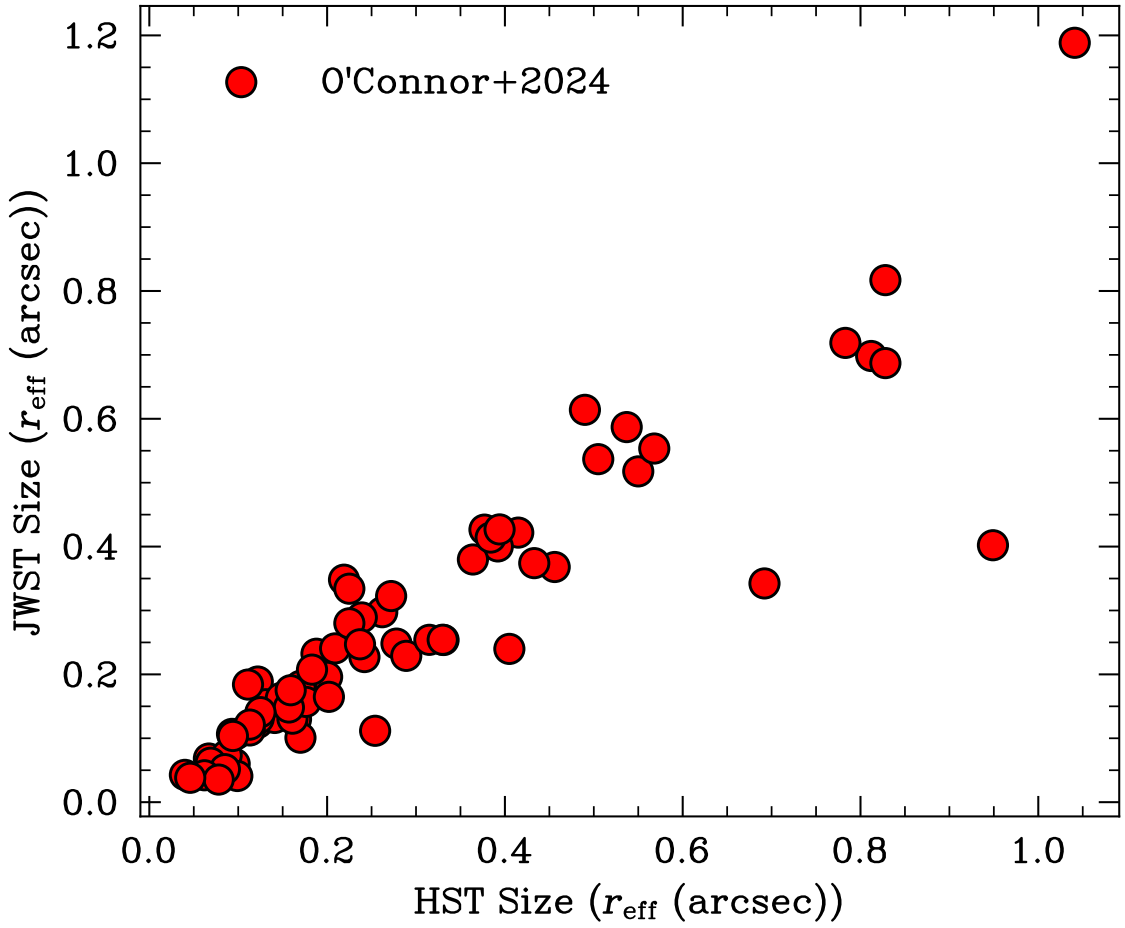

Figure 1: Measured half-light radii (r_e) of galaxies in the Blue Jay survey. The y-axis plots the size as measured using multi-gaussian expansion modelling that makes use of IMCASCADE and JWST imaging from the PRIMER survey (Cycle 1 GO 1837; PI:Dunlop), while the x-axis plots the r_e measured by van der Wel et al. (2014) using HST data.

Question 2

Next, outline your network architecture, including a schematic that shows how the neurons are distributed and connected. Remember, you need at least 5 neurons. This section is worth 10%.

Answer: Let's now define the architecture for our neural network. We will use a feed-foraward neural network (FFN) with the following structure:

- **Input Layer:** Takes in the x_i data points, array of length i.
- **Hidden Layer:** with k neurons, connected to the inputs. We will use a ReLU activation function. This is denoted layer $M = 0$.
- **Output Layer:** Outputs a single value for the regression model; we will also use a ReLU activation function, and this layer is denoted $M = 1$. Weights connecting the hidden layer to the output layer are denoted with the subscript j, where j goes from 1 to k where k is the number of neurons.

Let's expand on this: In terms of notation, we denote the layer by the superscript M where $M = 0$ is the hidden layer and 1 is the output layer.

Step 1: The input layer is the data points along x. In this case, the data points are represented by x_i.

Step 2: The k neurons in the hidden layer take the inputs x_i and compute a weighted sum over weights $w_k^{(0)}$ with the addition of a bias $b_k^{(0)}$ such that

$$z_k^{(0)} = w_k^{(0)} x_i + b_k^{(0)} \tag{1}$$

Or in vector notation,

$$\mathbf{Z}^{(0)} = x_i \mathbf{W}^{(0)} + \mathbf{B}^{(0)} \tag{2}$$

where the weights $\mathbf{W}^{(0)}$ and $\mathbf{B}^{(0)}$ have shapes $k \times 1$. The activation function then acts on z_k, in this case the ReLU function, which gives the activation $a_k^{(0)} = \text{ReLU}(z_k^{(0)})$. Or again in vector notation, we arrive at the activation $\mathbf{A}^{(0)}$.

Step 3: The activations $a_k^{(0)}$ of the hidden layer then act as the inputs to the output layer such that

$$z^{(1)} = w_j^{(1)} a_k^{(0)} + b^{(1)} \tag{3}$$

and therefore the activation for this layer is $y^{(1)} = \sigma\left(z^{(1)}\right)$. In vector notation we find

$$z^{(1)} = \mathbf{W}^{(1)} \mathbf{A}^{(0)} + b^{(1)} \tag{4}$$

where the matrix $\mathbf{W}^{(1)}$ has shape $1 \times k$ and $\mathbf{A}^{(0)}$ has shape $k \times 1$. The predicted value $y^{(1)}$ is then given by the activation of the output layer $y^{(1)} = a^{(1)} = \text{ReLU}(z^{(1)})$. A schematic of the architecture is shown in Fig. 2.

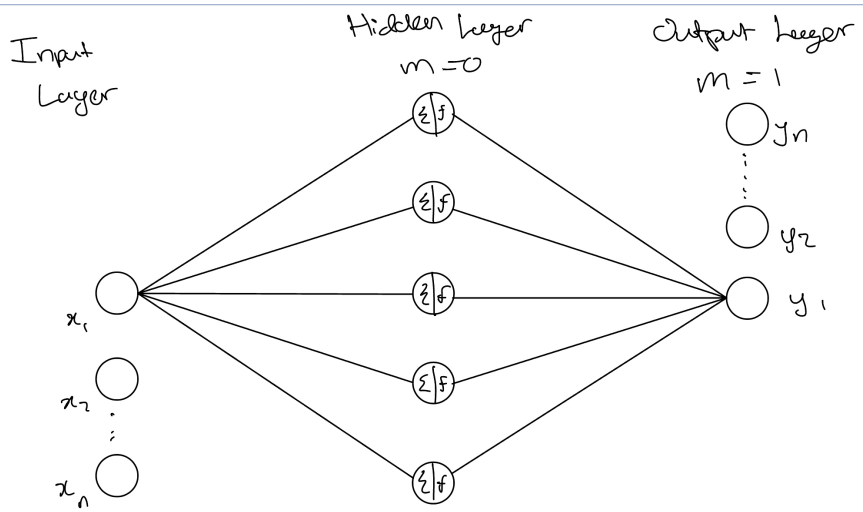

Figure 2: Neural network architecture.

Question 3

The third step is to derive the gradient for all neurons, specifically dL/dw_i and dL/db_i. This step is worth 30%. Be sure to apply the chain rule and keep in mind that your neurons may have multiple inputs, unlike the simpler example provided in the lecture.

Answer: We now wish to define the gradients for all neurons using the loss function which his the mean squared error (MSE) $L = 0.5(\hat{y} - y)^2$ where $\hat{y}$ is the predicted value from the neural network and y is the true value.

We first compute the gradient of the loss function with respect to the weights connecting the hidden layer to the output layer

$$\frac{\partial L}{\partial w_j^{(1)}} = \frac{\partial L}{\partial y^{(1)}} \frac{\partial y^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial w_j^{(1)}} = (y^{(1)} - y) \cdot a_k^{(0)} \tag{5}$$

since $\partial y^{(1)}/\partial z^{(1)} = 1$ for the ReLU(z) function where $z \geq 0$. Then, the gradient of the loss function with respect to the bias connecting the hidden layer to the output layer

$$\frac{\partial L}{\partial b^{(1)}} = \frac{\partial L}{\partial y^{(1)}} \frac{\partial y^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial b^{(1)}} = (y^{(1)} - y). \tag{6}$$

Now let's compute the gradient of the loss function with respect to the weights connecting the input layer to the hidden layer

$$\frac{\partial L}{\partial w_k^{(0)}} = \frac{\partial L}{\partial y^{(1)}} \frac{\partial y^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a_k^{(0)}} \frac{\partial a_k^{(0)}}{\partial z_k^{(0)}} \frac{\partial z_k^{(0)}}{\partial w_k^{(0)}} = (y^{(1)} - y) \cdot w_j^{(1)} \cdot x_i \tag{7}$$

and with respect to the bias

$$\frac{\partial L}{\partial b_k^{(0)}} = \frac{\partial L}{\partial y^{(1)}} \frac{\partial y^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a_k^{(0)}} \frac{\partial a_k^{(0)}}{\partial z_k^{(0)}} \frac{\partial z_k^{(0)}}{\partial b_k^{(0)}} = (y^{(1)} - y) \cdot w_j^{(1)}. \tag{8}$$

Additionally, for computational efficiency and to ensure the gradients do not become exceptionally large leading to an unstable network, we divide the gradients by m where m is the number of data points. In this way, the gradient is an average gradient for the training data.

Question 4

In the fourth step, show the training process in action by listing the weights, biases, gradients, outputs and losses over 10 steps. You should observe the loss decreasing monotonically if you set the learning rate to a small value, such as 0.1*. This part is worth 25%.*

Answer: *See the write up of the code for the neural network.* We initialise random weights for $\mathbf{W}^{(0)}$ and $\mathbf{W}^{(1)}$, while the initial biases $\mathbf{B}^{(0)}, b^{(1)}$ are set to 0. The weights w are then updated according to the usual

$$w = w - \eta \nabla L(w) \tag{9}$$

where η is the learning rate and $L(w)$ is the loss. Fig. 3 shows the fit to the data using the neural network where the learning rate $\eta = 0.1$, the number of steps is 10 and 5 neurons are used (5 neurons are used for all future runs of the network).

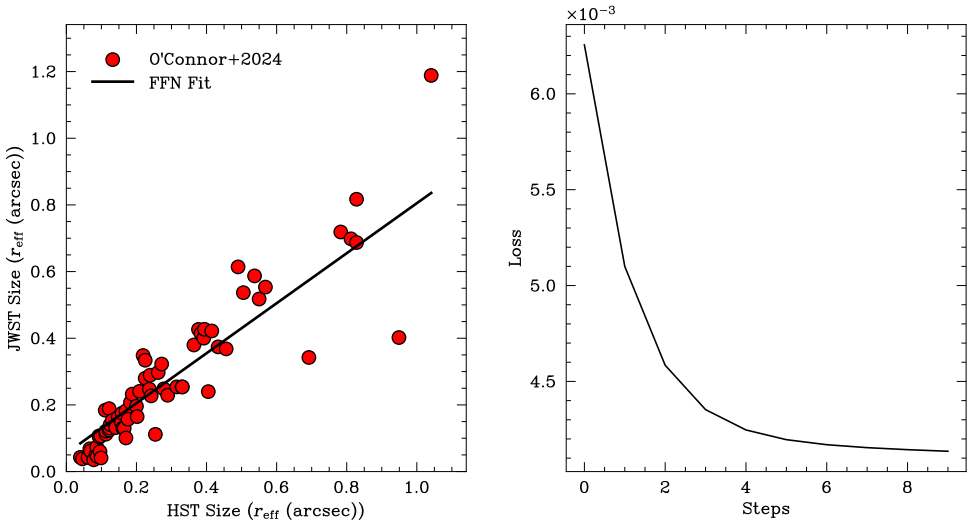

Figure 3: *(Left)* HST size vs. JWST size. The solid black line indicates the fit using the FFN. *(Right)* the loss as a function of steps, clearly the loss is decreasing with the number of steps for $\eta = 0.1$.

We now compute the loss for only 1 data point (for the sake of displaying the results in a table) and show that it is monotonically decreasing for $\eta = 0.1$ (see Fig. 4).

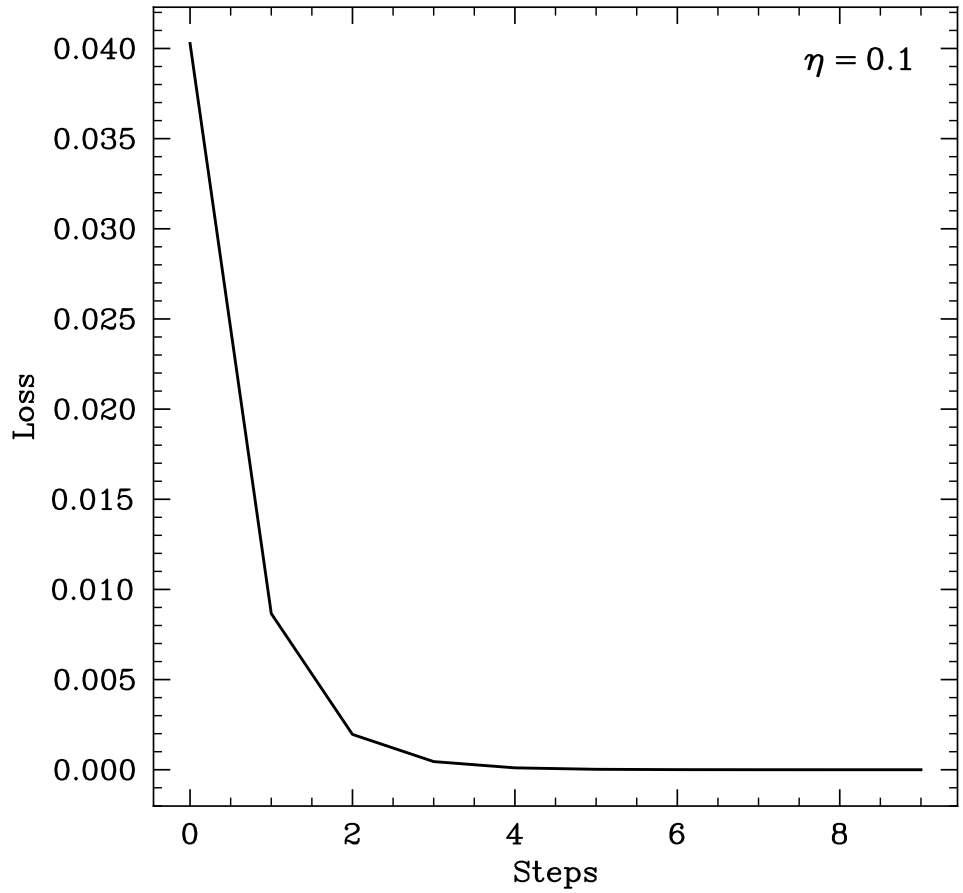

Figure 4: The loss as a function of steps for $\eta = 0.1$. Evidently, the loss is monotonically decreasing with the number of steps.

Finally, in the following tables (Table 1, 2, 3) we show the results of weight, bias, gradient, output and loss calculations for 10 steps for $\eta = 0.1$ (apologies the tables are annoyingly wide so they are landscape across the next few pages).

Step	$\mathbf{W_0} = w_k^{(0)} = (w_1, w_2, w_3, w_4, w_5)$	$\mathbf{B_0} = b_k^{(0)} = (b_1, b_2, b_3, b_4, b_5)$	$\mathbf{W_1} = w_j^{(1)} = (w_1, w_2, w_3, w_4, w_5)$	$b^{(1)}$
0	0.0979,0.4837,0.2784,0.2084,0.0556	0.,0.,0.,0.,0.	0.7456,0.7667,0.291 ,0.8184,0.4924	0.
1	0.0778,0.463 ,0.2706,0.1864,0.0423	-0.0212,-0.0218,-0.0083,-0.0232,-0.014	0.743 ,0.7536,0.2835,0.8128,0.4909	-0.0284
2	0.0685,0.4536,0.2671,0.1762,0.0362	-0.0309,-0.0317,-0.012 ,-0.0339,-0.0204	0.7423,0.7481,0.2803,0.8107,0.4905	-0.0415
3	0.0641,0.4492,0.2654,0.1714,0.0333	-0.0356,-0.0364,-0.0137,-0.039 ,-0.0235	0.7421,0.7456,0.2787,0.8099,0.4905	-0.0478
4	0.062 ,0.447 ,0.2646,0.1691,0.0319	-0.0378,-0.0386,-0.0146,-0.0414,-0.025	0.742 ,0.7445,0.278 ,0.8095,0.4904	-0.0508
5	0.061 ,0.446 ,0.2642,0.168 ,0.0312	-0.0389,-0.0397,-0.015 ,-0.0426,-0.0257	0.742 ,0.7439,0.2777,0.8093,0.4904	-0.0523
6	0.0605,0.4455,0.264 ,0.1675,0.0309	-0.0394,-0.0402,-0.0152,-0.0432,-0.026	0.7419,0.7436,0.2775,0.8093,0.4904	-0.053
7	0.0602,0.4453,0.2639,0.1672,0.0307	-0.0397,-0.0405,-0.0153,-0.0434,-0.0262	0.7419,0.7435,0.2774,0.8092,0.4904	-0.0533
8	0.0601,0.4452,0.2639,0.1671,0.0306	-0.0398,-0.0406,-0.0153,-0.0436,-0.0263	0.7419,0.7435,0.2774,0.8092,0.4904	-0.0535
9	0.0601,0.4451,0.2639,0.167 ,0.0306	-0.0398,-0.0406,-0.0153,-0.0436,-0.0263	0.7419,0.7434,0.2774,0.8092,0.4904	-0.0535

Table 1: Weights and biases after each training step for $\eta = 0.1$, rounded to 4 decimal places, for a single input x.

Step	$dL/d\mathbf{W_0}$	$dL/d\mathbf{B_0}$	$dL/d\mathbf{W_1}$	dL/dB_1
0	0.2008, 0.2065, 0.0784, 0.2204, 0.1326	0.2116, 0.2176, 0.0826, 0.2323, 0.1397	0.0264,0.1303,0.075 ,0.0561,0.015	0.2838
1	0.0928, 0.0941, 0.0354, 0.1015, 0.0613	0.0978, 0.0992, 0.0373, 0.107 , 0.0646	0.0069,0.055 ,0.0327,0.0202,0.0034	0.1316
2	0.0441, 0.0444, 0.0166, 0.0482, 0.0291	0.0465, 0.0468, 0.0175, 0.0507, 0.0307	0.0021,0.025 ,0.0151,0.0083,0.0009	0.0626
3	0.0212, 0.0213, 0.008 , 0.0231, 0.014	0.0223, 0.0224, 0.0084, 0.0243, 0.0147	0.0008,0.0117,0.0072,0.0037,0.0002	0.0301
4	0.0102, 0.0102, 0.0038, 0.0111, 0.0067	0.0108, 0.0108, 0.004 , 0.0117, 0.0071	0.0003,0.0056,0.0034,0.0017,0.0001	0.0145
5	0.0049, 0.0049, 0.0018, 0.0054, 0.0033	0.0052, 0.0052, 0.0019, 0.0057, 0.0034	0.0001,0.0027,0.0017,0.0008,0.	0.007
6	0.0024, 0.0024, 0.0009, 0.0026, 0.0016	0.0025, 0.0025, 0.0009, 0.0027, 0.0017	0.0001,0.0013,0.0008,0.0004,0.	0.0034
7	0.0012, 0.0012, 0.0004, 0.0013, 0.0008	0.0012, 0.0012, 0.0005, 0.0013, 0.0008	0. ,0.0006,0.0004,0.0002,0.	0.0016
8	0.0006, 0.0006, 0.0002, 0.0006, 0.0004	0.0006, 0.0006, 0.0002, 0.0006, 0.0004	0. ,0.0003,0.0002,0.0001,0.	0.0008
9	0.0003, 0.0003, 0.0001, 0.0003, 0.0002	0.0003, 0.0003, 0.0001, 0.0003, 0.0002	0. ,0.0001,0.0001,0. ,0.	0.0004

Table 2: Gradients after each training step for $\eta = 0.1$, rounded to 4 decimal places, for a single input x

Step	$\mathbf{A_0} = a_k^{(0)} = (a_1, a_2, a_3, a_4, a_5)$	$y^{(1)}$	Loss
0	0.0929, 0.459 , 0.2642, 0.1978, 0.0528	0.6859	0.0403
1	0.0527, 0.4177, 0.2485, 0.1537, 0.0262	0.5337	0.0087
2	0.0341, 0.3988, 0.2414, 0.1333, 0.0139	0.4647	0.0020
3	0.0253, 0.3899, 0.2381, 0.1237, 0.0081	0.4322	0.0005
4	0.021 , 0.3856, 0.2365, 0.1191, 0.0053	0.4166	0.0001
5	0.019 , 0.3836, 0.2358, 0.1168, 0.0039	0.4091	0.0000
6	0.018 , 0.3826, 0.2354, 0.1157, 0.0033	0.4055	0.0000
7	0.0175, 0.3821, 0.2352, 0.1152, 0.003	0.4037	0.0000
8	0.0173, 0.3819, 0.2351, 0.115 , 0.0028	0.4029	0.0000
9	0.0172, 0.3818, 0.2351, 0.1149, 0.0027	0.4025	0.0000

Table 3: Outputs and losses after each training step for $\eta = 0.1$, rounded to 4 decimal places, for a single input x

Question 5

Finally, increase your learning rate so that the loss does not decrease consistently. This demonstrates the importance of choosing an appropriate learning rate, and it ensures that your chosen problem and network are not overly simplistic. This step is worth the remaining 25%*.*

Answer: We again compute the loss for only 1 data point (for the sake of displaying the results in a table) and now show that it is not consistently decreasing when the learning rate is not chosen appropriately (i.e. the learning rate is too high). Fig. 5 plots the results when $\eta = 0.6$ and demonstrates that a learning rate is too high can cause the loss to increase rather than consistently decrease.

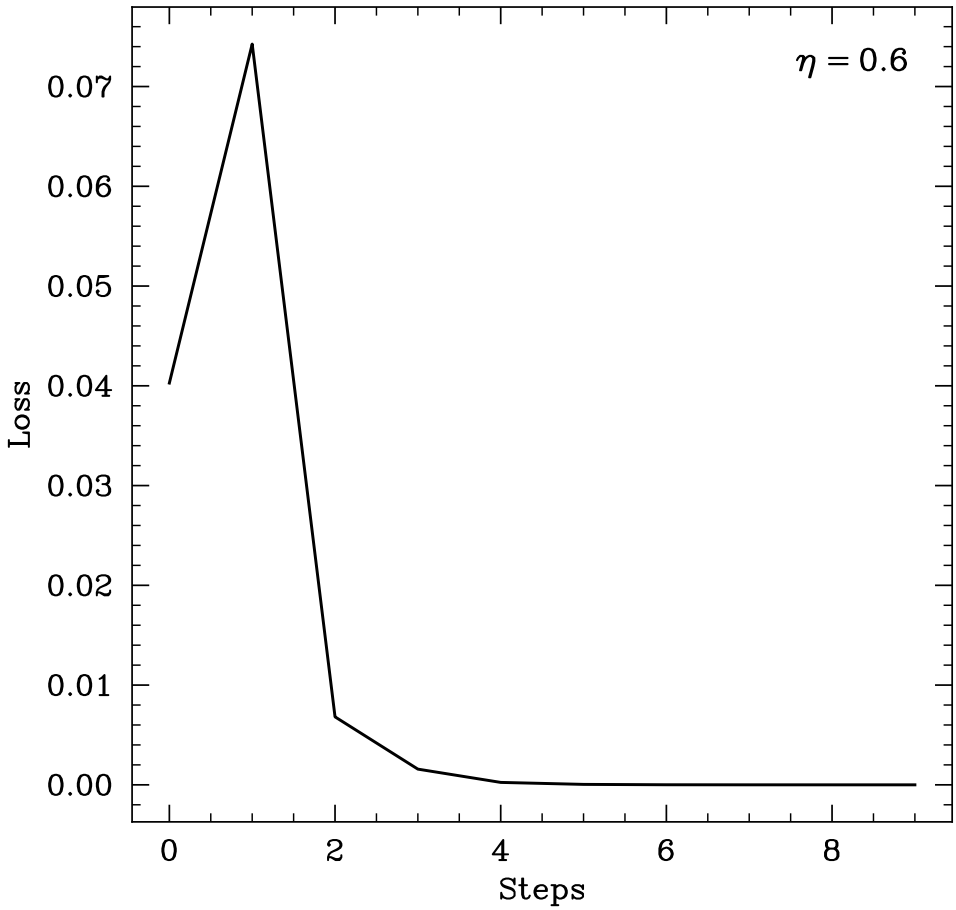

Figure 5: The loss as a function of steps for $\eta = 0.6$. When the learning rate is set too high, i.e. $\eta = 0.6$, the loss does not consistently decrease.

Again, in the following tables (Table 4, 5, 6) we show the results of weight, bias, gradient, output and loss calculations for 10 steps for $\eta = 0.6$.

Step	$\mathbf{W_0} = w_k^{(0)} = (w_1, w_2, w_3, w_4, w_5)$	$\mathbf{B_0} = b_k^{(0)} = (b_1, b_2, b_3, b_4, b_5)$	$\mathbf{W_1} = w_j^{(1)} = (w_1, w_2, w_3, w_4, w_5)$	$b^{(1)}$
0	0.0979,0.4837,0.2784,0.2084,0.0556	0.,0.,0.,0.,0.	0.7456,0.7667,0.291 ,0.8184,0.4924	0.
1	-0.0226, 0.3598, 0.2314, 0.0762,-0.024	-0.127 ,-0.1306,-0.0496,-0.1394,-0.0838	0.7298,0.6885,0.246 ,0.7847,0.4834	-0.1703
2	-0.0226, 0.5109, 0.2854, 0.0762,-0.024	-0.127 , 0.0286, 0.0073,-0.1394,-0.0838	0.7298,0.7373,0.2853,0.7847,0.4834	0.0609
3	-0.0226, 0.4619, 0.2664, 0.0762,-0.024	-0.127 ,-0.023 ,-0.0127,-0.1394,-0.0838	0.7298,0.7013,0.2659,0.7847,0.4834	-0.0091
4	-0.0226, 0.4843, 0.2749, 0.0762,-0.024	-0.127 , 0.0006,-0.0037,-0.1394,-0.0838	0.7298,0.7153,0.274 ,0.7847,0.4834	0.0246
5	-0.0226, 0.4753, 0.2715, 0.0762,-0.024	-0.127 ,-0.0089,-0.0073,-0.1394,-0.0838	0.7298,0.7092,0.2705,0.7847,0.4834	0.0113
6	-0.0226, 0.4791, 0.2729, 0.0762,-0.024	-0.127 ,-0.0048,-0.0058,-0.1394,-0.0838	0.7298,0.7117,0.272 ,0.7847,0.4834	0.017
7	-0.0226, 0.4775, 0.2723, 0.0762,-0.024	-0.127 ,-0.0065,-0.0064,-0.1394,-0.0838	0.7298,0.7106,0.2714,0.7847,0.4834	0.0147
8	-0.0226, 0.4782, 0.2726, 0.0762,-0.024	-0.127 ,-0.0058,-0.0062,-0.1394,-0.0838	0.7298,0.7111,0.2716,0.7847,0.4834	0.0156
9	-0.0226, 0.4779, 0.2725, 0.0762,-0.024	-0.127 ,-0.0061,-0.0063,-0.1394,-0.0838	0.7298,0.7109,0.2715,0.7847,0.4834	0.0152

Table 4: Weights and biases after each training step for $\eta = 0.6$, rounded to 4 decimal places, for a single input x.

Step	$dL/d\mathbf{W_0}$	$dL/d\mathbf{B_0}$	$dL/d\mathbf{W_1}$	dL/dB_1
0	0.2008, 0.2065, 0.0784, 0.2204, 0.1326	0.2116, 0.2176, 0.0826, 0.2323, 0.1397	0.0264,0.1303,0.075 ,0.0561,0.015	0.2838
1	0. , -0.2518, -0.09 , 0. , 0.	0. , -0.2653, -0.0948, 0. , 0.	0. ,-0.0813,-0.0655, 0. , 0.	-0.3854
2	0. , 0.0817, 0.0316, 0. , 0.	0. , 0.0861, 0.0333, 0. , 0.	0. ,0.0599,0.0325,0. ,0.	0.1167
3	0. , -0.0374, -0.0142, 0. , 0.	0. , -0.0394, -0.0149, 0. , 0.	0. ,-0.0233,-0.0135, 0. , 0.	-0.0561
4	0. , 0.015 , 0.0057, 0. , 0.	0. , 0.0158, 0.006 , 0. , 0.	0. ,0.0102,0.0057,0. ,0.	0.0221
5	0. , -0.0064, -0.0024, 0. , 0.	0. , -0.0067, -0.0026, 0. , 0.	0. ,-0.0042,-0.0024, 0. , 0.	-0.0095
6	0. , 0.0027, 0.001 , 0. , 0.	0. , 0.0028, 0.0011, 0. , 0.	0. ,0.0018,0.001 ,0. ,0.	0.0039
7	0. , -0.0011, -0.0004, 0. , 0.	0. , -0.0012, -0.0004, 0. , 0.	0. ,-0.0007,-0.0004, 0. , 0.	-0.0017
8	0. , 0.0005, 0.0002, 0. , 0.	0. , 0.0005, 0.0002, 0. , 0.	0. ,0.0003,0.0002,0. ,0.	0.0007
9	0. , -0.0002, -0.0001, 0. , 0.	0. , -0.0002, -0.0001, 0. , 0.	0. ,-0.0001,-0.0001, 0. , 0.	-0.0003

Table 5: Gradients after each training step for $\eta = 0.6$, rounded to 4 decimal places, for a single input x.

Step	$\mathbf{A_0} = a_k^{(0)} = (a_1, a_2, a_3, a_4, a_5)$	$y^{(1)}$	Loss
0	0.0929, 0.459 , 0.2642, 0.1978, 0.0528	0.6859	0.0403
1	0. , 0.2109, 0.17 , 0. , 0.	0.0167	0.0743
2	0. , 0.5135, 0.2782, 0. , 0.	0.5189	0.0068
3	0. , 0.4153, 0.2402, 0. , 0.	0.346	0.0016
4	0. , 0.4602, 0.2572, 0. , 0.	0.4242	0.0002
5	0. , 0.4422, 0.2503, 0. , 0.	0.3926	0.0000
6	0. , 0.4498, 0.2532, 0. , 0.	0.406	0.0000
7	0. , 0.4467, 0.252 , 0. , 0.	0.4005	0.0000
8	0. , 0.448 , 0.2525, 0. , 0.	0.4028	0.0000
9	0. , 0.4474, 0.2523, 0. , 0.	0.4018	0.0000

Table 6: Outputs and losses after each training step for $\eta = 0.6$, rounded to 4 decimal places, for a single input x.

References

van der Wel A., et al., 2014, , 788, 28