

Gunnar Rätsch Benchmark Datasets

K S Ananth

Task 3

1 Overview:

Gunnar Rätsch's benchmark datasets are widely used in the machine learning community for evaluating algorithms. There are 13 datasets in total which are considered as the benchmark datasets. We shall use these datasets to evaluate neural networks and the back propagation algorithm. For doing so we shall train the neural network with 10 different instances of the dataset and figure the test_set_error of the trained model for all 10 instances and calculate the average_test_error \pm standard_deviation. The datasets are can be found at https://github.com/tdiethe/gunnar_raetsch_benchmark_datasets and is downloadable. All the datasets are in one single .mat file which can be extracted and used.

Some details before the start of the training and analysis: all the models use the mean square error function as the cost function (cross entropy needs data modification) and as all the datasets have output in the form of 1 or -1, Tanh function is used as the activation function for all layers and models.

2 Datasets:

2.1 Dataset Information:

Dataset	Train Set	Inputs	Test Set
Banana	400	2	4900
Breast Cancer	200	9	77
Diabetes	468	8	300
German	700	20	300
Heart	170	13	100
Image	1300	18	1010
Ringnorm	400	20	7000
Splice	1000	60	2175
Thyroid	140	5	75
Titanic	150	3	2051
Twonorm	400	20	7000
Waveform	400	21	4600

Table 1: Datasets Details

2.2 Loss vs Number of Epochs:

The images 1 and 2 shows the loss vs number of epochs for the 12 datasets shown above.

These plots are a single instance out of the 10 total done to get the average test set error and the standard deviation of the neural network. All the models here are run for 500 epochs, with different learning rates and different neural networks. The details regarding this is in Table: 2.

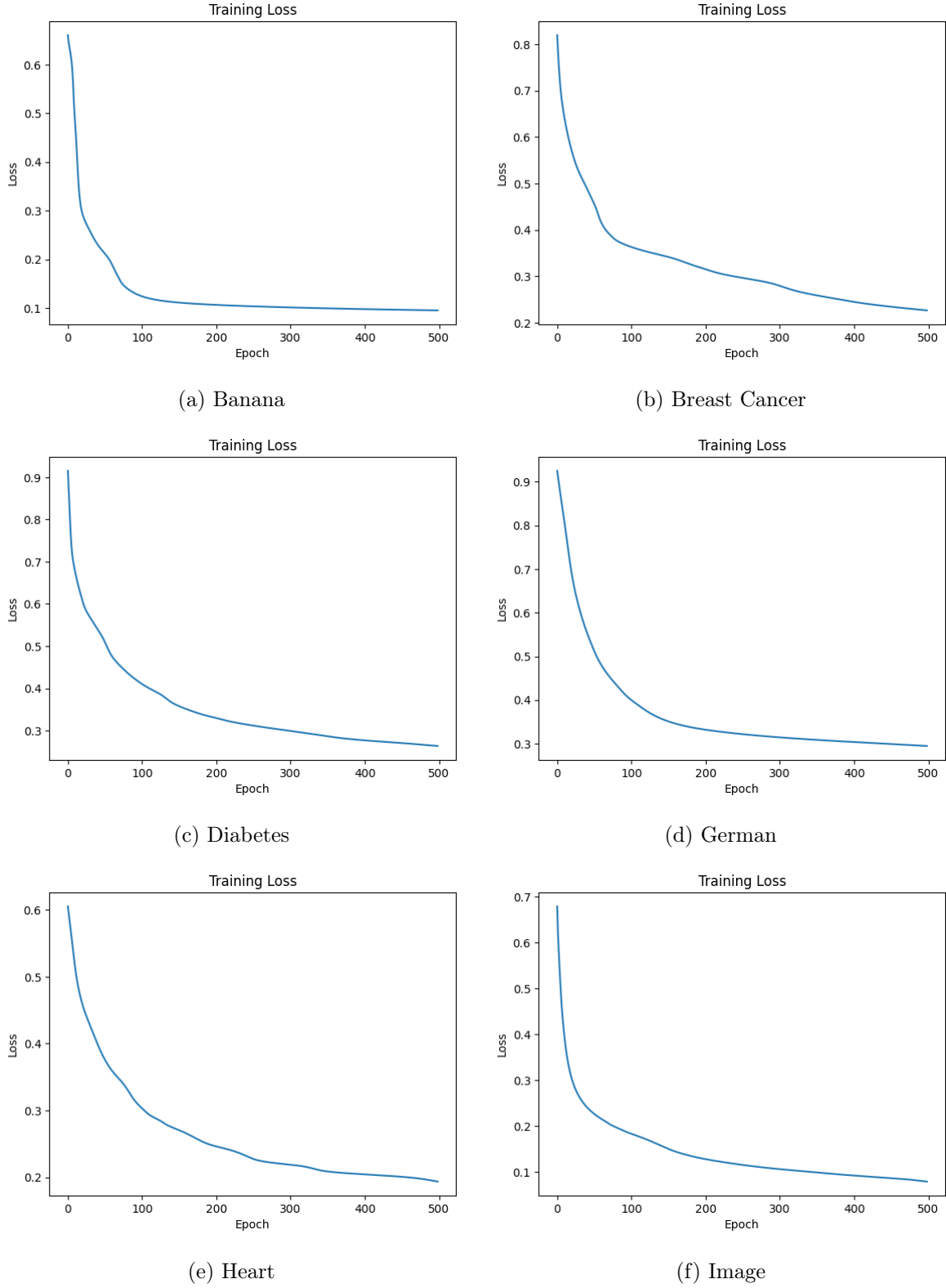
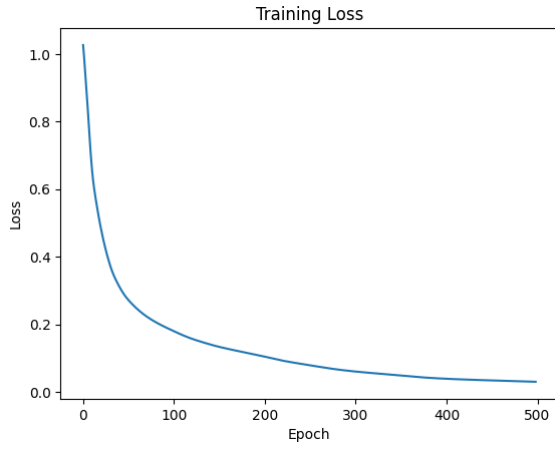
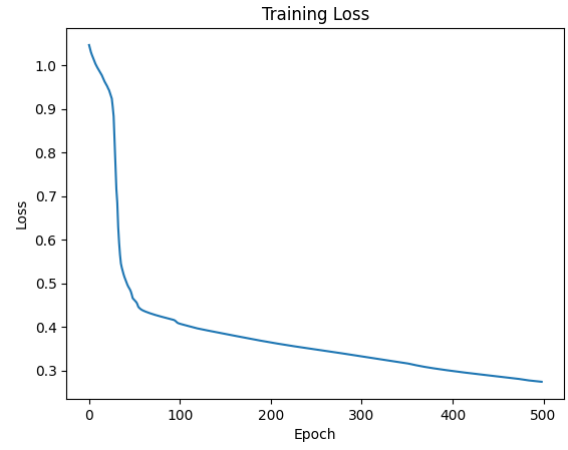


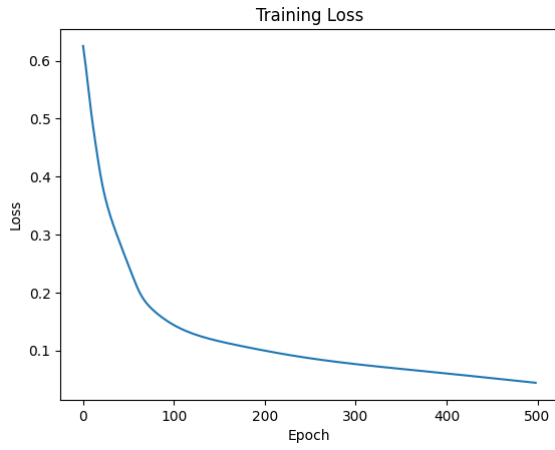
Figure 1: Datasets



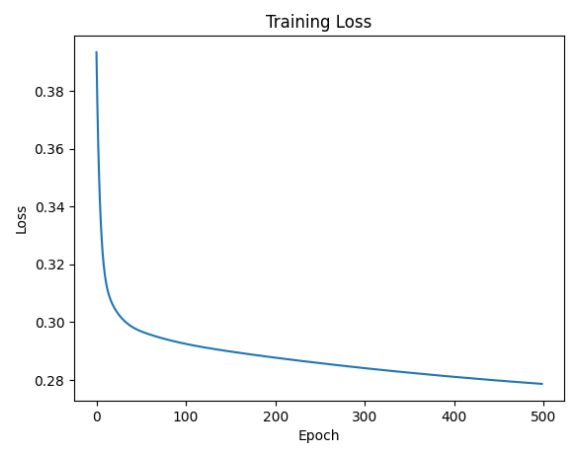
(a) Ringnorm



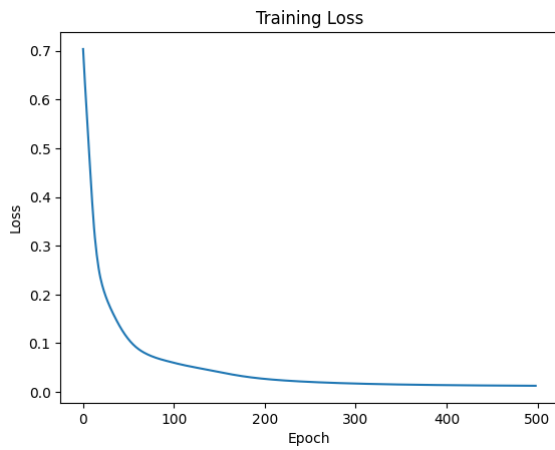
(b) Splice



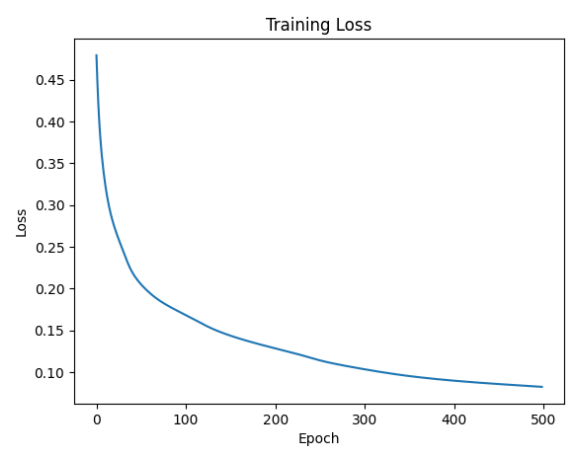
(c) Thyroid



(d) Titanic



(e) Twonorm



(f) Waveform

Figure 2: Datasets

3 Test Set Error

All the average test set errors with it's standard deviation is given below in Table: 2 These values obtained are produced by choosing the best from repeated evaluation by

Dataset	Test Set Error	Neural Architecture	Learning Rate	Time Taken (s)
Banana	10.63 ± 0.2	[2, 32, 24, 1]	0.1	9.34375
Breast Cancer	26.75 ± 3.49	[9, 27, 1]	0.2	3.1875
Diabetes	24.57 ± 1.72	[8, 16, 1]	0.2	7.4375
German	24.73 ± 1.7	[20, 16, 1]	0.2	11.125
Heart	16.9 ± 1.3	[13, 13, 1]	0.2	0.21875
Image	7.58 ± 1.23	[18, 18, 1]	0.3	21.25
Ringnorm	22.18 ± 2.02	[20, 16, 1]	0.4	3.9375
Splice	34.97 ± 4.00	[60, 12, 1]	0.2	54.65625
Thyroid	4.27 ± 1.96	[5, 15, 1]	0.1	0.40625
Titanic	22.83 ± 1.95	[3, 9, 1]	0.06	2.0625
Twonorm	5.02 ± 0.82	[20, 10, 1]	0.3	6.21875
Waveform	14.49 ± 1.23	[21, 7, 1]	0.3	3.15625

Table 2: Table with test set error

test and modifying the hyper parameters. I would be trying more of such modifications and will alter the values if any better found.

Note: The time in the table here signifies the CPU process time for the entire training process i.e. 500 epochs. It is the time taken for the training process of the plot shown in Figures: 1 and 2.

4 Mini-Batch Stochastic Gradient Method

Instead of using the entire dataset or a single example, Mini-Batch SGD uses a small, random subset of the data (mini-batch) to compute the gradient and update the parameters. The mini-batch size is typically a small number, such as 1, 10%, 20%, 50%.

An important change done in this is the decay of the learning rate as the number of iterations increase. The learning rate is first set at (η_0) and the learning rate is set to

$\eta = \frac{\eta_0}{1 + \text{decay} * \text{iteration}}$, where decay is set to be some small fraction which is to be fine tuned to get desired results. Here, we are supposed to find the number of iterations that is required to train the model and get the desired accuracies based on the previous test set error.

4.1 Algorithm:

The algorithm used to implement this method is given below.

Algorithm 1 Mini Batch Stochastic Gradient Method

Input: Training data $D = \{(x_i, y_i)\}_{i=1}^n$, $\epsilon = 1e^{-4}$, $\text{Loss}(\theta \pm \epsilon) \leftarrow$ Changes the parameter θ to $\theta \pm \epsilon$ and returns $C(\text{feedforward}(x), y)$.

```
1: Initialize network using Gaussian distribution with mean 0 and variance 1 to all weights and biases
2: for all l,j do
3:    $w_{ij}^l \leftarrow$  a small number,  $b_j^l \leftarrow$  small number
4: end for
5: /* Perform Gradient Check */
6: gradient_check( $x_1, y_1$ )
7: /* Training */
8:  $\kappa = n/10, n/5, n/2$ 
9: for each  $t = 1 \rightarrow$  number of iterations: do
10:   Sample  $\kappa$  samples randomly from training Dataset using a random seed
11:   for each training example  $(x, y) \in \kappa$  samples do
12:     /* Propagate inputs forward to find outputs */
13:     for each neuron i in input layer: do
14:        $a_i^0 \leftarrow x_i$ 
15:     end for
16:     for layer  $l=1$  to  $L$ : do
17:       for each neuron i in layer l: do
18:          $z_i^l \leftarrow \sum_j w_{ij}^l a_j^{l-1} + b_j^l$  {j  $\in \{1, 2, \dots, (\text{neurons\_layer}(l-1))\}$ }
19:          $a_i^l \leftarrow f_l(z_i^l)$ 
20:       end for
21:     end for
22:     /* Propagate deltas backwards from output layer to input layer */
23:     for each neuron i in output layer: do
24:        $\delta_i^L \leftarrow \frac{(\hat{y}_i - a_i^L)}{n} f'_L(z_i^L)$ 
25:        $\frac{\partial C_x}{\partial w_{ik}^L} \leftarrow \delta_i^L a_k^{L-1}, \quad \frac{\partial C_x}{\partial b_i^L} \leftarrow \delta_i^L$ 
26:        $\frac{\partial C}{\partial w_{ik}^L} \leftarrow \frac{\partial C}{\partial w_{ik}^L} + \frac{1}{n} \frac{\partial C_x}{\partial w_{ik}^L}, \quad \frac{\partial C}{\partial b_i^L} \leftarrow \frac{\partial C}{\partial b_i^L} + \frac{1}{n} \frac{\partial C_x}{\partial b_i^L}$ 
27:     end for
28:     for layer  $l=L-1$  to  $1$ : do
29:       for each neuron i in layer l: do
30:          $\delta_i^l \leftarrow f'_l(z_i^l) \sum_j \delta_j^{l+1} w_{ji}^{l+1}$  {j  $\in \{1, 2, \dots, (\text{neurons\_layer}(l+1))\}$ }
31:          $\frac{\partial C_x}{\partial w_{ik}^l} \leftarrow \delta_i^l a_k^{l-1}, \quad \frac{\partial C_x}{\partial b_i^l} \leftarrow \delta_i^l$ 
32:          $\frac{\partial C}{\partial w_{ik}^l} \leftarrow \frac{\partial C}{\partial w_{ik}^l} + \frac{1}{n} \frac{\partial C_x}{\partial w_{ik}^l}, \quad \frac{\partial C}{\partial b_i^l} \leftarrow \frac{\partial C}{\partial b_i^l} + \frac{1}{n} \frac{\partial C_x}{\partial b_i^l}$ 
33:       end for
34:     end for
35:     /* Update values of weights and biases using gradient descent */
36:     for all l,j,k: do
37:        $w_{jk}^l \leftarrow w_{jk}^l - \eta \frac{\partial C}{\partial w_{jk}^l}, \quad b_j^l \leftarrow b_j^l - \eta \frac{\partial C}{\partial b_j^l}$ 
38:     end for
39:   end for
40: end for
```

4.2 Results:

When the Mini-Batch Stochastic Gradient Method on the Gunnar Rätsch datasets, the results obtained are given in Tables: 3, 4, and 5.

4.2.1 10% of Dataset as Mini-Batch:

Dataset	Accuracy (%)	Initial eta	Iterations	Decay	Avg Time/Iteration (s)
Banana	88.06	0.01	2000	0.002	0.0061
Breast Cancer	74.03	0.01	4000	0.001	0.0010
Diabetes	77.34	0.04	3000	0.001	0.0054
German	76.67	0.1	4000	0.001	0.0032
Heart	83	0.03	8000	0.001	0.0017
Image	92.78	0.04	4000	0.001	0.0147
Ringnorm	77.8	0.08	4000	0.001	0.0053
Splice	66.07	0.04	6000	0.002	0.1218
Thyroid	96	0.008	6000	0.001	0.0003
Titanic	78.19	0.002	4000	0.002	0.0003
Twonorm	94.46	0.04	4000	0.001	0.0046
Waveform	85.13	0.04	4000	0.001	0.0046

Table 3: 10% of Dataset as κ

4.2.2 20% of Dataset as Mini-Batch:

Dataset	Accuracy (%)	Initial eta	Iterations	Decay	Avg Time/Iteration (s)
Banana	88.06	0.01	2000	0.002	0.0061
Breast Cancer	74.03	0.01	4000	0.001	0.0010
Diabetes	77.34	0.04	3000	0.001	0.0054
German	76.67	0.1	4000	0.001	0.0032
Heart	83	0.03	8000	0.001	0.0017
Image	92.78	0.04	4000	0.001	0.0147
Ringnorm	77.8	0.08	4000	0.001	0.0053
Splice	66.07	0.04	6000	0.002	0.1218
Thyroid	96	0.008	6000	0.001	0.0003
Titanic	78.19	0.002	4000	0.002	0.0003
Twonorm	94.46	0.04	4000	0.001	0.0046
Waveform	85.13	0.04	4000	0.001	0.0046

Table 4: 20% of Dataset as κ

4.2.3 50% of Dataset as Mini-Batch:

Dataset	Accuracy (%)	Initial eta	Iterations	Decay	Avg Time/Iteration (s)
Banana	88.37	0.015	1500	0.002	0.0138
Breast Cancer	75.32	0.01	2000	0.001	0.0052
Diabetes	75.67	0.04	1500	0.001	0.0115
German	77	0.1	2000	0.001	0.0082
Heart	82	0.03	2000	0.001	0.0032
Image	93.56	0.08	2500	0.001	0.032
Ringnorm	78.79	0.08	2000	0.001	0.0102
Splice	65.01	0.04	2000	0.002	0.0163
Thyroid	94.67	0.01	3000	0.001	0.0035
Titanic	77.78	0.002	2000	0.002	0.0037
Twonorm	94.79	0.04	2000	0.001	0.0101
Waveform	85.13	0.04	2000	0.001	0.01

Table 5: 50% of Dataset as κ

4.3 Plot of Obtained Results:

The plots of the results mentioned above is given in the tables: 6 and 7. These plots are the ones that were obtained when the results above were obtained.

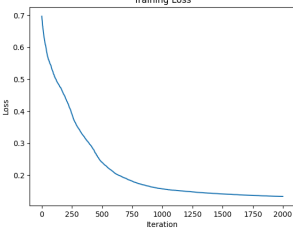
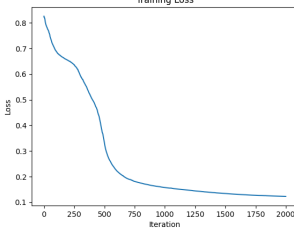
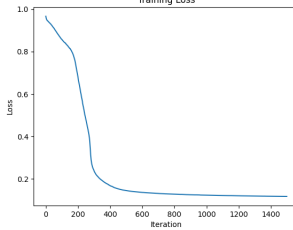
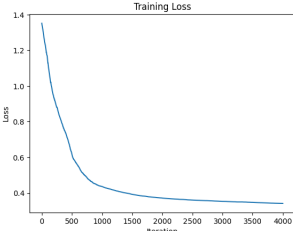
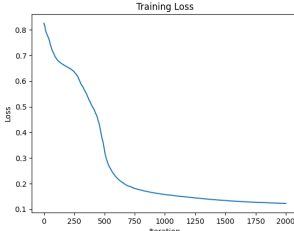
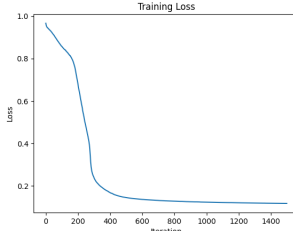
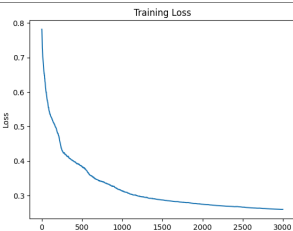
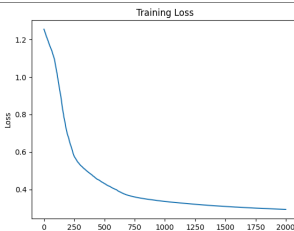
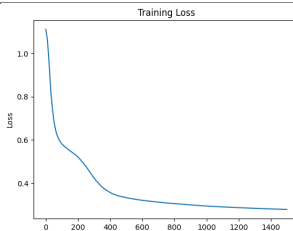
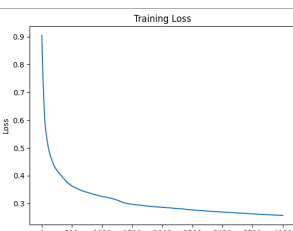
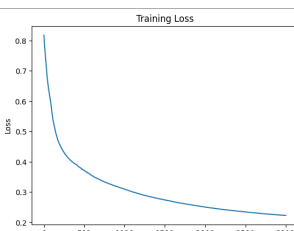
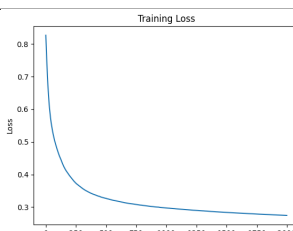

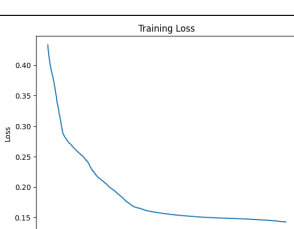
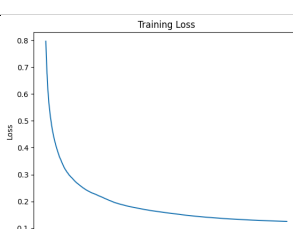
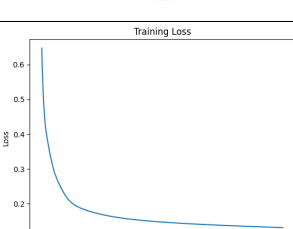
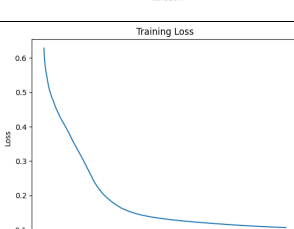
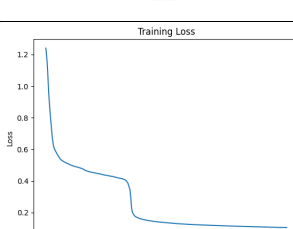
Dataset	10% of Dataset	20% of Dataset	50% of Dataset
Banana			
Breast Cancer			
Diabetes			
German			
Heart			
Image			

Table 6: Plots

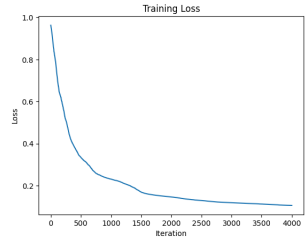
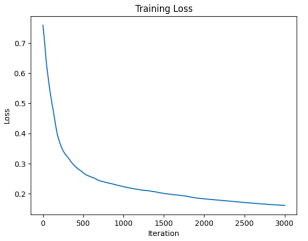
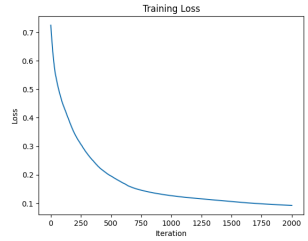
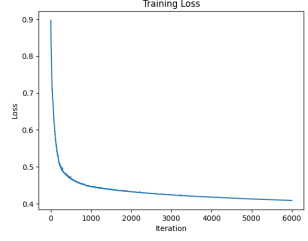
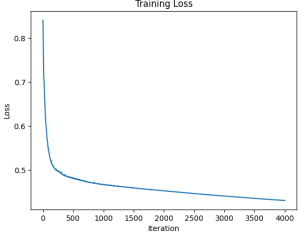
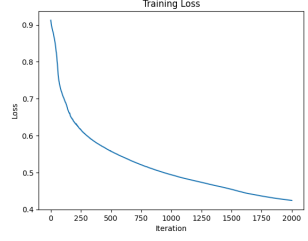
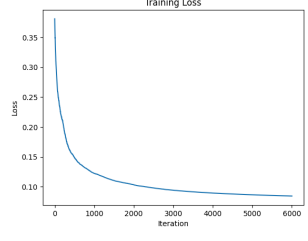
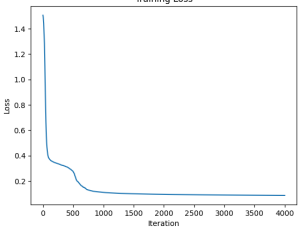
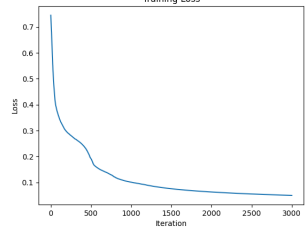
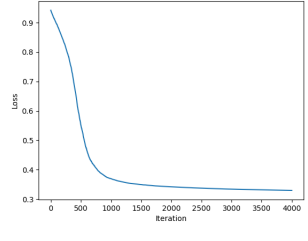
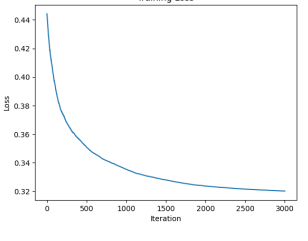
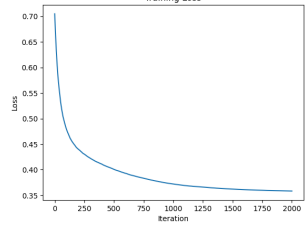
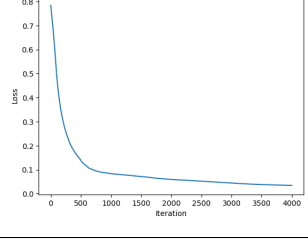
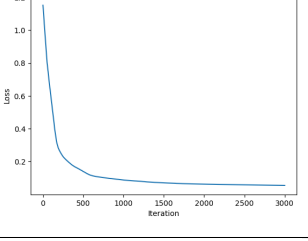
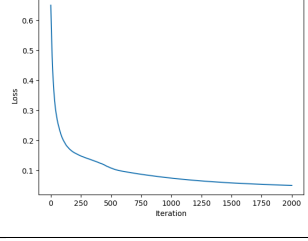
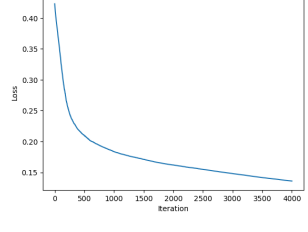
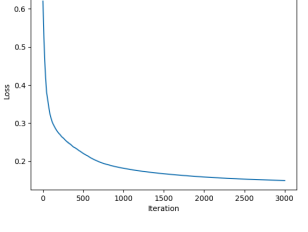
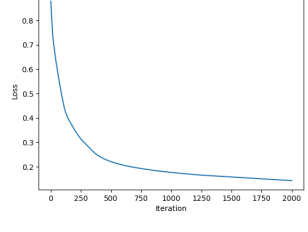
Dataset	10% of Dataset	20% of Dataset	50% of Dataset
Ringnorm			
Splice			
Thyroid			
Titanic			
Twonorm			
Waveform			

Table 7: Plots