Deep Learning Assignment 1- Report

Part 1: Deep vs Shallow

Task 1 Simulate a Function

I have used more than **two models** and **one function** as part of the assignment that satisfies the bonus point tasks.

I worked on three models, which were trained on two separate functions. The models I worked on, namely Model0, Model 1, and Model 2, with different parameters mostly over 500, are fully connected neural networks with seven, four, and one layers, respectively, which differ by the number of nodes.

The functions which I worked on are mentioned below.

1. **y=(sin(5*(pi*x)))/((5*(pi*x))**
2. **y=sign(sin(5*pi*X1))**

The learning rates for all the training were 0.001

The training process for the two functions resulted in convergence of the loss functions in all three models (deep, middle, shallow) within a reasonable number of iterations. As demonstrated in Figure 1, the Mean Squared Error (MSE) of the models is displayed for each epoch of the (sin(5*(pix)))/((5(pix)) simulation, with convergence being achieved after 2000 iterations. Figure 2 then provides a comparison between the predicted and actual values for the (sin(5(pix)))/((5(pi*x)) function, highlighting the models' accuracy.

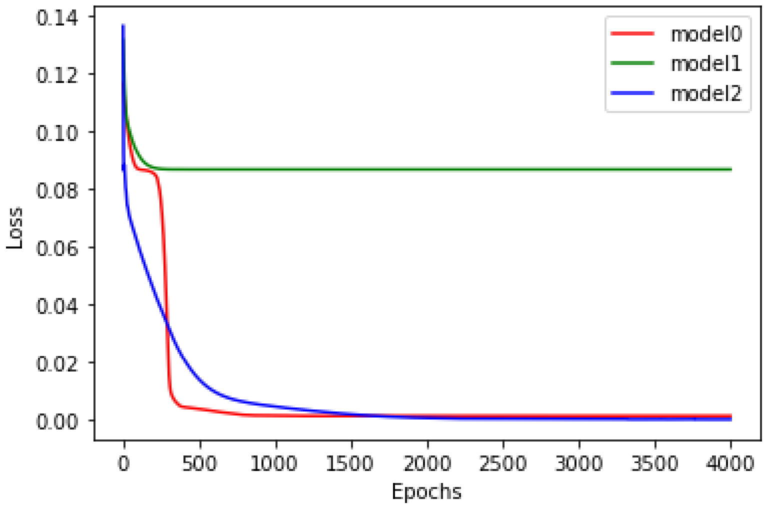

Figure 1

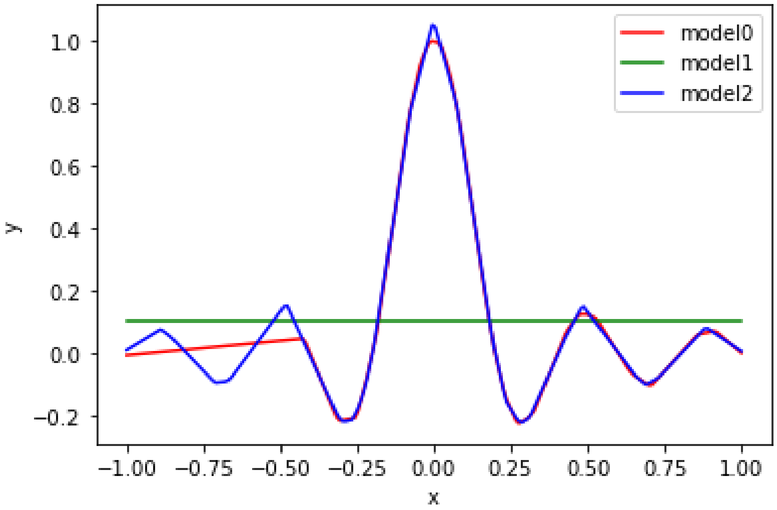

Figure 2

The results of the training process for the three models (deep, middle, shallow) for the sign(sin(5piX1)) simulation are shown in Figure 3, where the Mean Squared Error loss is depicted. The accuracy of the models is further demonstrated in Figure 4, which presents the comparison between the predicted and actual values for the sign(sin(5piX1)) function. Despite some inaccuracies at the edges of the x-axis, the models were able to correctly predict outputs for unseen inputs, as seen in the center of the graph. The deeper the model, the fewer misclassifications were observed at the extremes of the x-axis.

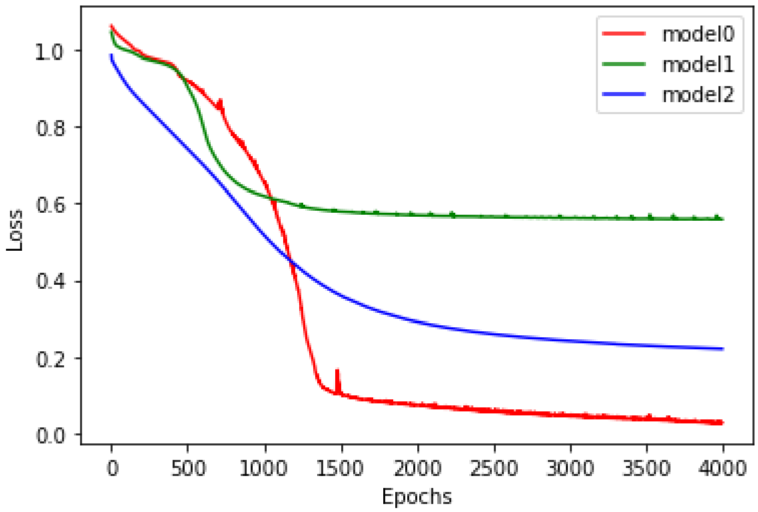

Figure 3

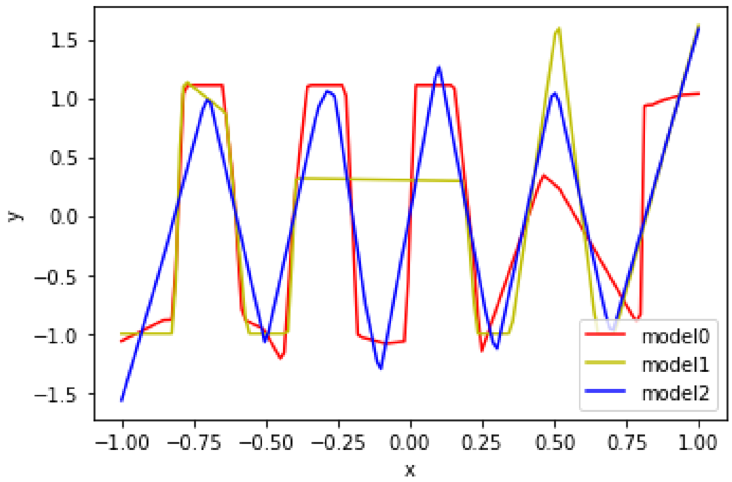

Figure 4

Task 1-2: Train on Actual Tasks

Two Convolutional Neural Network (CNN) models were created utilizing the MNIST dataset, with 60,000 training data points and 10,000 testing data points, and a batch size of 10. The training data was shuffled for improved learning, while the testing data remained unaltered. Both models employed fully connected layers and utilized max pooling with the RELU activation function. The number of nodes in each layer varied between the two models, and a learning rate of 0.001 was used for all models. Figure 5 presents the loss that occurred during the training of both model1 and model2, showing convergence around the 50 epoch mark for both models.

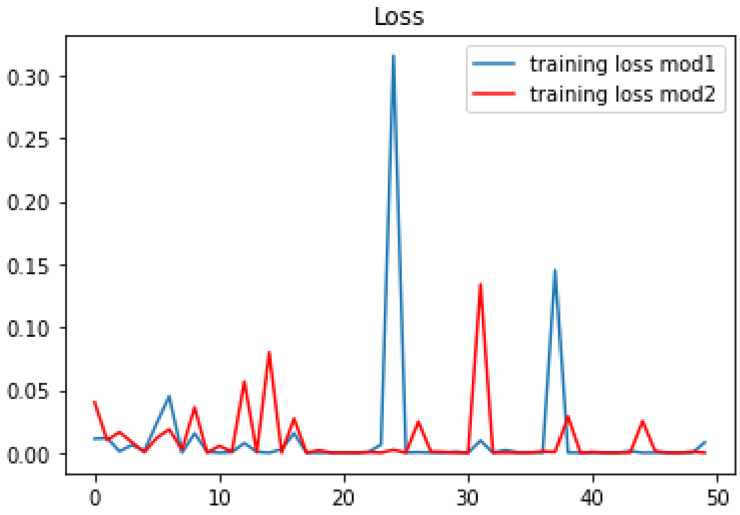

Figure 5

The training accuracy of the two models can be observed in Figure 6, which displays their performance on both the training and test datasets. As expected, the models exhibit higher accuracy on the training data compared to the test data.

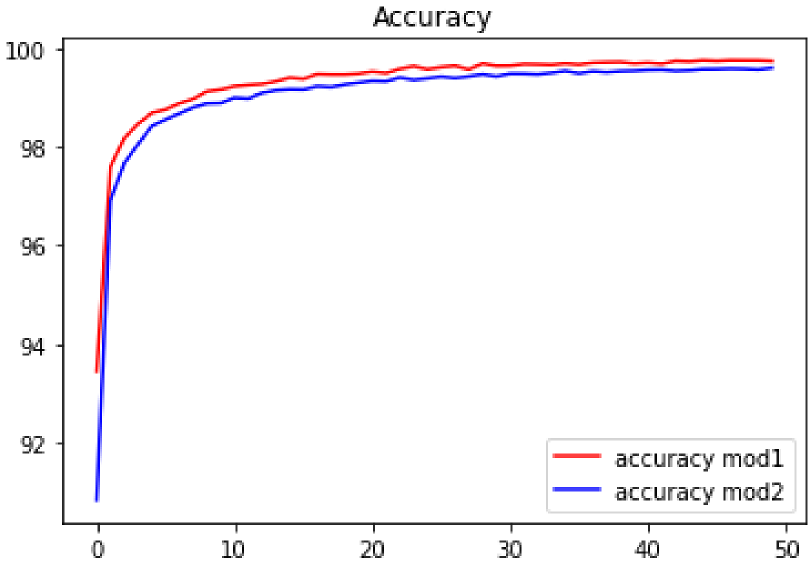

Figure 6

Part 2 Optimization

Task 1: Visualize the Optimization Process

An efficient Deep Neural Network, made up of three fully connected layers and 57 parameters, was trained to replicate the function (sin(5*(pix)))/(5(pi*x)). As seen in Figure 7, the second layer of the network is depicted, while the overall model optimization process can be viewed in Figure 8. The optimization was executed using the Adam optimizer and involved eight training series, each lasting 40 epochs, during which the model's weights were recorded at set intervals. The PCA implementation was employed for dimensionality reduction. The learning rate was kept consistent at 0.001 for all models.

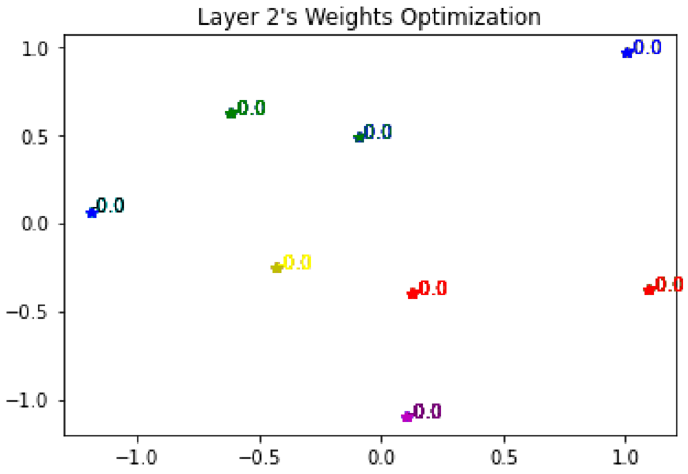

Figure 7

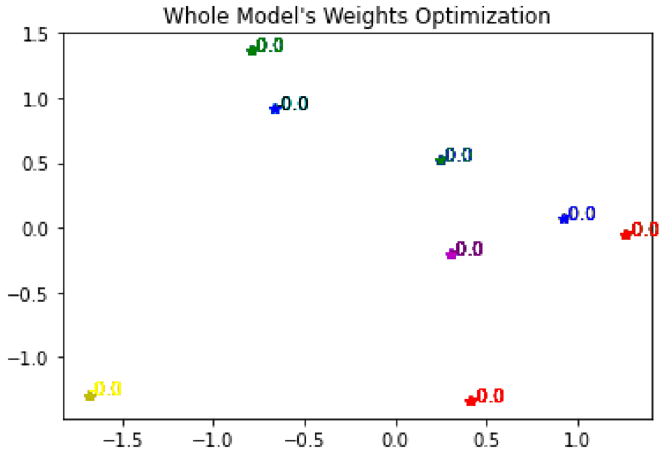

Figure 8

Figures 7 and 8 visually demonstrate the optimization of the weights within the Deep Neural Network as it undergoes the training process. The optimization was carried out using the backpropagation method and the figures show the gradual improvement of the weights over time. These figures provide a clear illustration of the fine-tuning of the network as it learns and adapts during the training phase.

Task 2: Observe Gradient Norm during Training

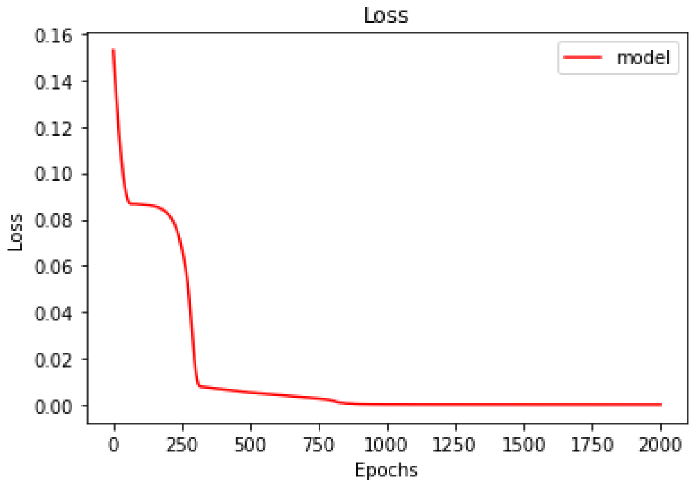

Figure 9

The performance of the model during training is represented in Figure 9, which displays the loss experienced by the model for each epoch. Additionally, Figure 10 shows the gradient norm for each epoch, with the spikes in the graph corresponding to the changes in slope depicted in Figure 9. It's important to note that there seems to be a deviation from the norm in Figure 10's graph.

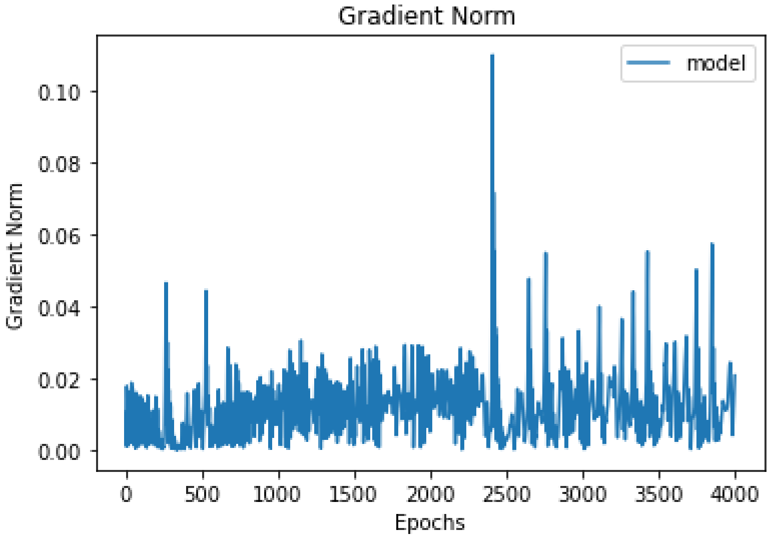

Figure 10

Part 3: Generalization

Task 1: Can Network fit random Variables

The Deep Neural Network was trained using the MNIST dataset, which was selected for this purpose. The network consisted of 3 hidden layers and underwent 2000 rounds of training on the MNIST dataset. To optimize the network, the Adam optimizer was used, and all models had a learning rate of 0.001.

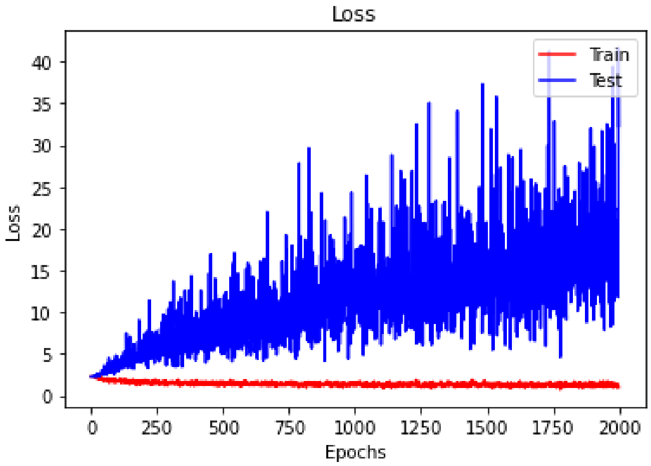

Figure 11

The results seen in Figure 11 show that while the training loss was relatively low, the loss on the test data, which had not been encountered during training, was significantly higher. This highlights the challenge in accurately fitting the random labels with the current model.

Task 2: Parameters Vs Generalization

The MNIST dataset was selected as the training and testing data for this study. Ten feedforward Deep Neural Networks were constructed, each featuring two hidden layers. The models were designed to cover a range of complexity, with the number of parameters ranging from hundreds to millions. The learning rate for all of the models was set to 0.001, and the Adam Optimizer was utilized for optimizing the Neural Network.

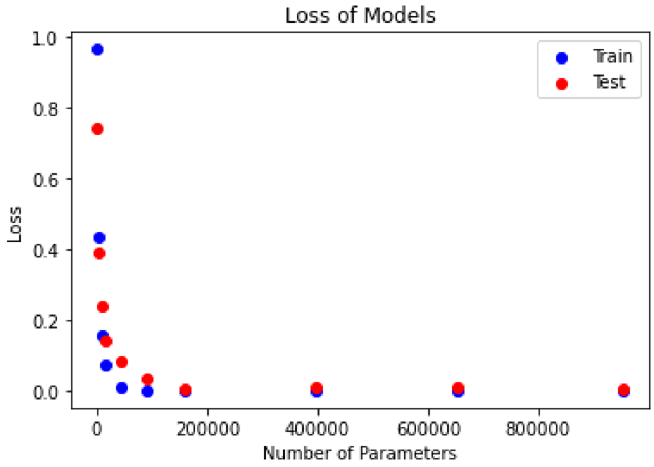

Figure 12

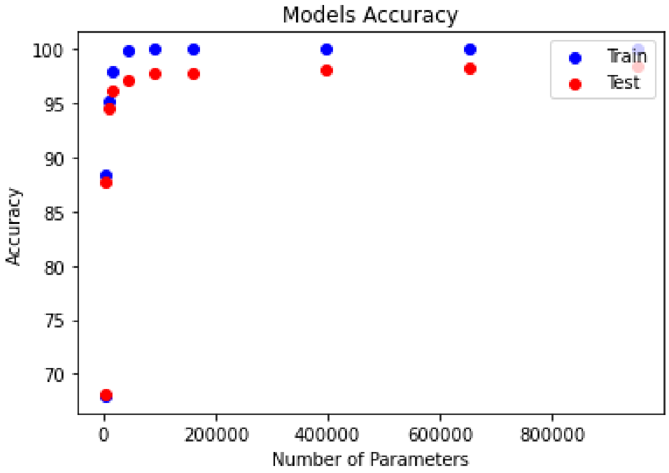

Figure 13

The results displayed in Figures 12 and 13 indicate that there is a positive correlation between the number of parameters in a Deep Neural Network model and its loss and accuracy. As the number of parameters increases, the model's loss decreases, and accuracy improves. However, there appears to be a saturation point beyond which adding more parameters only results in minor improvements in the model's performance. It is also evident from the figures that the models perform better on the training data than on the unseen testing data, which suggests overfitting may be present.

Task 3: Flatness VS Generalization

Part 1

The performance of two Deep Neural Networks was compared on the MNIST dataset. The first network had a batch size of 64, while the second network had a batch size of 1024. Both models utilized the Adam Optimizer with a learning rate of 0.001 for optimization. To evaluate the results, a linear interpolation was used between the parameters of both models (theta1 and theta2) with alpha being the interpolating factor.

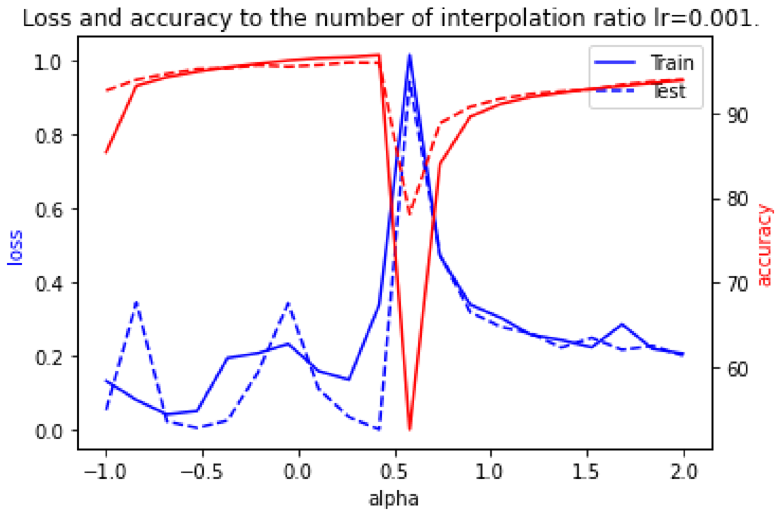

Figure 14

Figure 14 displays the loss, precision, and linear interpolation alpha even during development of two models with learning rates of 0.001 and 0.01 correspondingly.

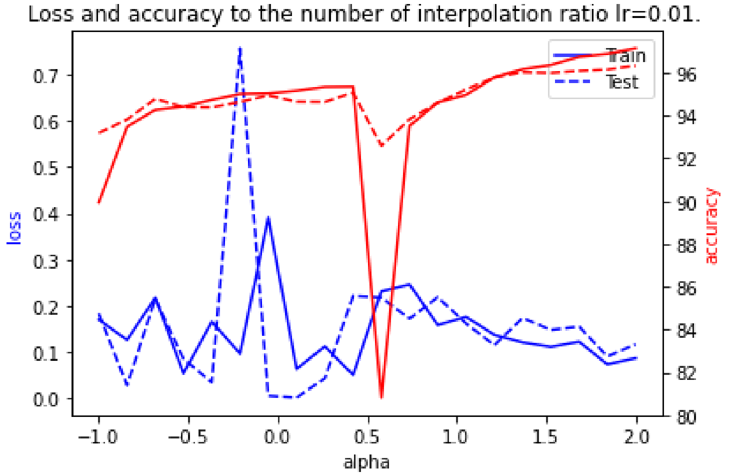

Figure 15

Part 2: Flatness Vs Generalization

The module was tested and provided with training using the MNIST data set. Five identical Deep Neural Networks, each with two secret layers and 16630 parameters, were trained in batches varying from five to one thousand. For the optimization technique, the Adam Optimizer is implemented, and all simulations have a learning rate of 0.001.

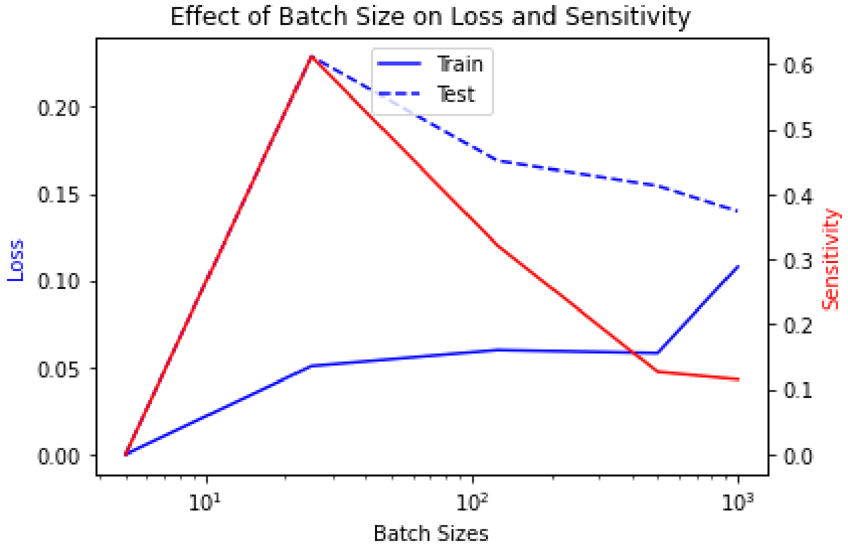

Figure 16

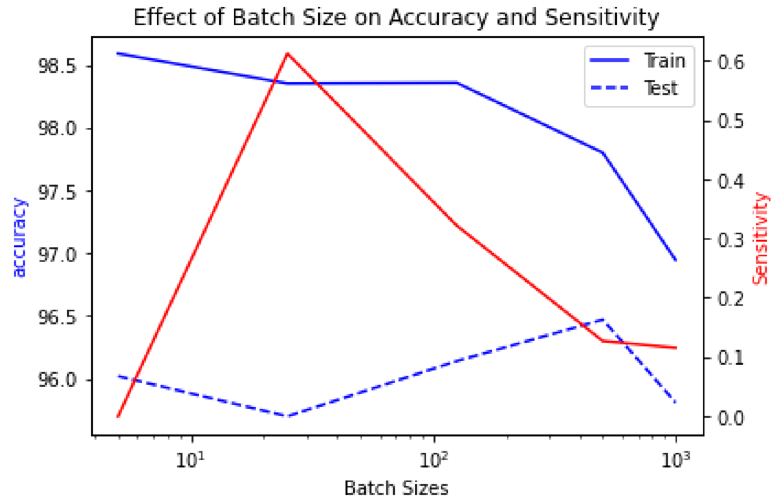

Figure 17

Following training, the accuracy and loss for the training sample and test sample for each of the different models were determined. Next, the forbenius norm of the gradient approach was used to evaluate the models' responsiveness.

Figures 16 and 17 demonstrate how sampling rate, loss, and susceptibility have an impact on accuracy and sensitivity. Sensitivity decreases with batch size. Therefore, we draw the conclusion that the system will perform best whenever the sample size is between 102 and 103.