DEEP LEARNING HOMEWORK-1

TASK-1-1

SIMULATE A FUNCTION:

- Used more than 2 models
- Used more than 1 function

On two distinct functions, three distinct models, each with a different number of parameters, were trained. Models 0, 1, and 2 are fully connected neural networks with seven, four, and one layers, respectively, with 571, 572, and 572 parameters. Every layer has a different number of nodes. The simulated functionalities included

i)y=(sin(5*(pi*x)))/((5*(pi*x))

ii) y=sign(sin(5*pi*X1))

The average learning rate was 0.001.

-Within an acceptable number of training cycles, all models for the two functions experienced loss function convergence. The Mean Squared Error (MSE) of three models (deep, intermediate, and shallow) for each epoch of the training for the simulation of (sin(5*(pi*x)) is shown in Figure 1 below. After 2500 iterations, convergence is attained. The (sin(5*(pi*x))/((5*(pi*x)) function's anticipated and actual values are shown in Figure 2.

FIGURE-1

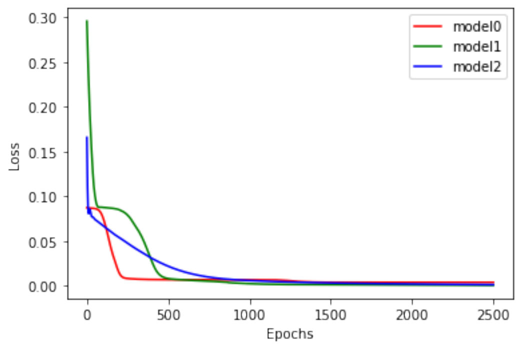
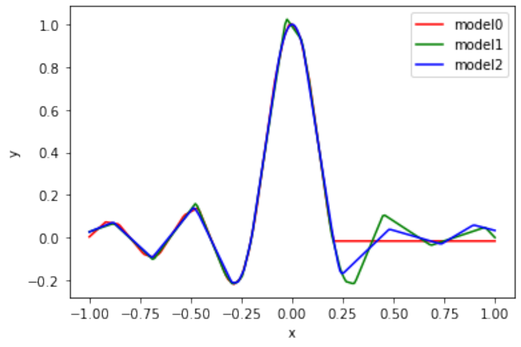


FIGURE-2



The Mean Squared Error (MSE) loss that happened during the training of the three models (deep, intermediate, and shallow) for the simulation of sign(sin(5*pi*X1)) is shown in Figure 3. The sign(sin(5*pi*X1)) function's anticipated and actual values are shown in Figure 4. All three models discovered that when given an unknown input, the accuracy function could produce the desired results. While the graph's models performed well in the middle, the x-extreme axis's ends include misclassifications. The failure at the graph's edges is reduced the deeper the model is.

FIGURE-3

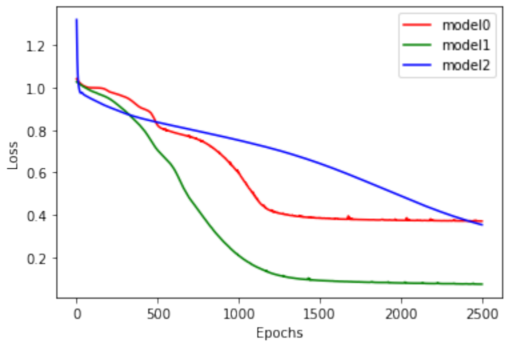
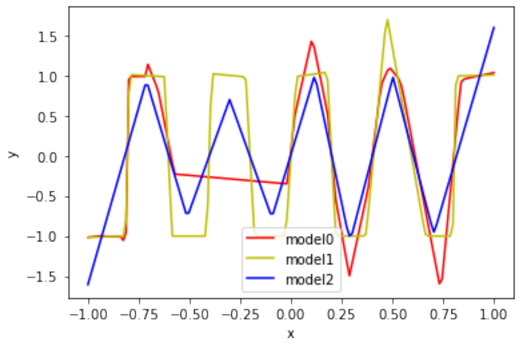


FIGURE-4

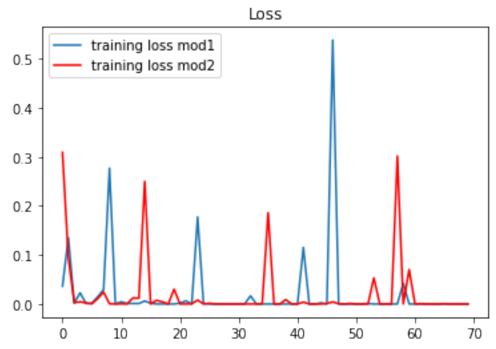


TASK-1-2

TRAIN ON ACTUAL TASKS

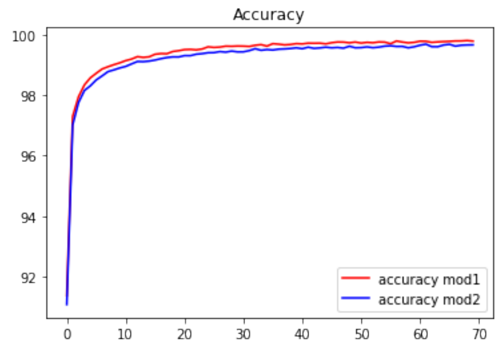
Convolutional neural networks (CNN) were used to create two models utilizing 60,000 training and 10,000 test data from the MNIST dataset, with a batch size of 10. Testing data is not jumbled, but training data is. The two models are completely interconnected, and in order to pool the data, we utilized the max pool function using RELU as the activation function. Each tier of every network has a different number of nodes. The average learning rate was 0.001. The loss that happened during the training of models 1 and 2 is shown in figure 5. For two models, convergence is often attained after 70 epochs.

FIGURE-5



The accuracy of the two models on training and test sets is shown in Figure 6 during model training. As predicted, we can see that two models regularly outperform each other on training data compared to test data.

FIGURE-6

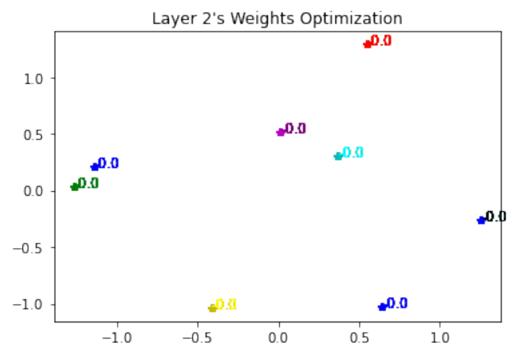


TASK-1-2

1-2 Visualize the Optimization Process

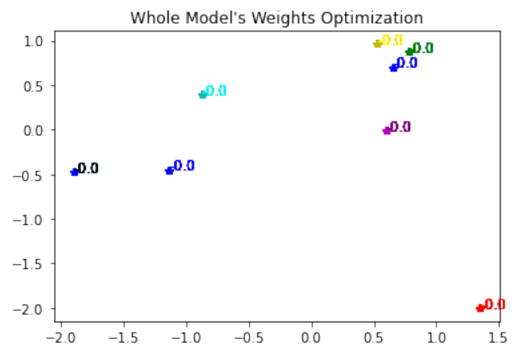
The function (sin(5*(pi*x))/((5*(pi*x))) was trained using a Deep Neural Network with three fully connected layers and 57 parameters. Figures 7 and 8 demonstrate the network's second layer and overall model optimization, respectively. The network's optimization was carried out using the Adam optimizer. There were eight separate training series, each with 60 training epochs, and weights for the model were frequently gathered. PCA application results in the dimension reduction. All models have a learning rate of 0.001.

FIGURE-7



Backpropagation is used to optimize the weights inside the network that is trained throughout the training phase, as seen in Figures 7 and 8. The figures demonstrate the gradual fine-tuning as the weights are gradually optimized.

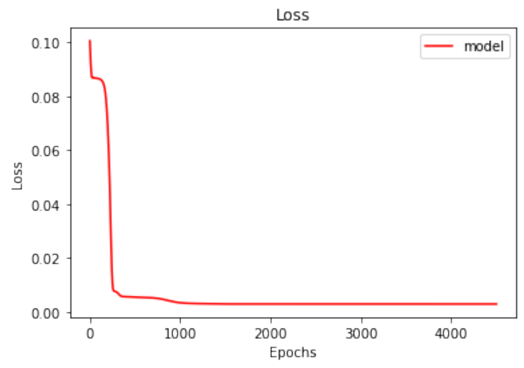
FIGURE-8



1-2 GRADIENT NORM OBSERVATION

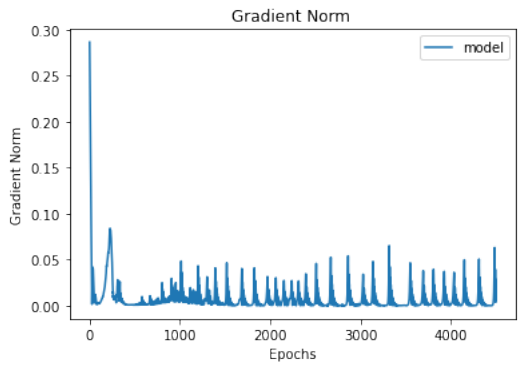
Figure 9 displays the model's loss for each training period.
While each epoch's gradient norm is shown in figure 10.

FIGURE-9



The slope variations in figure 9 and each of the spikes in figure 10 are related. The graph 10 exhibits an anomaly.

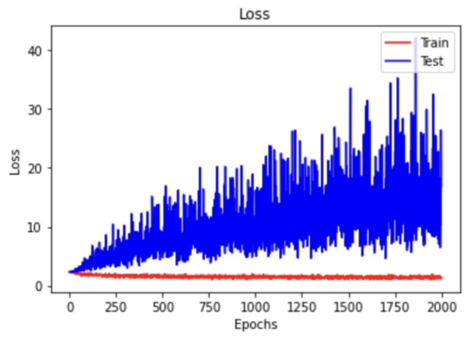
FIGURE-10



PART-3

1-3 CAN NETWORK FIT RANDOM LABELS

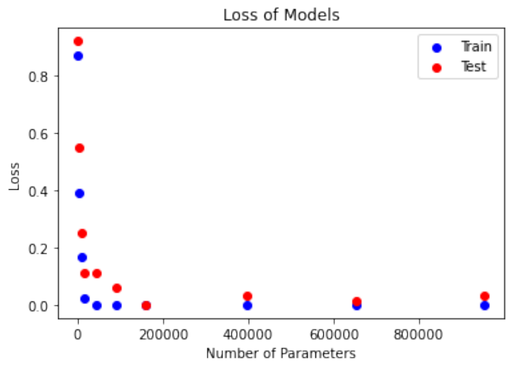
Figure- 11



1-3 Number of parameters v.s Generalization

The training and testing dataset used was the MNIST dataset. With three hidden layers, a feed-forward deep neural network was put into use. On the MNIST dataset, it underwent 2000 training iterations. The learning rate for all models is 0.001. The neural network's optimization method uses the Adam optimizer.

FIGURE-12

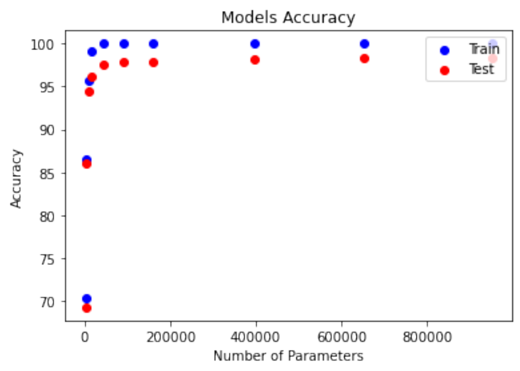


If we look at Figures 12 and 13, adding more parameters to the model reduces loss while also improving accuracy. However, after a certain number of parameters, there is little to no progress in the model from iteration to iteration. The model scarcely improves with more parameters included.

-

If we look at Figure 13, we can see that when the models are run on the training dataset instead of the testing dataset, they get greater accuracies and lower loss values.

FIGURE-13

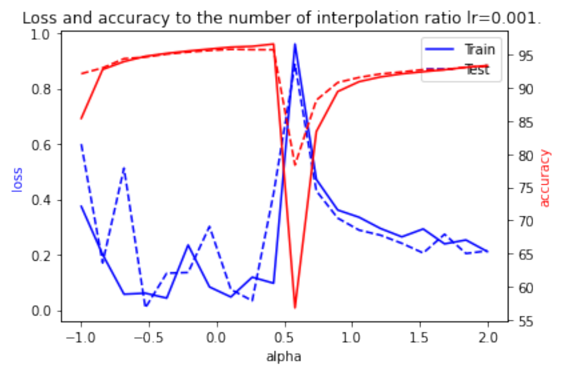


Task 3: Flatness VS Generalization

PART-1

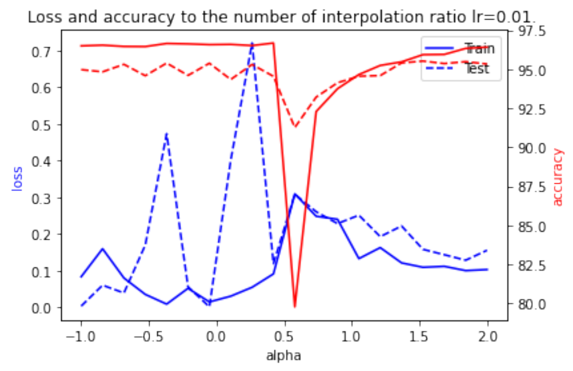
The training and testing data set was determined to be the MNIST dataset. With two distinct batch sizes of 64 and 1024, two Deep Neural Networks were put into use. The learning rate for the entire model was set at 0.001. The Neural Network was optimized using the Adam Optimizer. Theta1 and theta2 are the parameters of the model 1 and model 2, respectively, while alpha is the linear interpolation between them.

FIGURE-14



Figures 14 and 15 compare the loss, accuracy, and linear interpolation alpha for two models that were trained at learning rates of 0.001 and 0.01, respectively.

FIGURE-15



1-3 Part 2 Flatness Vs Generalization

The module was tested and trained using the MNIST data set.
Five identical Deep Neural Networks, each with two hidden layers and 16630 parameters, were trained in batches ranging from five to one thousand. For the optimization process, the Adam Optimizer is utilized, and all models have a learning rate of 0.001.

FIGURE-16

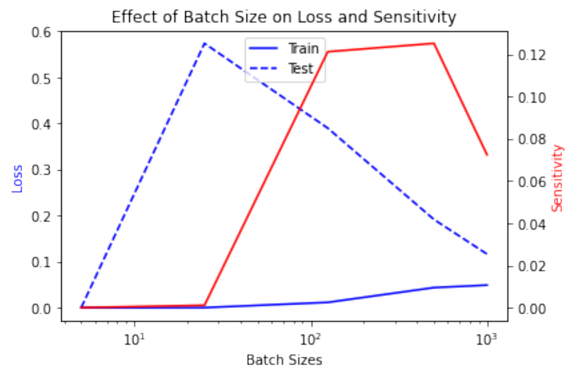
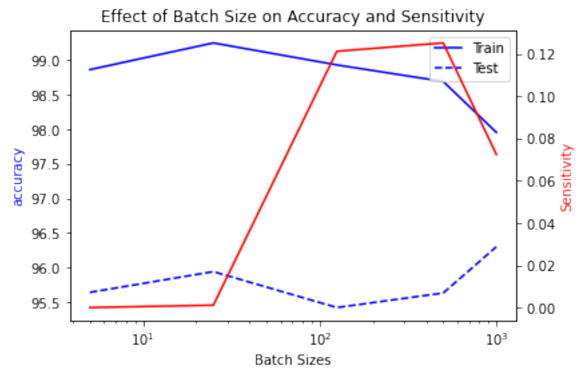


FIGURE-17



Following training, the accuracy and loss for the training dataset and test dataset for each of the five models were determined. Next, the forbenius norm of gradient approach was used to determine the models' sensitivity.

-

Figures 16 and 17 demonstrate how batch size, loss, and sensitivity have an impact on accuracy and sensitivity. Sensitivity decreases with batch size.
As a consequence, we draw the conclusion that the network will perform best when the batch size is between 102 and 103.

GitHub link - https://github.com/SandeepKOmmanaboyina/Deeplearning8430