Deep Learning HomeWork - 1

anvithayerneni/8430-DeepLearning-1 (github.com)

1-1 Simulate Function

I've developed three models in accordance with the specified guidelines, incorporating enhanced regularization through the addition of a weight decay parameter within the neural network's cost function. This inclusion aims to reduce the weights progressively during the backpropagation phase. Each model is trained with a specific number of parameters and is designed to perform distinct tasks.

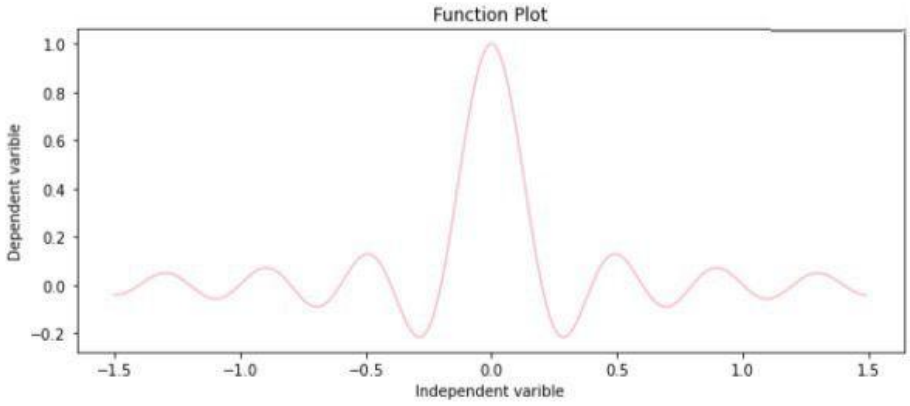
Model 1: This model, consisting of seven densely connected layers, utilizes RMSProp for optimization, MSELoss for evaluating performance, and features weight decay for regularization and learning rate 0.001(1-e3).

Model 2 is constructed with four dense layers, holding a total of 571 parameters. It utilizes the Mean Squared Error Loss (MSELoss) to assess loss and employs RMSProp for optimization purposes. The learning rate for this model is fixed at 0.001(1-e3), and it incorporates the leaky ReLU activation function to introduce non-linearity. Additionally, a weight decay of 0.0001(1-e4) is integrated into the model to support regularization.

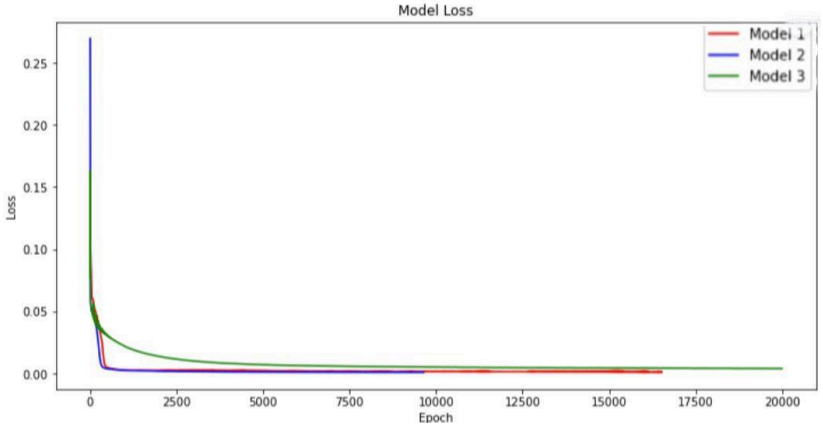
Model 3 is designed with a simpler structure, comprising just one dense layer but maintaining the same number of parameters, 571. It follows a similar setup for loss calculation and optimization, using MSELoss and RMSProp, respectively. The learning rate remains at 0.001, and like Model 2, it features leaky ReLU for activation. A weight decay of 0.0001 is also part of this model's design, aiding in the regularization process. Both models showcase an approach to learning that balances complexity and efficiency, with Model 2 offering a more layered approach compared to Model 3's streamlined design.

Function1:

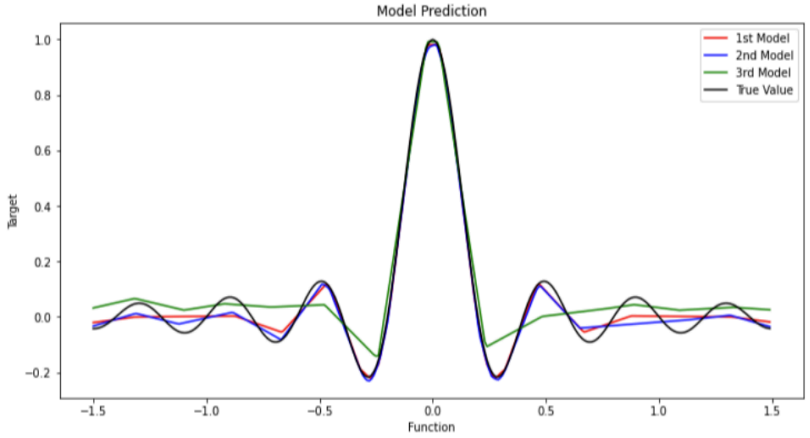
- Function 1 is represented by the mathematical formula (sin{5Π x}/{5Π x}).
- The following is a graphical depiction of Function 1, illustrating the relationship between the independent variable(x) and the dependent variable as defined by this function.



Demonstration:
All models exhibit convergence patterns when a maximum number of times is reached or when the number of classes decreases sharply. The losses incurred by each model are shown in the graph provided, showing the epoch values on the x-axis and the corresponding losses on the y-axis. This graph shows the loss function behavior of the three models.

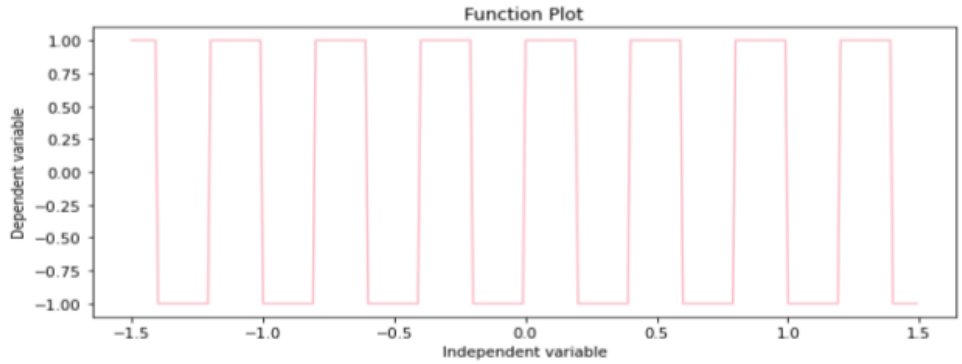


The graph provided below illustrates an evaluation between the true values and the predictions made by three models.



Findings:
The results say, Model1 & Model2 achieve convergence faster than the Model3, which only reaches convergence upon completing its maximum allotted epochs. The accompanying graph demonstrates that the additional layers in Models 1 and 2 contribute to reduced loss values and improved efficiency in learning the function. This highlights the importance of multiple layers in enhancing the speed and precision of the learning process.

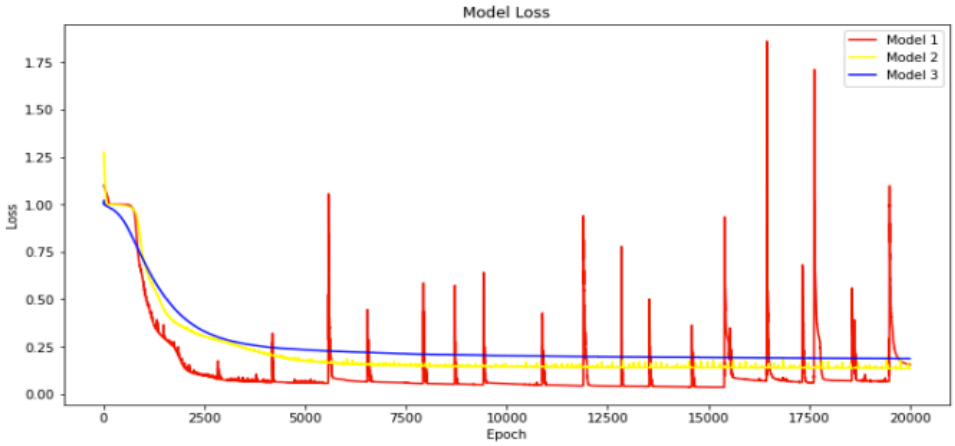
Function2:
Function 2 is defined by the equation sgn{sin(5Πx)\5Πx)}. This section presents a visual representation of Function 2, showcasing the correlation between the independent variable(x) and the dependent variable as determined by this specific function.



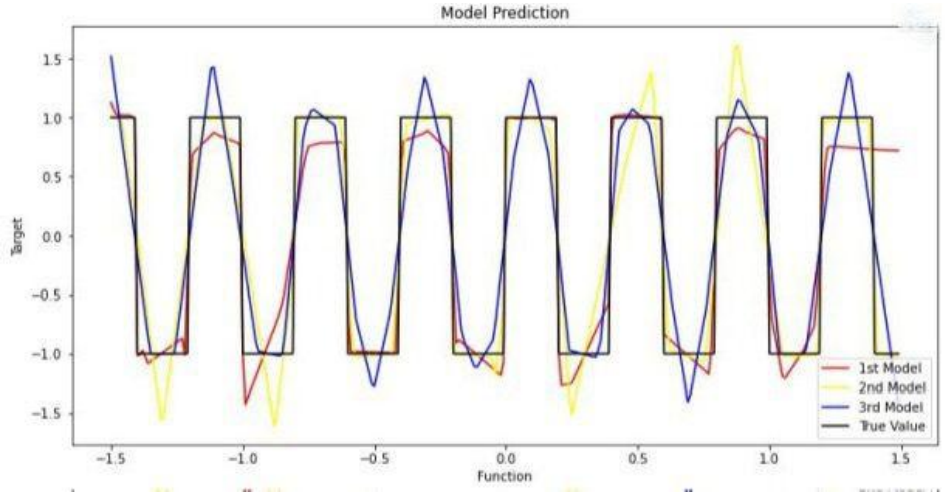
Demonstration:

Once the maximum epoch limit is reached or a significant decrease in learning rate occurs, convergence is achieved by each model. To further showcase this, a graphical representation of the loss for all three models is presented.

This graph maps epoch values ,loss values effectively illustrating the behavior of the loss function during training.



The chart presented below shows a comparative analysis between the true values and the predictions made by the three models.



Findings:
Before reaching convergence, each model hits the maximum epoch limit, indicating that the function poses a complex learning challenge for all three models. Among them, Model 1 achieves the lowest loss, proving to be the most efficient in learning and marginally surpassing Model 2 in performance. Conversely, Model 3 does not manage to secure a low loss and fails to converge within the allocated epoch range. This reinforces the notion that neural networks with a higher number of layers generally exhibit superior learning capabilities and quicker convergence.

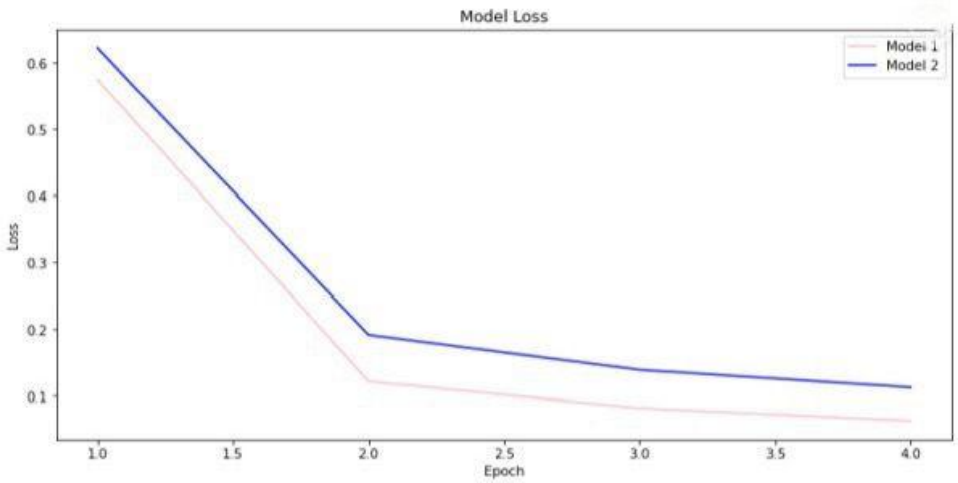
1-1 Training Actual Tasks

These models have undergone training using the MNIST dataset, which comprises 60,000 training samples and 10,000 testing samples.

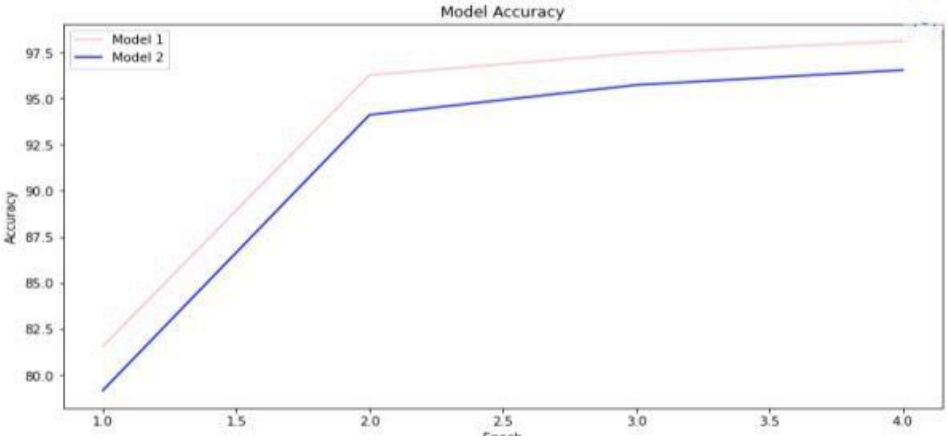
- The architecture of Model 1 incorporates a LeNet Convolutional Neural Network, incorporating a series of layers such as two 2D convolutions, each followed by a 2D max pooling layer and ReLU activation, as well as two dense layers with ReLU activation.
-
- Model 2 presents a unique Convolutional Neural Network algorithm: It consists of two initial 2D convolution layers that are activated by ReLU. Following these, there are two more 2D convolution layers paired with 2D max pooling layers and ReLU activation. A dropout layer is added after each of these. Finally, the model ends with two dense layers - the first one activated by ReLU and including dropout, and the last one utilizing Log_softmax activation for producing the output. This comprehensive approach consistently engages in the best possible results.

The training regimen for both models included the following hyperparameters:
- A learning rate set at 0.01.
- Momentum configured to 0.5.
- The use of Stochastic Gradient Descent as the optimization technique.
- A batch size of 64.
- A training duration of 10 epochs.
- Cross Entropy as the chosen loss function.

Below sections will detail the training loss experienced by Model 1 and Model 2.



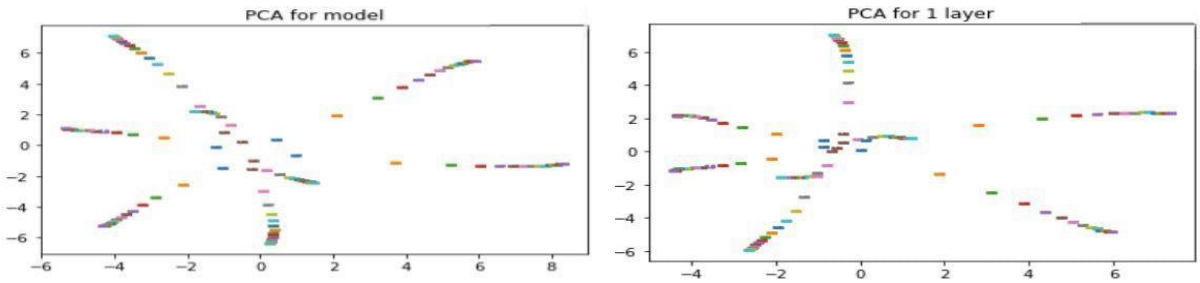
Model 1 & 2 Accuracy :



Findings:
Model 1 showcases enhanced performance over Model 2, thanks to its refined CNN architecture inspired by the LeNet framework. This is highlighted by its reduced loss and greater training precision. The superior results of Model 1 are credited to its carefully optimized architecture, contrasting with the custom CNN configuration employed in Model 2.

1-2 Visualize the optimization process

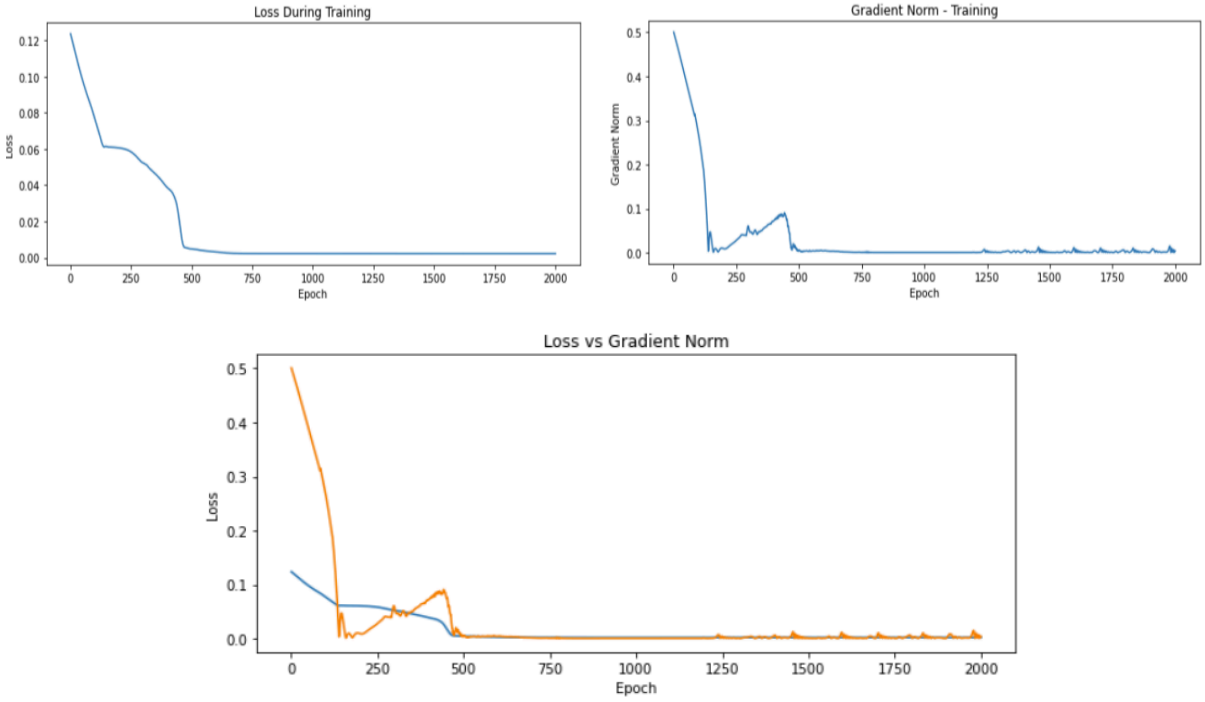
To effectively train the model, weights are collected at every third epoch in a repetitive cycle of eight. These weights are then condensed to two dimensions using Principal Component Analysis (PCA). The MNIST dataset, with 60,000 training samples and 10,000 testing samples, was utilized to train this specific model. This model is designed with three dense layers and uses cross entropy loss for its loss calculation. To optimize its performance, the Adam optimizer is used, with a learning rate set at 0.0004. With a batch size of 1000, the training sessions implement the ReLu function as the activation mechanism.



The chart illustrates how the dimensionality of the model weights was decreased after applying the Principal Component Analysis (PCA) technique. Initially, each model had 8,240 parameters or weights. Following eight training cycles, each consisting of 45 epochs, and the subsequent application of PCA, the weight dimensions were significantly diminished, as shown in the graph provided above.

1-2 Observe gradient norm during training

The chart illustrates the norm of the gradient calculated throughout the epochs, utilizing the function (sin{5Π x}/{5Π x}). Given the small size of the input, the model underwent training over epochs instead of iterations. Employing this function to compute both the gradient norm and the loss facilitated the recycling of the same computation process. Below are three graphs: one showing the gradient norm over the epochs and another depicting the loss over the epochs and versus.

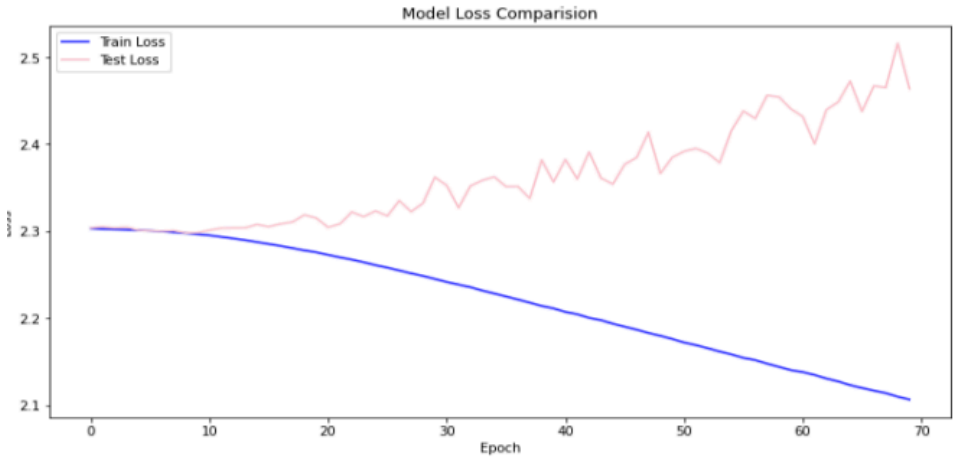


Findings:
The model's training process resulted in effective convergence. After the first 100 epochs, there's a minor uptick in the gradient, evident from the initial stabilization of the loss followed by

a more gradual decline. This continues until the loss levels off once again, around the 500-epoch mark, as depicted in a separate graph.

1-3 Generalization

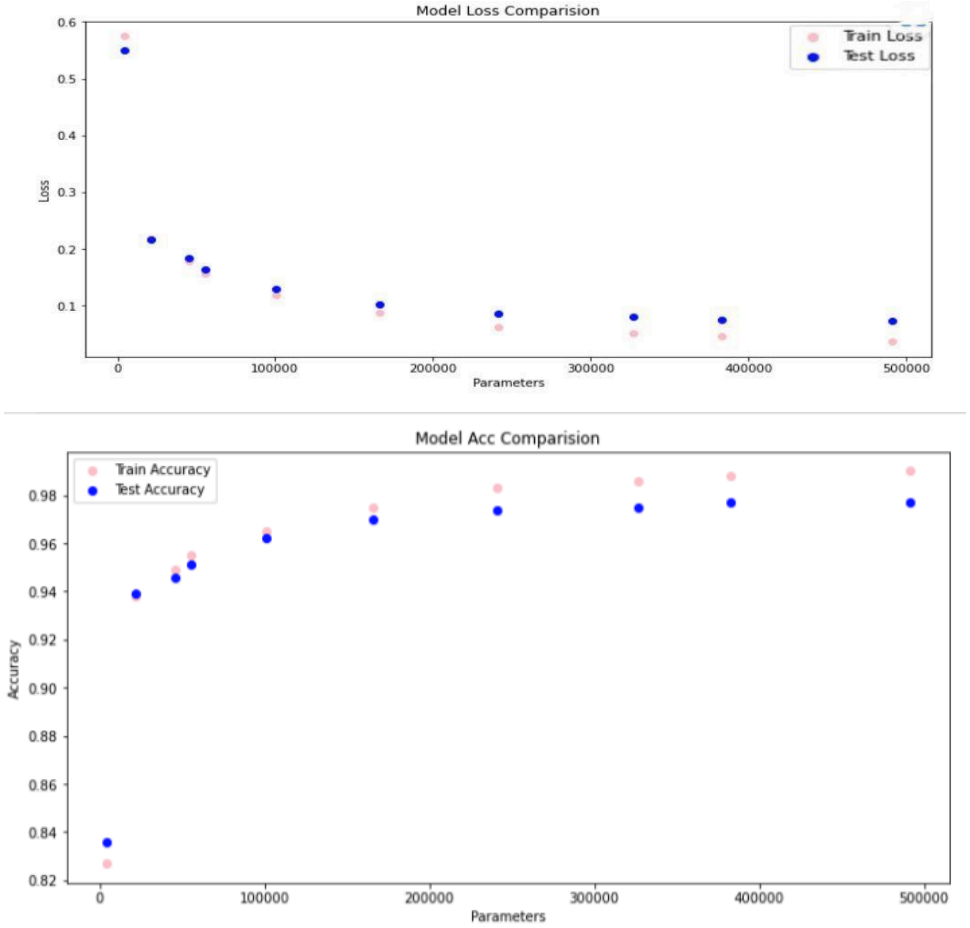
Can a network fit random labels?
To create this, 60,000 samples in a dataset were used, with an additional 10,000 samples reserved for testing. The architecture we chose was a CNN (LeNet), which incorporates key components such as a 2D convolution layer, followed by a 2D max pooling layer that is activated by a ReLU function. Furthermore, there is another 2D convolution layer, followed by a 2D max pooling layer and activated by ReLU, as well as a dense 2D layer with ReLU activation. Finally, the model has a dense 2D layer that is also activated by ReLU. Some key settings includes 0.0001 as learning rate value, Adam optimizer was used,batch size:100 for both training and testing, 100 epochs, and cross entropy as the method for calculating loss.



The training process, conducted on randomly distributed data, experienced a slow pace due to the necessity for the model to adapt to randomly assigned labels. Throughout the training, there was a noticeable effort by the model to internalize these labels, which resulted in a gradual decrease in training loss. Nonetheless, as training advanced through more epochs, a continual increase in test loss was observed alongside a decreasing training loss, as illustrated in the associated graph. This divergence between training and testing loss, which widened with more epochs, signals an overfitting issue, where the model excessively adjusts to the training dataset.

Number of parameters vs Generalization :
By utilizing the MNIST dataset, consisting of 60,000 training samples and 10,000 testing samples, this model was effectively trained. With a three dense layer architecture, a cross-entropy loss function, and the Adam optimizer, the model boasts a competitive performance. The learning rate was set at 0.001, while a batch size of 50 was used. The ReLU activation function was also implemented to further enhance the model's capabilities. In order to experiment with the model's size, each dense layer's input and output were

approximately doubled, resulting in an increase in the number of parameters requiring training. For a comprehensive understanding, the accompanying graphs illustrate the loss and accuracy comparisons among various model sizes.



The graphs clearly show that an increase in the model's parameters tends to widen the gap between training and testing loss/accuracy, with testing loss plateauing sooner than training loss or accuracy, a clear sign of overfitting. This phenomenon occurs because a larger parameter count makes the model more prone to overfitting, despite it achieving better training accuracy and lower training loss. Minimizing the discrepancy between training and testing loss/accuracy is crucial to combat overfitting. The graphs suggest a trend towards overfitting, yet limitations in computational resources prevented further expansion of the model's parameters.

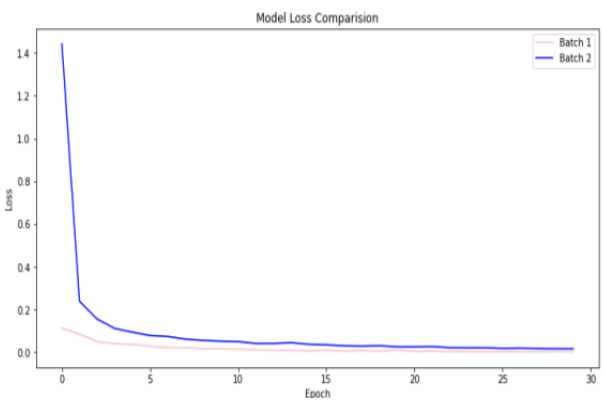
1-3 Flatness vs Generalization

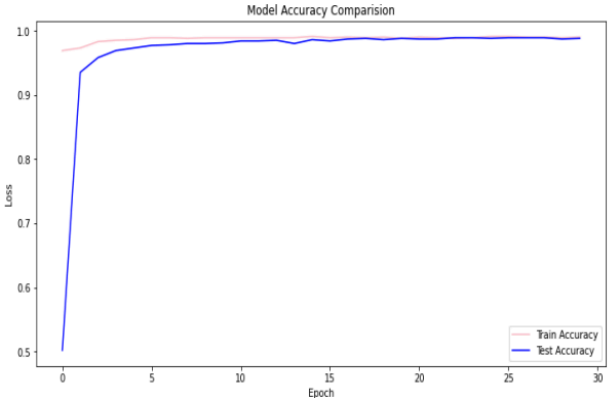
Part 1

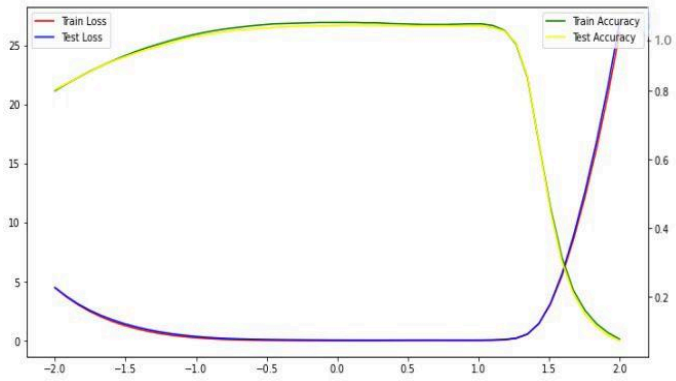
The model undergoes training using a dataset that includes 60,000 samples for training and 10,000 samples for testing, all sourced from the MNIST dataset. Its structure features three dense layers and two convolutional layers, employing Cross Entropy Loss as its loss measurement criterion. Stochastic Gradient Descent (SGD) serves as the optimization technique, with learning rates adjusted to 0.001 and 0.01. The model experiments with batch

sizes ranging from 100 to 500, while ReLU is the chosen activation function. During the training phase, weights from models trained across different batch sizes are aggregated to determine an interpolation ratio. This process leads to the creation of 50 distinct models, each bearing newly calculated weight values. The performance of these models, in terms of both loss and accuracy, is then documented and visually represented on a singular graph for analytical comparison.

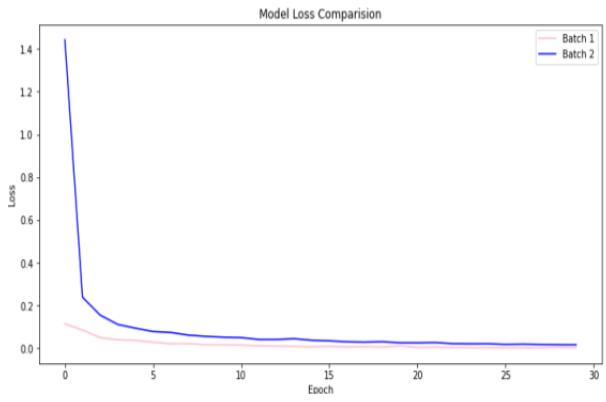
Lr: 1e-2

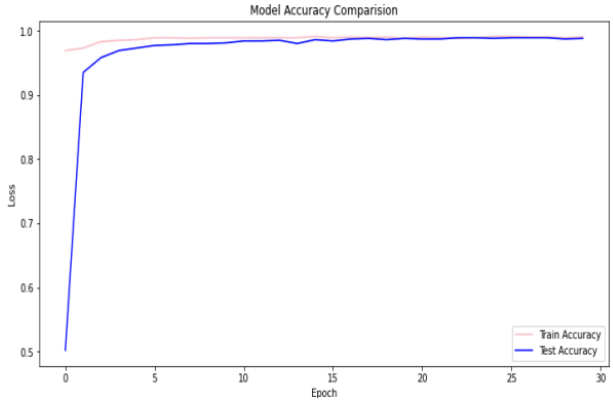


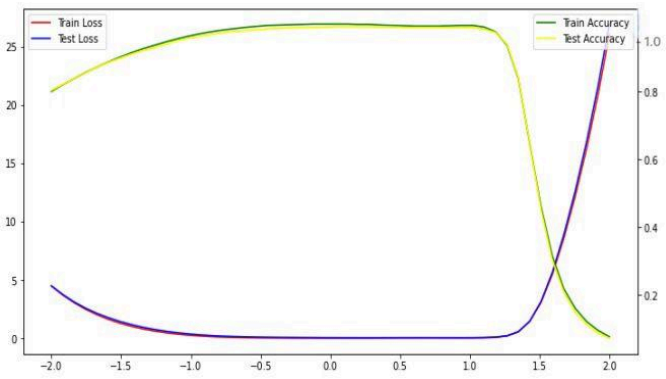




Lr 1e-3:





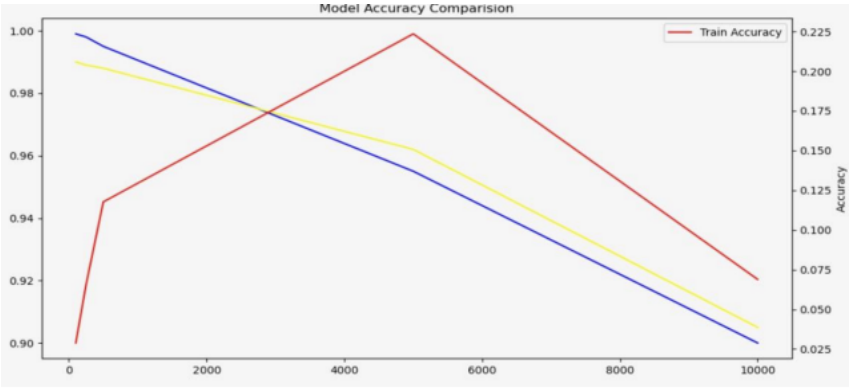


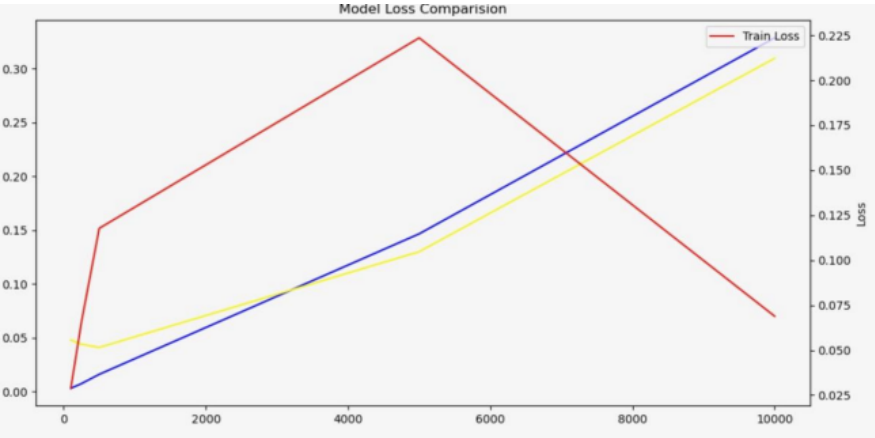
Findings:

The results of the experiment clearly indicate a decrease in both training and testing accuracy when different learning rates are utilized. However, there is a significant improvement in accuracy when the alpha value is set to 1.5, as determined by the interpolation ratio equation.This finding suggests that while all machine learning models engage in interpolation between data points, having a larger number of parameters than available data can result in the model overfitting and simply memorizing the data, leading to interpolation among these points.

Part 2

The described model, trained on a dataset comprising 60,000 MNIST samples and tested against 10,000 samples, features a structure with two convolutional layers and three fully connected layers, utilizing ReLU for activation. It employs Stochastic Gradient Descent (SGD) as the optimization technique with a learning rate of 0.001 and processes data in batches of 50. The Cross Entropy Loss function is used for evaluating performance.





Analysis of the graphs indicates that the network's responsiveness diminishes as the batch size is increased. The peak sensitivity is observed at approximately 4,000 batches, beyond which there is a noticeable decline in sensitivity, leading to reduced accuracy and higher loss. Therefore, it's observed that the network's effectiveness wanes when the batch size surpasses 5,000 epochs.