CPSC – 8430 Deep Learning

Reek Majumder

Part HW 1_1 : Deep Vs Shallow

Simulate a Function

Simulated Function 1
Code file- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/blob/main/Section1/HW1_1SimulatedFunc1_final/SimFunc1_Main.py
Simulated Function 2
Code file:- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/blob/main/Section1/HW1_1SimulatedFunc1_final/SimFunc2_Main.py

In this Assignment we have used three DNN models and used it to train functions

- $\frac{\sin(5\pi x)}{5\pi x}$ [Figure 1]
- $Sgn(\sin(5\pi x))$ [Figure 4]

Three Model Details

Features	**Model 0**	**Model 1**	**Model 2**
Dense Layers	7- Dense Layer	4- Dense Layer	1- Dense Layer
Activation Function After each layer	“Leaky_relu”	“Leaky_relu”	“Leaky_relu”
Total No. of parameters	573	574	573
Optimizer	Adam	Adam	Adam
Loss Function	Mean Squared Error (MSE)	Mean Squared Error (MSE)	Mean Squared Error (MSE)
Learning Rate	0.0004	0.0004	0.0004
Weight Decay	0.0001	0.0001	0.0001

- All the above models uses Adam optimizer with Weight decay which is a regularization techniques and avoids model overfitting.
- All the models were executed for 20,000 epochs for both the simulated function.
- For Simulated Function 1 = ($\frac{\sin(5\pi x)}{5\pi x}$) we find all three models to reach minimum loss faster than Simulated Function 2 =($Sgn(\sin(5\pi x))$) [Figure 2 vs Figure 5]
- For Simulated Function 1 we tested for less number of maximum epochs to find the convergence for three models [Figure 2a]
  Model 0 Convergence around 150 epochs
  Model 1 Convergence around 110 epochs
  Model 2 Convergence around 100 epochs
- We plotted the trained model with the actual function to understand how good the trained model fits the actual function. [Figure 3 and 6]

We observe that for complex function like simulated function 2 we see model 2 with 1 dense layer is trapped in local minima and the loss value does not decreases as low as for other models that is zero

Image Folder: - https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/tree/main/Section1/HW1_1SimulatedFunc1_final/result_final

Results For $\frac{\sin(5\pi x)}{5\pi x}$

Function Plot 1

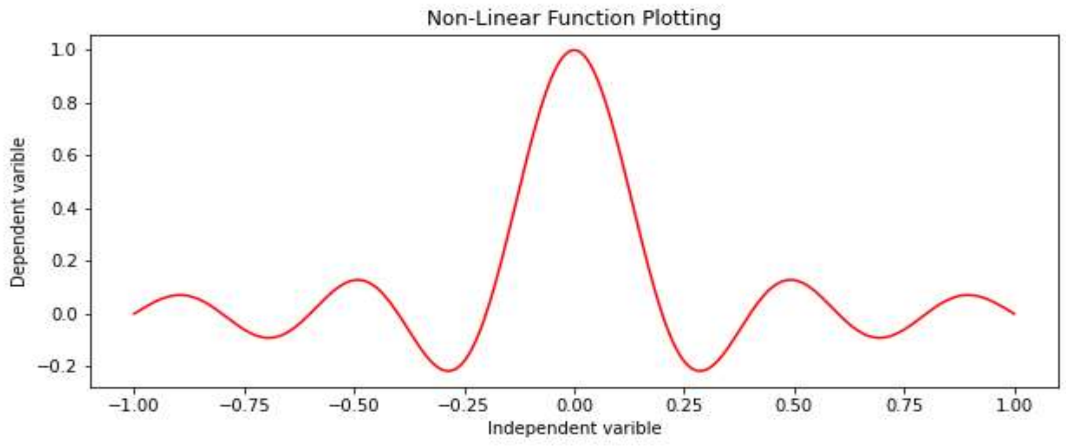

Figure 1: - Function plot for $\frac{sin(5\pi x)}{5\pi x}$

Loss Vs Epoch

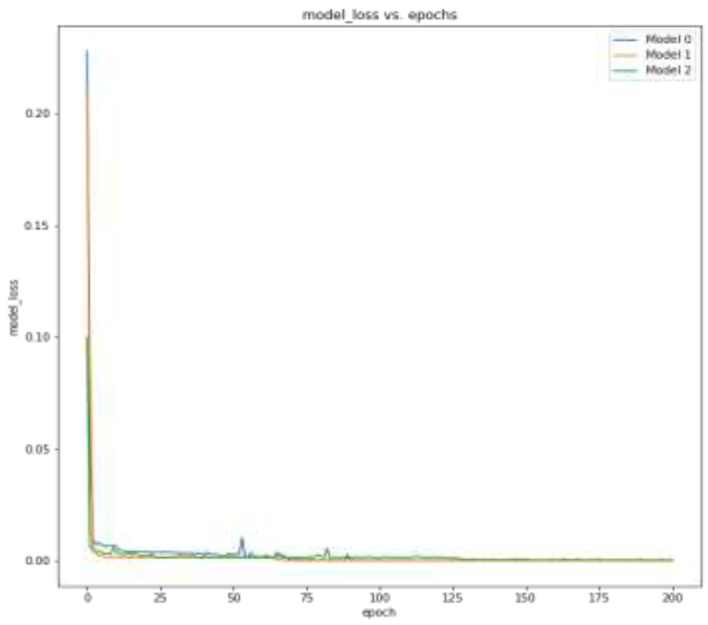

*Figure 2: - Model Loss vs number of epochs (Actual number of epochs is number in the graph *10)*

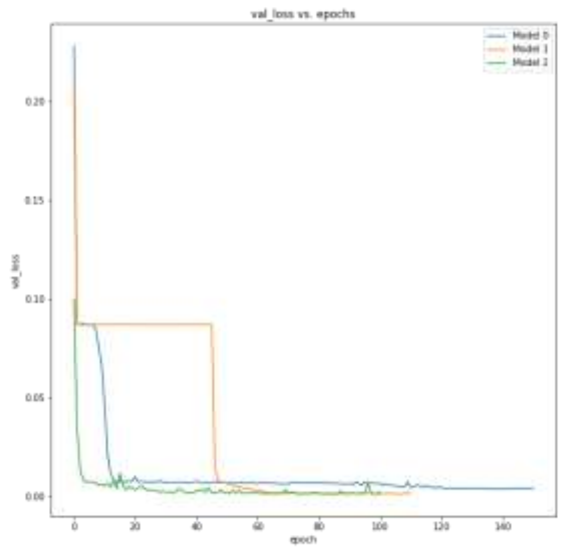

Figure 2a: Loss vs epoch with [convergence condition difference < 0.000001)

Predicted Model Vs Actual Function

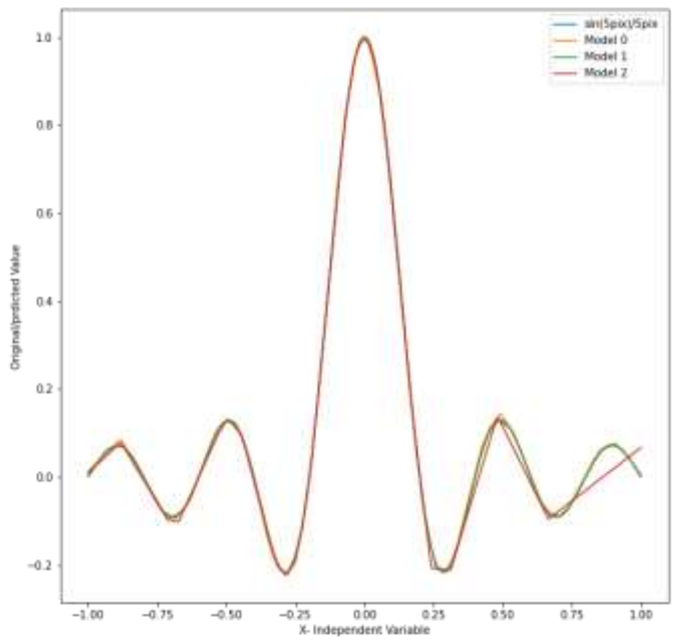

Figure 3:- Fitted three models with Actual Functions

Image Folder: https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/tree/main/Section1/HW1_1SimulatedFunc1_final/result_final2
Result for $Sgn(\sin(5\pi x))$

Function Plot

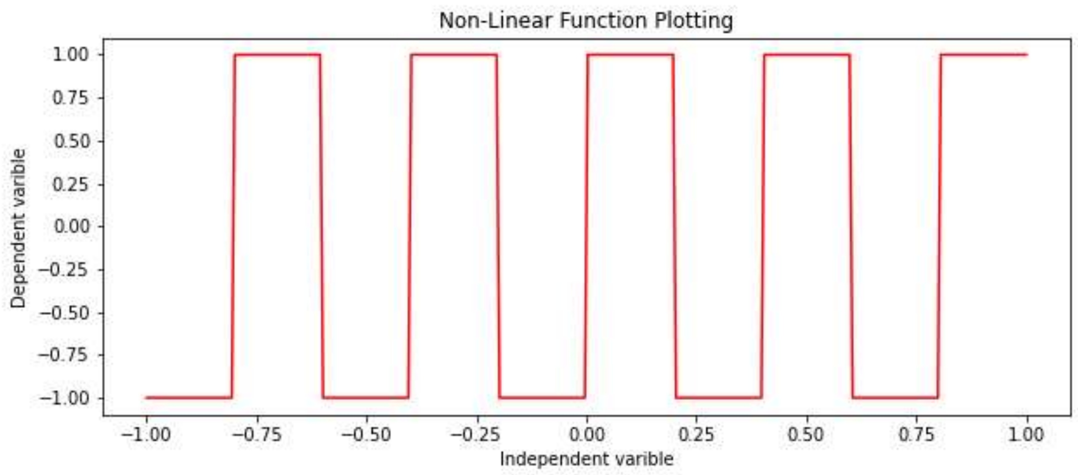

Figure 4: Function Plot for $Sgn(sin(5\pi x))$

Loss Vs Epochs

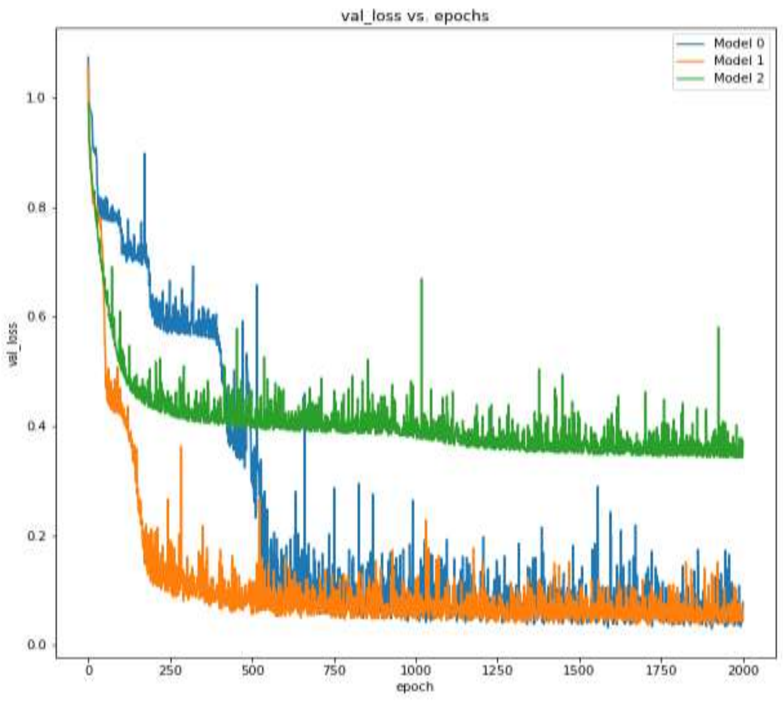

*Figure 5: Loss vs epochs for $Sgn(sin(5\pi x))$ [Actual Epoch is epoch * 10]*

Predicted Models Vs Actual Plot

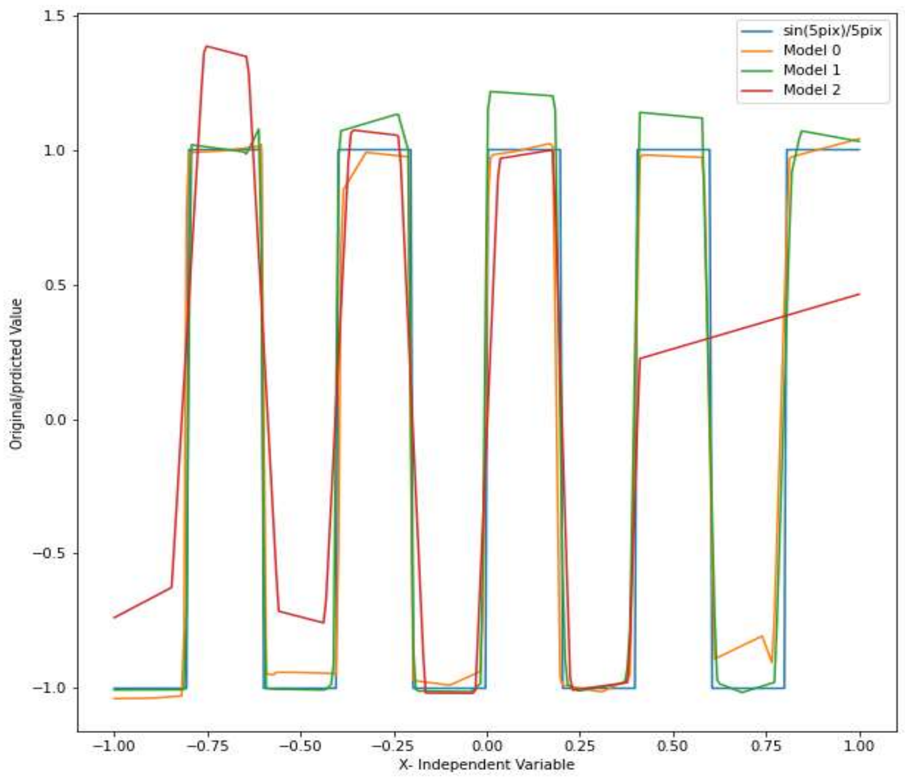

Figure 6:Fitted 3 models with actual function

Code file:- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/blob/main/Cifar/Cifar3Models_hw_1_3.ipynb
Image File:- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/tree/main/Cifar/reek_cifar_3Models_100_eps_alpha

Train on Actual Task: - CIFAR 10 dataset with three CNN models

For the Actual task we have trained three CNN models structured above on the Cifar 10 Dataset. We can Observe lowest values of loss yields highest value of accuracy. And models reach a constant loss and accuracy after 20 epochs. [Figure 7 and Figure 8]

Feature	**CiFar Model 1**	**Cifar Model 2**	**Cifar Model 7**
Number of Parameters	36458	46842	430102
Number of Convolution Layer (Layer No, Input channel , Output Channel, Kernel Size, Stride, Padding)	1-Convolution (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=1)	5- Convolution Layer (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=1) (**Layer2**, 16,32, kernel=(3,3), Stride=1, padding=1) (**Layer3**, 32,64, kernel=(3,3), Stride=1, padding=1) (**Layer4**, 64,32, kernel=(3,3), Stride=1, padding=1) (**Layer5**, 33,16, kernel=(3,3), Stride=1, padding=1)	2- Convolution Layer (**Layer1**, 3,32, kernel=(3,3), Stride=1, padding=1) (**Layer2**, 32,64, kernel=(3,3), Stride=1, padding=1)
Dense Layers (No, Input size, output Size)	1-Dense Layer (**Layer 1**,16*15*15,10)	1-Dense Layer (**Layer 1**,16*1*1,10)	2-Dense Layer (**Layer 1**,4096,100) (**Layer 2**,100,10)
Pool (Pool Size)	Max-Pool 2D (2,2)	Max-Pool 2D (2,2)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU), Softmax(dim=1)	Rectilinear Linear Unit (ReLU)	Rectilinear Linear Unit (ReLU)
Loss Function	Cross Entropy Loss	Cross Entropy Loss	Cross Entropy Loss
Optimizer Function	Adam	Adam	Adam
Learning Rate	0.0004	0.0004	0.0004
Weight Decay	0.0001	0.0001	0.0001
Gamma Scheduler	0.1	0.1	0.1
Schedular Step Size	10	10	10

	Train Loss	**Test Loss**	**Train Accuracy**	**Test Accuracy**
Model 0	1.919	1.924	55.16	54.356
Model 1	0.926	0.966	67.54	65.846
Model 2	0.980	0.9532	74.056	73.570

Results for Three Cifar CNN Models

Model Loss Vs Epochs

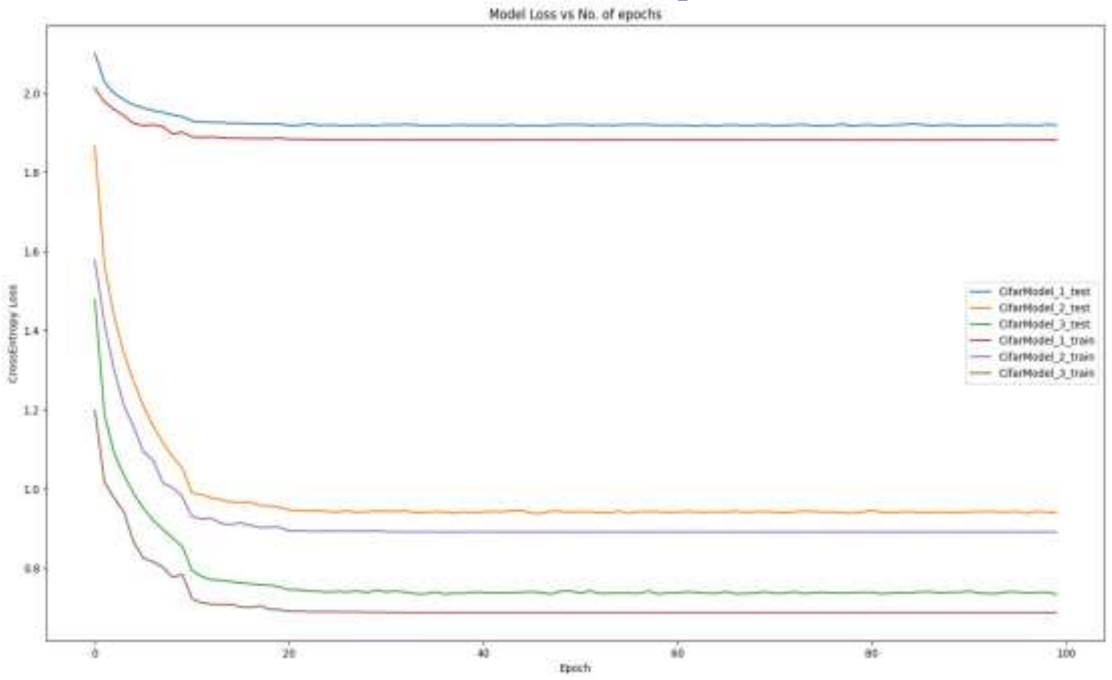

Figure 7:- Comparing Loss for three CNN models on Cifar10 dataset for 100 epochs

Model Accuracy Vs Epochs

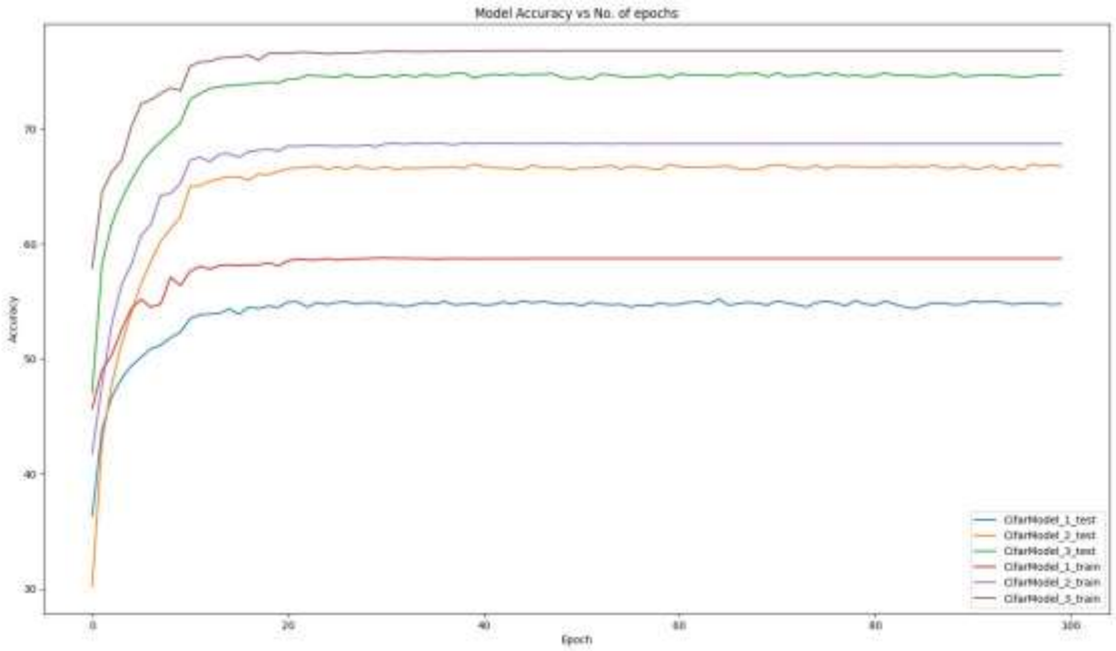

Figure 8: Comparing Accuracy of three trained CNN models for Cifar10 dataset for 100 epochs

File Location In Repo:-

Part HW 1_2: OPTIMIZATION

Code File :- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/blob/main/Cifar/cifar%2010_wts%20_16_feb_Final-Copy1.ipynb
Image folder:- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/tree/main/Cifar/reek_weights_trial_80iter_final2
For this section our goal is to observe the optimization Process.
For this we have used a model With Six Convolution Layer and two Dense Layer.
For each convolution layer is paired with Rectilinear Linear Unit(ReLU) and MaxPool 2d layer with Size (2,2).

Model Structure

Feature	**Cifar Model 21**
Number of Parameters	11085
Number of Convolution Layer **(Layer No, Input channel , Output Channel, Kernel Size, Stride, Padding)**	6-Convolution (**Layer1**, 3,8, kernel=(3,3), Stride=1, padding=1) (Layer**2**, 8,10, kernel=(3,3), Stride=1, padding=1) (Layer**3**, 10,20, kernel=(3,3), Stride=1, padding=1) (Layer**4**, 20,25, kernel=(3,3), Stride=1, padding=1) (Layer**5**, 25,16, kernel=(3,3), Stride=1, padding=1)
Dense Layers **(No, Input size, output Size)**	2-Dense Layer (**Layer 1**,16,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU)
Loss Function	Cross Entropy Loss
Optimizer Function	Adam
Learning Rate	0.0004
Weight Decay	0.0001
Gamma Scheduler	0.1
Schedular Step Size	10

Visualize the Optimization Process

Algorithm: -

- During our training phase we have saved the weights of each epoch as an vector by iterating through all the named parameters of the model and extracting weights from the model and storing it to the list of weights.
- We have executed the model for 8 times for 50 different epochs and created a data frame with number of total epochs vs number of weight column.
- Then we apply Principal Component Analysis (PCA) with 2 Dimensions on the data frame .
- We choose the points of every 3rd epoch of each time for plotting the weights in 2D Graph.
- Secondly we display the weights of whole model

Weights For Every Third Epoch

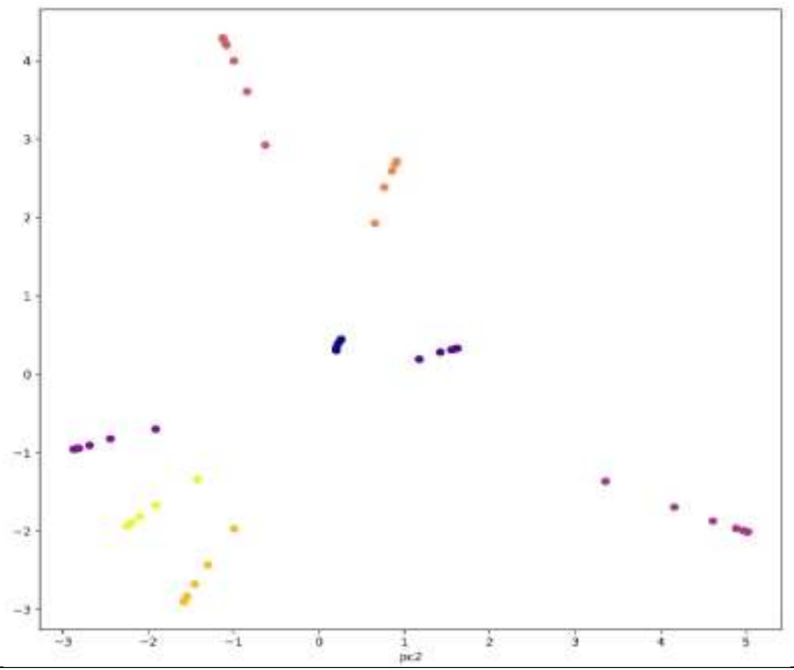

Figure11:- Displaying weights of every 3rd epoch for each time

Weights for whole model

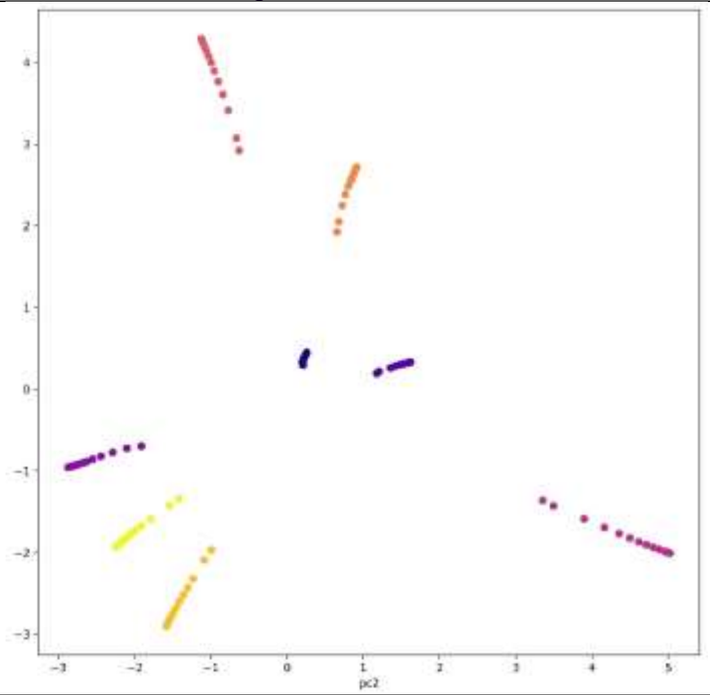

Figure 12:- Displaying weights of the whole models during each epoch and time

OBSERVE GRADIENT NORM

Code file :- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/blob/main/Section1/HW1_1SimulatedFunc1_final/Hw1_simulated_function1_withgrad_norm_final.ipynb
Image Folder:- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/blob/main/Section1/HW1_1SimulatedFunc1_final/result_final/Model%201_grad_norm%20vs%20loss.PNG

For this part of the assignment, I have observed the optimization process for simulated function 1, $\frac{\sin(5\pi x)}{5\pi x}$
And Model 1 from part 1 with 4 dense layers

Features	Model 1
Dense Layers	4- Dense Layer
Activation Function After each layer	"Leaky_relu"
Total No. of parameters	574
Optimizer	Adam
Loss Function	Mean Squared Error (MSE)
Learning Rate	0.0004
Weight Decay	0.0001

We observe that the fluctuations in grad norm reduces once the model reaches local minima.

What will happen when grad norm is equal to 0?
If grad norm is 0 the model will not learn in that step. If the grad_norm is 0 loss .backward won't change the parameters of the model.

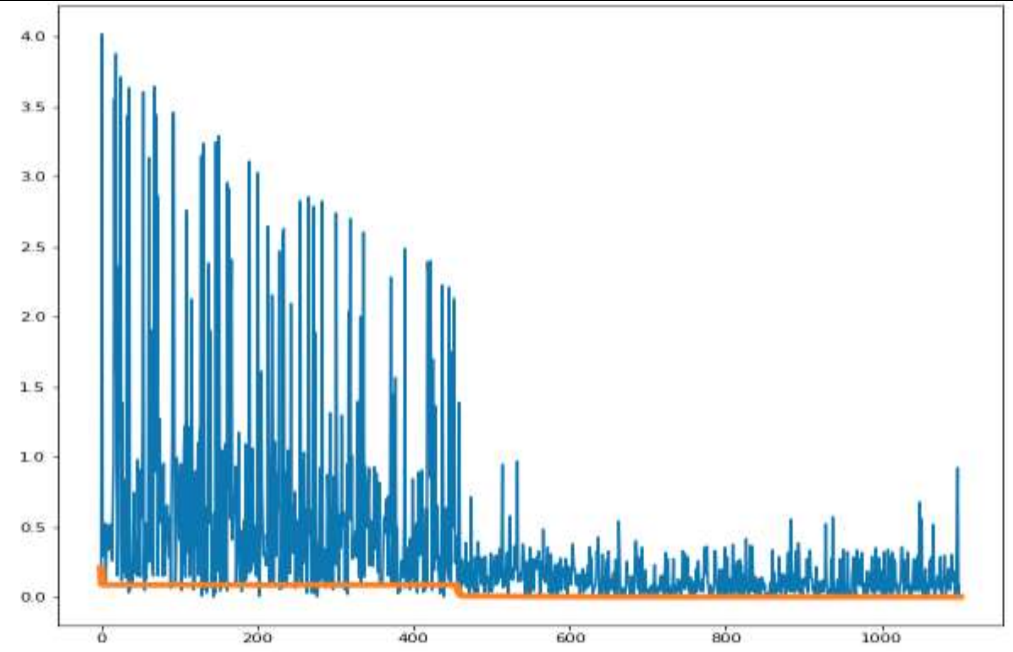

Figure 13: Grad_norm vs epoch and loss vs epoch

Minimal Ratio Vs Loss

For our chosen model we didn’t get grad norm =0.

Features	**Model 1**
Dense Layers	2- Dense Layer
Activation Function After each layer	“ReLu”
Optimizer	Adam
Loss Function	Mean Squared Error (MSE)
Learning Rate	0.0001

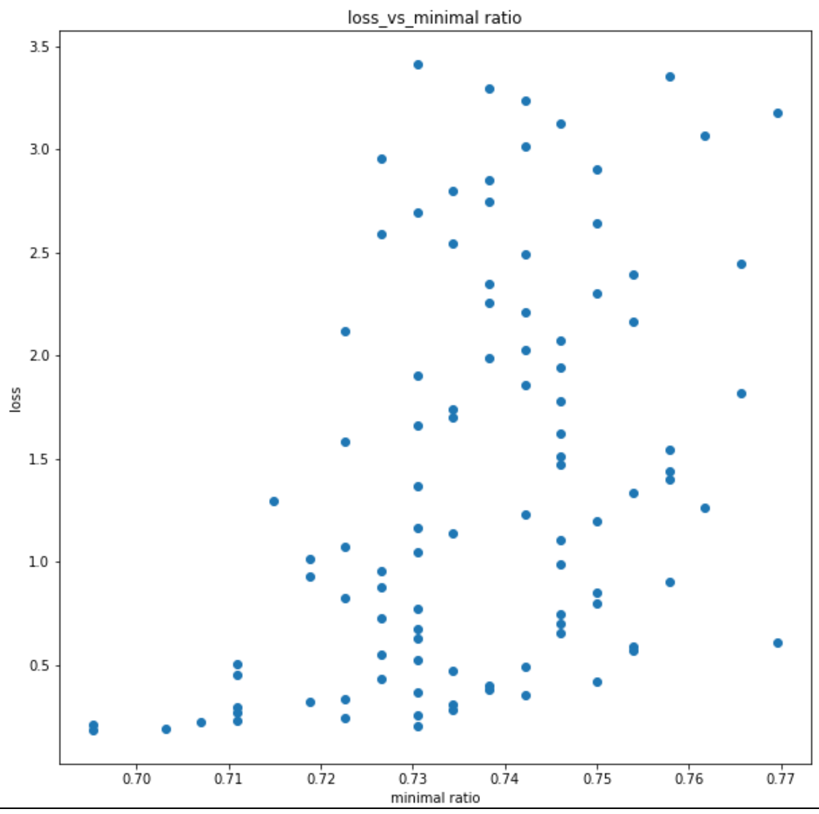

Figure :Loss Vs Minimal Ratio

GENERALIZATION

Image Folder:- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/tree/main/Mnist

Code File:- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/blob/main/Mnist/RandomMinist_Shuffle.py

In this part of the assignment, we train a deep learning model with MNIST dataset where we have randomized the labels in training dataset with respect to images and observe the behavior of the model on its learning process with random labels and compare its performance against testing data whose labels are properly labelled.

Our DNN models has 256 hidden nodes

Model Structure

Feature	RandomMnist
Number of Parameters	149002
Dense Layers **(No, Input size, output Size)**	3-Dense Layer (**Layer 1**,1024,128) (**Layer 2**,128,128) (**Layer 3**,128,10)
Activation Layer	Rectilinear Linear Unit (ReLU)
Loss Function	Cross Entropy Loss
Optimizer Function	Adam
Learning Rate	0.0001

We can observe the model can learn from random label dataset. But the model will perform poorly for the testing data. Because of this loss in training decreases but loss in testing increases for 1000 epochs.

The following graph was achieved on training the above model with the given shuffled training dataset and untampered dataset. [Figure 14]

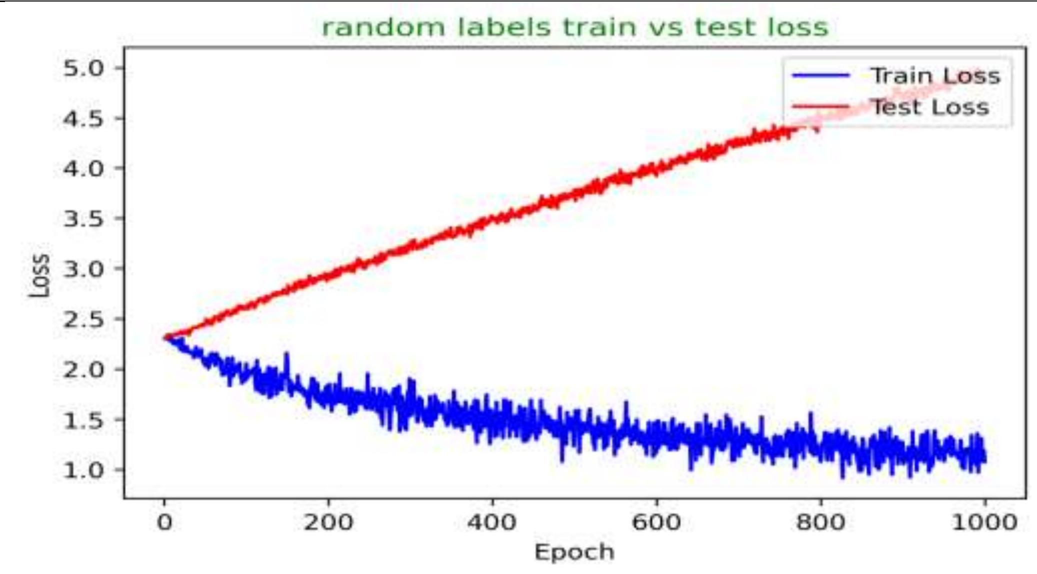

Figure 14:- Training Loss Decreases with epochs while testing loss increases

Number of Parameters Vs Generalization

Code File:- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/blob/main/Cifar/Cifar10Models_final_notebook_3_1.ipynb

Image Folder :- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/tree/main/Cifar/reek_cifar10_model_10Models_finalResult

- In this exercise we train our 10 CNN models for Cifar10 Dataset having the same structure but varying the values of CNN input and output channel and dense layer input and output parameters.
- We can observer from Figure below that as number of parameters increases, the model has better performance that is lower loss and higher accuracy. Hence, we can conclude that as the number of parameters increases, the performance of the model increases while having a same model structure.

Each model is trained for Cross Entropy Loss with Adam optimizer with following Hyperparameters

Hyper parameter	
Feature	**Parameter**
Loss Function	Cross Entropy Loss
Optimizer Function	Adam
Learning Rate	0.0004
Weight Decay	0.0001
Gamma Scheduler	0.1
Schedular Step Size	10

Feature	**Cifar Model 21**
Number of Parameters	11085
Number of Convolution Layer **(Layer No, Input channel, Output Channel, Kernel Size, Stride, Padding)**	6-Convolution (**Layer1**, 3,8, kernel=(3,3), Stride=1, padding=1) (Layer**2**, 8,10, kernel=(3,3), Stride=1, padding=1) (Layer**3**, 10,20, kernel=(3,3), Stride=1, padding=1) (Layer**4**, 20,25, kernel=(3,3), Stride=1, padding=1) (Layer**5**, 25,16, kernel=(3,3), Stride=1, padding=1)
Dense Layers **(No, Input size, output Size)**	2-Dense Layer (**Layer 1**,16,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU)

Feature	**Cifar Model 22**
Number of Parameters	12029
Number of Convolution Layer **(Layer No, Input channel, Output Channel, Kernel Size, Stride, Padding)**	6-Convolution (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=1) (Layer**2**, 16,10, kernel=(3,3), Stride=1, padding=1) (Layer**3**, 10,20, kernel=(3,3), Stride=1, padding=1) (Layer**4**, 20,25, kernel=(3,3), Stride=1, padding=1) (Layer**5**, 25,16, kernel=(3,3), Stride=1, padding=1)
Dense Layers **(No, Input size, output Size)**	2-Dense Layer (**Layer 1**,16,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU)

Feature	**Cifar Model 23**
Number of Parameters	13656
Number of Convolution Layer **(Layer No, Input channel, Output Channel, Kernel Size, Stride, Padding)**	6-Convolution (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=1) (Layer**2**, 16,10, kernel=(3,3), Stride=1, padding=1) (Layer**3**, 10,32, kernel=(3,3), Stride=1, padding=1) (Layer**4**, 32,20, kernel=(3,3), Stride=1, padding=1) (Layer**5**, 20,16, kernel=(3,3), Stride=1, padding=1)
Dense Layers **(No, Input size, output Size)**	2-Dense Layer (**Layer 1**,16,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU)

Feature	**Cifar Model 24**
Number of Parameters	16904
Number of Convolution Layer **(Layer No, Input channel, Output Channel, Kernel Size, Stride, Padding)**	6-Convolution (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=1) (Layer**2**, 16,25, kernel=(3,3), Stride=1, padding=1) (Layer**3**, 25,20, kernel=(3,3), Stride=1, padding=1) (Layer**4**, 20,25, kernel=(3,3), Stride=1, padding=1) (Layer**5**, 25,16, kernel=(3,3), Stride=1, padding=1)
Dense Layers **(No, Input size, output Size)**	2-Dense Layer (**Layer 1**,16,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU)

Feature	**Cifar Model 25**
Number of Parameters	29254
Number of Convolution Layer **(Layer No, Input channel, Output Channel, Kernel Size, Stride, Padding)**	6-Convolution (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=1) (Layer**2**, 16,64, kernel=(3,3), Stride=1, padding=1) (Layer**3**, 64,20, kernel=(3,3), Stride=1, padding=1) (Layer**4**, 20,24, kernel=(3,3), Stride=1, padding=1) (Layer**5**, 24,16, kernel=(3,3), Stride=1, padding=1)
Dense Layers **(No, Input size, output Size)**	2-Dense Layer (**Layer 1**,16,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU)

Feature	**Cifar Model 26**
Number of Parameters	52004
Number of Convolution Layer (Layer No, Input channel, Output Channel, Kernel Size, Stride, Padding)	6-Convolution (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=1) (Layer**2**, 16,128, kernel=(3,3), Stride=1, padding=1) (Layer**3**, 128,20, kernel=(3,3), Stride=1, padding=1) (Layer**4**, 20,30, kernel=(3,3), Stride=1, padding=1) (Layer**5**, 30,16, kernel=(3,3), Stride=1, padding=1)
Dense Layers (No, Input size, output Size)	2-Dense Layer (**Layer 1**,16,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU)

Feature	**Cifar Model 27**
Number of Parameters	63054
Number of Convolution Layer (Layer No, Input channel, Output Channel, Kernel Size, Stride, Padding)	6-Convolution (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=1) (Layer**2**, 16,128, kernel=(3,3), Stride=1, padding=1) (Layer**3**, 128,20, kernel=(3,3), Stride=1, padding=1) (Layer**4**, 20,64, kernel=(3,3), Stride=1, padding=1) (Layer**5**, 64,16, kernel=(3,3), Stride=1, padding=1)
Dense Layers (No, Input size, output Size)	2-Dense Layer (**Layer 1**,16,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU)

Feature	**Cifar Model 28**
Number of Parameters	74754
Number of Convolution Layer **(Layer No, Input channel, Output Channel, Kernel Size, Stride, Padding)**	6-Convolution (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=1) (Layer**2**, 16,128, kernel=(3,3), Stride=1, padding=1) (Layer**3**, 128,20, kernel=(3,3), Stride=1, padding=1) (Layer**4**, 20,100, kernel=(3,3), Stride=1, padding=1) (Layer**5**, 100,16, kernel=(3,3), Stride=1, padding=1)
Dense Layers **(No, Input size, output Size)**	2-Dense Layer (**Layer 1**,16,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU)

Feature	**Cifar Model 29**
Number of Parameters	83854
Number of Convolution Layer **(Layer No, Input channel, Output Channel, Kernel Size, Stride, Padding)**	6-Convolution (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=1) (Layer**2**, 16,128, kernel=(3,3), Stride=1, padding=1) (Layer**3**, 128,20, kernel=(3,3), Stride=1, padding=1) (Layer**4**, 20,128, kernel=(3,3), Stride=1, padding=1) (Layer**5**, 128,16, kernel=(3,3), Stride=1, padding=1)
Dense Layers **(No, Input size, output Size)**	2-Dense Layer (**Layer 1**,16,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU)

Feature	**Cifar Model 30**
Number of Parameters	130122
Number of Convolution Layer **(Layer No, Input channel, Output Channel, Kernel Size, Stride, Padding)**	6-Convolution (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=1) (Layer**2**, 16,128, kernel=(3,3), Stride=1, padding=1) (Layer**3**, 128,20, kernel=(3,3), Stride=1, padding=1) (Layer**4**, 20,128, kernel=(3,3), Stride=1, padding=1) (Layer**5**, 128,16, kernel=(3,3), Stride=1, padding=1)
Dense Layers **(No, Input size, output Size)**	2-Dense Layer (**Layer 1**,16,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU)

Feature	**Cifar Model 31**
Number of Parameters	425194
Number of Convolution Layer **(Layer No, Input channel, Output Channel, Kernel Size, Stride, Padding)**	6-Convolution (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=1) (Layer**2**, 16,256, kernel=(3,3), Stride=1, padding=1) (Layer**3**, 256,128, kernel=(3,3), Stride=1, padding=1) (Layer**4**, 128,64, kernel=(3,3), Stride=1, padding=1) (Layer**5**, 64,32, kernel=(3,3), Stride=1, padding=1)
Dense Layers **(No, Input size, output Size)**	2-Dense Layer (**Layer 1**,32,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU)

Loss Vs Number of Parameter

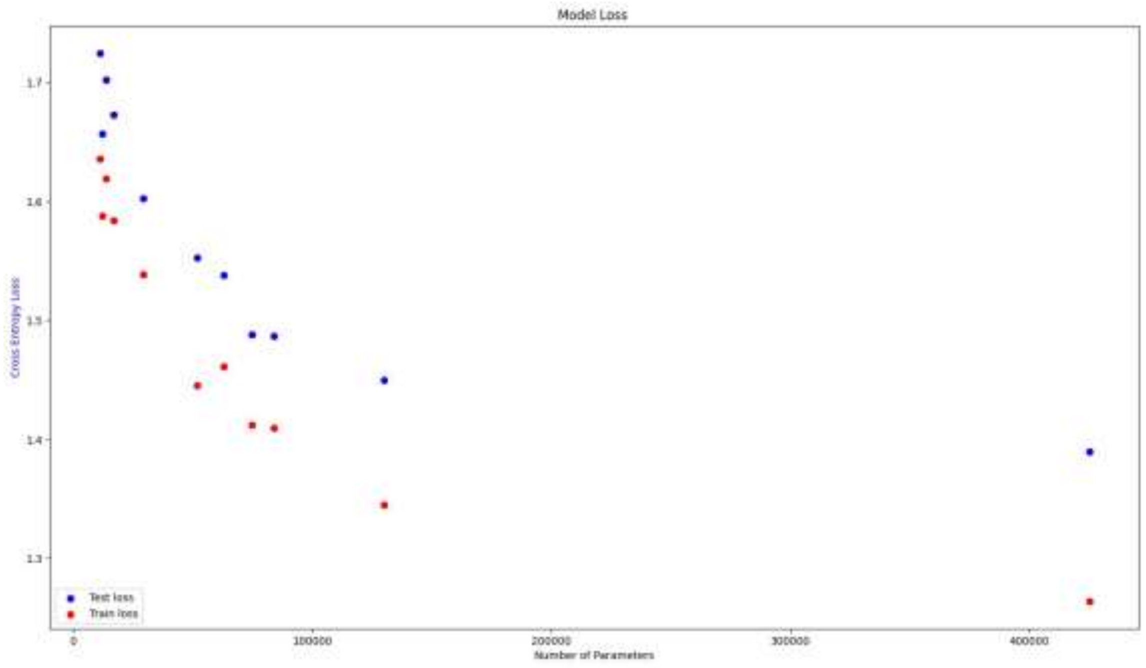

Figure 15: Decrease in loss with Increase in Parameters for Model 21 to Model 31

Accuracy Vs Number of Parameter

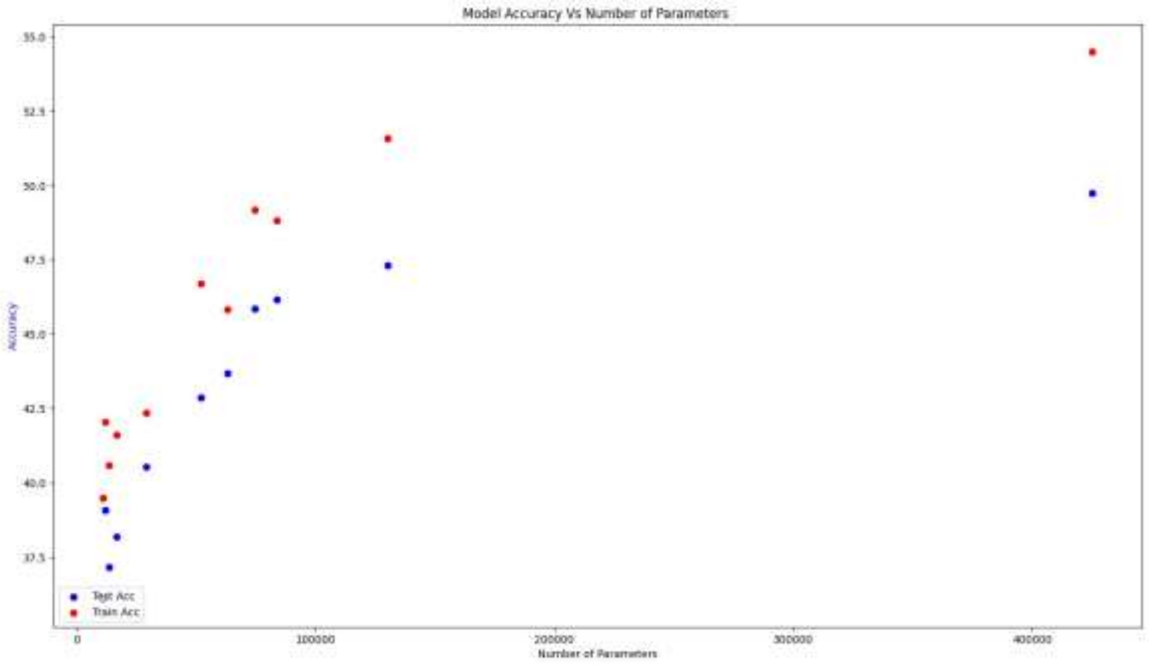

Figure 16: Increase in Accuracy with increase in parameters for Model 21 to Model 31

Flatness Vs Generalization

For this section of the assignment, we have used a deep learning model with one convolutional layer and one dense Layer from Cifar Model 1 and tested it for different Batch Size and Learning Rate.

Batch Size Code File:- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/blob/main/Cifar/Batch_sizes_models_hw_3_2_3_3.ipynb

Batch Size Result Folder:- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/tree/main/Cifar/batch_based_problems_final_eps_25

Feature	**Cifar Model 1**
Number of Parameters	36458
Number of Convolution Layer (Layer No, Input channel , Output Channel, Kernel Size, Stride, Padding)	1-Convolution (**Layer1**, 3,16, kernel=(3,3), Stride=1, padding=0)
Dense Layers (No, Input size, output Size)	2-Dense Layer (**Layer 1**,3600,10)
Pool (Pool Size)	Max-Pool 2D (2,2)
Activation Layer	Rectilinear Linear Unit (ReLU), Softmax(dim=1)
Loss Function	Cross Entropy Loss
Optimizer Function	Adam
Weight Decay	0.0001
Gamma Scheduler	0.1
Schedular Step Size	10

Training the above model for batch size [16,64,256,1024,4096]

- We trained the same model Cifar Model 1 on different batch sizes 16,64,256,1024,4096.
- During the training of each model, we calculated their sensitivities using Frobenius norm along with the loss, accuracy.
- Previous Step helped us to analyze the impact of increasing batch size which increases training and testing loss while reducing with sensitivity of the model. (Figure 17) *[Flatness and Generalization Part 2]*
- It also helped us to analyze the impact of increasing batch size which decreases training and testing accuracy while reducing the sensitivity of the model. (Figure 18) *[Flatness and Generalization Part 2]*
- Post Training the above models we extract the parameters in vector format for best model trained on batch size 64 and 1024 *[Flatness and Generalization Part 1]*
- We created 21 numbers between -2 to 2 as alpha (α) values
- We calculated the new parameter θ for different values of alpha using the formula
  θ = (1-α) * batch_64_parameter + α * batch_1024_parameter
- Each 21 vectors of θ values are converted into an model parameter.

- New 21 models are tested for training and testing loss and model Sensitivity (Figure 19)
- New 21 models are tested for training and testing accuracy and model Sensitivity (Figure 19)

We observe from the below graphs than accuracy and sensitivity of models decreases with the increase in batch size. However, loss increases with the increase in batch size.

Loss, Sensitivity and Batch Size

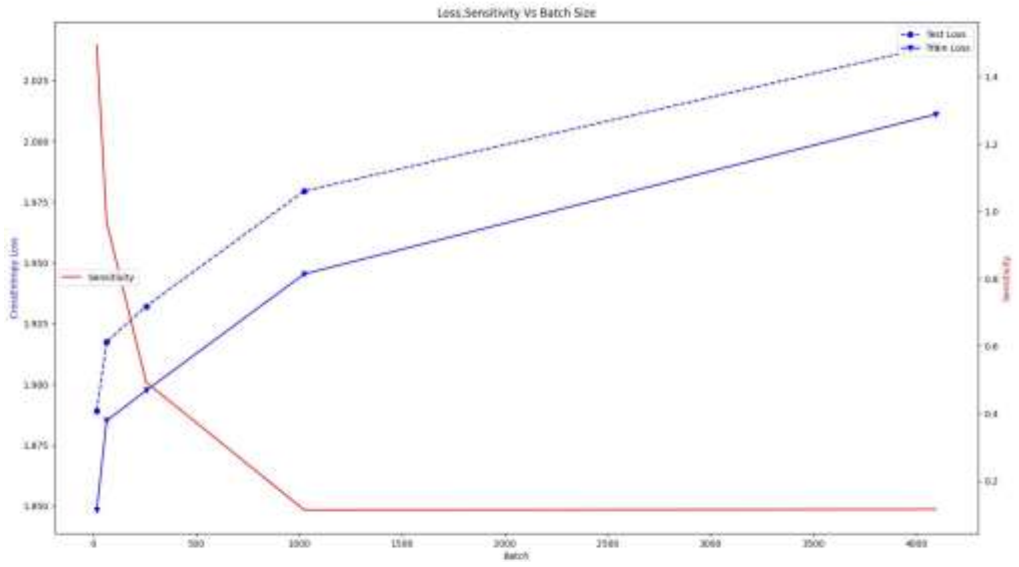

Figure 17: Increase in Training and testing loss and decrease in sensitivity with increase in Batch Size

Accuracy, Sensitivity and Batch Size

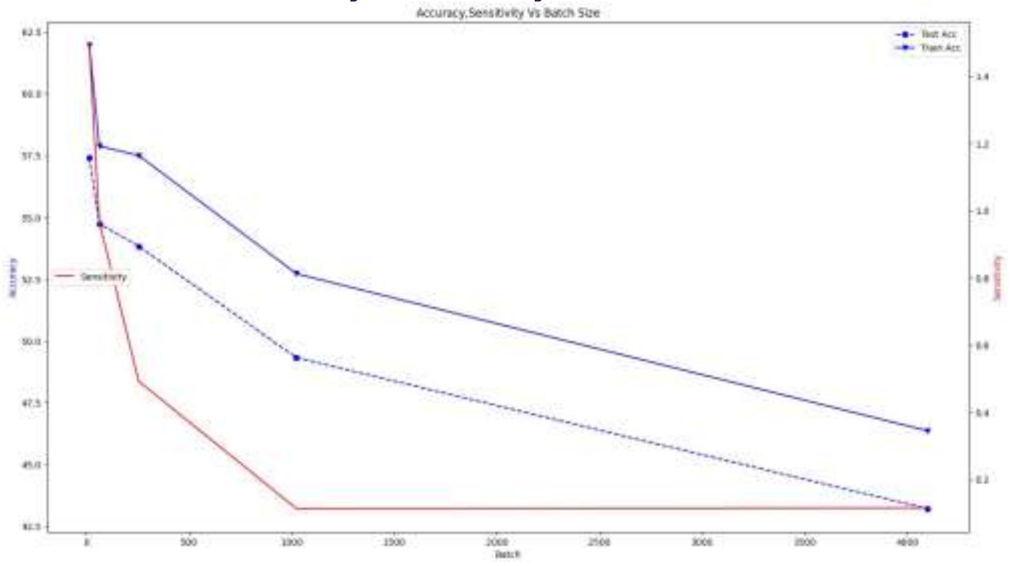

Figure 18: Decrease in training and test accuracy and model sensitivity with increase in batch Size

Training and Testing Loss and Accuracy for different values of alpha while comparing between batch size 64 and 1024

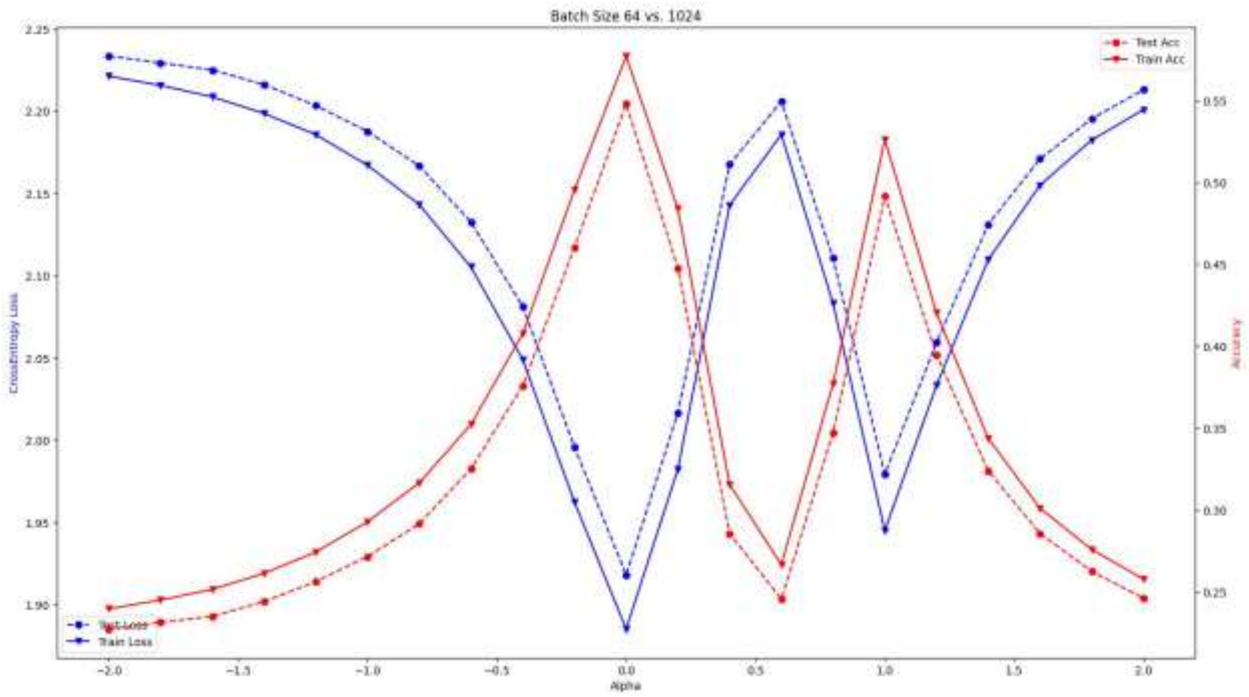

Figure 19: Change in Accuracy and loss based on alpha values

Learning Rate Analysis

Code File:- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/blob/main/Cifar/lr_alpha_hw_3_2_3_3.ipynb

Result Folder:- https://github.com/reek129/CPSC_8430--Deep_Learning_HW1/tree/main/Cifar/lrs_based_problem_final_eps_25

Similarly, we did for learning rates [0.1,0.01,0.001,0.0001,0.000001]

- We trained the same model Cifar Model 1 on different Learning Rate 0.1,0.01,0.001,0.0001,0.00001
- During the training of each model, we calculated their sensitivities using Frobenius norm along with the loss, accuracy.
- Previous Step helped us to analyze the impact of increasing learning rate which increases training and testing loss and decreases with sensitivity of the model. (Figure 20) *[Flatness and Generalization Part 2]*
- It also helped us to analyze the impact of increasing learning rate which decreases training and testing accuracy and model sensitivity. (Figure 21) *[Flatness and Generalization Part 2]*
- Post Training the above models we extract the parameters in vector format for best model trained on learning rate 0.001 and 0.01
- We created 21 numbers between -2 to 2 as alpha (α) values
- We calculated the new parameter θ for different values of alpha using the formula
  θ = (1-α) * lrs_0_001_parameter + α * lrs_0_01_parameter
- Each 21 vectors of θ values are converted into a model parameter.
- New 21 models are tested for training and testing loss and model Sensitivity (Figure 22)
- New 21 models are tested for training and testing accuracy and model Sensitivity (Figure 22)

We observe from the below graphs than accuracy and sensitivity of models decreases with the increase in learning rate. However, loss increases with the increase in learning rate.

Loss, Sensitivity Vs Learning Rate

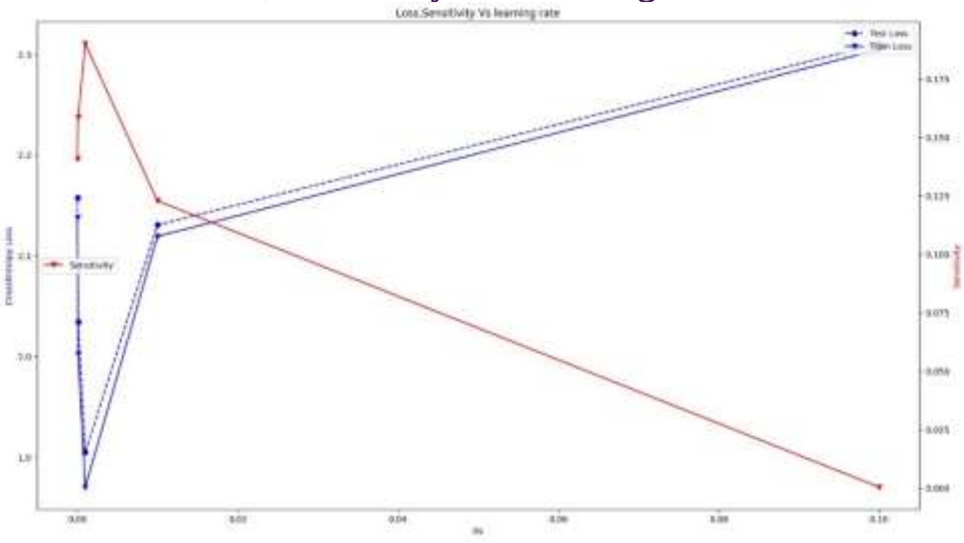

Figure20: Increase in training and testing loss and decrease in sensitivity with increase in learning rate

Accuracy, Sensitivity and Learning Rate

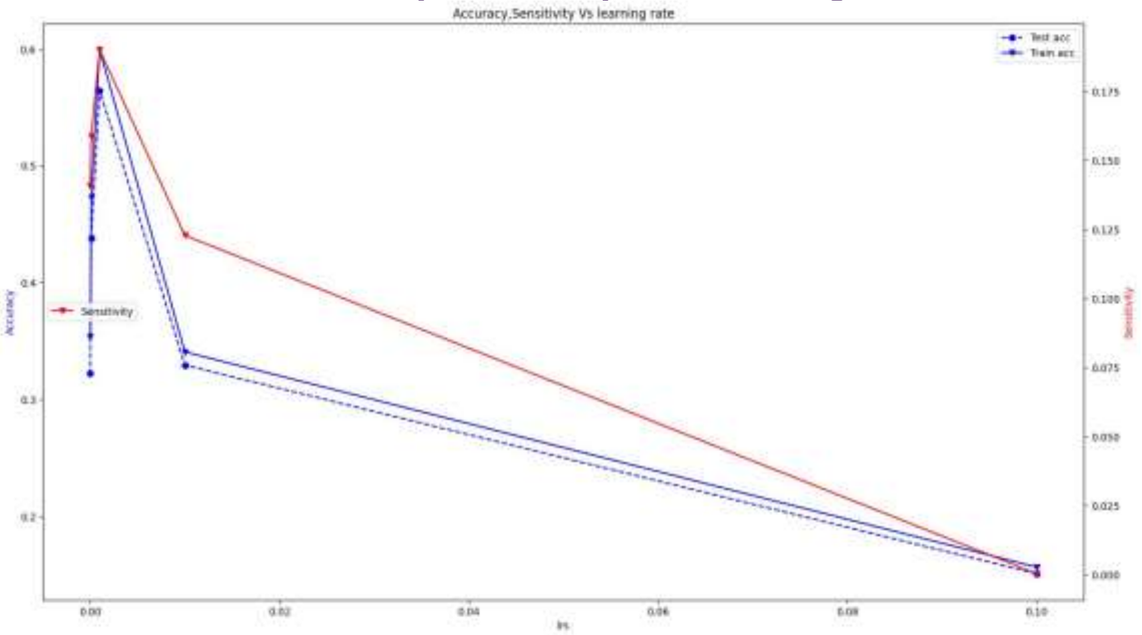

Figure21: Decrease in training and testing accuracy and sensitivity with increase in learning rate

Training and Testing Loss and Accuracy for different values of alpha while comparing between learning rate 0.01 and 0.001

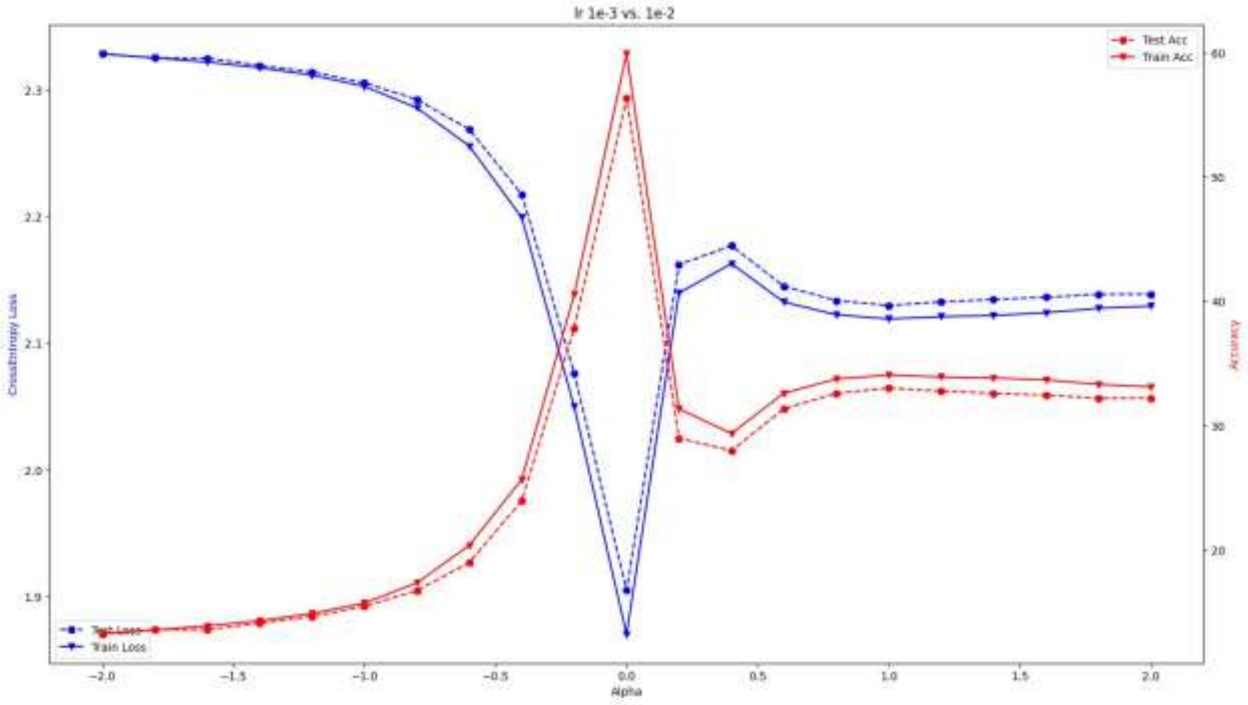

Figure 22 : Change in accuracy and loss based on different alpha values