**Code Explanation and Conclusion**

**Code Explanation:**

**Importing Necessary Module**

The code begins by importing necessary libraries and modules. These include PyTorch , torchvision, math, sklearn and matplotlib. All the above modules are for their respective features.

**ArcLoss Class Definition**

The “ArcLoss” class is defined as a subclass of “nn.Module”. This class implements the ArcLoss function for classification. The Arcloss Class contains different functions and parameters.

* “\_\_init\_\_” method: This method initializes the parameters of the ArcLoss function. It takes the following parameters:
* “embedding\_size”: The number of input features (dimension of the input).
* “num\_class”: The number number of classes.
* “scale”: Scaling factor as prescribed by Arcloss Creator.
* “margin”: Margin parameter as prescribed by Arcloss Creator.

“forward” method: This method defines the forward pass of the ArcLoss function. It takes the following parameters:

* “inputs”: Input data (features).
* “labels”: Target labels.

Within the forward method:

* “logits”: Computes the normalized input features. This ensures that each feature vector has unit norm.
* “arc”: The arc cosine (arc) of the logits is computed using torch.acos. This ensures that the arc cosine is within the range [0, π].
* “target\_cos”: The target cosine (target\_cos) is computed by adding the margin (self.margin) to the arc cosine and then taking the cosine of the result.
* “one\_hot”: A one-hot tensor (one\_hot) is created to mark the ground truth class for each sample.
* “output”: The final output is computed by applying the target cosine to the one-hot tensor (for the ground truth classes) and using the original cosine similarity for other classes. This is scaled by the scale parameter.

**Loss Computation:**

The output tensor is passed through F.cross\_entropy along with the ground truth labels to compute the ArcFace loss. F.cross\_entropy combines softmax and cross-entropy loss in a single function call. It takes care of computing the softmax function over the logits (scaled output), and then computes the cross-entropy loss between the softmax probabilities and the ground truth labels.

**Backward Pass:**

The backward pass is not explicitly implemented because F.cross\_entropy automatically computes gradients during the backward pass.

**Transforms and Dataset Loading:** Defines the data transformations and loads the CIFAR-10 dataset using torchvision. We have also implement some code to select balanced sample from the dataset in order to save time for training and testing while development.

**Model Definition:** Defines the neural network model (“Net”) using “nn.Module”. It consists of convolutional layers followed by fully connected layers. We have used multi Layer Network for better accuracy.

**Model Initialization:** Creates an instance of the model (“Net”) and moves it to the appropriate device (CPU or GPU).

**Criterion and Optimizer Definition:** Defines the ArcLoss criterion and the Adam optimizer. These function will be used through the training process to calculate the loss and optimize the model while training.

**Training Loop:** Executes the training loop over a specified number of epochs. Within each epoch, it iterates through batches of data, computes the loss using the ArcLoss criterion, backpropagates the gradients, and updates the model parameters using the optimizer. It also gives us the Test Accuracy after every Epoch. It also stores the ongoig loss and accuracy after every epoch inoreder to plot graphs.

**Plotting Graph:** Using the Data collected while Training Process, we plot Loss Curve Graph and Accuracy Curve Graph.

**Total Accuracy and Precision:** We Finally Test the Model on our Test Dataset to calculate the Final Accuracy and Precision of Model.

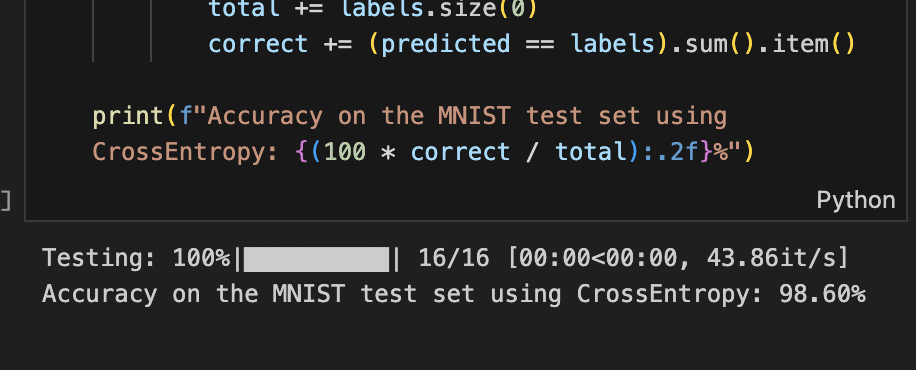
**Confusion Matrix Plot:** In order to understand the Performance of the Model in better way, we plot confusion Matrix.

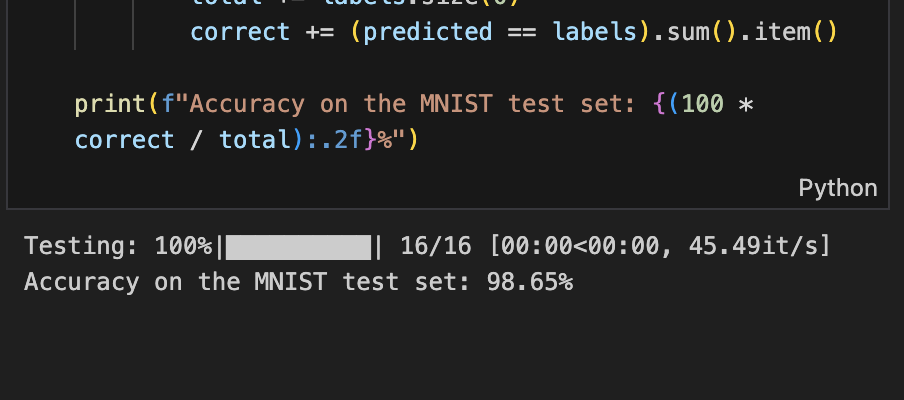
**CIFAR-10 and MNIST:**

Both the Code are very similar. The required changes in the DataLoader Function has been done in order to load respective datasets. Also, the Model Layers are updates as per Dataset. The Cifar dataset model network is deeper and accepts multi-channel input for color image. Finally Similar Graphs has been plotted for both the dataset.

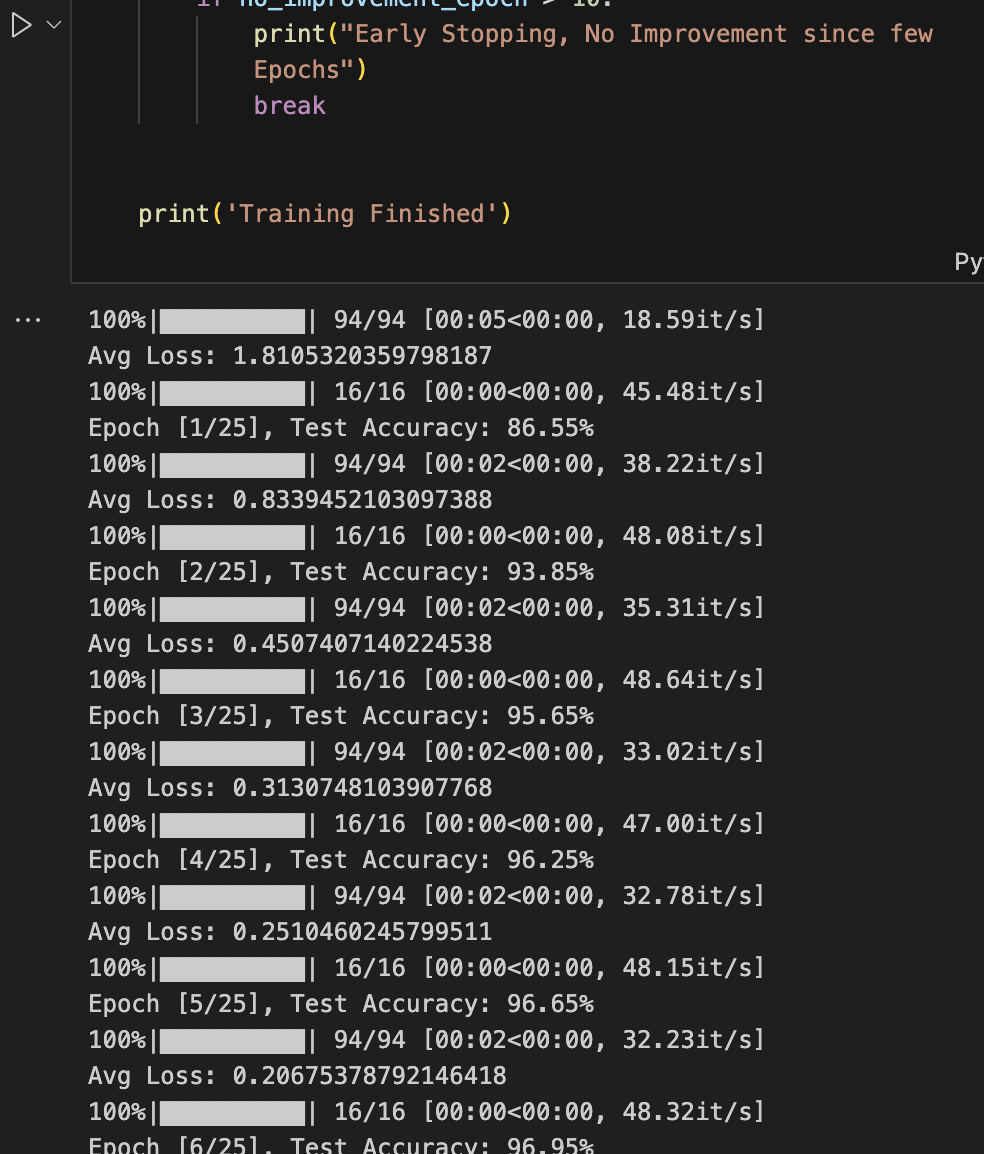
**Conclusion:**

After Working on the code for extensive period and running the code hundreds of time while making changes in order to achieve good accuracy. It became very clear that how important a Loss Function is for Model Training and Accuracy, and even a slight mistake or change in the loss function can severely affect the model performance.  
While completion of the assignment, initially we were getting a very poor accuracy because of mistakes in implementing the loss function. Finally, slowly but surely we kept on making changes but understanding what is going wrong and finally were able to get good accuracy.   
For our testing, we were comparing the accuracy of our Model with ArcLoss function vs The Model using standard CrossEntropyLoss Function. After final implementation, we observed that the Model using Arcloss is giving slighltly better result.   
The result is very limited and anything cannot be stated confidently, but we observed a small performance improvement by using Arcloss on MNIST dataset.

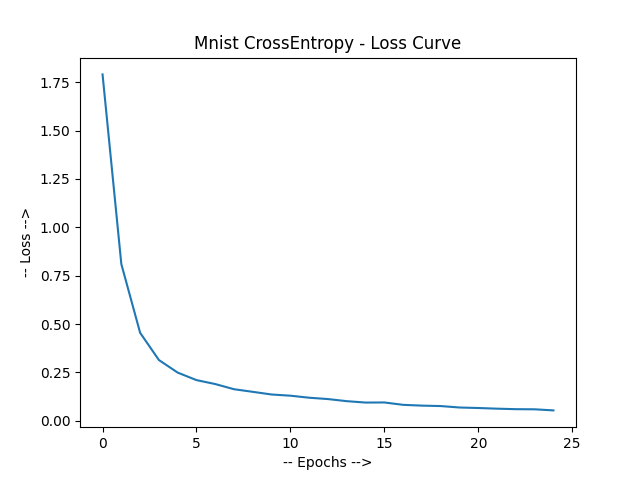
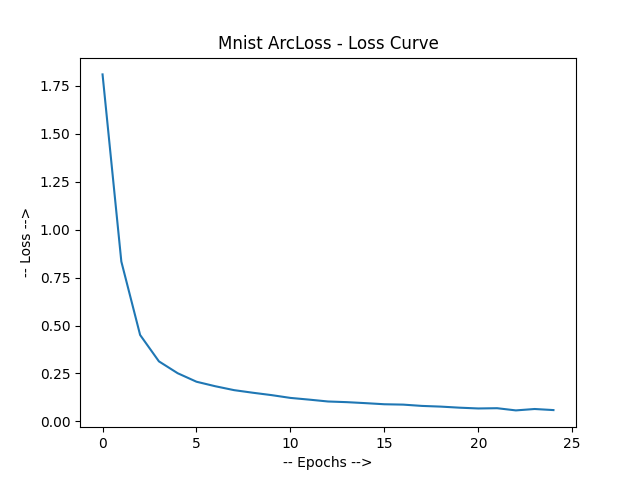
Using CrossEntropyLoss  


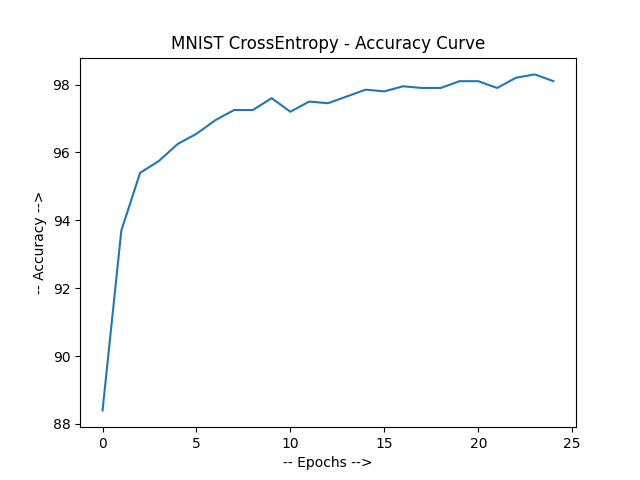
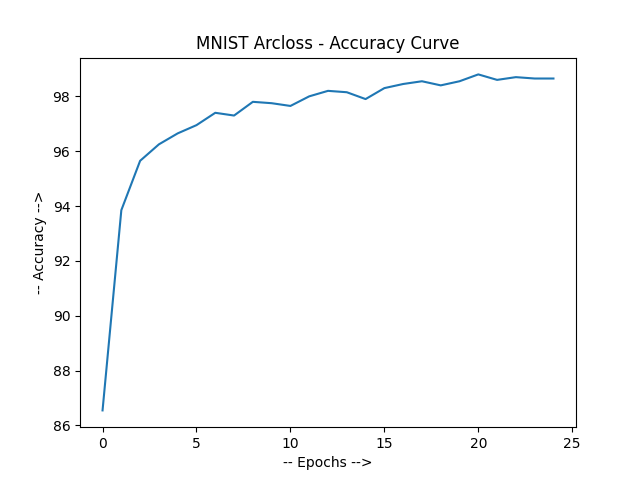
Using Arcloss  


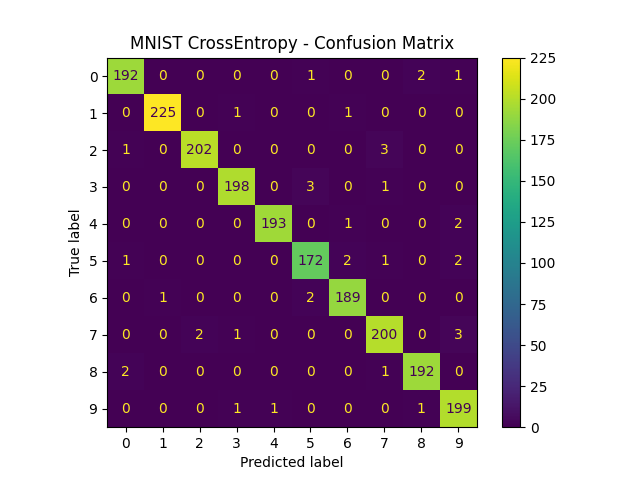
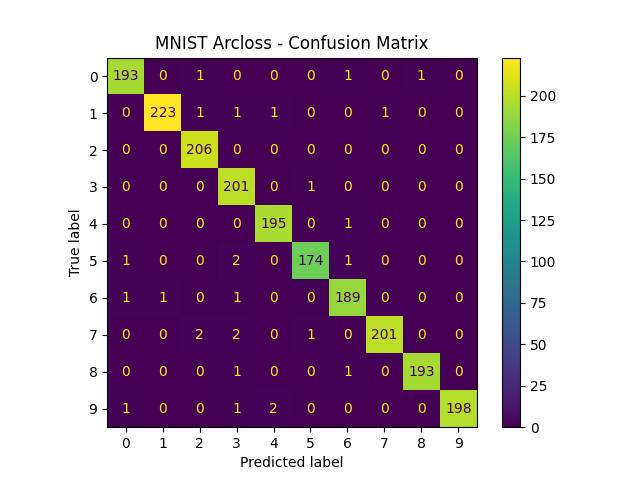
**Output Screenshots:**  
  
Code Execution Screenshots

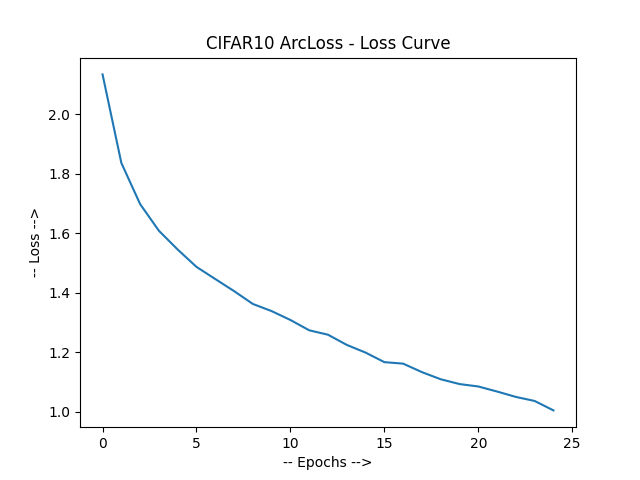


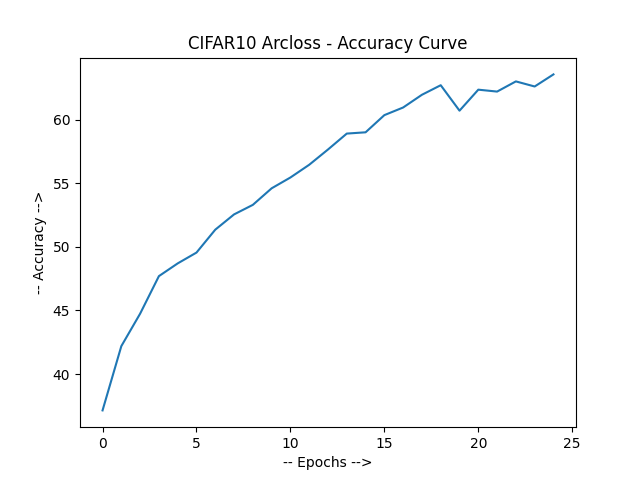
**MNIST Graph:**

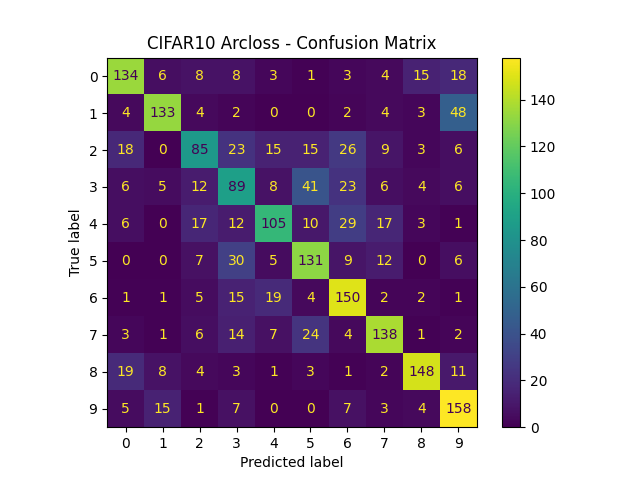






**CIFAR-10 Graphs**  






**References:**

ArcLoss Function

https://github.com/deepinsight/insightface/blob/master/recognition/arcface\_torch/losses.py

Understanding Code in Paper:

https://github.com/deepinsight/insightface/tree/master

CIFAR 10 Classification:

https://github.com/NvsYashwanth/CIFAR-10-Image-Classification

Understanding CIFAR-10 dataset:

https://www.kaggle.com/c/cifar-10/

MNIST Classification:

https://ashleyycz.medium.com/mnist-digit-classification-in-pytorch-302476b34e4f

Understanding MNIST Dataset:

https://www.kaggle.com/datasets/hojjatk/mnist-dataset