Task1: Training masked auto encoder

Roll No: M22AI567

Name: Krishna Kumari Ravuri

Steps Followed:

Data Preprocessing

- Downloaded PASCAL VOC2007 DATASET and Imported all the required libraries
- Read all the images of the JPEGIMAGES folders, resized and add Gaussian noise and stored it the numpy array
- Split the data into 2 separate set one with train-val-test (80-10-10) and other with train-val-test (70-10-20)

Auto Encoder with diff bottleneck dimensions

- Implemented auto encoder model with bottleneck dimension 256 and train-val-test split(80-10-10)
- Implemented auto encoder model with bottleneck dimension 256 and train-val-test split(70-10-20)
- Implemented auto encoder model with bottleneck dimension 128 and train-val-test split(80-10-10)
- Implemented auto encoder model with bottleneck dimension 64 and train-val-test split(80-10-10)
- Implemented auto encoder model with bottleneck dimension 32 and train-val-test split(80-10-10)
- Implemented auto encoder model with bottleneck dimension 16 and train-val-test split(80-10-10)

Auto Encoder with diff masking percentage

- Implemented auto encoder model with bottleneck dimension 256 and train-val-test split(80-10-10) and masking 40%
- Implemented auto encoder model with bottleneck dimension 256 and train-val-test split(80-10-10) and masking 60%
- Implemented auto encoder model with bottleneck dimension 256 and train-val-test split(80-10-10) and masking 80%
- Implemented auto encoder model with bottleneck dimension 256 and train-val-test split(80-10-10) and masking 20%

All models report:

Dimensions			No.of Epochs	Learning Rate	Optimizer	Loss Function	Total Loss	MSE Score	MAE Score
Bottleneck dimension	Train-val-test split	Masking							
256	80-10-10		10	0.001	Adam	MSE	0.005210531875491142	0.003840572194583255	0.05069612145480171
256	70-10-20		10	0.001	Adam	MSE	0.005374384578317404	0.003907464314758137	0.051380640717423515
128	80-10-10		10	0.001	Adam	MSE	0.005215854849666357	0.0038399761380639936	0.05048543205635392
64	80-10-10		10	0.001	Adam	MSE	0.005234170705080032	0.00392601884068947	0.05122225314337198
32	80-10-10		10	0.001	Adam	MSE	0.005172363948076963	0.0038479782569872784	0.0506547736619417
16	80-10-10		10	0.001	Adam	MSE	0.005332760978490114	0.003966723053978846	0.05139177435319714
256	80-10-10	20%	10	0.001	Adam	MSE	0.015371019951999187	0.03218873687124333	0.08852559972640443
256	80-10-10	40%	10	0.001	Adam	MSE	0.0333862341940403	0.0459326162879082	0.14270168569654168
256	80-10-10	60%	10	0.001	Adam	MSE	0.04507621377706528	0.10778944855542302	0.2594821607389239
256	80-10-10	80%	10	0.001	Adam	MSE	0.01551081147044897	0.03822550883536183	0.10393287201234934

justify your choice of best:

Why 80-10-10 split is better?

Implemented auto encoder model with bottleneck dimension as 256 for both the splits

Model autoencoder_256_80_10_10
Total Loss :0.005210531875491142
MSE Error: 0.003840572194583255
MAE Error :0.05069612145480171

Model autoencoder_256_70_10_20
Total Loss: 0.005374384578317404
MSE Error: 0.003907464314758137
MAE Error: 0.051380640717423515

Loss with 80-10-10 split less compared to 70-10-20 split.

Why 256 bottleneck dimension is better?

Implemented auto encoder model with different bottleneck dimension 256, 128,64,32, and 16

Model autoencoder_256_80_10_10
Total Loss 0.005210531875491142
MSE Error 0.003840572194583255
MAE Error 0.05069612145480171
Model autoencoder_128_80_10_10
Total Loss 0.005215854849666357
MSE Error 0.0038399761380639936
MAE Error 0.05048543205635392
Model autoencoder_64_80_10_10
Total Loss 0.005234170705080032
MSE Error 0.00392601884068947
MAE Error 0.05122225314337198
Model autoencoder_32_80_10_10
Total Loss 0.005172363948076963
MSE Error 0.0038479782569872784
MAE Error 0.0506547736619417
Model autoencoder_16_80_10_10
Total Loss 0.005332760978490114
MSE Error 0.003966723053978846
MAE Error 0.05139177435319714
Loss with 256 bottleneck dimension is less compared to other bottleneck dimensions

Why 20% masking is better?

Implemented auto encoder model with different masking percentages 20%,40%,60% and 80%

Model autoencoder_20per_Masking
Total Loss 0.015371019951999187
MSE Error 0.03218873687124333
MAE Error 0.08852559972640443
Model autoencoder_40per_Masking
Total Loss 0.0333862341940403
MSE Error 0.0459326162879082
MAE Error 0.14270168569654168
Model autoencoder_60per_Masking
Total Loss 0.04507621377706528
MSE Error 0.10778944855542302
MAE Error 0.2594821607389239
Model autoencoder_80per_Masking
Total Loss 0.05551081147044897
MSE Error 0.03822550883536183
MAE Error 0.10393287201234934
Loss with 20% bottleneck dimension is less compared to other masking percentages

What is the final model Dimensions?

Based on the all models loss for different splits , bottleneck dimension and masking percentage selected final model dimensions as follows:
Split: 80-10-10
Bottleneck Dimension: 256 and Masking percentage: 20%

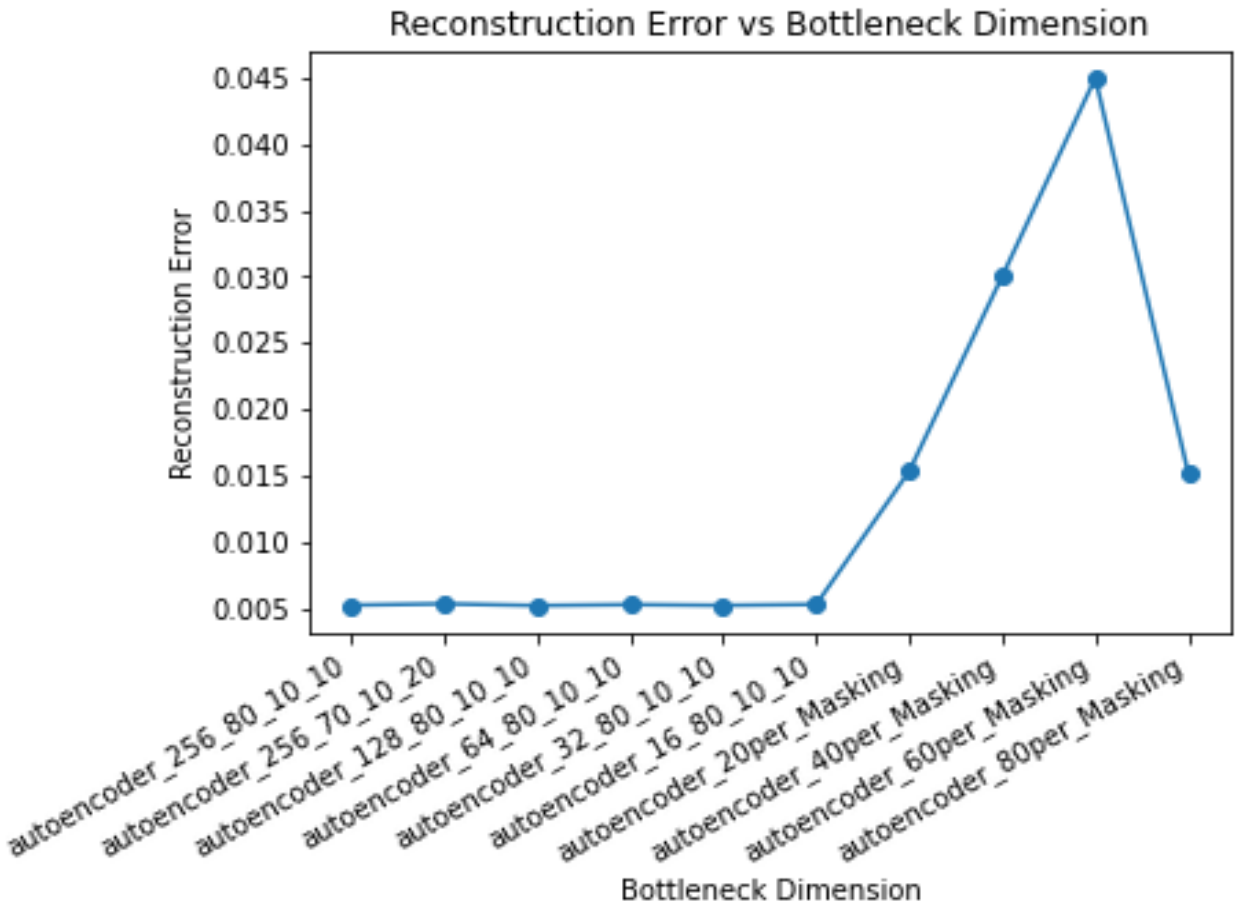
Saved the final model to use it for the feature extraction in the task2.

Observations about results, any additional analysis over results

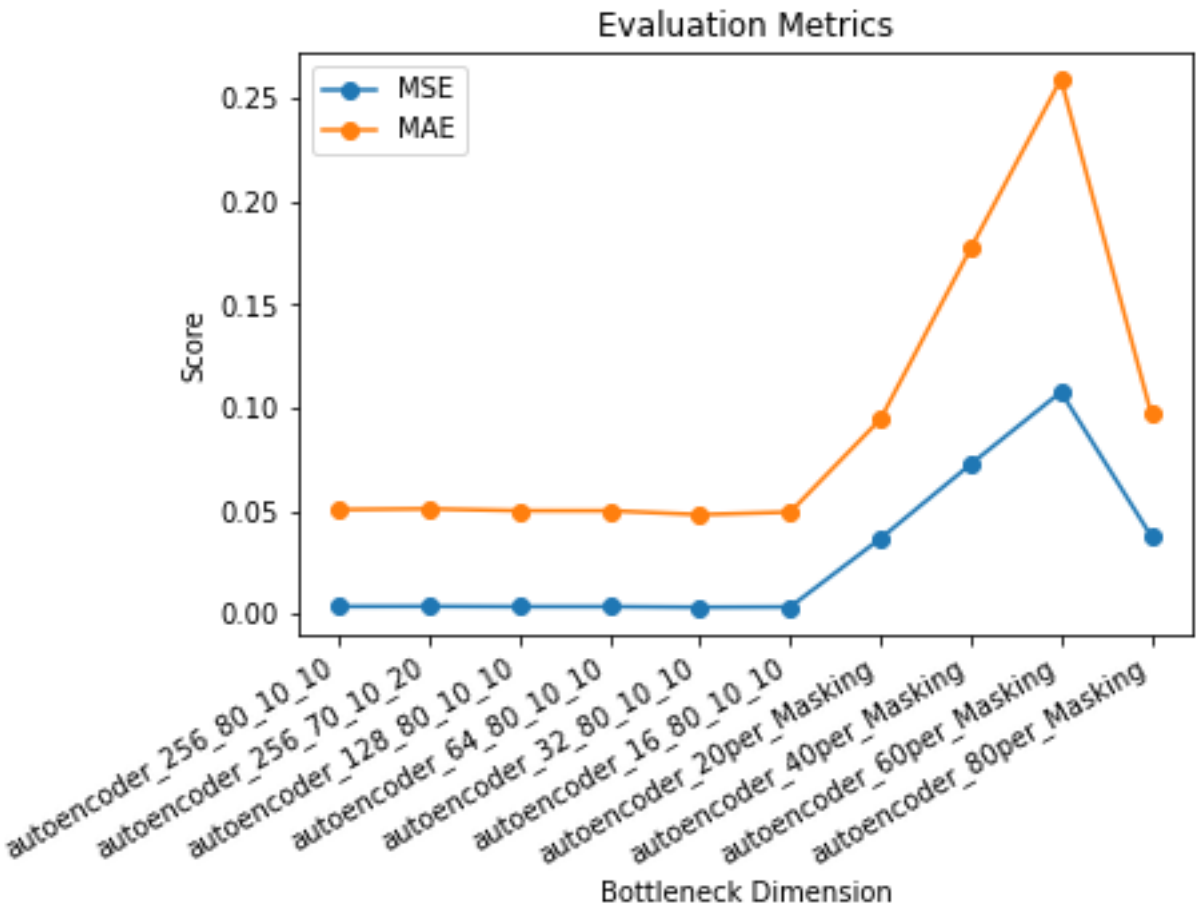
Why loss is high and denoised image is not clear?

Performance of the hardware: Due to less RAM capacity in my personal system.
Memory limitations: My personal computer has limited available memory.
Due to above reasons:
I have resizing an image to a smaller size(64,64), some details and fine-grained information may be lost. This loss of information can make it more challenging for the autoencoder to accurately denoise the image since it has fewer details to work with and less number of layers as well.
Along with this I have tried with rectangular masking also which is not giving better results.

Reconstruction error plot for every auto encoder model:



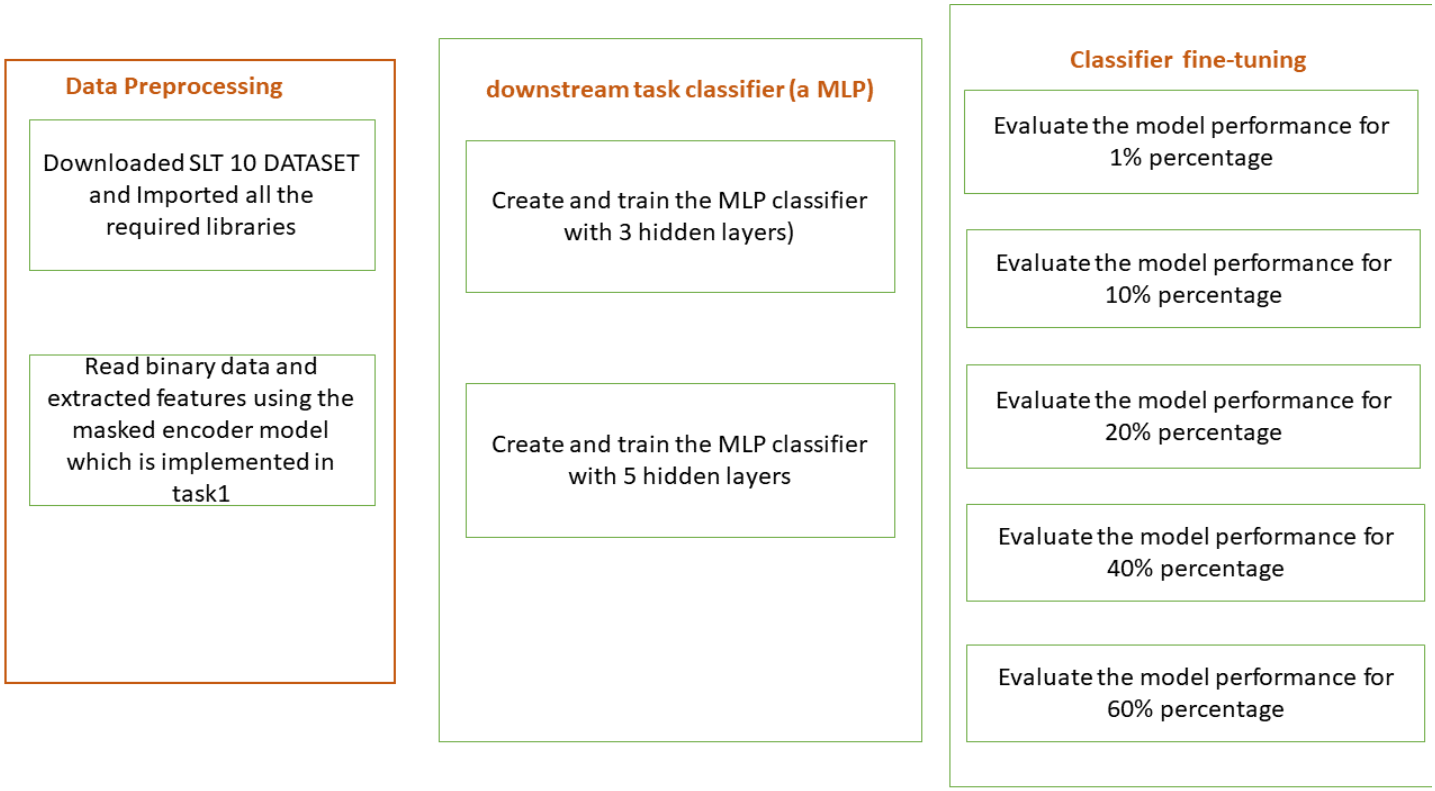
MSE (mean square error), MAE (mean absolute error)



Visualize and compare the original images, masked images, and reconstructed images

Task2: Fine-tuning a pre-trained auto encoder on STL-10 dataset

Steps Followed:

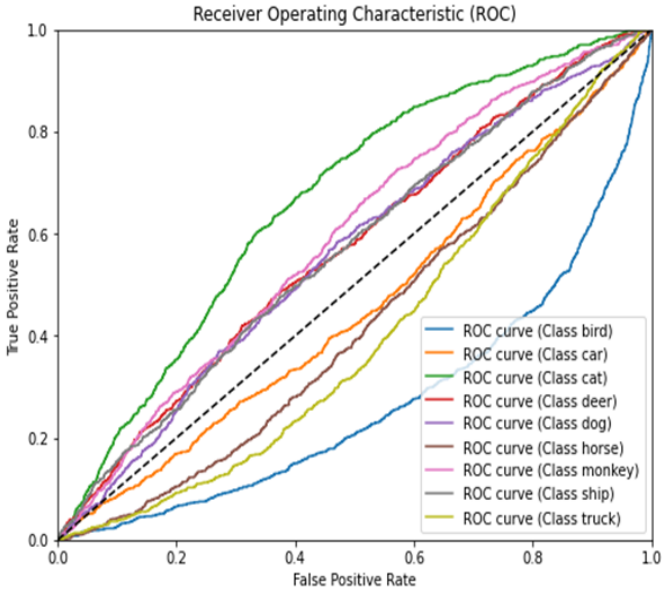


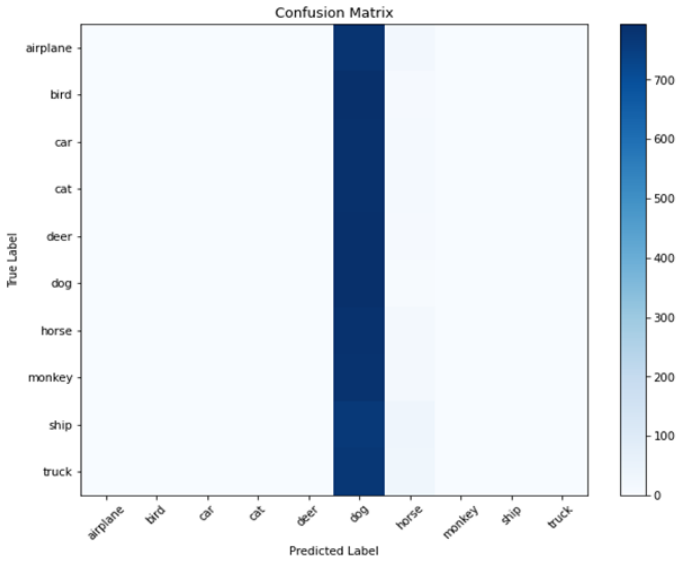
Model Performance details:

downstream task classifier (a MLP) with (100% training samples)	Accuracy	
	3 Hidden layers	5 Hidden layers
hidden_layer_sizes = (100, 100, 100)/(100, 100, 100,100,100)	0.373	0.373375
hidden_layer_sizes =(200, 200, 200)/(200, 200, 200,200,200)	0.389875	0.399875
hidden_layer_sizes = (1000, 1000, 1000)/(100, 100, 100,100,100)	0.41475	0.3885
fine-tune the classifier with % of train samples		
1%	0.100375	0.10025
10%	0.265375	0.28025
20%	0.324625	0.310125
40%	0.354	0.334
60%	0.378375	0.36175

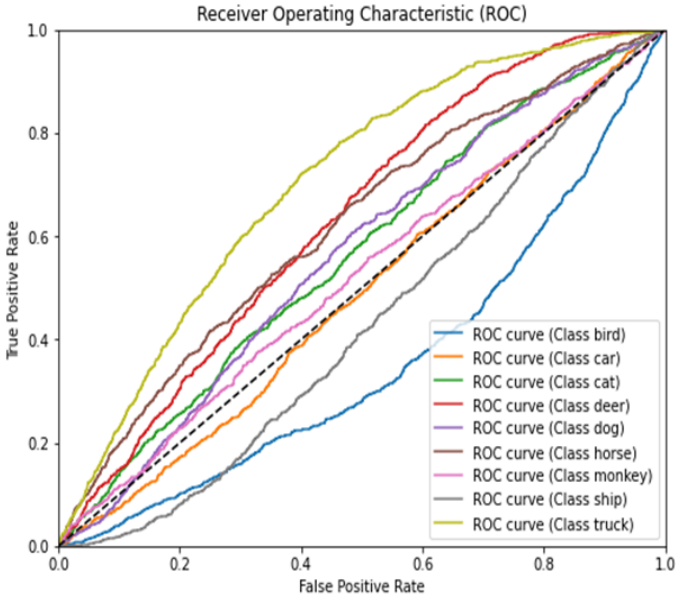
confusion matrix and AUC-ROC curve:

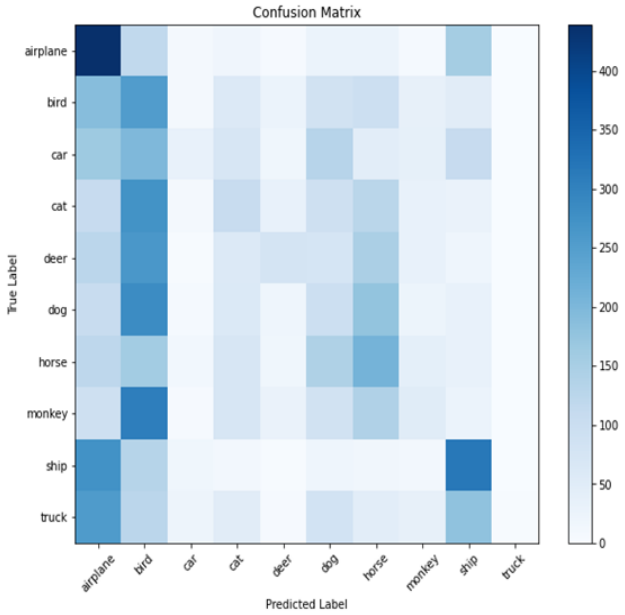
ROC curve and confusion matrix for 1% training sample with 3 hidden layers



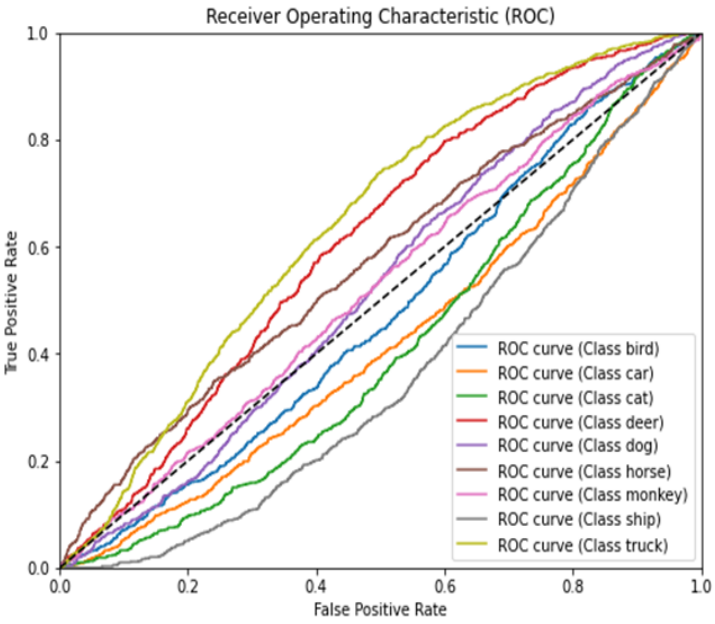


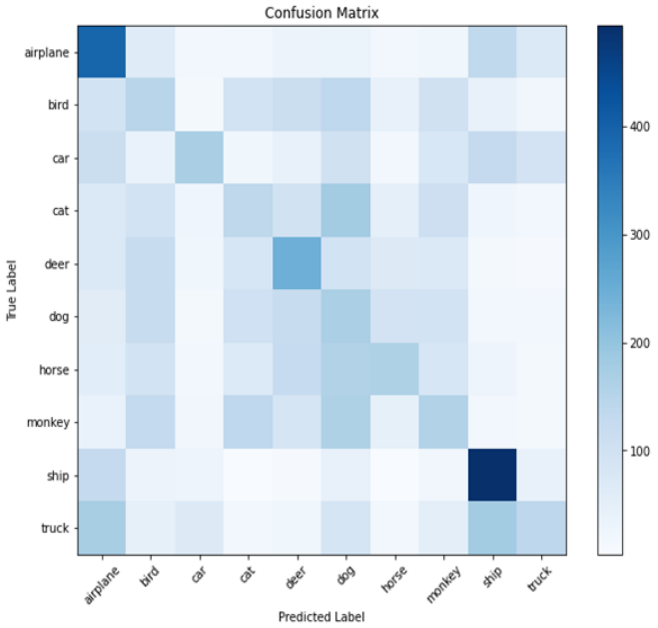
ROC curve and confusion matrix for 1% training sample with 5 hidden layers



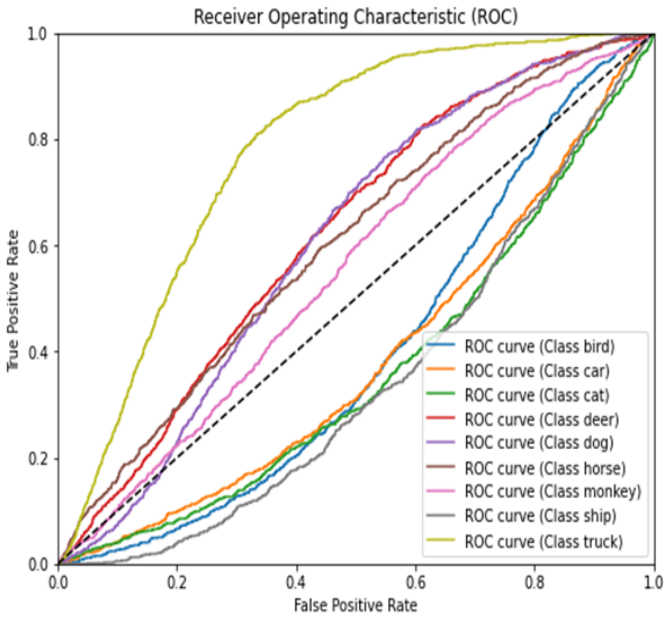


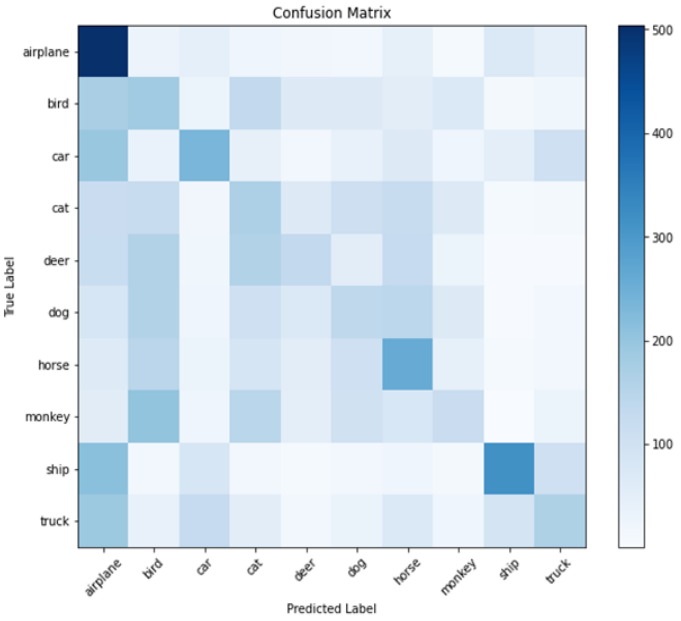
ROC curve and confusion matrix for 10% training sample with 3 hidden layers



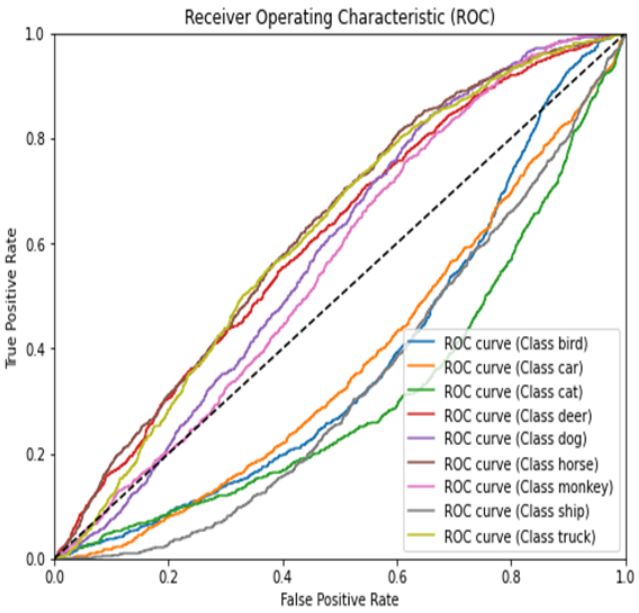


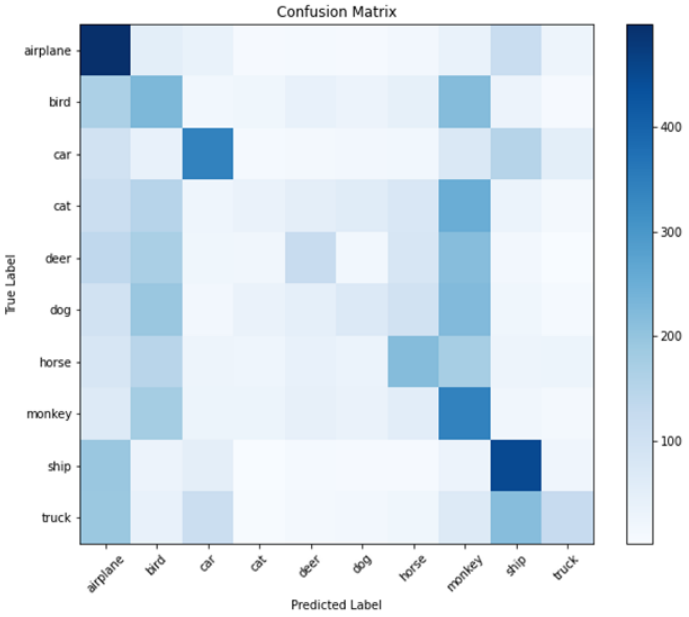
ROC curve and confusion matrix for 10% training sample with 5 hidden layers



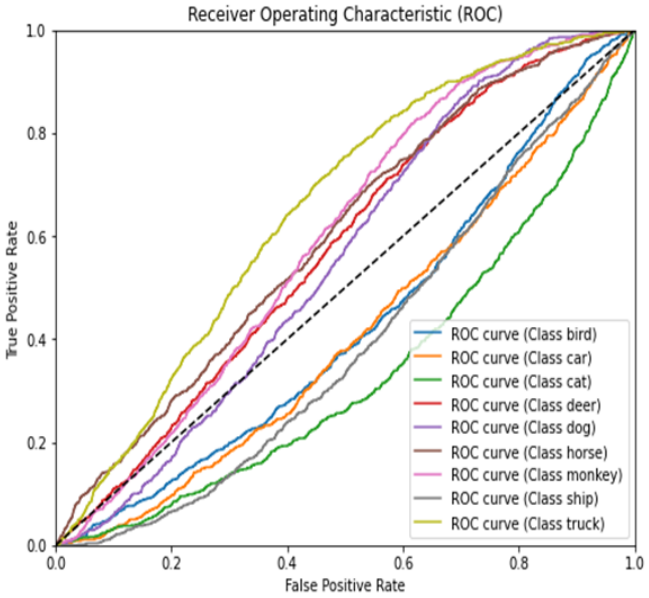


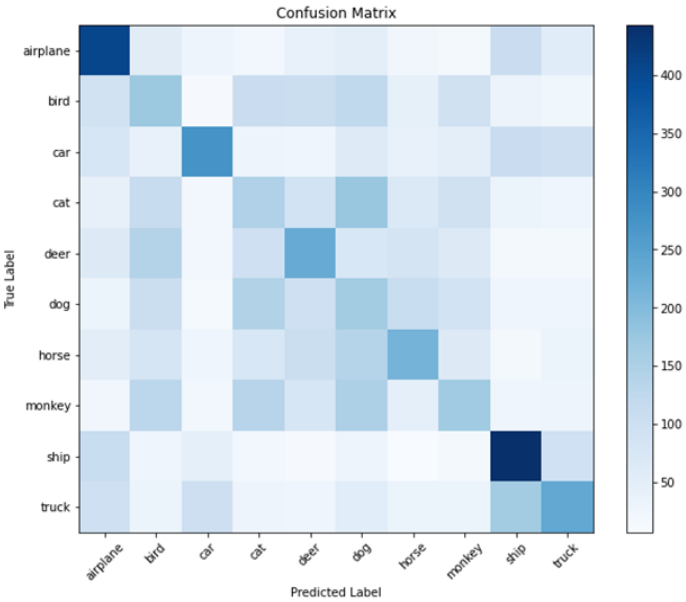
ROC curve and confusion matrix for 20% training sample with 3 hidden layers



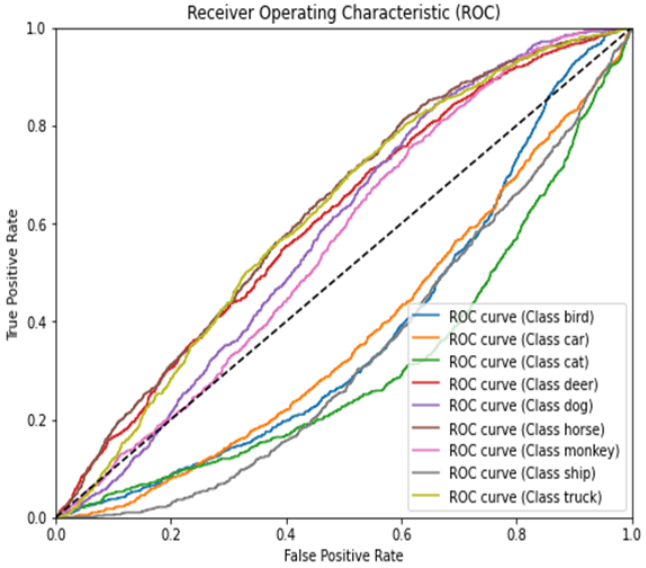


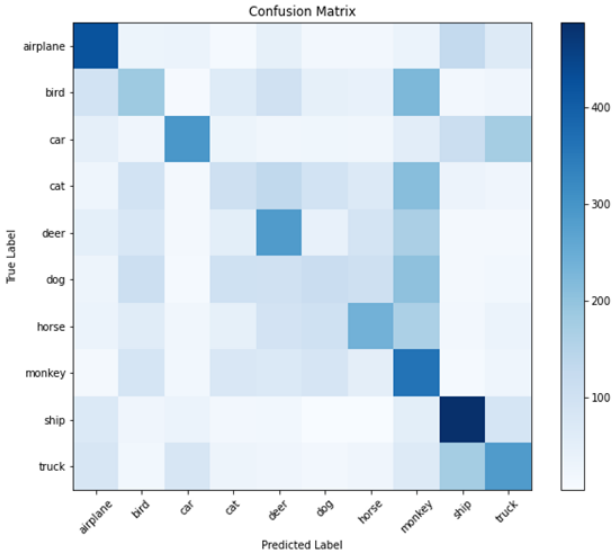
ROC curve and confusion matrix for 20% training sample with 5 hidden layers



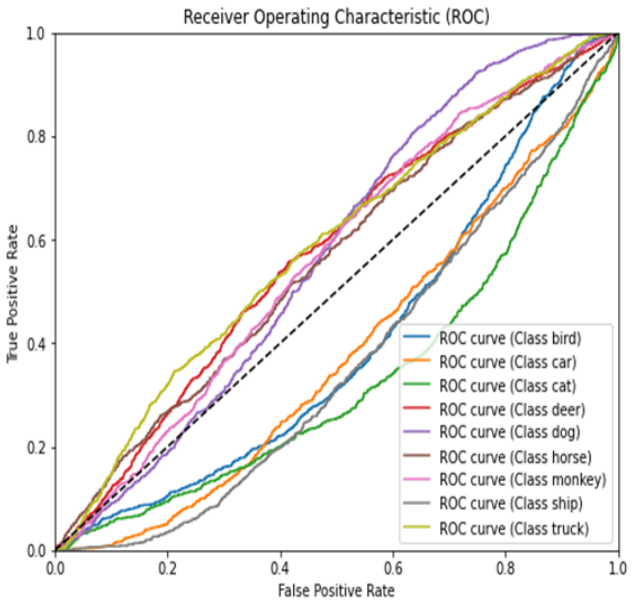


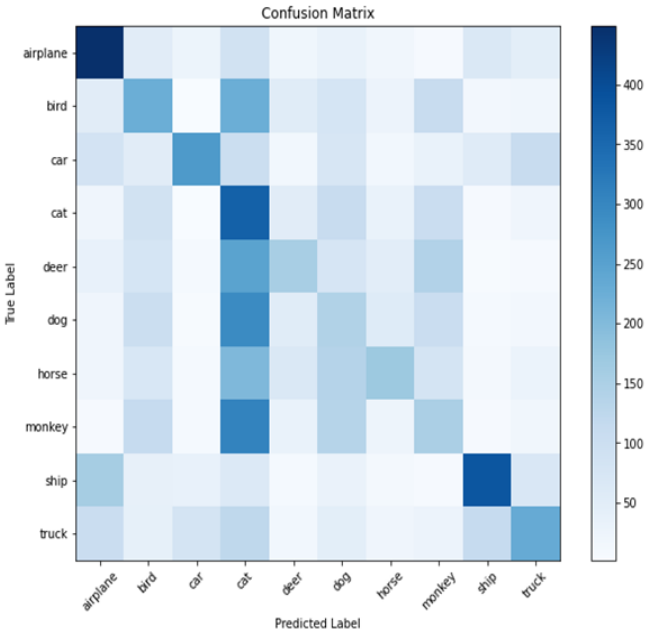
ROC curve and confusion matrix for 40% training sample with 3 hidden layers



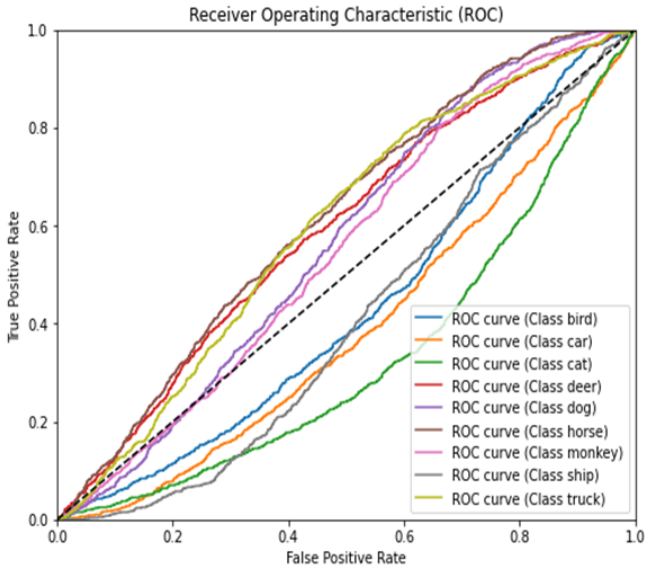


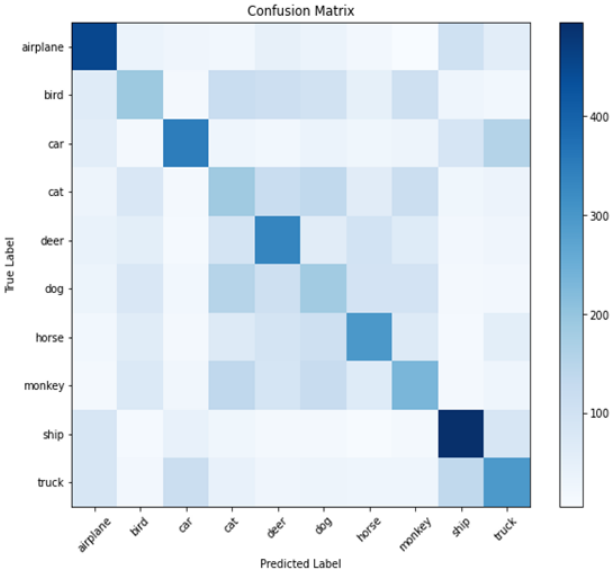
ROC curve and confusion matrix for 40% training sample with 5 hidden layers





ROC curve and confusion matrix for 60% training sample with 3 hidden layers





ROC curve and confusion matrix for 60% training sample with 5 hidden layers