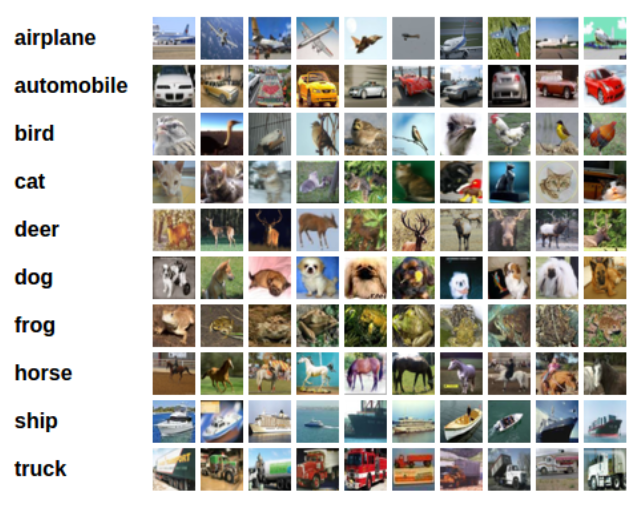
**Report for Assignment 6**

**-Debanshu Biswas (M20MA053)**

**Neural Network using PyTorch for Classification Task**

We used pytorch to implement neural network for classification task. We will use CIFAR10 dataset. It has the classes: ‘airplane’, ‘automobile’, ‘bird’, ‘cat’, ‘deer’, ‘dog’, ‘frog’, ‘horse’, ‘ship’, ‘truck’. The images in CIFAR-10 are of size 3x32x32.



The process in program is as follows,

* We loaded and normalized the CIFAR10 training and test data set. For this we used torch vision here.
* Then we defined Convoluation Neural Network.
* Then defined loss function.
* Train the the network on the training data
* Test the network on the test data

The output of torchvision datasets are PILImages of range [0,1]. And the transformed then to Tensors of normalized range [-1,1].

And we get the training as,



We get overall accuracy on 10000 test 53%

And for individual,

Accuracy for class plane is: 35.1 %

Accuracy for class car is: 69.5 %

Accuracy for class bird is: 44.5 %

Accuracy for class cat is: 21.3 %

Accuracy for class deer is: 56.2 %

Accuracy for class dog is: 49.6 %

Accuracy for class frog is: 47.8 %

Accuracy for class horse is: 61.1 %

Accuracy for class ship is: 91.6 %

Accuracy for class truck is: 27.7 %

**Neural Network using PyTorch for Regression Task**

I used dataset “heart\_failure\_clinical\_records\_dataset.csv” which I imported from Kaggle.com. Here we are trying to find platelets from creatinine\_phosphokinase data. Then we build a neutal network like,

class MLP(nn.Module):

    #Multilayer Perceptron for regression.

  def \_\_init\_\_(self):

    super().\_\_init\_\_()

    self.layers = nn.Sequential(

      nn.Linear(13, 64),

      nn.ReLU(),

      nn.Linear(64, 32),

      nn.ReLU(),

      nn.Linear(32, 1)

    )

  def forward(self, x):

      #Forward pass

    return self.layers(x)

We used ReLU function for pointwise non linearity. Then we used our designed neural network on our dataset. Then we get result for different epochs,

Starting epoch 1

Loss after mini-batch 1: 0.043

Loss after mini-batch 11: 0.426

Loss after mini-batch 21: 0.442

Loss after mini-batch 31: 0.461

Loss after mini-batch 41: 0.452

Loss after mini-batch 51: 0.472

Starting epoch 2

Loss after mini-batch 1: 0.047

Loss after mini-batch 11: 0.432

Loss after mini-batch 21: 0.450

Loss after mini-batch 31: 0.434

Loss after mini-batch 41: 0.485

Loss after mini-batch 51: 0.434

Starting epoch 3

Loss after mini-batch 1: 0.049

Loss after mini-batch 11: 0.458

Loss after mini-batch 21: 0.435

Loss after mini-batch 31: 0.423

Loss after mini-batch 41: 0.417

Loss after mini-batch 51: 0.480

Starting epoch 4

Loss after mini-batch 1: 0.051

Loss after mini-batch 11: 0.436

Loss after mini-batch 21: 0.417

Loss after mini-batch 31: 0.452

Loss after mini-batch 41: 0.447

Loss after mini-batch 51: 0.438

Starting epoch 5

Loss after mini-batch 1: 0.043

Loss after mini-batch 11: 0.442

Loss after mini-batch 21: 0.427

Loss after mini-batch 31: 0.428

Loss after mini-batch 41: 0.424

Loss after mini-batch 51: 0.451