**Neural Network and Deep Learning**

**Coursework Report**

1. READING DATASET AND CREATING DATALOADERS: -

Importing the Libraries: -

Firstly, we will import all the useful libraries like torch (it provides Tensor computation with strong GPU acceleration and Deep neural networks built-in system), torchvision (package consists of popular datasets, model architectures, and common image transformations), NumPy (mathematical functions, random number generators, linear algebra routines, Fourier transforms, etc), matplotlib (creating static, animated, and interactive visualizations) etc.

Declaring functions: -

We will first download the dataset in the framework---

For this we will use Pytorch torch vision to download the dataset and then we will def the function get\_data\_loader() to load data set then we will create a function load\_data\_mnist to load dataset in 4 batches. Another function we will create set\_axes for visualization.

Loading DATASET: -

Using CUDA for graphical processing an approach called general purpose computing on GPUs, we will load data set in train and test each of batch size 256 using function load\_data\_minst

1. CREATING MODEL

Stemming Part –

Takes as an input an image of particular size and divides it into non-overlapping patches. Each patch has its own dimension. Each patch is then vectorized, and transformed to a feature vector. All the features are stored in matrix.

Backbone Part-

Here, we are using two backbones with sequential model and activation function as RELU.

The model architecture is as follows:

Text

Description automatically generated

1. CREATE LOSS AND OPTIMIZER

Now we will define loss function and optimizer while experimenting with learning rate so in order to get best possible accuracy and minimum time to reach to global minima.

We found out that with learning rate of 0.0026, the accuracy of model with number of epoch being 65 is 88.49 which seems to be a good standard.

Here we will use CrossEntropyLoss() which measures the performance of a classification model whose output is a probability between 0 and 1.

We will define accuracy with 2-dimension y\_hat(matrix) and second dimension is the prediction of each class

We will take the sum of number of correct predictions.

1. Training model.

A picture containing chart

Description automatically generated

We will train the model with dataset and evaluate accuracy of model. We will find out that training and testing accuracy will never be same as occurrence of loss is organic in nature. Our aim is to minimize the loss, thus after creating suitable model and after certain number of epochs we get to the point where training accuracy is as good as **88.49percent**.

1. Final model accuracy.

Final model accuracy after epoch 65 is

Train accuracy = 0.9801

Test accuracy = 0.8849

Train loss : 0.0558

Graphical user interface, chart, application

Description automatically generated with medium confidence

