Note: For this assignment, you are not allowed to search online for any kind ofimplementation.

Objectives: In this assignment, you will write the code for training the Neural Networks for classification. The goals of this assignment are as follows.

- Understand how neural networks can be used for the classification of images
- Understand how a model can be created, trained, validated, and saved in PyTorch
- Understand how to read/write and process images in python.

Classify Images using Fully Connected Neural Networks

In this part, you will create a complete neural network architecture in PyTorch consisting of multiple layers. You are required to report results obtained from different experiments on the MNIST dataset. ***Submit the similar architecture of network that have best results in your experiments.***

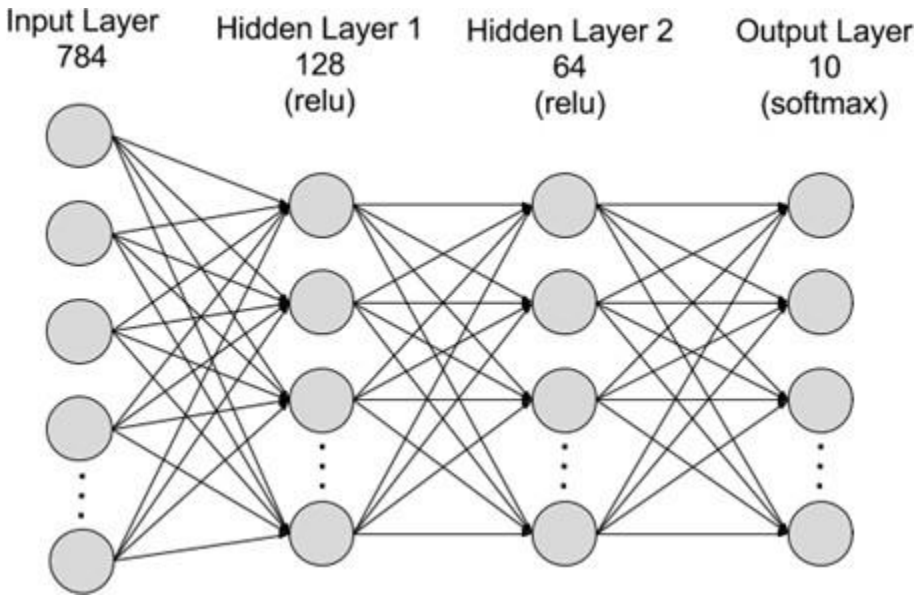

Fig. Representing a simple neural network for MNIST digits classification [source]

MNIST Dataset:

The dataset is attached with the assignment. This dataset contains *60000* training and *10000* test samples. Each sample is a grayscale image of size 28x28. There are 10 classes in which each sample is an image of one of the digits (0 to 9). Please note that in the MNIST dataset there are 10 categories. If we randomly guess one category, there is a 1/10 probability that it would be correct. Therefore, you cannot (theoretically) make a classifier that performs worse than that. If you get less than 10% accuracy in any of your experiments, you can safely assume that you are doing something fundamentally wrong. If your final results are less than 20% in terms of accuracy, your solution(s) will not be graded.

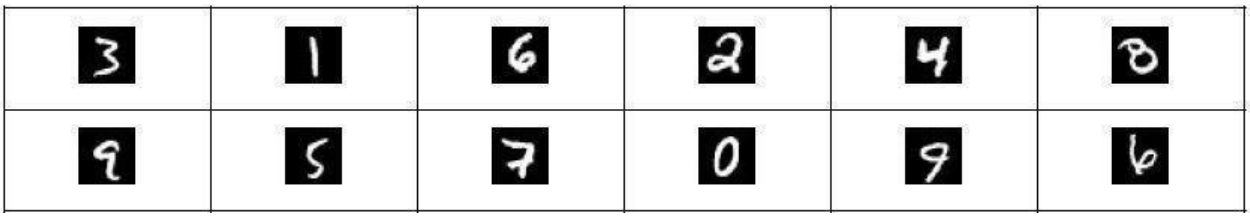

Figure: Sample Images from the Dataset

Steps:

1. Data loading and normalization

Load MNIST images using `loadDataset()` function. Function should accept the path of the dataset, size of training, validation and testing data, batch size and shuffle as input. Function should load training and testing data and then select data samples using torch.utils.data.Subset() according to given training, validation and test size. Function should return loaded training, validation and testing data. Function should also normalize the data with zero mean and 0.5 variance. Figure out the images optimal resizing parameters and load all the training and testing data with that size, zero mean per batch and 0.5 variance.

2. Initialize Network

You have to create a function to initialize the network. Parameters should be entered by the user. If your system has a GPU you can turn it on to improve performance. You can use pytorch built-in functions to create and initialize the network.

```
net = init_network(no_of_layers, input_dim, neurons_per_layer, dropout)
```

neuron_per_layer should be a list having elements describing the number of neurons in each hidden layer. Last element of the list should be the dimension of output (In our case 10). For example if you pass following parameters to this function:

```
net = init_network(2, 784, [100, 50, 10],0.2)
```

It should return you the network architecture with parameters initialized:

```
Size of net(1).w = 784x100

Size of net(1).b = 100

Size of net(2).w = 100x50

Size of net(2).b = 50

Size of net(3).w = 50x10

Size of net(3).b = 10
```

For more information about dropout:

https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

3. Training

Now create a function to train the initialized network. You also have to keep track of loss and accuracy on training and validation data for each epoch to display loss and accuracy curves.

```
net = train(net, train_set_x, train_set_y, valid_set_x, valid_set_y, learning_rate, training_epochs, loss_func, optimizer)
```

- This function returns a trained Neural Network `net, loss_array, and accuracy_array`

4. **Save Network**

   Write a function that saves your network so that it can be used later without training.

5. **Load Network**

   Write a function that loads your saved network so that it can be used without training. Function should return loaded model.

6. **Testing Step**

   Now create a function that uses a trained network to make predictions on a test set.

   ```
   pred = test(net, model,  test_set_x,  test_set_y)
   Function should return predictions of model.
   ```

7. **Visualize Results**

   Write a function that plot loss and accuracy curves and sample images and prediction made by model on them. Function should also plot confusion matrix, f1_score, and accuracy on test data. Review `sklearn.metrics` for getting different metrics of predictions.

8. **Main Function**

   Now create a main function that calls all above functions in required order. Main function should accept "Path of dataset", "size of training data", "size of validation data", "size of testing data", "number of hidden layers", "list having number of neurons in each layer", "Loss function", "optimizer", "batch size", "learning rate", "Is GPU enable(By default false if you do not have GPU)", "drop out", "Is training (default is False)", "Visualize Results (Default is False)" and "training epochs" as input. If "Is training" is true, the function should start training from scratch, otherwise the function should load the saved model and make predictions using it. While performing experiments you can use Google Colab or Jupyter Notebook. Your code must print your name, roll no, your all best/default parameters, training, validation and testing losses, accuracies, f1 scores and confusion matrix on test data and accuracy and loss curves of training and validation data.

Report

For this assignment, and all other assignments and projects, you must write a report. In the report you will describe the critical decisions you made, important things you learned, or any decisions you made to write your algorithm a particular way. You are required to report the accuracy you achieved. For each experiment, you are required to provide analysis of various hyperparameters.

1. Plot loss and accuracy curves for both training and validation data with and without image normalization. Report the difference in their accuracy and loss curves. Also report accuracy and loss on test data.
2. Report the accuracy by changing number of neurons in the hidden layers, or number of hidden layers or changing loss functions, batch size, learning rate and ratio of training and testing data etc.
3. For your final experiment display actual images and predicted labels for both cases if prediction is correct or wrong. If prediction is wrong show on what basis the model predicted it wrong and also show the worst prediction from the model. For example, if the model has predicted 3 as 8, prediction is wrong but both digits are similar in shape. But if model has predicted 0 as 1 then this prediction can be worse as both digits do not have similar patterns in it. Also display confusion matrix on testing data and report Precision, Recall and F1 score align with loss and accuracy scores and curves.

4. Report what you learned from this assignment, your analysis and if you find something innovative or interesting in the conclusion section.
5. Discuss time effect on time taken by network in different experiments or when using cpu or gpu.

Note:

1. **Please refer to the classification tutorial with pytorch.**
2. **You will need a softmax activation function on the output layer and for hidden layers you can play with different activation functions available in pytorch.**
3. **If you find images having 3 channels (RGB) you can convert them to a single channel (Grayscale).**
4. **You can use sklearn, matplotlib and pytorch for this assignment. Using sklearn you can calculate f1_score, accuracy, and confusion matrix. Matplotlib can help you in plotting loss and accuracy curves.**
5. **Code must fulfill requirements and should be in modular form, however syntax of different functions given in this assignment are just to clear the direction; you can define functions or classes in your own way.**