EE5179:Deep Learning for Imaging
Programming Assignment 1: MNIST Classification using MLP (Due by Sep 11, 2025)

1 Instructions

- Program in python for this assignment.
- Post any doubts you have on moodle. This will be helpful to your peers as well.
- The assignment primarily involves coding an MLP from scratch. You will find several sources online doing this. Refrain from directly copying code from any of these sources. Rather, if you get help from any of these sources, cite them as part of a references section.
- You are allowed to use online and offline resources for assistance, but using AI models like ChatGPT, Gemini, or similar tools to write code is strictly prohibited. If you are caught doing so, you will receive a zero on the assignment.
- Submit the codes(the actual notebook and a PDF version) and report in a zip file titled PA1_RollNumber.zip in the submission link provided on moodle.
- This assignment is essentially an extension to the previous tutorial, so all the code/ideas implemented in the tutorial will come very handy.

2 Preliminaries

- It is recommended to use Google Colab (as in the tutorials) or Jupyter/iPython notebooks for the assignment. The notebook format is very convenient to work (and evaluate!) with and additionally, Colab provides GPU access as well.
- The dataset that you'll be working with is the MNIST digits dataset. It comprises of handwritten digits which are pre-processed to ensure that the digits are centered and size normalized. The aim is to code a complete

handwritten digit recognizer, train it on the train set comprising 60,000 images and test performance on the test set comprising 10,000 images.

- The dataset can be downloaded from here or you can make use of the inbuilt dataset from Pytorch.

Note: Check if the labels are in one-hot format, or appropriately convert them to one-hot format before training and testing the network.

3 Backpropagation from Scratch

The first task is to code the below network from scratch.

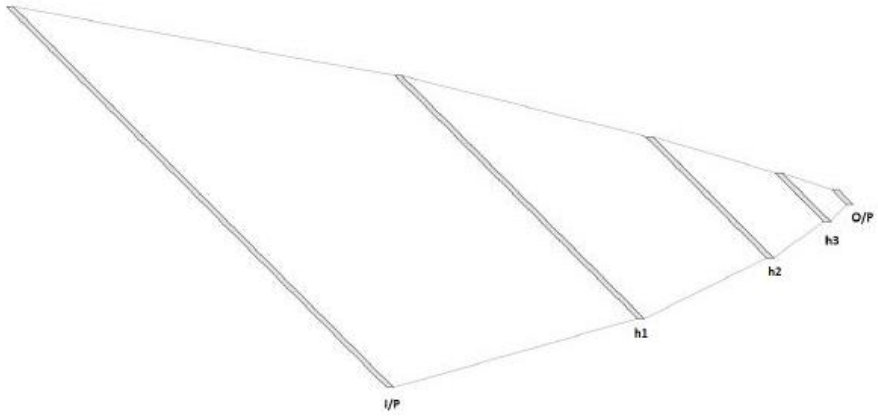

Figure 1: Handwritten Digit Recognizing MLP

As can be seen from the figure, the network comprises of an input layer, three hidden layers, followed by an output layer. Let the baseline model have the following architecture: I/P $\rightarrow$ $h_1(500)$ $\rightarrow$ $h_2(250)$ $\rightarrow$ $h_3(100)$ $\rightarrow$ O/P. Where the number of neurons are indicated beside the corresponding hidden layer.

The image is greyscale and has a size of 28×28. Ensure to flatten the image before training and testing.

Use sigmoid activation for all the hidden layers. Use linear activation for the output layer, but use a softmax to predict the probabilities of the input being a particular digit. The output layer comprises 10 neurons, one for each digit. Thus, $\hat{y}_i$, the output is a 10 dimensional vector which means that the true outputs should also be a 10 dimensional one-hot vector (i.e 1 in the index of the true class label and 0 everywhere else).

- You are free to try out other models besides the baseline if you have time, but prioritize on training the baseline model first.

- Use a cross entropy loss between the ground truth label y_i (a one-hot vector) and the predicted output $\hat{y}_i$.
- Perform backpropagation and forward pass for the neural network without using any library functions (you can use the built-in function for matrix multiplication, addition, and exponentials).
- Use the gradient descent algorithm to optimize the model parameters. Use a batch size of 64 and a learning rate η of 0.01. (i.e, update once after seeing 64 training samples). You are free to use packages such as dataloaders (from Pytorch) to organize the data in batches. Here is a good reference to get started with dataloaders.
- Train the model for 15 epochs. One epoch is when the network sees all of the training data once.
- Feel free to experiment with different training duration (number of epochs), learning rate, batch size and number of neurons per layer to arrive at a better model. Additional work such as these will be taken into account if the scores for the assignment are low in other places. Such experimentation is at the heart of Deep Learning, so it would be interesting to pursue this.
- In this assignment, initialize the biases with zeros but weights with a random sample from a uniform distribution as follows: $w_{ij} \sim \mathcal{U}[-M, M]$ where, $M = \sqrt{\frac{6}{N_i + N_o}}$, where N_i is the number of inputs to the weight tensor and N_o is the number of outputs in the weight tensor. This initialization is called the Glorot Initialization and is one that is popularly used. Read this to understand why this initialization scheme is popular.
- Submit the training and test loss plots depicting convergence against iterations. Plot the losses every 200 iterations (i.e 200 batch updates) or so for a clean plot.
- Report a confusion matrix or a classification report that describes the kind of errors made by the network.Looking at such statistics will help you better understand the network and focus on potential improvements.
- Lastly, report the net accuracy on the training and test sets by the network.

3.1 Activation

Repeat the above experiment with the following activation functions instead of the sigmoid activation.

- Tanh
- ReLU

Which activation performs the best?
Comment on the results observed.
Submit all the plots and accuracy metrics as in the baseline model.

4 Package

- Use Pytorch to mimic the best performing neural network in the previous part.
- All the hyperparameters (network architecture, batch size, learning rate, no of epochs) remain the same, however in this part you are free to use any packages/features (such as autograd) to simplify the different functions implemented.
- Instead of using Gradient descent, use the Adam optimizer to train this network. Here is a mathematical description of what ADAM does.
- Compare this with the network coded from scratch and comment on the results observed.
- Submit all the plots and accuracy metrics as in the baseline model.

4.1 Regularization

Add an L_2 regularization term to the loss function and train the modified network (with the best performing architecture). You are free to experiment with the regularization constant α (the extent of regularization).

Should you wish to, you can use available packages to implement the regularization.

Comment on the results observed.

Submit all the plots and accuracy metrics as in the baseline model.

5 Guidelines and Tips

- Ensure to vectorize the backpropagation and forward pass algorithm in a batchwise manner for the first part of the assignment. Failing to do this will not be penalized, however, this additional vectorization can save a lot on training time.
- Ensure to normalize the input image, before training and testing to help prevent potential overflow errors. The max value taken by a pixel is 255, so this information can be used to normalize the image.
- You can present any observations, plots, results and comments in the notebook itself instead of making a report if you wish.