CSE641 Deep Learning
Assignment-2 [60 marks]
Deadline: 23rd Feb 2023 EOD

General Instructions:

Use of any Deep Learning libraries like Pytorch or TF is not allowed for part-1 but you can use them in part-2 and part-3.

1. Each group member must do at least one of the following tasks. But all should know the working of all the tasks. (Recommended: Divide the sections among yourselves.)
2. For Plagiarism, institute policies will be followed strictly.
3. **Make sure to use Pickle or any other library to save all your trained models. There will not be enough time during the demo to retrain your model. This is a strict requirement.** You must upload your best models on Classroom to reproduce your results during the demo. If you cannot reproduce your results during the demo, no marks will be given. You are advised to prepare a well-documented code file.
4. You must submit Output.pdf, Code files (including both .py files and .ipynb files), and models dumped after training. Mention your sample outputs in the output.pdf. The Output.pdf should enlist all hyperparameters clearly for quick reference.
5. Submit code, models, and output files in ZIP format with the following name: A1_Member1_Member2_Member3.zip

PART I: OPTIMIZERS (25 marks)

Implement Optimizers on Multilayer Perceptron from scratch. DO NOT USE DL libraries in PART 1.

Extending the Part-3 of Assignment-1 on the same Fashion-MNIST dataset. Implement the following optimizers from scratch. You can reuse the same base code that you submitted for. **Use the best result configurations you obtained from Assignment-1 part 3, and replace SGD with the other optimizers discussed below:**

1. The following optimizers are to be implemented from scratch: **[4 * 5 = 20]**
    - **a.** Gradient Descent with Momentum
    - **b.** Nestrov's Accelerated Gradient
    - **c.** AdaGrad
    - **d.** RMSProp
    - **e.** Adam
2. For each of the optimizers you try above, generate the following plots: **[5 marks]**
    a. Loss plot - Training Loss and Validation Loss V/s Epochs.
    b. Accuracy plot - Training Accuracy, Validation Accuracy V/s Epochs)
    c. Analyze and Explain the plots obtained.

For PART 2 and 3, you can use DL libraries of your choice.

PART II: Convolution Neural Network (CNN) (20 marks)

Dataset: *MNIST*

If you are using PyTorch, then download the dataset from **torchvision.datasets** library, and if you use tensorflow, then use **tensorflow.keras.datasets** to load the data. **Don't download the data from any other source, as the train test split on different links may vary.**

Dataset description: The dataset contains 60,000 images in the training set and 10,000 in the test set. Each image is of 28 X 28 size digits between 0 to 9.

Task: Develop a neural network to identify the digits. Perform splitting of Train data in training, validation sets with an 80:20 ratio (random stratified) and use the test data from the test.csv file.

1. Visualise 5 random images from 5 different digits. **[5 marks]**
2. Setup1: Create a CNN architecture having **[4*2 = 8 marks]**
    a. a kernel size of 5×5, followed by kernel of 2x2. Use 10 feature maps for the first convolution layer and 20 for the second.
    b. Add pooling after each co nvolution.
    c. Add a linear layer with 50 neurons. Finally, add a classification head. Use softmax activation function at classification head.
    d. In the above setup you are free to test varying pooling techniques, LR rates, optimisers and activation functions, strides and padding.
3. Setup 2: Update setup 1 by adding a residual connection after one of the layers as you deem fit. **[2 marks]**
4. Generate the following plots for each setup: **[5 marks]**
    - Loss plot - Training Loss and Validation Loss V/s Epochs.
    - Accuracy plot - Training Accuracy, Validation Accuracy V/s Epochs
    - Analyze and Explain the plots obtained

PART III: Data Augmentation(15 marks)

Use the best model configuration from Part 2, try each of the below augmentations one by one on a random stratified subset 20% of the train split. Add the modifications to your original train set and observe change in performance (one augmentation at a time).

NOTE: DO NOT PERFORM ANY AUGMENTATION ON THE VAL OR TEST SET.

1. Positional Augmentation: **[2*5 = 10]**
    a. Resize your data. **[2 marks]**
    b. Left-right flip the original data. **[2 marks]**
    c. Rotate the original data by some degree. **[2 marks]**
    d. Add some Gaussian noise to the original data. **[2 marks]**
    e. Combine all the augmentation steps among the above four parts. **[2 marks]**
2. Generate the following plots for each augmentation: **[5 marks]**
    - Loss plot - Training Loss and Validation Loss V/s Epochs.
    - Accuracy plot - Training Accuracy, Validation Accuracy V/s Epochs
    - Analyze and Explain the plots obtained, specially the role different augmentation played in improving the accuracy if any.