Ramakrishna Mission Vivekananda Educational & Research Institute

Belur Math, Howrah, West Bengal

School of Mathematical Sciences, Department of Computer Science

Assignment-2: Deep Learning with MNIST and SVHN Datasets

M.Sc. Computer Science and Big Data Analytics Date: 16 Feb 2025

Course : **CS411: Applications of Computer Vision and Deep Learning**

Deadline: 10th-March-2025, 11:59 P.M.

Instructor: Jimut Bahan Pal Max marks: 140

Instructions: **Attempt all the questions**; **One single PDF should be submitted** alongside the **jupyter-notebook**. Submit the code and the PDF writings as **Name_ROLL.zip** and send it to the reply email on **jimutbahanpal@yahoo.com** AND **jpal.cs@gm.rkmvu.ac.in** as Carbon Copy (CC). You may also CC it to your other personal email address to check whether the email has gone to the address. **Submission after the deadline will fetch you -5 marks per day, so start early!**

1. This part of the assignment takes you through building a classification network pipeline using pytorch from scratch. You must use the Google Colaboratory platform or any other GPU compute resource and submit the notebook for this task. You can tune the model's parameters to run it in decent-sized images without further hurting its performance. Everything in this assignment is left to the student's choice to tune and play with for a better learning experience. Investing at least 15-20 hours for this assignment is required. The notebook should be self-explanatory, and you must demo the assignment in class using a loaded model that you will train and save over time. The focus should be on visualizing the qualitative results and easy demo within 5-10 minutes. Please practice the demo 1-2 times before the final one. (80)

   In this assignment, you will work with two widely used datasets: the **Modified National Institute of Standards and Technology (MNIST)** dataset of handwritten digits and the **Street View House Numbers (SVHN)** dataset. You will build, train, and evaluate Convolutional Neural Network (CNN) models using PyTorch. The assignment is designed to help you understand dataset handling, model building, training, evaluation, and visualization techniques.

   You may discuss the logic and approach with your peers, but you must write and submit your own code. Copying code from others will result in penalties.

   (a) Dataset Loading and Preprocessing **(20 Marks)**

      (a) Load the MNIST and SVHN datasets using PyTorch's `torchvision.datasets` module.

      (b) Set a seed value of **42** for reproducibility in NumPy, Python's `random` module, and PyTorch.

      (c) Split each dataset into training (70%), validation (10%), and testing (20%) sets.

      (d) Plot the first 2 images from the training, validation, and test sets for each dataset.

         - Label the images appropriately, show a distribution plot for the images in train, validation and test samples for MNIST and SVHN datasets.
         - You should have a total of **120 images** (2 samples $\times$ 10 classes $\times$ 3 splits $\times$ 2 datasets).

      This step ensures you understand the dataset structure and can visualize the data distribution across splits.

   (b) Building and Training the CNN Model **(30 Marks)**

      (a) Build a CNN model from scratch using PyTorch's `torch.nn` module. Your model should include:

         - Convolutional layers
         - Batch normalization

- ReLU activation
- Max-pooling layers
- A fully connected layer with **softmax activation** for classification
- The output size should be **10** for both MNIST and SVHN datasets.
- Do not use pre-trained models. Please ensure you make the image's intensity between 0 and 1 before passing into the model for each of the case, or you can use any other normalization technique of your choice, but it should be consistent across the datasets.

(b) Train the model on the MNIST training set for **50 epochs**.
- Use appropriate loss functions (e.g., Cross-Entropy Loss).
- Experiment with different **optimizers** (e.g., SGD, Adam), **learning rates**, **batch sizes**, and **activation functions**.
- Justify your choices for hyperparameters and architectures.

(c) Report metrics for each epoch:
- Accuracy, precision, and recall on the validation and test sets.
- Save these metrics in separate text files.

(d) Plot the training and validation metrics (e.g., loss and accuracy curves) over the 50 epochs.

(e) Repeat the above steps for the **SVHN dataset**.

This part helps you understand how to design and train a CNN model, experiment with hyperparameters, and evaluate performance.

(c) Model Evaluation and Visualization **(25 Marks)**

(a) Compare the performance of the trained models on MNIST and SVHN.
- Identify and visualize **misclassified examples** from the test sets for each class.
- Provide explanations for why these misclassifications might have occurred.
- Suggest potential improvements to the model.

(b) Implement a function to load the trained model and perform inference on the test sets.
- Dynamically demonstrate inference on random test samples from both datasets.
- Create **confusion matrices** for both datasets.
- Visualize **probability maps** (softmax scores) for random test samples.
- Analyze classification performance per class.

This step ensures you can critically evaluate model performance and interpret results.

(d) Dimensionality Reduction and Clustering Analysis **(25 Marks)**

(a) Apply **UMAP** (Uniform Manifold Approximation and Projection) for dimensionality reduction.
- Visualize the feature representations of the fully connected layer's output for both datasets, colored according to the classes in 2D plot.
- Analyze clustering patterns for MNIST and SVHN after training for 50 epochs.
- Compare how the SVHN and MNIST test samples differ in the UMAP visualization.

(b) Cross-dataset evaluation:
- Use the model trained on MNIST to classify SVHN test samples.
- Report UMAP visualizations for the SVHN test samples using the fully connected layer's output.
- Report metrics (accuracy, precision, recall) for this out-of-distribution (OOD) task.
- Repeat the process using the SVHN model to classify MNIST test samples.
- Analyze how the clustering patterns change when using the same model on different datasets.

This part helps you understand how models generalize (or fail to generalize) across datasets and how latent representations differ.

(e) Imbalanced Class Training and Analysis **(30 Marks)**

(a) Train the MNIST model using the following class distributions for 50 epochs:

- **Case 1**: 50 samples of class '1' and 1000 samples of class '2'.
- **Case 2**: 500 samples of class '1' and 1000 samples of class '2'.
- **Case 3**: Full samples of class '1' and full samples of class '2'.

(b) For each case:

- Report test metrics (accuracy, precision, recall) for the two classes on both MNIST and SVHN datasets on these two classes only.
- Show UMAP plots for the test samples passed through the fully connected layers.

(c) Repeat the above steps for the **SVHN dataset**.

(d) Analyze and describe:

- How the latent manifold (approximated by UMAP) changes across the three cases.
- Why the clustering patterns vary and how class imbalance affects model performance.

This step helps you understand the impact of class imbalance on model performance and latent representations.

(f) Report and Submission **(20 Marks)**

(a) Write a report summarizing your findings:

- Include all plots, metrics, descriptions and visualizations numbered according to the questions.
- Provide explanations for misclassifications, clustering patterns, and performance variations.
- Justify your choices of hyperparameters, architectures, and improvements.

(b) Submit your code as a Jupyter notebook with clear comments and explanations.

The report ensures you can communicate your findings effectively and document your work. If you are not so comfortable with latex, please use HTML for reporting, as done in the previous assignment. Please ensure that you save the model files with proper naming convention and backup them later, so that you can use them for demo.
