Neural Networks and Convolutional Neural Networks
COL 774: Assignment 2

Released on: 10th September, 2024

Due Date [Part 1 and 2]: 7.00PM, 22nd September, 2024 [Sunday]

- **Copyright Claim:** This is a property of Prof. Rahul Garg Indian Institute of Technology Delhi, any replication of this without explicit permissions on any websites/sources will be considered violation to the copyright.
- This assignment has two main parts - 1. Neural Network and 2. Convolutional Neural Networks.
- All the announcements related to the assignment will be made on Piazza. The access code is wbd8avkblhb. You are strongly advised to have a look at the relevant Piazza post regularly.
- You are advised to use vector operations (wherever possible) for best performance as the evaluation will be timed.
- You should use Python 3 for all your programming solutions.
- Your assignments will be auto-graded, make sure you test your programs before submitting. We will use your code to train the model on the train set and predict on the test set.
- Input/output format, submission format and other details are included. Your programs should be modular enough to accept specified parameters.
- You may submit the assignment **individually or in groups of 2**. No additional resource or library is allowed except for the ones specifically mentioned in the assignment statement. If you still wish to use some other library, please consult on Piazza. If you choose to use any external resource or copy any part of this assignment, you will be awarded D or F grade or **DISCO**.
- You should submit work of your own. You should cite the source in the report, if you choose to use any external resource for the competitive part i.e. Part 1 Section (c) and (d) and Part 2 Section (c). You will be awarded D or F grade or **DISCO** in case of plagiarism.
- **Late submission policy**: For each late day from the deadline, there will be a penalty of 7%. So, if you submit, say, 4 days after the deadline, you will be graded out of 72% of the weightage of the assignment. Any submission made after 7 days would see a fixed penalty of 50%.
- For doubts, use Piazza. Send an email to Kaustubh R Borgavi for doubts that may disclose implementation details. You are also heavily encouraged to report your best objective function score for Part 1 Section () and (d) and Part 2 Section (c) through mailing.

History of updates

1. **Part 2: Convolutional Neural Networks (Score: 50 Points, Due date: 22nd September, 2024)** This part of the assignment is to get you started with Convolutional Neural Networks (CNN) and deep learning. You are going to experiment and get an experience working with the popular deep learning framework `PyTorch`. A2.2 has been divided into three parts, in which you will perform the image classification task. (a) Implement a simple CNN architecture and evaluate performance on the given handwritten script images dataset (same as the one used for A2.1) for binary classification of two language scripts: *Bengali* and *Kannada*. (b) Implement the given CNN for 8-class classification on the same dataset comprising of all of its classes (c) Competitive part on the 8-class classification model where the grading would be done on the basis of the accuracy obtained on the Private dataset. In (c), you are free to experiment with the model architecture, optimisers and their corresponding hyperparameters with training under a time constraint. All the parts have to be implemented using `PyTorch`.

   You are allowed to use from the following modules. Make sure your scripts do not include any other modules. `os, torch, torchvision, numpy, matplotlib, cv2`. A `conda .yml` **LINK** file will be given to you set up your environment on the system such that the versions of the packages remain consistent with the ones that will be used during evaluations. You can install the environment using the given `.yml` file by running the following command: `conda env create --name a2_cnn_env --file=a2_cnn_env.yml`

   Selection of compute device: For all parts of this assignment, you will be training and testing your models on the `CPU` itself. Hence, please **DO NOT** load your models to the GPU.

   Prerequisites:

   - Getting started with the PyTorch
   - PyTorch tutorial on YouTube

   (a) **(17 Points) Binary classification using a CNN:** In this part, you will be implementing the given CNN in `PyTorch` [Refer architecture diagram given below] for carrying out binary classification. You are going to use the `Binary dataset` with **2 classes** of Handwritten word scripts namely: `Bengali` and `Kannada`. You will be training this CNN using `Pytorch` on the Public train set and then testing the performance of the model on the public test set.

   Dataset:

   - **Public Train Data**: Download. The `Binary dataset` consists 400 images per class in the train set. The dataset has been shuffled already, hence, make sure that you **do not shuffle** it.
   - **Public Test Data**: Download. The public test set is for measuring the performance of your model. It contains 100 images per class. The dataset has been shuffled already, hence, make sure that you **do not shuffle** it.
   - **Private Train and Test Data**: During evaluation we will be using the private train set to train the model and evaluate it on the private test set. The characteristics of the private dataset is similar to the given public dataset.

   Data Loader: You have been provided with a custom dataloader boilerplate code similar to the `preprocessor.py` script in the previous part of this assignment. Download link Use the `Trainloader.py` script to load the training dataset and the `TestDataloader.py` script to load the testing dataset. You will have to import the `CustomImageDataset` classes defined in these scripts in your submission scripts for loading the data. Please refer the `main` function defined in both the script to understand the usage of these Dataloaders in `PyTorch`.

   Model Architecture for Part (a):

   i. **CONV1** (2D Convolution Layer) in_channels = 1, out_channels = 32, kernel=3×3, stride = 1, padding=1.
   ii. **RELU** ReLU Non-Linearity
   iii. **POOL1** (MaxPool Layer) kernel_size=2×2, stride=2, padding=0.
   iv. **CONV2** (2D Convolution Layer) in_channels = 32, out_channels = 64, kernel=3×3, stride = 1, padding=1.
   v. **RELU** ReLU Non-Linearity

vi. **POOL2** (MaxPool Layer) kernel_size=2×2, stride=2.

vii. **FC1** (Fully Connected Layer) output = 1

Instructions:
CNN Specifications:

- Your model has been designed to be compatible with the input size of $[C = 1, H = 50, W = 100]$ which is handled by the `Trainloader.py` and `Testloader.py` scripts.
- Set `torch.manual_seed(0)` at the beginning of your training script.
- Set your model and input tensors to the datatype `float32` using the syntax `.float()` in `torch.`
- Use **Cross Entropy Loss** (`torch.nn.BCEWithLogitsLoss()`) function and **Adam Optimizer** `torch.optim.Adam(model.parameters(), lr=0.001)` and `Batch size=128`.
- Train your model for **8 epochs**.
- Save the model state dictionary by using `torch.save(net.state_dict(), "part_a_binary_model.pth")`, where `net` is the object of my network class.
- You will using the Public train set to train your model and public test set to test your model's performance.
- During evaluation, the private train set will be used to train your model and the private test set to test your model.
- Save the weights after each training epoch. The last saved weights file after training of 8 epochs should be used by your testing script which will carry out inferencing on the test set. The predictions from the test set should be saved as a `predictions.pkl` file [Refer next point on how save this file]. The predictions generated on the private test set after 8 epochs of training on the private train set will be used to compute the accuracy which will be used for grading.
- **Guidelines to submit the predictions:** You should use the **TrainDataLoader** link to initialize the train loader for loading the public training data and the **TestDataLoader** link for loading the public test set. While saving the predictions, ensure to save only the predicted labels on the public test set as **integer classes** [refer class idx in `public_train.csv` or `public_test.csv`] in the form of a 1-D `numpy` array. For ex: If `pred` is the 1-D array storing the predictions, then `pred[i]` is the predicted class idx for the test sample `i`. The length of the `pred` array is the number of samples in the test set. Save this numpy array as a pickle file by the name `predictions.pkl`. While implementing the assignment, you will be using the public test set to save the corresponding `predictions.pkl` file. While evaluation, we will be giving the private dataset inplace of the given public dataset to train and test the model.
- Do not shuffle train and test samples.
- The hyperparameters will be fixed for this part of the assignment.

Report: Here the points to be included in the report explaining your evalution of the model:

- Report public test accuracy and other observations.
- Line plot to show training loss on the y-axis and the epoch on the x-axis.
- Calculate the accuracy of public test data at the end of each epoch and plot a line graph with accuracy on y-axis and number of epochs on the x-axis.
- Compare the accuracy with the Neural Network and argue which one performs better in terms of accuracy, and training time. Comment on these observations.

(b) **(17 Points) Multi-class Classification** In this part, you will be implementing a CNN for the task of multi-class classification. The `Multi-class dataset` shared here contains 8 Indian language images. You will be training this CNN using `Pytorch` on the Public train set and then testing the performance of the model on the public test set.

Dataset:

- **Train Data**: Download. The `Binary dataset` consists 400 images per class in the train set. The dataset has been shuffled already, hence, make sure that you **do not shuffle** it.

- **Public Test Data**: Download. The public test set is for measuring the performance of your model. It contains 100 images per class. The dataset has been shuffled already, hence, make sure that you **do not shuffle** it.
- **Private Train and Test Data**: During evaluation we will be using the private train set to train the model and evaluate it on the private test set. The characteristics of the private dataset is similar to the given public dataset.

Model Architecture for Part (b):

i. **CONV1** (2D Convolution Layer) in_channels = 1, out_channels = 32, kernel=3×3, stride = 1, padding=1.
ii. **RELU** ReLU Non-Linearity
iii. **POOL1** (MaxPool Layer) kernel_size=2×2, stride=2, padding=0.
iv. **CONV2** (2D Convolution Layer) in_channels = 32, out_channels = 64, kernel=3×3, stride = 1, padding=1.
v. **RELU** ReLU Non-Linearity
vi. **POOL2** (MaxPool Layer) kernel_size=2×2, stride=2.
vii. **CONV3** (2D Convolution Layer) in_channels = 64, out_channels = 128, kernel=3×3, stride = 1, padding=1.
viii. **RELU** ReLU Non-Linearity
ix. **POOL3** (MaxPool Layer) kernel_size=2×2, stride=1, padding=0.
x. **FC1** (Fully Connected Layer) output = 512
xi. **RELU** ReLU Non-Linearity
xii. **FC2** (Fully Connected Layer) output = 8

Data Loader: You have been provided with a custom dataloader boilerplate code similar to the `preprocessor.py` script in the previous part of this assignment. Download link Use the `Trainloader.py` script to load the training dataset and the `TestDataloader.py` script to load the testing dataset. You will have to import the Dataset classes defined in these scripts in your submission scripts for loading the data. The `main` function defined in these two scripts shows an example usage of these scripts.

Instructions:

- Your model has been designed to be compatible with the input size of $[C = 3, H = 50, W = 100]$ which is handled by the Dataloder scripts.
- Set `torch.manual_seed(0)` at the beginning of your training script.
- Set your model and input tensors to the datatype `float32` using the syntax `.float()` in `torch.`
- Use **Cross Entropy Loss** `(torch.nn.CrossEntropyLoss())` function and **Adam Optimizer** `torch.optim.Adam(model.parameters(), lr=0.001)` and `Batch size=128` [Please note: While computing loss, you would have to convert the labels to the datatype `.long()` due to `PyTorch`'s internal implementation of the CrossEntropyLoss.].
- Train your model for **8 epochs**.
- Save the model state dictionary by using `torch.save(net.state_dict(), "part_b_multi_model.pth")`, where `net` is the object of my network class.
- You will using the Public train set to train your model and public test set to test your model's performance.
- During evaluation, the private train set will be used to train your model and the private test set to test your model.
- Save the weights after each training epoch. The last saved weights file after training of 8 epochs should be used by your testing script which will carry out inferencing on the test set. The predictions from the test set should be saved as a `predictions.pkl` file [Refer next point on how save this file]. The predictions generated on the private test set after 8 epochs of training on the private train set will be used to compute the accuracy which will be used for grading.
- **Guidelines to submit the predictions:** You should use the **TrainDataLoader link** to initialize the train loader for loading the public training data and the **TestDataLoader link** for loading the public test set. While saving the predictions, ensure to save only the predicted labels on the public test set as **integer classes** [refer class idx in `public_train.csv` or `public_test.csv`]

in the form of a 1-D `numpy` array. For ex: If `pred` is the 1-D array storing the predictions, then `pred[i]` is the predicted class idx for the test sample `i`. The length of the `pred` array is the number of samples in the test set. Save this numpy array as a pickle file by the name `predictions.pkl`. While implementing the assignment, you will be using the public test set to save the corresponding `predictions.pkl` file. While evaluation, we will be giving the private dataset inplace of the given public dataset to train and test the model.

- Do not shuffle train and test samples.
- The hyperparameters will be fixed for this part of the assignment.

Report: Same as Part(a)

(c) **(16 Points) Designing the CNN model for best accuracy:** In this task, you are required to design a Convolutional Neural Network (CNN) for 8-class classification that achieves the best possible accuracy on a held-out test set within 30 minutes of runtime. The task is competitive, and your grade will be based on the relative accuracy achieved by your model compared to other students. You are encouraged to experiment with different CNN designs [some suggestions are listed below], such as varying the number of hidden layers, units in each layer, loss functions, activation functions, and gradient descent variants, using the given Multi-class dataset from Part (b). After training, you will be checking the performance of your model on the public test set. You will writing a separate testing script which will load the latest weights after training and inference on the given test set. The predictions from the public/private test set should be saved in a predictions.pkl file, and the weights saved at the end of training or the 30-minute runtime will be used to compute the final accuracy. You should use the `trainloader.py` to initialize the train loader for loading the training data and the `testloader.py` for loading the public test set. While experimenting, you can use the public train set to train your model and the public test set to check the performance of your model. While evaluation this will be replaced by the private train and test set. Additionally, visualize the loss graphs for both training and validation to comment on the convergence of your chosen architecture, and compare your chosen architecture and training settings with other variants tried during experimentation to support your findings. Ensure that the training does not exceed the 30-minute limit, with the script overwriting the weights after each epoch. This is a competitive part in which you will be evaluated over a private dataset similar to the public dataset.

Here are some suggestions for innovative CNN models for your reference:

- AlexNet
- Residual Network based CNNs
- VGGNet

Report:

- Report your final model architecture along with training and feature engineering details of your best performing model on the public dataset.
- Explain your architecture, and comment why did you choose it over the others.
- Write about all the experimentation you performed along with their performance metrics (eg. loss values and public test accuracy etc). Also mention which ones did not work out. Also, argue about the possible reason behind their failure.
- Report your loss values after each epoch as a line plot vs epochs.

Submission Guidelines: You will have to submit the python scripts for all three parts as a zip file named as: `Entry1_Entry2.zip` on **Moodle**. You will submitting the report named as: `Entry1_Entry2.pdf` on **Gradescope**.

(a) For problem 2.2(a)

- Make sure you follow the same naming conventions and command line arguments for the files while submission:
  i. `part_a_train.py --train_dataset_root --save_weights_path`
    A. Here, `train_dataset_root` takes the path to the train dataset root (public/private) folder which will be given as an input to the `CustomImageDataset` class imported from `Trainloader.py` script. `save_weights_path` takes the path to the location where the `part_a_binary_model.pth` model weights will be overwritten after each epoch of training.

ii. `part_a_test.py --test_dataset_root --load_weights_path --save_predictions_path`

A. Here, `test_dataset_root` takes the path to the test dataset root (public/private) folder which will be given as an input to the `CustomImageDataset` class imported from `Testloader.py` script. `load_weights_path` takes the path to the location where the `part_a_binary_model.pth` model weights were saved after training for 8 epochs for loading. `--save_predictions_path` takes the path to the location where the `prediction.pkl` file will be saved after inferencing on the test set.

- During evaluation, we will be training and testing your model on the private dataset and check if your predictions match with our predictions on the same model on the private dataset.
- The codes will be autograded, hence ensure that your submission is following the given submission format.
- For any error in the code or in the submission format, a penalty will be applied depending on the issue.

(b) For problem 2.2(b)

- Make sure you follow the same naming conventions and command line arguments for the files while submission:

i. `part_b_train.py --train_dataset_root --save_weights_path`

A. Here, `train_dataset_root` takes the path to the train dataset root (public/private) folder which will be given as an input to the `CustomImageDataset` class imported from `Trainloader.py` script. `save_weights_path` takes the path to the location where the `part_b_multi_model.pth` model weights will be overwritten after each epoch of training.

ii. `part_b_test.py --test_dataset_root --load_weights_path --save_predictions_path`

A. Here, `test_dataset_root` takes the path to the test dataset root (public/private) folder which will be given as an input to the `CustomImageDataset` class imported from `Testloader.py` script. `load_weights_path` takes the path to the location where the `part_b_multi_model.pth` model weights were saved after training for 8 epochs for loading. `--save_predictions_path` takes the path to the location where the `prediction.pkl` file will be saved after inferencing on the test set.

(c) During evaluation, we will be training and testing your model on the private dataset and check if your predictions match with our predictions on the same model on the private dataset.

(d) The codes will be autograded, hence ensure that your submission is following the given submission format.

(e) For any error in the code or in the submission format, a penalty will be applied depending on the issue.

(f) For problem 2.2(c)

- Make sure you follow the same naming conventions and command line arguments for the files while submission:

i. `part_c_train.py --train_dataset_root --save_weights_path`

A. Here, `train_dataset_root` takes the path to the train dataset root (public/private) folder which will be given as an input to the `CustomImageDataset` class imported from `Trainloader.py` script. `save_weights_path` takes the path to the location where the `part_c_model.pth` model weights will be overwritten after each epoch of training until the training is completed or by the end of 30 minutes (whichever is sooner).

ii. `part_c_test.py --test_dataset_root --load_weights_path --save_predictions_path`

A. Here, `test_dataset_root` takes the path to the test dataset root (public/private) folder which will be given as an input to the `CustomImageDataset` class imported from `Testloader.py` script. `load_weights_path` takes the path to the location where the `part_c_multi_model.pth` model weights were saved after training for loading. `--save_predictions_path` takes the path to the location where the `prediction.pkl` file will be saved after inferencing on the test set.

(g) During evaluation, we will be training and testing your model on the private dataset to compute the final accuracy. This is a competitive part and the grading will be relative to the accuracy obtained in comparison to other students.

(h) The codes will be autograded, hence ensure that your submission is following the given submission format.

(i) Your training should complete within **maximum runtime of 30 minutes**. You can fix the number of epochs in order to stop the training before this. Also, please overwrite the weights after each epoch so that the last saved weights after training or after 30 minutes (whichever is sooner) will be used to compute to the final accuracy for grading using the testing script.

(j) For any error in the code or in the submission format, a penalty will be applied depending on the issue.

Grading Scheme:

- For 2.2(a) and 2.2(b): you will be evaluated based on the correctness of your implementation. It will be evaluated based on the loss and accuracy obtained after training and testing on the private dataset.
- For 2.2(c) you will be evaluated based on the accuracy over private test dataset. This part is competitive and the grading will be in comparison to other students.