EECE 571T - Assignment 2

Bhushan Gopaluni, Matthew Yedlin

Due Date: March 22, 2023 - 11:59 pm

Late submissions will not be accepted. In case of a medical or family emergency, please inform the instructors.

Deliverables: Please upload the completed version of the jupyter notebook file (.ipynb) on Canvas. You need to complete the sections marked with a *ToDo* tag in the notebook. The jupyter notebook that you submit must contain the cell output from a clean execution (restart kernel and run all cells sequentially). Please do not upload other files such as datasets.

Runtime: This assignment includes only PyTorch coding questions. You will need GPU, for which we recommend using Google Colaboratory. You can easily unzip the assignment2 folder, which contains a jupyter notebook and a CSV file. Upload the folder on your google drive and run the notebook.

1 Image Classification using CNNs - 6 points

In this question, we want to use convolutional neural networks (CNNs) to do image classification on the MNIST dataset, a collection of 28×28 images of handwritten digits from 0 to 9. Our CNN takes one image as input and predicts the digit it represents. I.e, each digit is a class. This question is mostly a tutorial on neural networks in PyTorch, which is organized into five subsections.

1.1 Dataset and Dataloader - 0 points

PyTorch has an efficient and user-friendly API for loading and parsing data. Get familiar with the PyTorch dataloader as it will be an inseparable part of your assignments and projects. For clarity, the dataloaders return a python dictionary throughout the assignment.

1.2 Train, Validation, Test Split - 3 points

MNIST does not provide a validation set. split the training data into a train set and a validation set. You can use the validation set for hyperparameter tuning. Use the test set only for reporting the final results.

1.3 Network - 0 points

The jupyter notebook file provides two sample networks, one using a pre-defined network called ResNet, and one implemented from scratch. Try to use both of them and get familiar with neural networks in PyTorch. The networks expect a python dictionary as input and return a dictionary as output, to be consistent with other modules such as dataloaders and the training script.

1.4 Training - 0 points

We have provided you with the script for training the network, calculating the loss, backpropagating the gradients, and transferring the data between CPU and GPU. Please read this part of the code thoroughly.

Think about the loss function and compare it with other classification losses. Also, you can change hyper-parameters like batch size and learning rate. Train the network for at least five epochs to see meaningful changes in the loss, but avoid training for too long as you have limited time and GPU resources.

1.5 Evaluation - 3 points

Now we want to evaluate our trained network. We are only interested in the accuracy of the network on the test set. But for comparison, report the accuracy of the model on train, validation, and test sets. Also, plot the input images and output of your network for a few samples in the test set. You can use the function **plot_images()** provided for you in the **Helper Functions** section. Note that you should use your testing data only for evaluation, not for training or hyperparameter tuning.

2 Anomaly Detection using CNNs - 12 points

In this question, given three images, we want to detect the image representing a different digit. The CNN takes three different images from the MNIST dataset as inputs, where two of them have the same digit and the other one has a different digit. The network predicts the index of the image with a different digit. For example in Fig 1, the first image (from left to right) has a different digit. Therefore, the label of this sample is 0, which is the index of the first image. Note that we do not have access to the digit of each image. We only know the index of the different image. Also, in this question, we are considering only digits 0 to 4.

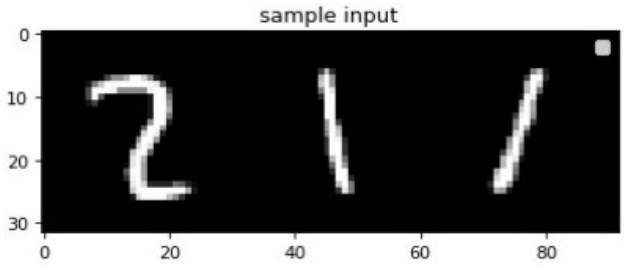

Figure 1: Sample input for question 2. The figure contains three images (from left to right): [(digit:2, index:0), (digit:1, index:1), (digit:1, index:2)]. The label for this data is 0, which is the index of the image with a different digit.

2.1 Dataset and Dataloader - 0 points

The dataloader returns three randomly chosen images and the index of the image with a different digit. It is guaranteed that two and only two of the images represent the same digits. The batch size is set to 128, but feel free to change it.

2.2 Train, Validation, Test Split - 1 point

Split the data into train, validation, and test sets. Print the number of samples in each dataset.

2.3 Network - 6 points

Implement your CNN from scratch. **Do not use pre-defined networks.** The network should take a python dictionary as input and return a dictionary as output. More details in the jupyter notebook.

2.4 Training - 2 points

The training script is provided to you similar to question 1. But you should define a suitable loss function that matches the output of your network and helps the network to learn.

2.5 Evaluation - 3 points

Same as question 1, report the accuracy of the model and plot the input images and output of your network for a few samples in the test set.

3 Sentiment Analysis using RNNs - 12 points

The IMDB dataset contains 50000 movie reviews. Each review is labeled as either positive (1) or negative (0). We want to implement a recurrent neural network (RNN) that takes each review (series of words) as input and predicts the sentiment behind it, i.e, whether it is a positive or a negative review.

3.1 Load and Tokenize Data - 0 points

We use a preprocessed version of the IMDB dataset done by Affandy Fahrizain, which is attached to your assignment folder. We then assign a unique token (number) to each word, so we can apply mathematical operations to them.

3.2 Dataloader - 0 points

The dataloader returns a python dictionary containing reviews and their labels. Also, the data is split into the train, validation, and test sets.

3.3 Network - 6 points

Implement your network based on Long Short-Term Memory (LSTM) blocks. The network should take a python dictionary as input and return a dictionary as output. More details in the jupyter notebook.

3.4 Training - 3 points

The training script is provided to you similar to question 1. But you should define a suitable loss function that matches the output of your network and helps the network to learn. It usually takes more epochs for RNNs than CNNs to converge. Train the model for at least 10 epochs.

3.5 Evaluation - 3 points

Report the accuracy of the model on the train, validation, and test sets. Also, print the input review and output of your network for one positive example and one negative example in the test set. Choose qualitative examples that show the effectiveness of your network.