Assignment 1

Fully Connected and CNN's

Submission deadline: 15.02.2024

General Instructions:

- Submission via moodle, must include code (and all necessary files to run it).
- The code must be written in Python 3.6 or higher, and must run.
- Avoid external libraries other than torch and numpy.

Assignment:

Fashion Mnist is a dataset with 60K 28x28 greyscale images. The images belong to 10 categories of clothing articles. In this exercise you will design and train a network on this dataset.

In this exercise you will design 4 networks, showing the difference between Feed Forward and CNN and the importance of network depth.

Assignment (Code):

1. Net_1: Fully Connected, 3 classes, 2 weighted layers.

   For this network, use only classes {0,1,2} of the dataset. Design and train a 2 layer fully connected network. The network should reach about 80% accuracy and use at most 50K parameters.

2. Net_2: Fully Connected, 7 classes, 2 weighted layers.

   Same as (1), but use classes {0,…6}. Reach as high accuracy as you can. Use no more than 50K parameters.

3. Net_3: Fully Connected, 7 classes, 4 weighted layers.

   Same as (2), with 4 fully connected layers instead of 2. Reach as high accuracy as you can. Use no more than 50K parameters.

4. Net_4: CNN, 7 classes:

   Design a CNN for the dataset with the 7 labels {0,…6}. Train to reach better performance than Net_3. The network's weighted layers should be: 3 convolutional layers, and one fully connected layer in the end to flatten the tensors. Use no more than 50k parameters.

Note: the architecture limitations apply only to weighted (trainable) layers. Regularization, normalization, activation, and max-pooling layers are by definition not trainable layers, and you can use them freely.

Resources (Code):

For your convenience, you may use the following code, available with the assignment, in the file FMnist.py:

1. Train_loader is a torch iterator that downloads a Fashion items dataset; a set of 28x28 greyscale images with 10 categories of clothing articles.
2. Test_loader is the same, for the test set
3. Splice_batch receives data (X), labels (Y) and a number of classes, and removes every data entry beyond the number of classes given.
4. Count_parameters receives a model and returns the number of weighted parameters that model uses. Use this function to check yourself.
5. View_data_sample receives one of the dataloaders and plots the images.

Assignment (Report):

1. Report your architecture, training time and performance on the test set. Do this for each network.
2. Add a plot showing train error, test error, and accuracy as a function of the number of epochs. Do this for each network.
3. Explain why the performance of Network 1 is better than that of Network 2.
4. Explain why the performance of Network 3 is better than the performance of Network 2.
5. Explain why the performance of Network 4 is better than the performance of Network 3.
6. Conclude, by questions 3,4,5 above, which architecture is best suited for the FMNIST classification problem.

Submission:

Your submission should include:

- All the networks you were required to design, and the code used to train them.
- A saved file of your trained model, and a code that loads and tests it.
    - See: https://pytorch.org/tutorials/beginner/saving_loading_models.html

- A PDF assignment report.

Good luck!