Deep Learning # 2
Universität Bern

Hamadi Chihaoui(hamadi.chihaoui@unibe.ch)

Due date: 25/04/2023

1 Introduction

In this assignment you need to upload a zip file to ILIAS which includes: 1) A Jupyter Notebook file Assignment2.ipynb completed with code and answers and 2) a Jupyter Notebook exported to HTML (File / Export Notebook as / HTML). The zip file name must be FirstName LastName.zip. If your implementation requires auxiliary functions, you must implement those functions inside a corresponding .py file. Please state your name at the beginning of the notebook.

1.1 Notes on code and submission quality

In addition to answering the different questions, you are also expected to provide well written submissions. Here are some recommendations to take into consideration.

- Please answer the question in the same order as in the assignment and use the same question numbers.
- Don't answer the questions in the code comments. Use the text cells in your notebook.
- Remove clutter such as unused code lines instead of turning them into comments.
- Make sure the right execution order of the notebook cells is from top to bottom. A TA should be able to reproduce your results by simply clicking "Run All" without having to guess which cells should be executed first.

Poorly written submissions might result in points deduction.

2 Problem

In this assignment, our goal is to train a traffic sign detection model that output the **list** of the traffic signs present in a given picture. You are given a dataset of some traffic sign image patches (from 6 categories) and some image patches without a traffic sign that represent the `no_sign` class. The image patches have a resolution of 64x64 pixels.

Figure 1: A representative of each class in the dataset. The first 6 classes represent some traffic sign categories, while the last class is the `no_sign` class.

You will train a multi-class (7 classes) classification model and then use it to output the traffic signs that are present on some bigger images of 512x512 pixels.

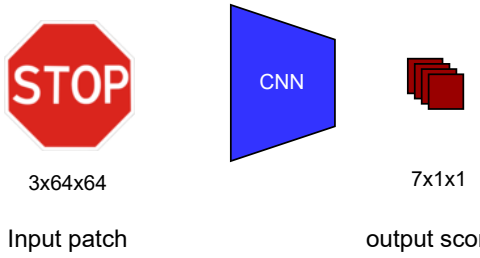

Figure 2: Scheme for training a multi-class classifier. We train on image patches of fixed size and get 7 scores as output for every patch. Each score is relative to one class.

Tasks:

1. **[Total 70 pts] Train a convolutional neural network for multi-class classification.**

   You will implement and train a convolutional neural network for multi-class classification of image patches (see Fig 2). To access the data, first extract `data.zip`. Folders `train` and `val` contain the train and validation datasets respectively.

   - **[15 pts]** Create Dataset and DataLoader objects for provided training and validation data (folders train and val). Visualize few images from each class.
   - **[15 pts]** Implement a convolutional model according to the definition below:
     – Convolutional layer, kernel size 3x3, stride 1, 32 channels
     – Max Pooling layer, kernel size 3x3, stride 2, ceil mode=True
     – Activation function LeakyReLU, slope 0.2
     – Convolutional layer, kernel size 3x3, stride 1, 64 channels
     – Max Pooling layer, kernel size 3x3, stride 2

– Activation function LeakyReLU, slope 0.2
– Convolutional layer, kernel size 3x3, stride 1, 64 channels
– Max Pooling layer, kernel size 2x2, stride 2
– Activation function LeakyReLU, slope 0.2
– Convolutional layer, kernel size 2x2, stride 1, 128 channels
– Activation function LeakyReLU, slope 0.2
– Convolutional layer, kernel size 3x3, stride 1, 256 channels
– Activation function LeakyReLU, slope 0.2
– Convolutional layer, kernel size 3x3, stride 1, 256 channels
– Activation function LeakyReLU, slope 0.2
– Convolutional layer, kernel size 1x1, stride 1, 7 (output) channels

- **[25 pts]** Write the training code and train the network you implemented.
  – Train for 40 epochs with batch size of 16.
  – Optimize the cross entropy loss.
  – Use Adam optimizer with learning rate 3e-4.
- **[15 pts]** Include plots for the training and validation losses and accuracies.

Figure 3: An example of a 512x512 image where we want to find the list of all existing traffic signs.

2. **[Total 20 pts] Run your fully-convolutional model on full images**

   Since our model is fully-convolutional and no layer needs a fixed size input, we can feed it with bigger images and get a spatial map of traffic sign scores for different image patches. Our traffic sign detector can only detect traffic signs of size 64x64. To detect bigger traffic signs, we will first resize our images to smaller scales and then run the network on different scales of the same image. Your task will be to implement a traffic sign detector using your trained classification model. You are given a skeleton of `TrafficSignDetector` class.

- **[10 pts]** Finish the implementation of `detect_single_scale` method of `TrafficSignDetector` class, that takes an input image and returns a set of all detected traffic sign classes using your trained model (details in the method comments)
- **[5 pts]** Finish the implementation of `detect_multi_scale` method of `TrafficSignDetector` class, that runs `detect_single_scale` on different sizes of the image.
- **[5 pts]** There are 4 full size images in the `full_images` folder. Initialize the detector with your trained model and run the multi-scale traffic sign detection on the images and output the detected traffic sign classes for every image. For example, for image `d.png` (see Fig 3), an ideal detection should output a list containing the class `stop`, `output=['stop']` .

3. **[Total 10 pts] Improve your model**

   Try to improve your model by tuning your architecture, hyper-parameters, regularization etc. Some examples of improvements: training parameters (e.g. optimizer, learning rate, scheduling, batch size), network architecture (e.g. number of layers, number of units, activation function, normalization layers), model regularization (e.g. data augmentation, dropout, weight decay, early stopping), test-time augmentation, etc.

   Train at least 3 changed configurations from scratch, describe the changes you made and report classification accuracy on the validation set from point 1.