Advanced Deep Learning 2022 Assignment 1

The deadline for this assignment is **May 10, 2021, 22:00**. You must submit your solution electronically via the Absalon home page.

Important rules and notes:

- All assignments in the course are individual:
  - You are not allowed to collaborate with anyone on the assignment and you are not allowed to communicate your solutions to other students.
  - You must not ask for help from anyone except the teachers and TAs on the course. On the other hand, we encourage you to use the exercise classes and the Absalon forum to get help. The exercise sessions exist to help you with the assignments, and you are welcome to ask any questions related to the teaching material and the assignments on the forum.
  - If your solution contains material from other sources than the assignment text, you must cite the source of the material and any changes you have made. This also applies to material from textbooks, Absalon, etc.
  - If your solution uses methods or notation which are not used in the course material, you must specify where you have found the method or notation.
  - If you are in doubt about plagiarism or citation rules, please ask the teachers or TAs.

  Please be very observant of these rules. We do not want any plagiarism cases, both for your and our sake.
- A solution consists of:
  - A PDF file with detailed answers to the questions, which may include graphs and tables if needed. Do *not* include your complete source code in the PDF file except if you are asked to do so. However, you can show important parts (i.e., a few lines) of your source code.
  - A .zip file with all your solution source code with comments about the major steps involved in each question (see below). Source code must be submitted in the original file format, not as PDF.
- *Important: Do not zip the PDF file*, since zipped files cannot be opened in the speed grader. Zipped pdf submissions will not be graded.
- Your PDF report should be self-sufficient. That is, it should be possible to grade it without opening the .zip file. We do not guarantee opening the .zip file when grading.
- Your code should be structured such that there is one main file (or one main file per question) that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions, classes, etc.

- Your code should include a README text file describing how to compile and run your program, as well as a list of all relevant libraries needed for compiling or using your code.
- Handwritten solutions will not be accepted, please use the provided LaTeX template to write your report.
- Figures should be readable. They should have proper captions (same holds for tables) and labels, a legend when appropriate. In general, write proper sentences in the report.
- You are only asked to perform tasks that make sense from a machine learning point of view. If you do something that that does not make much sense, it is most likely wrong.

1 Convolutional neural networks

The learning goal of this part of the assignment is to get more comfortable with implementing convolutional neural networks (CNNs) in PyTorch.

1.1 Basic CNN definition

Consider the notebook `Torch_Traffic_Signs_Basic_Template.ipynb`. It is a template for training a convolutional neural network (CNN) for classifying traffic signs on the same data as the competition described in [4]. Recognition of traffic signs is a challenging real-world problem of high industrial relevance. Traffic sign recognition can be viewed as a multi-class classification problem with unbalanced class frequencies, in which one has to cope with large variations in visual appearances due to illumination changes, partial occlusions, rotations, weather conditions, etc. However, humans are capable of recognizing the large variety of existing road signs with close to 100 % correctness – not only in real-world driving situations, which provides both context and multiple views of a single traffic sign, but also when looking at single images as those shown in Figure 1. Now the question is how good a computer can become at solving this problem.

The notebook is supposed to run with GPU support, for example, using *Google Colaboratory*. Even then, executing it will take some time.

The notebook misses the CNN definition.

Consider the following print of a model:

```
Net(
  (conv1): Conv2d(3, 64, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(64, 32, kernel_size=(5, 5), stride=(1, 1))
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
      ceil_mode=False)
  (conv3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
      ceil_mode=False)
  (fc3): Linear(in_features=512, out_features=43, bias=True)
)
```

1. Give a description of each part of the neural network in words.
2. Calculate and explain why the number of input features to the last layer (the linear layer) should be 512.

Figure 1: Examples from the traffic sign data set.

3. Implement the network in the notebook. After each convolutional layer, the ELU (exponential linear unit, [1]) activation function should be applied. Show the model definition in the report.

1.2 Batch normalization

The learning goal of this part of the assignment is gaining experience in using batch normalization and and in reading original scientific Deep Learning articles.

Read the following frequently cited paper [2], which is freely available online:

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

You will need algorithmic details of the described method for answering question below.

Add batch normalization to the model. Look up the corresponding PyTorch function in the documentation. Look at the Batch Normalization section in d2l for examples of where batch normalization can be applied to a CNN.

Please show in the report all places where you added it (include the layer before and after for each batch normalization layer).

It is an open discussion where to do the batch normalization relative to the non-linearities (activation functions). Here you have to do it as described in [2].

1.3 Optimizers

The current implementation uses the Adam optimizer.

Switch to a different optimizer and show the changes to the code in the report. Describe the main features of the optimizer, and argue for your choice of optimizer and hyperparameters.

1.4 Experimental architecture comparison

The learning goal of this part of the assignment is gaining experience in tweaking CNNs and in conducting an experimental comparison between CNN approaches.

The task is to empirically compare different CNN architectures. Take the notebook `Torch_Traffic_Signs_Basic_Template.ipynb` as a starting point.

You are supposed to investigate if the batch normalization and change of optimizer as implemented above improves the performance?

Training a neural network is a random process (random weight initialization, random mini-batches). Thus, single trials are not sufficient to establish that a modification improves the learning for a given task. Proper statistical evaluation is required. The least you can do is to repeat the training and evaluation of the baseline and your alternative system a couple of times and compare the mean/median performance of the approaches. For a scientific study, you would establish the statistical significance of your findings using a (typically non-parametric) statistical significance test. For this assignment, it is sufficient to perform three independent trials for each architecture (you are encouraged to do more, but that is optional).

Training time may be an issue, and longer training can give you better results. *Thus, start with the experiments in good time before the deadline.* Fix a maximum number of epochs for your experiments.

This should be *at least* 400 epochs. Remember to report this number, because the budget may influence the ranking of methods. Visualize the training progress.

In the report, you are supposed to describe what you did, present the results, discuss the results, and draw very careful preliminary conclusions. The report should contain a plot (remember proper axes labels etc.) showing the training progress vs. epochs.

1.5 Challenge (optional)

This part is optional and will not have an effect on the grading. Human performance on the traffic sign recognition task on the test set is 99.22 %, the best algorithm in [4] achieves 99.46,% using an ensemble of CNNs.

Now we challenge you: How good can you get a CNN to perform on this task? For example, you could try dropout; adding neurons, layers or connections to the given architecture; better training augmentation; starting with a pretrained model; test time augmentation; better preprocessing; and/or changing the optimizer and its hyperparameters.

The constraints are:

- The solution should be a single CNN, not a ensemble.
- The solution should be reproducible, that is, if you repeat the training a few times, you should get a solution of similar quality.
- Training the network should not take longer than a couple of hours on *Google Colaboratory*.
- The code should fit in a notebook not longer than twice the one handed out.
- The notebook should be self-contained, only common libraries may be imported. The fewer external libraries the better.
- You may look for inspiration online. However, please cite all sources of inspiration you used.

If you consistently get better than 99 %, your solution is good. If you consistently beat human performance, your solution is very good. If you consistently reach or beat 99.46,% your solution is excellent. Let me know if you get a solution that is good or better!

2 U-Nets

In this part of the assignment, we take a look at U-Nets, which are state-of-the-art in image segmentation, and at the concept of receptive fields, which is helpful for designing and understanding CNNs.

2.1 U-Nets

The learning goals of this part of this assignment are getting hands-on experience in using fully convolutional neural networks for segmentation in practice and to improve the understanding of PyTorch code.

We will consider the popular U-Net [3], which has become the state-of-the-art in medical image segmentation. Download the PyTorch U-Net implementation `https://github.com/PyTorchLightning/lightning-bolts/blob/master/pl_bolts/models/vision/unet.py`. Consider the notebook `x_ray_segmentation_ERDA.ipynb`, which applies the U-Net to the lung segmentation task in x-ray images described in [5].

Adapt the downloaded U-Net implementation:

- Replace the up-convolution by nearest neighbor upsampling. You can use `nn.Upsample` using the model `'nearest'`.
- Replace all activation functions by ELUs.

Compare the performance before and after these modifications. Describe the experimental setup and the results in the report. You are supposed to describe what you did, present the results, discuss the results, and draw very careful preliminary conclusions. The report should contain a plot (remember proper axes labels etc.) showing the training progress vs. epochs.

Keep the guidelines for experimental comparisons in Section 1.4 in mind. Ideally, you should evaluate the effect of each modification individually and in combination. However, if you are running out of compute time, it is sufficient to compare the results without and with both modifications.

References

[1] D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In Y. Bengio and Y. LeCun, editors, *International Conference on Learning Representations (ICLR)*, 2016.

[2] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[3] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015.

[4] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

[5] B. van Ginneken, M. B. Stegmann, and M. Loog. Segmentation of anatomical structures in chest radiographs using supervised methods: a comparative study on a public database. *Medical Image Analysis*, 10(1):19–40, 2006.