Assignment 2

Deadline: 11.15.2023 - 11.59pm

By the deadline, you are required to submit the following:

Report. Prepare a single PDF file that presents clear and concise evidence of your completion of each of the listed tasks. **Use the overleaf template available on iCorsi**.

Source code. Create a single zipped folder containing

1. A Python script **using the template available on iCorsi**, performing each of the specified tasks. If a task is accomplished in the code but not documented in the report, it will be considered incomplete. Therefore, make sure to thoroughly report all your work in the submitted report. **Jupyter notebook files will not be accepted**.
2. The paramters of the two trained model

General hints. The question marked with * is more challenging, and we recommend possibly leaving them as the last questions to solve. To obtain the maximum grade, not only should all exercises be completed correctly, but the plots must also be clear, have a legend, and have labels on the axes and the text should be written in clear English. For saving plots in high quality, consider using the command `matplotlib.pyplot.savefig`. Clarity of plots/text and code will account for 5 points.

Submission rules: As already discussed in class, we will stick to the following rules.

- Code either not written in Python or not using PyTorch receives a grade of 0.
- Submission between 00.00 am - 00.10 am of Thursday 16th will be accepted with no reduction if your counter of late submission is equal to 0. With 10% of reduction otherwise.
- If plagiarism is suspected, TAs and I will thoroughly investigate the situation, and we will summon the student for a face-to-face clarification regarding certain answers they provided. In case of plagiarism, a score reduction will be applied to all the people involved, depending on their level of involvement.

- If extensive usage of AI tools is detected, we will summon the student for a face-to-face clarification regarding certain answers they provided. If the answers are not adequately supported with in-person answers, we will proceed to apply a penalty to the evaluation, ranging from 10% to 100%.
- GPU usage is recommended for this assignment

1 Image classification using CNNs [90/100]

The CIFAR-10 dataset is a widely used collection of images in the field of computer vision. It consists of 60000 (50000 for the training and 10000 for the test) 32x32 color images across 10 different classes, with each class containing 6,000 images. These classes include common objects such as airplanes, automobiles, cats, and dogs. CIFAR-10 serves as a benchmark for image classification tasks and has been instrumental in developing and evaluating machine learning algorithms for image recognition. The task of this assignment is to classify the images

Note: in the python template we set a seed. Don't change it.

1.1 Data (20 pts)

1. (5 pts) Load the data (You can do it directly in PyTorch) and take some time to inspect the dataset. Observe at least one image per class and a histogram of the distribution of the images of the training and test set. What do you observe?

2. (5 pts) Assume you have downloaded the dataset using the variable `dataset_train`. Are the entries in the correct type for the DL framework in PyTorch? How can you arrive at a suitable format for your training pipeline? Answer this question by also providing clarification about

   (i) The type of each element of the dataset

   (ii) How we can convert it to a suitable type. **Hint:** have a look at frame 13 of Lecture 4

   (iii) The dimension of the image as a `torch.Tensor` object

   (iv) The meaning of each dimension of the images

3. (5 pts) When you arrive at this question you should have each entry as a `torch.Tensor` of shape (3 32, 32) and clear the meaning of each dimension. A good practice in DL is to work with features having mean 0 and standard deviation equal to 1. Convert the dataset of the images in this format. To do so, you can do it from scratch (not recommended) or use the function `torchvision.transforms.Normalize`[1]. If you go for this second option, don't forget that we have already transformed our dataset in the previous point, hence, it could be of help using the function `transforms.Compose`[2]. Do not overwrite code.

[1] Documentation
[2] Documentation

4. (5 pts) As you may observed, we only have train and test set. We need a validation set for hyperparameteres tuning. Create a validation set. Use 80% of data for the training set and 20% of the data for the validation set.

1.2 Model (10 pts)

Starting from the code provided during Lecture 6, define a ConvNet. You can **only** use

- Convolutional layers
- Max/Avg Pooling layers
- Activation Functions
- Fully connected layers

For each convolutional layer you can choose padding and stride, however, we recommend to choose padding $= 0$ and stride $= 1$. The other choices are up to you. You can also take some inspiration from famous ConvNets.

1.3 Training (60 pts)

1. (15 pts) Implement the training pipeline. Make sure to code the following
   - Print and record the current training loss and accuracy every n steps (choose n)
   - Print and record the current validation loss and accuracy every n steps (choose n)

   The validation loss will help you in hyperparameters tuning.
2. (13 pts) Train your model. With my choice of hyperparameters, the best test accuracy is above 70%, hence, a necessary condition to get full mark is to achieve an accuracy on the test set greater or equal than 70%.
3. (2 pts) **Save** the parameters of the trained model as NAME_SURNAME_1.pt
4. (10 pts) Plot the evolution of the train and validation loss in the same plot and comment them
5. (18 pts) Change the architecture as you like and try to increase the accuracy as much as possible. Try any ideas that come to your mind but try to *justify it*. Some hints
   - Add Dropout (Any other hyperparameter to tune?)
   - Change activation functions (GeLU is known to work well with images)
   - Make your CNN deeper
   - Add some regularization techniques
   - Change the optimizer
   - You can also use `EarlyStop` from Exercise 3

- Data augmentation

Points will be attributed as follows:

- Any successful try (test accuracy increase): 1 point
- Accuracy on test set $\geq 75\%$: 5 points
- Accuracy on test set $\geq 80\%$: 10 points
- Accuracy on test set $\geq 83\%$: 13 points

To get the remaining 5 points you should also justify your choices (e.g, "I used Dropoout because ... and hence I have increased epochs as I noted ..."). **Note:** Transfer learning is not allowed - otherwise you can complete this exercise easily and it is not useful for educational purpose.

6. (2 pts) **Save** the parameters of the trained model as NAME_SURNAME_2.pt

1.4 Bonus question* (5 pts)

Until now, we asked you to keep the seed fixed. It is now time to do some statistics. Train the model of point 1.3.5 10 times with 10 different seeds (Already specified in the Python template) and see if your best accuracy has been obtained by chance. Try to answer to the following questions:

- How do you compare two models to asses which is better?
- How robust is your model?

Questions [5 points]

After studying Chapter 8.6 of Dive into Deep Learning, summarize in 10 lines what ResNets are (architecture, residual blocks, skip connections, advance w.r.t ConvNet, ...)

Remarks

- In Exercise 1.3.2 I can get an accuracy of 72.2% on the test set in 439.56 seconds with local training using GPUs
- The choice Conv - Pool - Conv is not mandatory. Have you tried Conv - Conv - Activ - Pool - Conv - Conv - Activ - Pool - FC?
- A non-exhaustive list of my hyperparameters: learning rate: 0.03; Epochs: 5; Number of Conv Layers: 4; Batch size: 32