Assignment 2

Deadline: 4th November 2024 - 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**. **Be sure to include all the plots required. Be sure to write a report of no more than 6 pages**.

Source code. Submit 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 report all your work in the submitted report. **Jupyter notebook files will not be accepted**.

General hints. The question marked with * is more challenging, and we recommend possibly leaving it as the last question 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, 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. Best practices in coding will account for 5 points. Best practices for coding include commenting on each function and commenting on the code using the standard rules for Python.

Submission rules: As 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.
- Report & Code submitted after the deadline will receive a grade of 0.
- If plagiarism is suspected, TAs and I will thoroughly investigate the situation, and we will summon the students 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 students 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**

Image classification using CNNs

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. (0 pts) This can be helpful only if you use Python notebook for the assignments before downloading the Python script. Create two cells, in the first one, write `torch.manual_seed(manual_seed)` and in the second one write `torch.randint(1, 10 , (1,1))`. Do the following experiments: (i) Run the first and the second cell in a row. (ii) Run the first cell and two times in a row the second one. What do you observe? We don't need any answer, but please just keep this phenomenon in mind.

2. (5 pts) Load the data (You can do it directly in PyTorch) and take some time to inspect the dataset. Report here at least one image per class and an histogram of the distribution of the images of the training and test set. What do you observe?

3. (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 the slides

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

   (iv) The meaning of each dimension of the images

4. (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

[1] Documentation

dataset in the previous point, hence, it could be of help using the function `transforms.Compose`[2]. You can see an example in the tutorial linked above.

5. (5 pts) As you might have observed, we only have a train and test set. We need a validation set for hyperparameter tuning. Create a validation set by splitting the test set.

6. (20 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

   Note that the choice Conv - Pool - Conv is not mandatory. Have you tried Conv - Conv - Activ - Pool - Conv - Conv - Activ - Pool - FC? For each convolutional layer you can choose padding and stride, however, we recommend choosing padding $= 0$ and stride $= 1$. We highly recommend starting from the model provided during Lecture 6 and adapting it.

7. (35 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 hyperparameter tuning. Train your model. With my choice of hyperparameters, the best test accuracy is above 65%, hence, a necessary condition to get full marks is to achieve an accuracy on the test set greater or equal to 65%. You may want to follow some of these hints:

   - 4 epochs should be enough
   - Batch size = 32 could be a good starting point
   - A learning rate around 0.03 should be a good tradeoff between speed and stability
   - SGD should be a good optimizer

   Report here:

   - (5 pts) Your learning rate, and why? Why did you choose it? Why not higher/smaller?
   - (5 pts) Number of epochs: Why that number?
   - (25 pts) Some comments on your architecture: why that deep? why not deeper? why not smaller? (This seems more like a question related to the previous point. However, the model strictly relies upon the performance of the training)

[2] Documentation

8. (5 pts) Plot the evolution of the train and validation losses in the same plot and comment on them. Specifically focus on the last epochs, showing that no overfitting is detected

9. (15 pts) You may have noticed that it is quite likely to get overfitting in many cases, preventing it from going beyond 65% with accuracy. Change the architecture as you like and try to increase the accuracy as much as possible. The base architecture must remain a ConvNet. Try any idea that comes 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
   - Data augmentation
   - Batch normalization
   - .....

   Points will be attributed as follows:

   - Any successful try (test accuracy increase): 2 point
   - Accuracy on test set $\geq 70\%$: 3 points
   - Accuracy on test set $\geq 72\%$: 4 points
   - Accuracy on test set $\geq 73\%$: 5 points
   - ...
   - Accuracy on test set $\geq 78\%$: 10 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 purposes.

10. (5 pts*) Until now, we asked you to keep the seed fixed. Train **the model you defined at point 6.** 5 times with 5 different seeds[3] (Already specified in the Python template), each for 4 epochs. Don't change other hyperparameters. Observe the accuracy of the test set in each case. What can you say? (It should take 6 minutes with Kaggle GPUs)

[3] If you have more resources, you can even do it more than 5 times (but explain why you should do it)