Deep Learning Assignment

The goal is to optimize the performance of Deep Learning models in the CIFAR-100 dataset using the TensorFlow 2 library. comes with your .ipynb and .py files to check the results.

First, run and study the notebook "Task 3 TensorFlow 2 - CIFAR-100.ipynb". The notebook contains many references in the documentation of TensorFlow 2 which you should study extensively.

TF2 is the only acceptable ML framework for exercise (tensorflow.compat.v1 should not be used either).

Deep Learning Problems GPU acceleration is essential. From cloud solutions you can work in either Colab or Kaggle.

- Open it in Colab and make a copy on your drive
- Fork it to the kaggle kernel

If you have a graphics card you can work locally by downloading the ipynb file.
For all the implementations you will do you will work with some subcategories of CIFAR-100 that correspond in a unique way to your team, putting in the notebook as team_seed the number of your team.
In the notebook you will find in the section "Introduction and overview of the data set" the code that gives you the names of your classes as well as the relevant indexes that you must use in whatever implementation you do.
Start with notebook models and work on defining new models, either from scratch (transfer from scratch) or transfer learning. For each model include a brief description of its architecture and key features. Optimize by following the section "Improving performance with experiments" and / or additional optimizations.

Describe the optimization of your models, your choices and your conclusions.

Remarks on optimization:

- A model is optimized on its own, not on other models. For example, if you bring a state-of-the-art model with learning transfer, you will look at how much you can optimize it and not compare it to a very simple "from scratch" network. This does not mean that you can not make comparisons between models in tables and graphs (see next note).
- Optimization can refer to the performance in terms of metrics we use (correctness) in the control set but also to other properties related to education: required memory, number of parameters, training time, behavior in terms of overtraining etc. .K. You can show comparative results "before and after" of improving these quantity properties even for the same model, regardless of its absolute performance in terms of accuracy compared to other models.
- Try all the optimization features mentioned in the notebook.

The role of the different number of categories per group (20 to 80). A small number of categories generally means an easier problem but also less overall data and vice versa for a large number of categories. This obviously has an effect on runtime, performance and overtraining or not small and large networks.
We are interested in seeing what you achieve with 80 categories and tell us training times.
Notes on GPU performance. Here are the measurements we took for the run times per step (step) -lower is better- for the simple rugged and VGG16 of the notebook at Colab, Kaggle and a physical GeForce RTX 2080 Ti as well as their AI-Score -library ai-benchmark. The simple CNN has 128,420 trainable agents while the VGG16 has 14,765,988.

	Colab (Tesla T4)	Kaggle (Tesla P100)	RTX 2080 Ti
simple CNN (ms / step)	15	15	7
VGG16 (ms / step)	75	53	28
AI-Score	14248	20983	28368

Note the chosen cloudw. If you use a natural GPU, tell us its AI-Score using the ai-benchmark.

Additional questions. In addition to your previous reference to models, their optimization and performance, you can analyze the following topics if you have not already included them, always based on the experiments you performed:

- Transfer learning vs training from scratch (“from scratch ”)
- Effect of data augmentation
- Effect of the number of levels to be trained (fine-tuning) during the transfer of learning
- Effect of the number of data / classes on the performance of the model
- Effect of the learning rate (learning rate))
- Effect of the optimizer algorithm
- Effect of the batch size
- Effect of the image size (resize input)