The Flower Classification Challenge

This course focuses on the principles underlying deep learning. Therefore, the exercises typically ask you to re-implement known algorithms and consider somewhat simplified settings. Beyond knowledge of these foundations, a deep learning practitioner must also build up experience using the many powerful tools available in the real world. We organize this competition to give you the opportunity to start doing so.

Participation is optional and your submissions will not be graded.

Competing in the same teams as for the exercises is not mandatory. Each team can have a maximum of 3 students. Solo participation is also allowed. **Note that you will not be able to use identical team names as those used for the exercises.** The winners will be announced during the last flipped classroom session (28.01.2025) and will receive nothing but eternal glory, a certificate and a spot on the course webpage. To mimick a real-world setting, we give you a lot of freedom in how you tackle this challenge. However, with this freedom comes responsibility. So please play fair and make it easy for us to evaluate your submission. In doubt, please post your questions on the Competition thread in ILIAS, or contact us.

To get access to the code, use the following link: `https://classroom.github.com/a/BAGPaLJo`.

The Challenge:

In this challenge, you are to train a model to perform class prediction on a flower dataset. Specifically, given an image of a flower as input, your model must predict to which of 17 possible classes it belongs.

An example of the flower dataset

To encourage everybody to participate, we have two tracks in this competition:

- Fast networks, < 100k learnable parameters.
- Large networks, < 25M learnable parameters.

These two tracks shall be considered as separate competitions with separate winners. You may compete in either (or both) tracks.

You are allowed to implement any architecture, and manually or automatically tune the hyperparameters of the model. The following are just a few of the things you can experiment with:

- Learning rate and its optional scheduler
- Different optimizers and their hyperparameters
- Specialized model architectures
- Activation functions
- Regularization
- Data preprocessing
- Cross-validation to ensure you do not overfit to the validation set

Also, you do not have to shy away from more advanced techniques like warm-starting your model with weights from other pre-trained models, or self-supervised learning.[1] You are allowed to use all the scripts and tools you already know from the exercises. However, you are not limited to them.

Here are a few important topics concerning the competition:

- **Evaluation:**
  - The final performance will be measured in terms of classification accuracy of your model on an **unseen** test set. Assume we will head out and photograph some flowers.
  - To get a taste of how we shall evaluate your model, try running `evaluate_model.py`. The script loads your saved model from `models` folder and evaluates its accuracy on validation data loaded from the `dataset/test` folder.
  - To evaluate your model, we will populate this folder `dataset/test` with the unseen test data.
  - There will be a leaderboard on ILIAS's discussion forum, under the thread Competition, which will rank teams based on their classification accuracy. This will allow you to see how well your team is performing compared to others.
  - **You are allowed up to five submissions per track.** The result of your team's last submission will be considered as the final result for that track. Simply upload the link to the commit containing the model to be evaluated. We will evaluate your model and update the leaderboard accordingly after each submission.
- **Hardware:**
  - The fast networks can be trained on a CPU in a few minutes. The larger networks might require GPUs.
  - You may use any kind of hardware that is available to you. For example, Google Colab repeatedly offers a VM with a GPU for at most 12 hours at a time for free. We provide you a simple Colab Notebook to get started.
  - You are also free to use the pool computers.[2] You can use ssh to log in remotely to these computers as follows: `ssh yourpoolaccount@login.informatik.uni-freiburg.de`
    See `SetupGuideTFPool.pdf` in ILIAS for more info on how to use the pool computers.
    Also, see Pool-FAQ for further information.
- **Implementation Constraints:**
  - Your model should be written using PyTorch.
  - Do not modify the code inside `src/eval`. Your model will be evaluated using this code.
  - Other than that, you are free to extend/modify the baseline code given to you or write your own code from scratch.
- **Github Repository Constraints:**
  - **Keep your repository under 150 MB.**
  - Github does not allow files larger than 100MB to be tracked in repositories. **So if you are competing in the Large Networks track, make sure your trained model is under this limit.**
  - You are allowed to push only one trained model per track.
  - Your code has to be in the `main` branch before the deadline.
  - Number of pushes to the main branch is not limited.

As a starting point, we provide you with the following:

- `dataset.py` containing the code to download the original flower dataset and divide it into train/validation/test splits.[3] First thing you should do is run `python -m src.dataset`.
- `data_augmentations.py` containing two sample data augmentation pipelines.
- `training.py` containing code to train your model.
- `eval/evaluate.py` containing the code we shall use to evaluate your model. **Do not edit this file.**

[1]Frozen weights of pre-trained models count towards the limit of learnable parameters.

[2]Important: Your TF pool home directory disk quota is insufficient to install PyTorch. Please contact us as soon as possible to request additional storage space.

[3]You are free to consider different splits, use cross-validation, or even use all the data to train your final model(s).

- `main.py` containing the code to load the data, train, evaluate, and, optionally, save the model.
  You can run it from the root directory using `python -m src.main`.
- `cnn.py` containing `SampleModel`, as an example of what a (very basic) convolutional model looks like.
- A sample saved model in `src/models`. This file contains the weights of the `SampleModel` given above trained for 50 epochs using the default pipeline that is provided to you.
- `evaluate_model.py` which the organizers will use to evaluate your model.
  You can run it from the root directory using `python -m src.evaluate_model`.
- `requirements.txt` which contains the list of libraries required to run the given pipeline. **Please manually install torch and torchvision** (see `assignment_07.pdf` for instructions).

Your submission must include:

- **A fully functional training pipeline**. This must include the code for your model, data augmentations you use and the code for training the model.[4]
- `requirements.txt` updated with any additional libraries you use.
- **The file(s) with the saved weights of your trained model(s)**, similar to `models/sample_model`, in the `models` folder. The files must be saved as `fast_model` or `large_model`[5], depending on the track it belongs to.
- `submission.md` must be populated with the answers to the following questions:
  - The track(s) you are competing in.
  - The number of learnable parameters in your model(s). You can find this out using `torchsummary.summary`.
  - A *brief* description of your approach to the problem.
  - Command to run to train your model(s) from scratch with your hyperparameter settings and data augmentations. For example:
    ```
    python -m src.main --model SampleModel --epochs 50 --data-augmentation
    resize_to_64x64 --use-all-data-to-train
    ```
    You're free to edit `main.py` however you please to make this work (E.g., to accept hyper-parameters as options, or hard-code them). You can also add new files, if you wish. **We must be able to train your model by running a single command.**
  - Command to run to evaluate your saved model(s) with your data augmentation pipeline.
    For example:
    ```
    python -m src.evaluate_model -m SampleModel -p 'sample_model' -D 'dataset/test'
    -d 'resize_to_64x64' -v 'INFO'
    ```
    Again, you're free to edit `evaluate_model.py` however you want, **but ultimately, evaluation MUST be done by invoking `eval_model(...)` function in `eval/evaluate.py`. We must also be able to evaluate your model by running a single command.**
- *sample_submission.zip* contains a sample of what a submission must look like.

This project is due on 22.01.2025 23:59 CET.

[4]If you use model warm-starting or other such methods, that should also be included in this pipeline. We must be able to run your code and train your model from scratch, as you did.

[5]Github has a limit of 100MB for tracked files. Your `large_model` must hence be smaller than 100MB.