

## 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. The winners will be announced during the last flipped classroom session (03.02.2026) 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/ZsQvIazD>.

### 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 parameters.
- Large networks, < 25M 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.<sup>1</sup> 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 test data loaded from the `dataset/test` folder.
- To evaluate your model, we will populate this folder `dataset/test` with the unseen test data.
- For each submission, there can only be a **single** model (i.e., a single file) saved in the `models` folder.
- The model and augmentation intended for evaluation must be defined by `evaluation_model` and `evaluation_augmentation` variables in `cnn.py` and `data_augmentations.py`, respectively.
- To make a submission to either track, you should push your changes to the `fast` or `large` branch, corresponding to the track(s) you are participating in.
- You are allowed **up to five submissions per track**.
- You are allowed **only one submission every 24 hours per track**. You should plan your submissions accordingly. Avoid last-minute submissions.
- There is a **leaderboard** webpage, which will rank teams based on their classification accuracy. This will allow you to see how well your team is performing compared to others.
- The leaderboard results are preliminary; the final results will be announced after the deadline.
- You are **not** allowed to collect/use additional data for training. This rule applies to custom pre-trained models as well (i.e., pre-trained by you). You are not required to enforce this rule on publically available pre-trained models.
- After the deadline, all submissions will be evaluated to ascertain compliance with the rules.
- We reserve the right to disqualify any team that violates the rules or engages in unfair practices.

- **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 like Google Colab, Kaggle, etc.
- For convenience, we have provided a **Google Colab notebook** that you can use to train your model. Feel free to change it to your needs.

- **Implementation Constraints:**

- Your model should be written using PyTorch.
- Your code must be compatible with `Python>=3.12.3`.
- You are allowed to use additional dependencies, but you must keep the existing packages in `requirements.txt` with the same specified versions.
- You are, obviously, free to use any GPU(s) you have access to. However, your code must support running on CPU.
- Your environment during evaluation will be setup using `pip install -r requirements.txt --extra-index-url https://download.pytorch.org/whl/cpu`. Make sure that your `requirements.txt` works with this command. **We will not be able to evaluate your submission if it does not work with this command.**
- Your environment must be deterministic (as much as possible). Please refer to lines 15-20 in `src/evaluate_model.py` for an example of how to do this.
- The model and augmentation intended for evaluation must be defined by `evaluation_model` and `evaluation_augmentation` variables in `cnn.py` and `data_augmentations.py`, respectively.
- Any modification to the code inside `src/eval` or `src/evaluate_model.py` will be overwritten during evaluation.

---

<sup>1</sup>Frozen weights of pre-trained models count towards the limit of learnable parameters.

- 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:**
  - 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 100MB.**

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.<sup>2</sup> 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.**
- `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 model saved as `models/sample_model`. 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.

Your submissions 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.<sup>3</sup>
- `requirements.txt` updated with any additional libraries you use.
- **The file with the saved weights of your trained model**, similar to `models/sample_model`.
- `submission.md`, submitted to the `main` branch, must be populated with the answers to the following questions:
  - A *brief* description of your approach to the problem.
  - Command to run to train your model from scratch with your hyperparameter settings and data augmentations. There is an example in `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.**

The competition's deadline is **28.01.2026 23:59 CET**.

---

<sup>2</sup>You are free to consider different splits, use cross-validation, or even use all the data to train your final model(s).

<sup>3</sup>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.