The final project is part of the oral exam, which means you are **not allowed to work in groups**. The purpose of this project is for you to you get hands-on experience on most topics of the course and to show that you can present and explain the results of your work. To get access to the template please use the following github classroom link
`https://classroom.github.com/a/dRo0fptr`.[1]

The first 15 minutes of the oral exam will be about your project. You will first present your approach and the results, and then we will discuss these together. To give the project presentation some structure, you will have to prepare a few slides. Your slides should consist of a motivation slide, slides detailing your approach (2-3) as well as slides for your results (2-3). *Don't go overboard with your slides.* They are intended to make your presentation coherent, but your presentation should only be **7-8 minutes** to still have some time for discussions. It is important that *your evaluation builds the basis for discussion and* ***scientifically analyzes*** *which are the important aspects and characteristics of your approach*—you should present your findings in a convincing manner.

Optimization of a Convolutional Neural Network

Your task is to automatically improve and analyze the performance of a neural network for a *flower classification*[2] dataset. How you improve the performance of the given network is up to you **as long as it is by means of AutoML**. In addition, let us assume that you would like to deploy your model to a system with some hardware constraints (e.g. model size) or the application has some further fine-grained constraints on precision (e.g., in a medical application). Therefore, your AutoML system has to also adhere to constraints provided by users.

In the end, you should convince us that you indeed improved the performance of the network when compared to the default approach. To this end, you could consider one or several of the following:

- Apply HPO to obtain a well-performing hyperparameter configuration (e.g., BO or EAs);
- Apply NAS (e.g., BOHB or DARTS) to improve the architecture of the network;
- Extend the configuration space to cover preprocessing, data augmentation and regularization;
- Apply one or several of the speedup techniques for HPO/NAS;
- Apply multi-objective optimization;
- Apply meta-learning, such as algorithm selection or warmstarting, to improve the performance;
- Apply a learning to learn approach to learn how to optimize the network;
- Determine the importance of the algorithm's hyperparameters;

Please note that you do not have to apply all of these methods – pick the ones that you think are most appropriate. To evaluate your approach please choose the way you evaluate well; you could consider the following:

- Measure and compare against the default performance of the given network;
- Plot a confusion matrix;
- Plot the performance of your AutoML approach over time;
- Apply a statistical test.

[1] This template prohibits groups.
[2] `http://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html`

You are allowed to use all scripts and tools you already know from the exercises; however, you are not limited to them. Overall, you should respect the following constraints:

- **Metric:**
  - The final performance has to be measured in terms of top-3 classification accuracy on a validation set. **However, your found configurations have to satisfy some constraints, such as, e.g. have a maximal network size of $2{\cdot}10^7$ and a precision of at least 0.39 (at least on validation set).**
- **Experimental Constraints:**
  - Your code for making design decisions should run no longer than 86 400 seconds (without additional validation) on a single machine.
  - Training a network can take at most 50 epochs.
  - You can use any kind of hardware that is available to you. For example, you could also consider using Google Colab (which repeatedly offers a VM with a GPU for at most 12h for free) or Amazon SageMaker (which offers quite some resources for free if you are a first-time customer). *Don't forget to state in your slides what kind of hardware you used!*
- **Implementation Constraints:**
  - You can freely extend the baseline implementation provided to you. However, your search space should always include the given default network.
  - You are not allowed to use manual tuning to achieve better performance. All improved design decisions have to be made (somehow) automatically.
  - You need to provide a script "AutoML.py" running your optimizer that accepts at least the following arguments:
    `--constraint_max_model_size [int]`
    `--constraint_min_precision [float]`
- **Grading Guidelines:**
  - We grade your projects based on: sound motivation for the chosen approach, achieved performance, convincing presentation, scientific workflow, reproducible results and answers to our questions.
  - We expect that you find at least a configuration that has a top-3 accuracy ≥ 0.8.
  - To earn a 2.0 as a grade, we expect at least that:
    - You apply the correct approaches and tools you learned about in the lecture.
    - Your approach does not violate different constraints on network sizes (in particular, $2 \cdot 10^7, 5 \cdot 10^7, 1 \cdot 10^8$).[3]
  - To earn a grade better than 2.0, we expect at least that:
    - You have your own ideas on how to improve an approach from the lectures. For example two years ago, a student combined hyperparameter importance and BOHB in an interesting way and showed that it improved the overall performance.
    - Your found solutions adhere to the network size constraint and different constraints on minimal precision (in particular, 0.39, 0.40, 0.42, at least on validation set).

As a starting point, we provide a repository containing:

[3] If these constraints are too easy for your implementations, you can also use harder constraints (i.e., constraints on small network sizes). In any case, we might assess your submission with other network sizes as constraints.

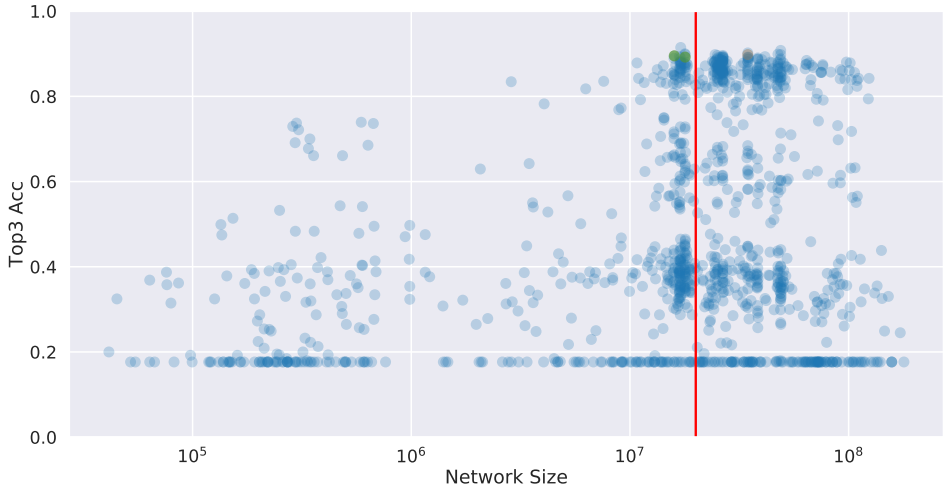

Figure 1: Evaluated configurations on the configuration space with a configuration budget 10 times larger than your allowed budget. The y-axis gives the top-3 accuracy in percent and the x-axis the network size in terms of the number of parameters of each model. The red line shows the network-size constraint of $2 \cdot 10^7$. Orange dots adhere only to the precision constraint of 0.39. The green dots adhere to both constraints.

- A cropped and downsized version of the original flower classification dataset. The images are only of size 16×16.
- A default parameterized network to optimize (written in pytorch)
- An example script to show you how to train and evaluate a network based on the default configuration
- An example script that shows you how to use BOHB in SMAC.

This project is due on September 13th 2021. Students should submit their PDF for the presentation to `biedenka@cs.uni-freiburg.de` (Freiburg) / `deng@tnt.uni-hannover.de` (Hannover). **No teams are allowed for the final project.**