General Setup The project can be worked on in **groups of up to 3 students**. The purpose of this project is for you to get hands-on experience on topics from the lecture and to show that you can present and explain the results of your work in a scientific manner. The projects are intended to give you the flexibility to explore AutoML solutions to interesting problems. To get access to the project template repository please use the following GitHub link:
`https://classroom.github.com/a/T_Fzxg5j`

Exam "AutoML Student Conference" On the 4th of September we will organize a virtual conference on `https://app.gather.town/` jointly with all students from Freiburg and Hannover. For the conference your group should create a poster and will have to present their results to their respective professor. The presentation should be 5 minutes and is followed by a Q&A. While not giving your presentation you are encouraged to visit other posters and exchange ideas with your peers. After the conference you have time (see the timeline below) to incorporate feedback to improve your poster before you have to submit the final version.

Timeline

- Implement your solution for a project (Due 25.08.23)
- Create a poster to explain your approach & result (Due 25.08.23)
- Review posters of other teams (25.08.23 - 01.09.23)
- Present the poster in a virtual poster session (**September 4th, 14:00 - 18:00**)
- Revise your poster based on the feedback you received (04.09.23 - 08.09.23)

Grading

Your grade is determined by:

- your groups approach & results;
- your involvement in the poster presentation & discussion;
- the reviews you provide.

Ensure that you have:

- worked scientifically;
- equally contributed to the project;
- clean and well-documented code;
- a well structured poster;
- a clear message and insights;
- good illustrations, figures and plots;
- a comprehensive short pitch;
- a reasonable summary of the presented approach;
- a list of strong and weak points of the poster;
- provided constructive feedback.

Projects

Biologists have spent decades trying to identify plant species in the wild. Can machine learning leverage such curated datasets to help the current and future biologist's tasks? They had previously outsourced this task but their compute bill was too high, though the solution found was a good working solution. These biologists want to hire your team to improve on the previous solution using a CNN, at a lower total compute required. To this end, you can use all your AutoML knowledge that you gained during the semester.

The overall task is to automatically improve and analyze the performance of a convolutional neural network intended to have high accuracy for classification tasks. The dataset for this task is concerned with accurately labeling invasive plant species. Since optimization of neural networks for image tasks can be computationally expensive you should consider one of the following options when solving the task[1]:

Option I: Multi-fidelity Optimization of a Convolutional Neural Network

For this problem setup, your goal is to leverage multi-fidelity optimization to find high-performing configurations within a small AutoML budget. To get you up and running, the template repository comes with:

- example code based on SMAC3 that allows you to run a multi-fidelity optimizer similar in behavior to BOHB;
- all necessary links and code to load the data;
- an example search space defining the CNN and its training setup;
- a simplified version of the dataset for ease of use;
- baseline(s) (including a default configuration) to let you easily judge the quality of your approach.

In the end, you should convince us that you indeed improved the performance of the network when compared to performance thresholds reached by baseline configurations. To do so, your possible solution approaches could

- customizing the search space by
  - separately consider architectural and other hyperparameters;
  - jointly optimize architectural as well as other hyperparameters;
  - incrementally optimize parts of the search space
- apply NAS methods on top of multi-fidelity approaches;
- explore the impact of different fidelities;
- automatically determine the best fidelity to use;
- optimize the hyper-hyperparameters of a multi-fidelity method;

Please note this list is not exhaustive, nor do you have to apply all of these methods. Pick and choose the approaches that you have learned in the lecture that you think are most appropriate, but the overall theme has to incorporate multi-fidelity optimization for this task.

You have to show that you can evaluate your approach scientifically. Thus you should carefully consider how you present your results, report how many seeds you evaluate, and show the performance variance when comparing algorithms. As you have to present your results in a poster

[1]Note: While these are presented as separate options, you are free to solve the overarching tasks by combining the approaches.

presentation, pay close attention to the plots that you produce. They have to be easy to understand, straightforward to read and interpret.

You are allowed to use all scripts and tools that you know from the lecture and exercises as well as all template code we provide, but you are not limited to them. Overall, you have to respect the following constraints:

- maximum runtime of 6 hours for the final AutoML run (excluding validating the results on the test set);
- training a model uses at most 20 epochs;
- k-fold cross-validation is used (with a max of k=3);
- you only use the *balanced* version of the dataset (the example code shows you how to do so);
- the maximal image size you use is 32x32;
- test performance has to be reported with the above constraints;
- design decisions have to be made automatically;
- you can use any kind of hardware that is available to you (e.g. Kaggle[2] or Colab[3]) but you have to mention which hardware you used clearly.

Grading guidelines for the coding portion of the exam For the coding part we expect the following.

- Your code is well documented;
- You adhered to the reproducibility[4] checklist;
- It is straight forward to find a configuration that reaches a test accuracy higher than 60%;
- An optimized configuration reaches a test accuracy over 65%;
- You should have implemented your own idea and not just (re-)using code and tools

[2] `https://www.kaggle.com/code`
[3] `https://colab.research.google.com/`
[4] `https://drive.google.com/file/d/1Ztits2BG7_BvsEshoGhhx4wXMTTOZ_86/view?usp=sharing`

Option II: Meta-Learning – Using Prior Data to Warmstart Optimization

Under this problem setup, your goal is to leverage existing configuration data to make the overall optimization process as efficient as possible. The template repository provides:

- example code that demonstrates how to warmstart the AutoML tool SMAC;
- a dataset from which to warmstart; This dataset consists of the bottom 80% of configurations of 5 seeded multi-fidelity runs, only taking the configurations evaluated at the maximum epoch fidelity.
- code to setup the already preprocessed data;

In the end, you should convince us that you indeed improved the performance of the network when compared to any of the points in the warmstarting dataset and the hidden top 20%. To do so, your possible solution approaches could

- configure the optimizer with data from the warmstarting dataset;
- shrink the search space based on the meta-dataset;
- combine your approach with multi-fidelity methods;
- explore the impact of different hyperparameters to prune the configuration space;
- built your own surrogate function based on the meta-learning data;
- optimize the hyperparameters of your method;

Please note this list is not exhaustive, nor do you have to apply all of these methods. Pick and choose the approaches that you have learned in the lecture that you think are most appropriate, but the overall theme has to focus on meta-learning by warmstating the optimization for this task.

You have to show that you can evaluate your approach scientifically. Thus you should carefully consider how you present your results, report how many seeds you evaluate, and show the performance variance when comparing algorithms. As you have to present your results in a poster presentation, pay close attention to the plots that you produce. They have to be easy to understand, straightforward to read and interpret.

You are allowed to use all scripts and tools that you know from the lecture and exercises as well as all template code we provide, but you are not limited to them. Overall, you have to respect the following constraints:

- a maximum runtime of 6 hours for the final AutoML run;
- training a model uses at most 20 epochs;
- k-fold cross-validation is used with $k <= 3$; [5]
- you only use the *balanced* version of the dataset (the example code shows you how to do so);
- the image size you are required to use is 32x32;
- test performance has to be reported with the above constraints;
- design decisions have to be made automatically;
- you can use any kind of hardware that is available to you (e.g. Kaggle[6] or Colab[7]) but you have to mention which hardware you used clearly.

[5] You may use less but beware of over-fitting
[6] `https://www.kaggle.com/code`
[7] `https://colab.research.google.com/`

Grading guidelines for the coding portion of the exam For the coding part we expect the following.

- Your code is well documented;
- You adhered to the reproducibility[8] checklist;
- It is straight forward to find a configuration that obtained at least 60% test accuracy;
- A well optimized configuration obtained at least 65% test accuracy with your method;
- You demonstrated that your method outperforms the baseline which does not utilize warstarting;
- You implemented your own idea beyond simply using the provided code

[8] `https://drive.google.com/file/d/1Ztits2BG7_BvsEshoGhhx4wXMTTOZ_86/view?usp=sharing`

Option III: Multi-Objective Optimization of a CNN

Under this problem setup, your goal is to find the configuration that yields the best accuracy while also optimizing other objectives that make the resulting networks as efficient as possible. The template repository provides:

- example code to run a multi-objective optimizer (see also `https://automl.github.io/SMAC3/main/examples/3_multi_objective/index.html`) to optimize for accuracy and other cost objectives such as training time and MACs[9] used.
- an example scalarization weight for the objectives
- code to setup the already preprocessed data and run the baseline pipeline
- a default configuration that specifies a baseline performance to beat

In the end, you should convince us that you indeed improved the performance of the network when compared to the Pareto front that the default configuration may be a part of. That is, under the given budget, finding a solution that dominates the test set performance of the default configuration is the minimum goal. To do so, your possible solution approaches could

- customize the search space by
  - separately considering architectural and other hyperparameters;
  - jointly optimize architectural as well as other hyperparameters;
  - incrementally optimize parts of the search space
- apply multi-objective NAS methods;
- explore the impact quality) of different objective[10] in terms of optimization cost and solution quality;
- explore the effect of scalarization type and the weights
- combine multi-objective and multi-fidelity optimization for efficiency;
- optimize the hyperparameters of your method;

Please note this list is not exhaustive, nor do you have to apply all of these methods. Pick and choose the approaches that you have learned in the lecture that you think is most appropriate, but the overall theme has to incorporate multi-objective optimization for this task, as such you have to present pareto fronts and explain relevant intuition and methods for selecting the final configuration.

You have to show that you can evaluate your approach scientifically. Thus you should carefully consider how you present your results, report how many seeds you evaluate, and show the performance variance when comparing algorithms. Needless to say, the evaluation protocol and appropriate use of the validation and test sets should be done carefully and reported. As you have to present your results in a poster presentation, pay close attention to the plots that you produce. They have to be easy to understand, and straightforward to read[11] and interpret.

You are allowed to use all scripts and tools that you know from the lecture and exercises as well as all template code we provide, but you are not limited to them. Any code snippet/script generated from LLMs should be credited as an in-line comment or docstring (Github Copilot usage could be acknowledged in the README of the project code). Overall, you have to respect the following constraints:

[9] `https://en.wikipedia.org/wiki/Multiply-accumulate_operation`

[10] Using all the objectives given in the example is not necessary and neither is the list of objectives given a restriction. Any number and kind of objectives may be used, as long as justification is provided with the final solution dominating the default configuration on the objectives provided.

[11] Note that plots should be legible when formatted for poster inclusion.

- maximum runtime of 8 **hours** for the final AutoML run (excluding validating the results on the test set);
- training a model uses at most 20 epochs;
- k-fold cross-validation is used with $k = 3$;
- you only use the *balanced* version of the dataset (the example code shows you how to do so);
- the image size you are required to use is 32x32;
- test performance has to be reported with the above constraints;
- design decisions have to be made automatically;
- you can use any kind of hardware that is available to you (e.g. Kaggle[12] or Colab[13]) but you have to mention which hardware you used clearly.

Grading guidelines for the coding portion of the exam For the coding part we expect the following.

- Your code is well documented;
- You adhered to the reproducibility[14] checklist;
- It is straight forward to find a configuration that reaches a test accuracy of 60% without being *significantly* worse on the other objectives (cost, MACs) than the baseline;
- Your well optimized configuration dominates the default configuration on all 3 objectives on the test set;
- The found optimized configuration should lie on a Pareto front that achieves a Hypervolume measure >13200 for a reference point (error=0.95, cost=21600, macs=1000);
- The optimized configuration achieves a test accuracy of over 65% while dominating the default configuration;
- You implemented your own idea and not just (re-)using code and tools

[12] `https://www.kaggle.com/code`
[13] `https://colab.research.google.com/`
[14] `https://drive.google.com/file/d/1Ztits2BG7_BvsEshoGhhx4wXMTTOZ_86/view?usp=sharing`

Grading

Approach & Implementation

- **Overall at most 50 points**
- Points will be reduced if
  - Only reusing existing code without implementation of novel new idea beyond the lecture: -15 points
  - Accuracy less than 60% : -15 points
  - Accuracy less than 65%: -10 points
    (not accounted for if "Accuracy less than 60%" already applied)
  - No code documentation (docstrings, ReadMe and so on): -5 points
  - Reproducibility list not taken into account: -10 points
  - We cannot run your code easily: -10 points
- Bonus points
  - for novel and innovative approaches: +20 points

Reviewing of Others Posters

- **Overall at most 10 points**
- Points will be reduced if
  - Feedback is not constructive: -5 points
  - Feedback in bullet points without full sentences: -5 points

Poster Presentation

- **Overall at most 40 points**
- Points will be reduced if
  - Idea is not convincing or too trivial: -10 points
  - Poster is not well structured: -10 points
  - Figures are too small: -5 points
  - No figures for illustrating the approach: -5 points
  - Presentations longer than 6min: -5 points
  - No clear message: -5 points
  - No clear insights from the results: -5 points