General constraints for code submissions Please adhere to these rules to make our and your life easier! We will deduct points if your solution does not fulfill the following:

- If not stated otherwise, we will use exclusively Python 3.6.
- If not stated otherwise, we expect a Python script, which we will invoke exactly as stated on the exercise sheet.
- Your solution exactly returns the required output (neither less nor more) – you can implement a `--verbose` option to increase the verbosity level for developing.
- Add comments and docstrings, so we can understand your solution.
- (If applicable) The `README` describes how to install requirements or provides addition information.
- (If applicable) Add required additional packages to `requirements.txt`. Explain in your `README` what this package does, why you use that package and provide a link to it's documentation or GitHub page.
- (If applicable) All prepared unittests have to pass.
- (If applicable) You can (and sometimes have to) reuse code from previous exercises.

Now that you have learned about one shot model and differentiable architecture search. you will use these concepts to implement a specialized NAS method.

1. **Differentiable Architecture Search** [14 points]
   In this second part of the exercise you will run DARTS for finding an optimal CNN architecture on MNIST. The search model is defined in `model_search.py`. It contains three stacked cells: reduction-normal-reduction. The architecture search problem is to find an optimal operation out of $\mathcal{O} = \{conv\_3x3, max\_pool\_3x3, avg\_pool\_3x3, Identity\}$ in each edge of these cells. The number of intermediate nodes is 2.

   (a) In order to create the architecture continuous relaxation, we need to define a *MixedOp*, which is a convex combination of the operations outputs connecting two nodes in the cells. It is defined as follows: [6pt.]

   $$x^{(j)} = \sum_{i<j} \tilde{o}^{(i,j)}(x^{(i)}) = \sum_{i<j} \sum_{o \in \mathcal{O}} \frac{e^{\alpha_o^{(i,j)}}}{\sum_{o' \in \mathcal{O}} e^{\alpha_{o'}^{(i,j)}}} o(x^{(i)})$$

   Based on this formulation you have to fill in `model_search.py` in order to compute the output tensor $x^{(j)}$ of the MixedOp.

   Your implementation should satisfy the test in `test_darts_mixed_op.py`.

   (b) Having defined the search model, we now need to run the DARTS optimization loop (Algorithmi 1, Slide 3 of Topic 2). We will use the first-order approximation. Your task is going to be only to write the lines of code that compute the architectural updates in `train_search.py`. [6pt.]

   Your implementation should satisfy the test in `test_darts_architect.py`.

   (c) Afterwards, you should be able to run `python train_search.py` without any errors. This will conduct the search for 5 epochs and write in a directory named `logs/` the output logs and a file with the optimal architecture configuration. [2pt.]

   In the end, to generate a visualization of the found cells run `python visualize.py`. This should generate two `.pdf` files named `normal.pdf` and `reduction.pdf`. Push the contents written in `logs/` together with the two `.pdf` files generated by the visualization script to your github repository.

   NOTE: Running `train_search.py` on a GPU machine takes a few minutes. This might scale to more than 1h when running on a CPU machine.

2. **Code Style** [1 point]
   On every exercise sheet we will also make use of `pycodestyle`[1] to adhere to a common python standard. Your code will be automatically evaluated on every push and you will be informed if the test fails. To check it locally, first run `pip install pycodestyle` and then run `pycodestyle --max-line-length=120 src/` to check your source file folder. Alternatively run `make checkstyle`

[1]former pep8

This assignment is due on 19.01.22 (14:00). Submit your solution for the tasks by uploading your code, PDF files and log file to your groups Github repository. The PDF has to include the name of the submitter(s).