Training Free Neural Architecture Search

TAs: Niccolò Cavagnero, Luca Robbiano

email: niccolo.cavagnero@polito.it, luca.robbiano@polito.it

OVERVIEW

The advent of machine learning represents one of the major turning points of our development as a species, often compared with the discovery of fire or electricity. Indeed, standard approaches for the design of deep neural networks heavily rely on trial-and-error procedures to tune the hyperparameters. However, this process, which highly depends on the experience of the designer, is extremely time-consuming, and, most importantly, cannot guarantee the optimality of the final design. An approach to solve this issue is to implement an algorithm of Neural Architectures Search (NAS) that identifies the most performing architecture within a pool of available options. Yet, standard NAS approaches are based on algorithms that are required to train all the tested candidates to assess their performance. Therefore, NAS pipelines are usually very expensive in terms of time and computational resources. To solve this problem, the research community is recently focusing on the adoption of proxies to score the architectures at initialization, thus avoiding the heavy and slow training phase and achieving consistent speedups even with limited hardware resources. In this project you will get familiar with the NAS literature implementing the first training-free NAS algorithm developed in the community.

GOALS

The goals of this project are the following:

- Acquisition of the capability of understanding, implementing and evaluating a scientific paper.
- Getting familiar with NAS literature, understanding points of strength and weaknesses of different methods.
- Development of a variation of the original model.

STEPS

1. Implement the method used by “Neural Architecture Search without Training” to score networks at initialization. You can find a preprint of the paper here: http://arxiv.org/abs/2006.04647
   Note: Be sure to read the final version of this paper, v3. The original preprint (v1) significantly differs from the final published version.
2. Implement the NASWOT search algorithm and:
   a. Run experiments based on the training free score on the NATS search space.
   b. Run the same experiments picking the optimal network at each step. This means using the validation accuracy of the networks.
3. Propose a variation and implement it. Run the same experiments with your variation. It is strongly recommended to ask one of the TAs before implementing it, to be sure it is feasible and can be accepted as a proper variation.
4. Deliver working PyTorch scripts for all the required steps.
5. Write a complete PDF report. The report must contain: a brief introduction, a related works section, a methodological section for describing the algorithms you are going to use, an experimental section with all the results and discussion. End the report with a brief conclusion.

Note:

- Each experiment must be repeated at least 30 times, reporting averages, standard deviations and time required for the search. See the “Before starting” paragraph to avoid waste of computational time.
- The networks must not be re-trained, for both time and reproducibility reasons, only the accuracies reported in the benchmark must be taken into account.
- All the three vision datasets must be considered in the experiments.

SUGGESTED VARIATIONS

Variation 1

Implementation and evaluation of a different search algorithm. We suggest evolution-based methods for their simplicity.

Variation 2

Implementation and evaluation of different training-free metrics (at least two). See below for additional resources.

DETAILS

Details and useful information for completing the project follows.

Before starting

In order to ensure reproducibility and to spare time, we **strongly** suggest computing ahead-of-time the training-free metrics for all the possible architectures, for example saving them in a .csv file. Since you are required to report the time for each experiment, remember to store the time required to compute the metrics for each network. Once all the scores have been computed, you won't need a GPU anymore to run your experiments.

The datasets

NATS-Bench is a dataset of trained networks for the development of NAS algorithms, which contains approximately 15k architectures trained and evaluated on three different Vision datasets: CIFAR10, CIFAR100 and ImageNet16-120 (which is ImageNet scaled to a 16x16 resolution and reduced to 120 classes). The datasets fully explores two search spaces: topological and size. You only need to run experiments on the topological search space. Each network is characterized by a computational cell stacked multiple times to form the whole architecture. The cells are represented by a DAG, where edges represent the operators and each node is the sum of all the feature maps transformed by the edges pointing to the node. The number of edges is not constrained, while the maximum number of nodes is set to 4. The possible operators are : conv3x3, conv1x1, avgpool3x3, skip connection and zeroize (no_op).

Note: while reading papers and code, you might find that this dataset is often referenced as NAS-Bench-201. While NATS and NAS-Bench-201 are not exactly the same thing, they can be considered equivalent for the scope of this project.

Examples of questions you should be able to answer at the end of the project:

- What are the main limitations of a standard NAS pipeline?
- Which are the main paradigms in terms of search algorithms?
- Which are the points of strength and the weaknesses of such paradigms?
- What is a search space? Which types of search space do exist?
- Given a genotype, how is it translated into a full architecture in cell-based NAS?
- Describe a given search algorithm (among the ones in the suggested papers).

Resources

NATS-Bench: Benchmarking NAS Algorithms for Architecture Topology and Size
http://arxiv.org/abs/2009.00437

Neural Architecture Search without Training
http://arxiv.org/abs/2006.04647

Python xautodl package
https://github.com/D-X-Y/AutoDL-Projects

```
    pip install xautodl
```

ImageNet16-120
Download with:

```
    wget 'https://www.dropbox.com/s/o2fg17ipz57nru1/?dl=1' -O \
        ImageNet16.tar.gz
```

PyTorch Dataset implementation:

```
    from xautodl.datasets.DownsampledImageNet import ImageNet16
```

NATS Bench
https://drive.google.com/file/d/17_saCsj_krKjlCBLOJEpNtzPXArMCqxU
It can be downloaded with the commands

```
    pip install gdown>=4.4.0
    gdown 17_saCsj_krKjlCBLOJEpNtzPXArMCqxU
```

or

```
    wget 'https://www.dropbox.com/s/pasubh1oghex3g9/?dl=1' -O \
        'NATS-tss-v1_0-3ffb9-simple.tar'
```

PyTorch implementation:
https://github.com/D-X-Y/NATS-Bench

```
    pip install nats_bench
```

Regularized Evolution for Image Classifier Architecture Search (useful for Variation 1)
https://arxiv.org/abs/1802.01548

Other training-free metrics (useful for Variation 2)
http://arxiv.org/abs/2101.08134