Differentiable Neural Architecture Search

Vasily Ivanov and Sergey Kastryulin (equal contribution)

[1] SkolTech, Moscow
va.ivanov@skoltech.ru
[2] SkolTech, Moscow
snk4tr@gmail.com

1 Problem statement

Tabular ML is a field of research that is long-term dominated by "boosting on trees" family of algorithms. Datasets in the field can significantly vary in size and structure. There is no stable and reusable neural solution that can beat tree boostings and random forests, though it is sometimes possible to get an improvement for particular dataset using ResNet or attention based solutions (Gorishniy et al. 2021). In this setting neural architecture search can be a prominent approach since it can give ANN-based model desired domain adaptation allowing to replace hand-made design of neural network for each domain by single design of the search space for the particular architecture.

2 Differentiable NAS

Differentiable neural architecture search is a family of algorithms that realize the search of architecture (or design of the direct acyclic graph of the model) in a way that the search procedure becomes differentiable and utilize gradient based methods.

3 NAS schemes

Apart from differentiable neural architecture search schemes there exists plethora of solutions since you can parameterize neural network as hyper-parameter, so available NAS algorithms include but not limited to:

1. Bayesian black box optimization
2. Different versions of random search in parameter space
3. RL-based architecture optimization
4. Evolutionary algorithms

We discard those in the scope of our task. It leaves us with those differential architecture search methods to optimize for model graph:

3.1 Differentiable Architecture Search - DARTS

Basically DARTS (Liu, Simonyan, and Yiming Yang 2018) is a way to optimize all variants of the layer in the same time. We replace layer or module we want to optimize with the softmax mixture of variants from the search space of the model. After optimization we select specific variant via simple argmax on the mixture weights.

P-DARTS is a progressive version of the same algorithm (Chen et al. 2019). Its main feature would be that we start optimization procedure with smaller number of cells to optimize and later increase it number to approximate perfect DAG solution. Main feature of this method would be relative speed of convergence.

PT-DARTS is a perturbation based modification of DARTS algorithm (Wang et al. 2021). Main idea is that we eliminate paths every step of algorithms to identify most important ones. After complete run on all edges we optimize for nodes. Output is pruned DAG of the model. In the wild this algorithm usually works better or on pair with other variants of DARTS.

3.2 Proxyless NAS

To optimize for architecture using this method (Cai, Zhu, and Han 2018) we first build hypergraph - graph of the model with all possible submodules. In order to optimize this large model we binarize so called architecture-weights in order to optimize for one selected path only and keep model size reasonable. This method allows to reduce memory footprint for the model because it optimize only one path in the graph and can lead to better results than DARTS in some cases. Though speed of convergence is comparably lower.

3.3 Stochastic Neural Architecture Search - SNAS

This method (Xie et al. 2018) is based on random selection of subsets of nodes in adjacency matrix that describes hypergraph. After optimization remaining edges and vertices is a desired graph. The method is rather hard to implement and expected to converge later than other methods.

3.4 Dirichlet Neural Architecture Search

This method treats weights of paths in hypergraph as random variables, modeled by Dirichlet distribution in the end to end manner. On some benchmarks it achieves state of the art results.

3.5 GDAS

GDAS (Dong and Yi Yang 2019) implements an alternative way to model a differentiable NAS. The authors claim the method to be superior compared with evolution- and RL-based methods. There are no direct comparisons and discussions about other differentiable NAS methods, which makes it hard to assess its competitive advantages. Nevertheless, the authors claim the method to be able to achieve prominent results while being remarkably sample efficient. The method is based on the fully differentiable optimization scheme where the potential operation candidates are sampled from a pre-defined discrete search space using Gumbel softmax trick (Gumbel 1954). Instead of using the whole DAG, GDAS samples one sub-graph at one training iteration, accelerating the searching procedure. The main search is performed to obtain so-called computation cells, which are combined with a pre-defined procedure to obtain a final network.

4 Tabular Architecture

We decided to keep to using mlp-resnet blocks for the search space of our model because these architectures show strong persistent performance (Gorishniy et al. 2021) on tabular datasets.

5 Tabular datasets

We decided to keep validation same as in the tabular paper (Gorishniy et al. 2021). For the starting experiments we are looking into single big classification dataset like those are used in NAS tabular papers just for the sake of usability. We went with Covertype, Higgs Small, Otto Group Products, Adult and Churn Modelling.

Dataset	Folder Name	Type	Num Features	Train Size
Covertype	covtype	multiclass	54	371847
Higgs Small	higgs-small	binclass	28	62751
Otto Group Products	otto	multiclass	93	39601
Adult	adult	binclass	7	26048
Churn Modelling	churn	binclass	10	6400

Table 1: Properties of used datasets

6 Algorithm Selection

Since tabular data samples are usually significantly smaller than for example images and model will surely fit modern GPUs with usable batch size, there is

no imminent need to minimize size of model memory footprint so I don't think we should go for the "pruned" random subgraph variants just for the sake of it. Performance-wise and from practical point of view literature suggests GDAS and PT-DARTS as a performing near SOTA approaches and since we don't need to reduce memory footprint we consider going with PT-DARTS as our first solution.
Later after serious consideration we abandoned this idea because we went with automl/NASLib repo and it became really hard to integrate new opt methods into experimental pipeline.
We compared DARTS, GDAS and DrNAS methods in our NAS experiments.

7 Experiments

7.1 Tabular Baseline

At first strong simple tree-based baseline for each dataset was used. We went with Random Forest algorithm as simplest to use and powerful model for the tabular data. In order to run this experiment you should run experiments/run_experiment_1.sh file from experiments folder in the repo. Resulting metrics will be in the /home/experiments/rf.pickle file.

7.2 Search space

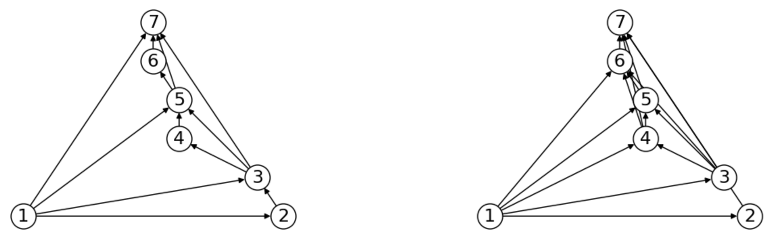
For differential NAS we had to create proper search space. In the field it is considered effective to use cell-based design for the search space. Two types of cells were tested - Darts cells (Liu, Simonyan, and Yiming Yang 2018) and ASR bench cell. Later performed better - most likely because of the less complicated structure and some connections being already dropped.

7.3 Layer candidates for the edges

We used these layers on the edges of cells:

1. Linear layer
2. Zero layer
3. ResNet MLP layer
4. MLP bottleneck layer
5. Transformer layer

Promising approach was the transformer layer but during experiments it appeared that transformer layer is basically too powerful - too hard to train and too easy to overfit, to use it on edges of hypergraph. This approach seems to work as standalone solution but it seems like it can't be combined with smaller capacity layers, because it is getting smallest possible weights, even with pretraining when running vs other operations. Also training process of the final architecture



(a) ASR bench basic cell (b) DARTS basic cell

Fig. 1: Basic cells for the NAS Search Space

often collapses and leads to bad local minima and requires very thorough approach. You can reproduce this experiment via experiments/run_experiment_2.sh Because of the significant amount of full-connected layers it seems that network is very sensitive to initialisation and learning rate schedule.

In order to recreate this experiment you should run experiments/run_experiment_3.sh file from experiments folder in the repo. For the ASR cell results you will need to run next experiment.

method	Covertype	Higgs	Otto	Adult	Churn Modelling
random forest	82.6992	71.3870	75.6948	84.2945	85.3500
Transformer_op	36.46	52.856	13.009	76.377	79.65
Darts_cell_op	-	52.856	72.697	76.377	79.65
ASR_cell_op	-	47.144	65.999	76.377	79.65

Table 2: Accuracy for the experiments with baseline rf, different cells, darts, transformers

7.4 Tabular NAS Algorithm Search

In order to recreate this experiment you should run experiments/run_experiment_4.sh file from experiments folder in the repo. According to the results DARTS provides solid solution and no approach here is guaranteed to work with custom edge operations and custom search space, though I'm not sure if it's due low quality of the library or algroithms are very unstable in general. Almost all benchmarks in these papers come from CIFAR10 and it doesn't give much hope for the generalisation of the approach.

algorithm	Covertype	Higgs	Otto	Adult	Churn Modelling
Darts	-	47.144	65.999	76.377	79.65
GDAS	-	47.144	26.05	76.384	79.65
DrNAS	-	52.856	57.482	76.377	79.65

Table 3: Accuracy for the different algorithms

8 Conclusions

Seems like for smaller tasks DARTS can still beat other approaches and Random Forest, though I couldn't reproduce best quality results because of the time pressure.

9 Discussion

Current state of the differentiable NAS procedures suggests them being unstable and the lib I based my research on - seems like really low quality with dozens of major bugs and simple constructions failing on every attempts of out of the box use. I only see the practical point in Tabular Nas in case of really profound tailoring of the search space. Otherwise edge operations move each other out of the balance and make network almost impossible to train.
I had to limit my experiments because of the deadline and repeatability issues within NASLIB.

References

Cai, Han, Ligeng Zhu, and Song Han (2018). "Proxylessnas: Direct neural architecture search on target task and hardware". In: *arXiv preprint arXiv:1812.00332*.

Chen, Xin et al. (2019). "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation". In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1294–1303.

Dong, Xuanyi and Yi Yang (2019). "Searching for a robust neural architecture in four gpu hours". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1761–1770.

Gorishniy, Yury et al. (2021). "Revisiting deep learning models for tabular data". In: *Advances in Neural Information Processing Systems* 34, pp. 18932–18943.

Gumbel, E.J. (1954). *Statistical Theory of Extreme Values and Some Practical Applications: A Series of Lectures*. Applied mathematics series. U.S. Government Printing Office. URL: `https://books.google.ru/books?id=SNpJAAAAMAAJ`.

Liu, Hanxiao, Karen Simonyan, and Yiming Yang (2018). "Darts: Differentiable architecture search". In: *arXiv preprint arXiv:1806.09055*.

Wang, Ruochen et al. (2021). "Rethinking architecture selection in differentiable NAS". In: *arXiv preprint arXiv:2108.04392*.

Xie, Sirui et al. (2018). "SNAS: stochastic neural architecture search". In: *arXiv preprint arXiv:1812.09926*.