THE MAXIMUM A POSTERIOR ESTIMATION OF DARTS

Jun-Liang Lin, Yi-Lin Sung*, Cheng-Yao Hong*, Han-Hung Lee, Tyng-Luh Liu*

Institute of Information Science, Academia Sinica

ABSTRACT

The DARTS approach manifests the advantages of relaxing the discrete problem of network architecture search (NAS) to the continuous domain such that network weights and architecture parameters can be optimized properly. However, it falls short in providing a justifiable and reliable solution for deciding the target architecture. In particular, the design choice of a certain operation at each layer/edge is determined without considering the distribution of operations over the overall architecture or even the neighboring layers. Our method explores such dependencies from the viewpoint of maximum a posterior (MAP) estimation. The consideration takes account of both local and global information by learning transition probabilities of network operations while enabling a greedy scheme to uncover a MAP estimate of optimal target architecture. The experiments show that our method achieves state-of-the-art results on popular benchmark datasets and also can be conveniently plugged into DARTS-related techniques to boost their performance. Our code is available at https://github.com/MAP-DARTS/MAP-DARTS.

***Index Terms*—** Neural Architecture Search, AutoML, Computer Vision, Deep Learning

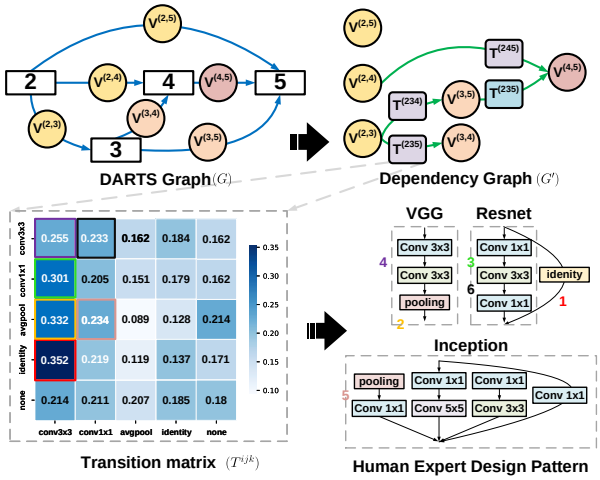

Fig. 1: Illustration of a cell-based search space and the corresponding directed acyclic graph (DAG): The operation from the intermediate node i to node j can be seen as a vertex $v^{(i,j)}$in the dependency graph G'. Note that each node in G is plotted as a box, while in G' as a circle. The transition matrix contains the knowledge that some operations have higher probability to co-occur, such as the human-design models.

1. INTRODUCTION

Neural architecture search (NAS) has attracted much attention as a new paradigm for automatic design of network models. The exploration has mostly centered around computer vision tasks, including image classification [1], object detection [2] and semantic segmentation [3].

A common drawback of most existing NAS algorithms is that the performances of their resulting networks could vary considerably and it indeed takes several runs to uncover a good architecture. Such inconsistency could be greatly mitigated if a NAS technique is blessed with a rough idea of knowing where to look for promising candidates in the search space. Radosavovic *et al*. [4] propose designing network design space to discover a subspace with a high concentration of good models among the original layer-based space. They propose to refine large design space by manually invoking restrictions to improve the space quality, such as sharing bottleneck ratio and increasing stage width. However, it would become laborious to manually find intuitive induction to pose restrictions in cell-based search space, due to its complicated topology compared to layer-based search space. We instead propose to focus on constructing proper DARTS priors to guide the automatic network design.

Motivated by the observation that there exists clear patterns between consecutive operations in human-designed models [5, 6], our approach explicitly encodes the operation-level priors in the training. Specifically, we consider an arbitrary pair of adjacent directed edges (in a one-shot supernet) as a "transition" from the operation on the leading edge to the which on the following edge. Therefore, we construct the "dependency graph", in which we can easily incorporate operations prior information. We illustrate our idea in Figure 1. The nodes of the dependency graph correspond to architecture weights on the original supernet and the edges represent transition matrices, each of which encodes the dependencies between operations of consecutive nodes.

*Same contributions.

We introduce a new NAS formulation that the maximum a posteriori (MAP) reasoning is incorporated into both the optimization and search phases. It works on a dependency graph of network operations, which is constructed from the coupled DARTS graph. The MAP estimate is obtained by locally evaluating a node-wise posterior probability for choosing a specific operation and by globally estimating the likelihood of a resulting architecture. The design leads to a greedy k-best algorithm, which implicitly relies on message passing driven by the learned transition probabilities. A novel use of our method is to leverage the above-described priors to regularize the learning of transition matrices, which greatly improves the reliability of our NAS technique. The posterior probabilities from the MAP formulation can be exploited to design a top-k greedy algorithm to reliably and effectively uncover the target architecture network. As a result, our method yields satisfactory performance on popular benchmark datasets, and also can be conveniently applied to improve other DARTS-based methods.

2. OUR METHOD

We briefly describe the basic concept of DARTS and then detail the proposed MAP formulation. The focal discussion is to justify that the effectiveness of our method can be greatly boosted by exploring a useful prior of learning operation transition matrices from good neural network architectures.

2.1. MAP-DARTS

The DARTS graph G illustrated in Figure 1 indeed induces a *one-shot supernet* [3] of the cell level. Besides the coupling issue of sharing parameters $\boldsymbol{\alpha}^{(i,j)}$ across the numerous network architectures embodied in the one-shot graph, recent research efforts [7] have pointed out that the DARTS optimization often tends to favor *strong* paths and result in nonuniform exploration over all feasible architectures. Furthermore, its strategy to decide the final cell architecture is locally and independently determined by choosing the strongest operation of each edge. The scheme is counter-intuitive in that the decision on $o^{*(i,j)}$ inherently depends on what operations have already been included in those preceding edges. To tackle all these challenging issues, we propose a new NAS formulation that the *maximum a posteriori* (MAP) reasoning is incorporated into both the optimization and search phases to yield an improved DARTS framework, named as MAP-DARTS.

Rather than solely working on the graph G of one-shot supernet, our method also constructs an auxiliary graph G' whose nodes $\{v^{jk}\}$ correspond to directed edges (j,k) in G. Note that v^{jk} will be used to denote not only the corresponding node in G' but also the underlying random variable. The probability of $v^{jk} = o \in \mathcal{O}$ is expressed by

$$p_o^{jk} = P\{v^{jk} = o \mid o \in \mathcal{O}\} \tag{1}$$

and $\mathbf{p}^{jk} = (p_{o_1}^{jk}, p_{o_2}^{jk}, \ldots, p_{o_C}^{jk})^\intercal$ is used to denote the probability distribution of all the network operations at node v^{jk}. In addition, there exists a directed edge (ij, jk) from v^{ij} to v^{jk} in G' if and only if the two corresponding directed edges (i,j) and (j,k) are present in G. In this case, v^{ij} is a parent node of v^{jk}, whose set of ancestor nodes in G' will be denoted hereafter as V^{jk}. In MAP-DARTS, the decision of assigning a particular operation $o \in \mathcal{O}$ to v^{jk} can be made only until all ancestors in V^{jk} have completed the operation assignment. That is, the decision explores the inherent dependency of operations from V^{jk} and can be achieved by evaluating the node-wise posterior probability.

$$o^{*jk} = \arg\max_{o \in \mathcal{O}} P(v^{jk} \mid V^{jk} = \mathbf{o}^{jk}, D) \tag{2}$$

where D is the underlying training data and the expression $V^{jk} = \mathbf{o}^{jk}$ symbolizes the operation assignment among the ancestor nodes of v^{jk}. The proposed method first learns the node-wise transition matrices of network operations, or adopts pre-trained priors of transition matrices, and then seeks a *maximum a posteriori* (MAP) estimate of the optimal architecture based on (2).

Let T^{ijk} be the matrix of transition probabilities between all possible operations of nodes v^{ij} and v^{jk} where (ij, jk) is an directed edge in G'. Then T^{ijk} can be written as

$$T^{ijk} = \begin{pmatrix} T^{ijk}_{o_1,o_1} & T^{ijk}_{o_1,o_2} & \cdots & T^{ijk}_{o_1,o_C} \\ T^{ijk}_{o_2,o_1} & T^{ijk}_{o_2,o_2} & \cdots & T^{ijk}_{o_2,o_C} \\ \vdots & \vdots & \ddots & \vdots \\ T^{ijk}_{o_C,o_1} & T^{ijk}_{o_C,o_2} & \cdots & T^{ijk}_{o_C,o_C} \end{pmatrix} \tag{3}$$

where each $T^{ijk}_{o,o'}$ is the transition probability from the event $v^{ij} = o$ to $v^{jk} = o'$ and $\sum_{o' \in \mathcal{O}} T^{ijk}_{o,o'} = 1$. With (1) and (3), we write, in matrix form, the node-wise operation transitions of G as

$$\mathbf{p}^{jk} = \sum\nolimits_{i=0}^{j-1} \beta^{ij} \times T^{*ijk} \times \mathbf{p}^{ij} \tag{4}$$

where T^{*ijk} denotes the transpose of T^{ijk}, $\sum_{i=0}^{j-1} \beta^{ij} = 1$ and β^{ij} weighs, among the j ancestors, the influence of ancestor v^{ij} to v^{jk}. It is worth mentioning that the right-hand-side of (4) is indeed the (accumulated) message passed from the upstream parent nodes, and (4) is an efficient reduction of the following message propagation:

$$\mathbf{p}_{t+1}^{jk} = (1-\lambda)\mathbf{p}_t^{jk} + \lambda \sum\nolimits_{i=0}^{j-1} \beta^{ij} \times T^{*ijk} \times \mathbf{p}_t^{ij} \tag{5}$$

where $0 < \lambda < 1$ is the update weight, t is the iteration index and the reduction follows from the imposed information propagation order given by the child-parent relations. During the learning stage, we aim to solve the nested optimization and learn only $\boldsymbol{\alpha}^{(i,j)} \in \boldsymbol{\alpha}_I$, where $\boldsymbol{\alpha}^{(i,j)} \in \boldsymbol{\alpha}_I$ correspond to those $v^{ij} \in G'$ without any ancestors. Their respective $\mathbf{p}^{ij}$ can be readily computed via taking softmax over $\boldsymbol{\alpha}^{(i,j)} \in \boldsymbol{\alpha}_I$. All the remaining $\mathbf{p}^{ij}$ from G' can be obtained via (4). In realizing

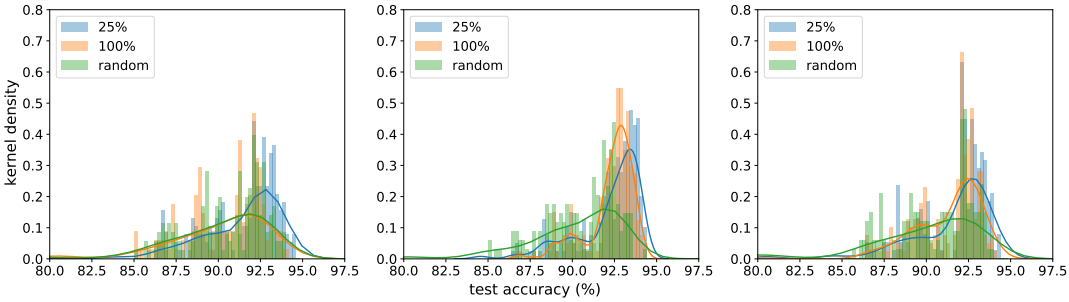

Fig. 2: The distribution of models sampled from different priors. *Left*: Non-weighted priors construct by all models in the NAS-Bench-201 space. *Middle*: Weighted priors construct by all models in the NAS-Bench-201 space. *Right*: Weighted priors construct by 100 sampled models in the NAS-Bench-201 space.

the above steps, we still need an efficient way to decide the parameters $\{\beta^{ij}, T^{ijk}\}$ in (4). Our approach treats them as learnable parameters. To ensure the optimization leads to stable NAS performance, we construct general priors of T^{ijk} from generic neural networks of good performance to provide regularization constraints.

2.2. Transition matrix priors

Learning with prior regularization has the advantage of preventing the optimization from straying to unexpected outcomes. However, it also adds constraints to the optimization and may hinder the model learning when using inappropriate priors. Taking the two aspects into consideration, we adopt a performance-driven strategy to prefer sampling *good* network models from the search space of a NAS benchmark, detailed in the experimental results, and use them to construct the priors.

Let P^{ijk} be the prior matrix of the same size as the transition matrix T^{ijk}. Performance-driven sampling is used to repeatedly generate M models from the benchmark search space. The prior matrices can be updated by

$$P^{ijk}_{o_X,o_Y} = \sum\nolimits_m w_m \times \mathbf{1}[o^{ij}_m = o_X \wedge o^{jk}_m = o_Y] \quad (6)$$

where o_X and o_Y are specific operations, w_m is the weight based on model m's performance and o^{ij}_m denotes the operation on edges (i, j) of model m. Upon the completion of sampling for priors, we *normalize* P^{ijk} to ensure each row sum to 1 . In learning T^{ijk} in (4), each transition matrix is initialized by its prior P^{ijk} and optimized with the additional regularization term $\|T^{ijk} - P^{ijk}\|_2$.

3. EXPERIMENTS

3.1. Characteristic of priors

To fairly compare different settings and fast retrieve robust results, we choose to analyze our algorithm on the NAS-Bench-201 space [8]. It is a public benchmark for most of the NAS algorithms. As a simplified version of the search space in cell-based NAS, the cell contains 6 edges, 5 operations on each edge, and a total $5^6 = 15625$ models in the space. We first use all models in the NAS-Bench-201 space and select top-quality models based on their performances at 10th epoch, using (6) (with weights being 1) and normalization to compute the priors. To evaluate the quality of priors, we use the priors to sample another 200 sets of models and observe their distribution. Figure 2 *Left* shows the distribution of models sampled from the priors built with different ratios of top models. The models sampled from top-25% priors perform better than the models sampled from top-100% priors and the models with random sampling significantly. It indicates that models sampled from the prior constructed with better models tend to have better performance.

To construct more robust priors, we weight models according to their rank. To be more specific, we give better models higher weights and worse models lower weights. Note that the weights decay linearly from 1.5 to 0.5 across models. Also, we sparsify the transition matrices to make low confidence operations not be selected. Figure 2 *Middle* shows the results with weighted priors, and both priors have advancements. The top-100% priors improve significantly and perform similarly as top-25% priors after weighted. Furthermore, this property still holds even if we construct priors with few models. Figure 2 *Right* shows the results with the priors construct with only 100 models. It suggests that with weighted and sparsification, we can get equal quality priors under different ratio of top models, and large amount of models for ranking is unnecessary. In a larger search space, priors may become sparser and reduce the diversity of models to be searched. In this case top-100% priors can make good use of all the sampled models and provides reliable results.

3.2. CIFAR10 and ImageNet

We deploy our framework on DARTS-based methods on the CIFAR10 dataset. With the concern of efficient searching, we select the works with lower search costs such as DARTSV1

Architecture	Test Err. (%)	Param. (M)	Search Cost (GPU-days)
DARTSV1 [9]	3.00 ± 0.14	3.3	0.4
PC-DARTS [10]	2.67 ± 0.07	3.6	0.1
FairDARTS [7]	2.54 ± 0.05	3.32	0.4
MiLeNAS [11]	2.51 ± 0.11	3.87	0.3
SGAS [12]	2.66 ± 0.24	3.80	0.25
MAP-DARTSV1	2.70 ± 0.07	4.0 ± 0.27	0.4
MAP-PC-DARTS	2.53 ± 0.04	3.4 ± 0.36	0.1
MAP-PC-DARTS*	**2.48**	3.32	0.1

Table 1: Results of different architectures on CIFAR10. The MAP module can be applied to various DARTS based methods. (* the best result)

Architecture	Test Err. top-1	Test Err. top-5	Param. (M)	Cost
PC-DARTS (CIFAR10) [10]	25.1	7.8	5.3	0.1
PC-DARTS (ImageNet) [10]	24.2	7.3	5.3	3.8
FairDARTS [7]	24.4	7.4	5.3	3
MiLeNAS (CIFAR10) [11]	24.7	7.6	5.3	0.3
SGAS (CIFAR10) [12]	24.1	7.3	5.4	0.25
MAP-PC-DARTS (CIFAR10)	23.8	7.0	5.8	0.1
MAP-PC-DARTS (ImageNet)	**23.3**	7.0	6.3	3.8

Table 2: Results of different architectures on ImageNet.

and PC-DARTS. Different from the original algorithm, we replace architecture parameters with our dependency net, which contains trainable α_I and transition matrices. To construct priors for initializing the transition matrices, we sample 1000 models from DARTS search space and train each for 30 epochs on CIFAR10. The priors can be construct following (6) with $w_m = 1.5 - ranking(m)/1000$ and a sparsification threshold of 0.1 after normalization. The results are shown in Table 1. We report the mean and standard deviation testing error of 5 individual runs. Benefit from the MAP, we got a better architecture compared with previous works.

In our experiments on ImageNet, we use the priors constructed on CIFAR10. The supernet training also follows the setting of PC-DARTS and the details can be found in Appendix *Training details*. The performances of our approach are shown in Table 2. Note that the architectures searched on CIFAR10 and ImageNet itself are both evaluated. The model searched on CIFAR10 is used to assess the transferability of our method. We can observe that MAP-PC-DARTS attained state-of-the-art results in both settings.

warm-up/total epoch	0/0	5/10	15/50
PC-DARTS	-	4.28	2.67
MAP-PC-DARTS	2.77	2.71	**2.53**

Table 3: Results (in error rate) of different searching epoch settings on CIFAR10.

4. DISCUSSION

4.1. Priors properties analysis

We discuss whether priors can accelerate searching. Table 3 shows the results of PC-DARTS and MAP-PC-DARTS under different searching settings. We use A/B to denote warm-up epochs/total epochs. In the 0/0 case, PC-DARTS has no meaningful results without training, but MAP-PC-DARTS can derive a model with only 2.77% error rate. In the 5/10 case, MAP-PC-DARTS continues to improve the results, while PC-DARTS still does not find better architecture. In general, with the knowledge provided by priors, we can accelerate NAS and meanwhile obtain reliable results.

4.2. Transferability

In the experiments on ImageNet, we observe a good transferability on both the searched model and the priors from CIFAR dataset. It suggests that good architectures can perform well across different dataset. This observation is consistent with the previous work [13] and increase the reusability of priors. Moreover, we can construct priors with a smaller dataset and transfer them to a larger dataset to boost the searching process.

5. CONCLUSIONS

We tackle the problem of network architecture search by seeking a maximum a posteriori (MAP) estimate to the optimal target architecture. The proposed approach builds upon the *continuous* relaxation of the DARTS model and demonstrate its usefulness with satisfactory performance on benchmark datasets and reliable reasoning for locating the target architecture. A notable idea of our method is to focus on learning the node-wise transition matrices of network operations rather than the network parameters, which enable the proposed NAS technique bypasses the coupling dilemma caused by sharing network parameters over various architectures in the one-shot supernet. Our method brings a new paradigm to NAS in the real world that when we meet a new dataset, we can first sample some models as a "probe" into the search space, and then the rough results can help us guide the search process. Furthermore, the prior can reuse multiple times, which can become a feature of the AutoML service and reduce the cost to get reliable results. Our future work will focus on extending the MAP formulation to other non-DARTS NAS approaches.

6. REFERENCES

[1] Barret Zoph and Quoc V. Le, "Neural architecture search with reinforcement learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[2] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le, "NAS-FPN: learning scalable feature pyramid architecture for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 7036–7045, Computer Vision Foundation / IEEE.

[3] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Fei-Fei Li, "Auto-DeepLab: hierarchical neural architecture search for semantic image segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 82–92, Computer Vision Foundation / IEEE.

[4] Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He, and Piotr Dollár, "Designing network design spaces," *CoRR*, vol. abs/2003.13678, 2020.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 770–778, IEEE Computer Society.

[6] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "MobileNetV2: inverted residuals and linear bottlenecks," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 4510–4520.

[7] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li, "Fair DARTS: eliminating unfair advantages in differentiable architecture search," *CoRR*, vol. abs/1911.12126, 2019.

[8] Xuanyi Dong and Yi Yang, "NAS-Bench-201: extending the scope of reproducible neural architecture search," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. 2020, OpenReview.net.

[9] Hanxiao Liu, Karen Simonyan, and Yiming Yang, "DARTS: differentiable architecture search," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[10] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong, "PC-DARTS: partial channel connections for memory-efficient differentiable architecture search," *CoRR*, vol. abs/1907.05737, 2019.

[11] Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang, "Milenas: Efficient neural architecture search via mixed-level reformulation," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. 2020, pp. 11990–11999, IEEE.

[12] Guohao Li, Guocheng Qian, Itzel C. Delgadillo, Matthias Müller, Ali K. Thabet, and Bernard Ghanem, "SGAS: sequential greedy architecture search," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 2020, pp. 1617–1627.

[13] Chenxi Liu, Piotr Dollár, Kaiming He, Ross B. Girshick, Alan L. Yuille, and Saining Xie, "Are labels necessary for neural architecture search?," *CoRR*, vol. abs/2003.12056, 2020.