



TND-NAS: Towards Non-Differentiable Objectives in Differentiable Neural Architecture Search

Bo Lyu¹^a and Shiping Wen²^b

¹*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China*

²*Australian AI Institute, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo 2007, Australia*

Keywords: Neural Architecture Search, Reinforcement Learning, Non-Differentiable, Supernetwork

Abstract: Differentiable architecture search has gradually become the mainstream research topic in the field of Neural Architecture Search (NAS) for its high efficiency compared with the early heuristic NAS (EA-based, RL-based) methods. Differentiable NAS improves the search efficiency, but no longer naturally capable of tackling the non-differentiable objectives. Researches in the multi-objective NAS field target this point but requires vast computational resources cause of the individual training of each candidate architecture. We propose the TND-NAS, which discretely sample architectures based on architecture parameter α (without sampling controller), and directly optimize α by policy gradient algorithm. Our representative experiment takes two objectives (Parameters, Accuracy) as an example, we achieve a series of high-performance compact architectures on CIFAR10 (1.09M/3.3%, 2.4M/2.95%, 9.57M/2.54%) and CIFAR100 (2.46M/18.3%, 5.46M/16.73%, 12.88M/15.20%) datasets.

1 INTRODUCTION

Neural Architecture Search (NAS) aims at alleviating the tremendous labor of manual tuning on neural network architectures, which has facilitated the development of AutoML (Guo et al., 2021; Stamoulis et al., 2020; He et al., 2021). Recently, under the fast-growing in this area, NAS models have surpassed previous manually designed models in various research fields. As an intuitive method, Reinforcement Learning (RL) based NAS (Zoph and Le, 2017) employs the RNN controller to sample the architecture (represented by sequential indices), and the validation accuracy of each candidate architecture is viewed as the reward for policy gradient optimization.


By continuous relaxation of the search space, differentiable NAS researches make the loss function differentiable w.r.t architecture weights, thus the search can be processed directly by gradient-based optimization. Even with high search efficiency, differentiable NAS research rarely involves non-


differentiable objectives, e.g., energy, latency, or memory consumption.

Parallel to the explosive development of the differentiable NAS sub-field, the multi-objective NAS methods (MnasNet (Tan et al., 2019), DPP-Net (Dong et al., 2018)) dedicate to searching for the neural architectures in discrete space with the consideration of multi-dimensional metrics, including differentiable and non-differentiable ones. MNAS methods rely on the heuristic search strategy such that the computational overhead is huge.

Our method relies on the differentiable NAS as the main search framework, in which candidate operations are progressively eliminated base on the architecture parameters α after each stage of the search, meanwhile, the depth of the model is increased gradually (Chen et al., 2019), by which means to tackle the “depth gap”. Our contributions may be summarized as follows:

1) TND-NAS methods is capable to tackle the non-differentiable objectives under the differentiable search framework, that is, with the merits of high

^a <https://orcid.org/0000-0003-1595-8361>

^b <https://orcid.org/0000-0001-8077-7001>

efficiency in differentiable NAS and the objective compatibility of multi-objective NAS.

2) Through flexible and customized search configurations, the visualization of architecture evolving during the search process shows that our method can reach the trade-off among differentiable and non-differentiable metrics.

2 RELATED WORK

RL/EA-based NAS. Traditional architecture search methods start from employing Evolutionary Algorithm (EA) as the search strategy [8] (Stanley and R. Miikkulainen, 2002; Sun et al., 2019; Sun et al., 2020; Real et al., 2019; Liu et al., 2017; Lu et al., 2019; Han et al., 2021; Li et al., 2021) (Wang et al., 2021). In these works, high-performing network architectures are mutated, and less promising architectures are discarded. Recently, the significant success of RL-based NAS is first reported by (Zoph and Le, 2017; Zoph et al., 2018). ENAS (Pham et al., 2018) comes in the continuity of previous works (Zoph and Le, 2017; Zoph et al., 2018), it proposes the weight sharing strategy to significantly improve the searching efficiency. Overall, the RL/EA-based NAS methods heuristically search the architecture over a discrete domain and suffer from the efficiency issue.

Differentiable NAS. By continuous relaxation of the searchspace, DARTS (Liu et al., 2019) constructs the supernet by mixed-edge operation, the task of architecture search is treated as learning a set of continuous architecture parameters. DARTS and the followed works also suffer from the high GPU-memory overhead issue (Xu et al., 2021), which grows linearly w.r.t. candidate-set size. As a result, these works have to search with only a few blocks (cells), but evaluate the architecture with more blocks stacked, which undoubtedly brings in the “depth gap” issue (Chen et al., 2019) (Xie et al., 2021).

3 METHODOLOGY

3.1 Preliminary

In differentiable NAS method, to make the search space continuous, the operation in specified location is characterized by all candidate operations:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x) \quad (1)$$

According to the customs, under the differentiable supernet framework. Obviously, the task of architecture search then is to learn a set of vectors. Thus the search procedure is formulated as a bilevel optimization problem as the upper-level variable and weight parameter as the lower-level variable:

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{\text{val}}(w^*(\alpha), \alpha) \\ \text{s.t.} \quad & w^*(\alpha) = \arg \min_w \mathcal{L}_{\text{train}}(w, \alpha) \end{aligned} \quad (2)$$

To effectively solve this bilevel optimization problem, DARTS approximates the solution by alternate optimization.

3.2 Search Method

Our evaluation acceleration scheme follows the weight-sharing differentiable NAS, in which the sub-networks inherit the weight from the supernet. Importantly, the architecture parameters are not directly trained by gradient descent with differentiable loss function (cross-entropy loss), but rather treat as the sampling policy, which is trained by policy gradient algorithm (REINFORCE (Williams et al., 1992)) in discrete space, which follows the non-differentiable route. In terms of the weight parameters training, weight parameter is optimized by the gradient descent of cross-entropy loss:

$$\omega^*(\alpha) = \arg \min_{\omega} \mathcal{L}_{\text{train}}(\omega, \alpha) \quad (3)$$

In terms of the architecture sampling, for each index, the binary vector is sampled by multinomial distribution by probability vector.

$$p^{(i,j)} = \text{softmax}(\alpha^{(i,j)}) \quad (4)$$

$$g^{(i,j)} \sim \text{Multi}(p^{(i,j)}, 1) \quad (5)$$

the sub-network structured by inheriting the supernet's weights:

$$N_m = \mathcal{A}(g_m) \quad (6)$$

where N stands for the sub-network.

To optimize the policy architecture parameters, we empirically resort to the policy gradient algorithm (REINFORCE), the gradient of policy architecture parameters is estimated based on the reward of discretely sampled architectures, that is:

$$\begin{aligned} \nabla_{\alpha} \mathcal{J}_{\text{val}}(\alpha) &= \mathbb{E}_{N \sim \pi_{\alpha}(N)} [R_{\text{val}}(\omega^*, N) \nabla_{\alpha} \log(\pi_{\alpha}(N))] \\ &\approx \frac{1}{M} \sum_{m=1}^M (R_{\text{val}}(\omega^*, N_m) - b) \nabla_{\alpha} \log(\pi_{\alpha}(N_m)) \end{aligned} \quad (7)$$

Apparently, in terms of performance, the most common metric Accuracy can be directly employed as a reward. Much more attention needs to be addressed to the multi-objective scenario, in which the reward function needs to be designed based on real-world requirements. For example, resource-constrained scenarios need the trade-off between performance and efficiency, e.g., memory consumption, or inference latency. Motivated by Mnasnet[5], taking the Accuracy and Parameters as the objectives, we make the reward be linearly w.r.t Accuracy, but non-linearly w.r.t Parameters, which is treated as a reward-penalty factor in the non-linearly scalarization function:

$$R = \text{Acc} \cdot \left(\frac{\text{Params}}{P} \right)^{\beta} \quad (8)$$

Table 1: Hyper-parameters of search.

Batch size	128	Initial channels	16
Stage epochs	25	Layers	5,12,20
Pretrain epochs	10	Drop operation num	4,3,2
Optimizer (ω)	SGD	Weight decay	4e-5
LR	0.025	LR decay	cosine
Momentum	0.9	Reference of Parameters	4,4,3,2,1,8
Dropout ratio	0.3	Penalty coefficient(C10)	-0.60,-0.60,-0.60
Optimizer (α)	Adam	Penalty coefficient(C100)	-0.75,-0.85,-0.85
RL interval	1(each)	RL sampling batch size	10

Table 2: Hyper-parameters of evaluation on CIFAR10 and CIFAR100.

Batch size	128	Initial channels	24
Epochs	600	Layers	20
Optimizer	SGD	Weight decay	3e-4
Learning rate	0.025	LR scheduler	cosine
Momentum	0.9	Dropout ratio	0.3
Auxiliary towers weight	0.4	Cutout regularization len	16

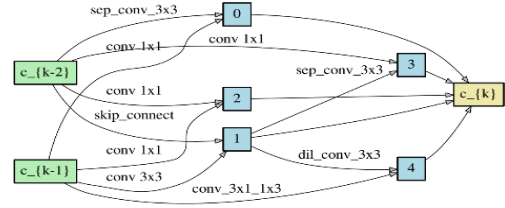
4 EXPERIMENTS

Our search processes are conducted on 2 popular image classification datasets, CIFAR10 and CIFAR100. During the search procedure, we keep the training set split into two equal-size subsets, one for weight parameters training and the other for architecture parameter validation.

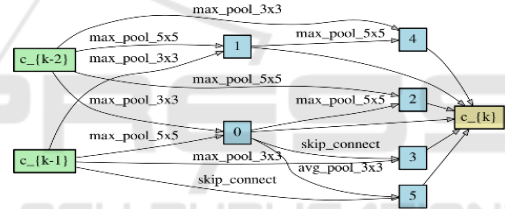
4.1 Architecture Search

The search experiment is performed with PyTorch 1.4 framework on 2 NVIDIA 1080Ti GPUs that each with 11GB memory. Our search hyper-parameters are presented in Table 1.

The visualization of our searched model (CIFAR10-S, under compression configuration for small-scale model) is shown in Fig.1(a), in which normal cell is in Fig.1(a), and reduction cell is in Fig.1(b). The nodes represent the feature maps (FMs), and edges stand for the connection that implemented by the searched operation.



(a) Normal cell.



(b) Reduction cell.

Figure 1: The searched result (CIFAR10-S) of TND-NAS. The Parameters of this architecture is 1.09M with 16 initial channels.

4.2 Architecture Evaluation

To evaluate the searched architectures, we randomly initialize its weights, train it from scratch, and report its performance on the test set. To be notice that the test set is blind for the architecture search or architecture training.

Table 3: Comparison of the evaluation results on CIFAR10 and CIFAR100.

Architecture	Test Err. (%)		Params (M)	Search Cost (GPU-days)
	C10	C100		
DenseNet-BC [41]	3.46	17.18	25.6	-
NASNet-A + cutout [6]	2.65	-	3.3	1800
AmoebaNet-A + cutout [22]	3.34	-	3.2	3150
AmoebaNet-B + cutout [22]	2.55	-	2.8	3150
Hierarchical Evolution [23]	3.75	-	15.7	300
PNAS [42]	3.41	-	3.2	225
ENAS + cutout [7]	2.89	-	4.6	0.5
DARTS (first order) [8]	3	17.76	3.3	1.5
DARTS (second order) + cutout [8]	2.76	17.54	3.3	4
SNAS + mild constraint + cutout [43]	2.98	-	2.9	1.5
SNAS + moderate constraint + cutout [43]	2.85	-	2.8	1.5
SNAS + aggressive constraint + cutout [43]	3.1	-	2.3	1.5
ProxylessNAS + cutout [29]	2.08	-	5.7	4
RelativeNAS + cutout [39]	2.34	15.86	3.93	0.4
P-DARTS CIFAR10 + cutout [17]	2.5	16.55	3.4	0.3
P-DARTS CIFAR100 + cutout [17]	2.62	15.92	3.6	0.3
P-DARTS CIFAR10 (large) + cutout [17]	2.25	15.27	10.5	0.3
P-DARTS CIFAR100 (large) + cutout [17]	2.43	14.64	11	0.3
TND-NAS CIFAR10 (S) + cutout	3.3	-	1.09	1.3 [†]
TND-NAS CIFAR10 (M) + cutout	2.70	-	3.2	1.3 [†]
TND-NAS CIFAR10 (L) + cutout	2.54	-	9.57	1.3 [†]
TND-NAS CIFAR100 (S) + cutout	-	18.3	2.46	1.3 [†]
TND-NAS CIFAR100 (M) + cutout	-	16.73	5.46	1.3 [†]
TND-NAS CIFAR100 (L) + cutout	-	15.20	12.88	1.3 [†]

[†] Performed on 2 NVIDIA 1080Ti GPU each with 11G memory for 0.65 day

Our evaluation on CIFAR10/CIFAR100 follow the experimental training setting in P-DARTS, hyperparameters setting is shown in Table 2.

For comparison, some state-of-the-art approaches are reported in Table 3, including the manually designed models and outstanding NAS models. Our experiment reaches a series of models on CIFAR10: extremely compact model (S) of 3.3% test error with 1.09M Parameters, moderate model (M) of 2.7% with 3.2M Parameters, and large model (L) of 2.54% with 9.57M Parameters, which has shown the flexibility of TND-NAS. Our search experiment on CIFAR100 also achieves the promising results: 18.3% test error with 2.46M Parameters (S), 16.73% with 5.46M Parameters (M), 15.2% with 12.88M Parameters (L). As demonstrated, TND-NAS is comparable with the P-DARTS, but much more diversified in model scale.

4.3 Search Cost Comparison

TND-NAS costs merely 0.65 days on 2 NVIDIA 1080Ti GPUs, each with only 11G memory. Compared with previous promising multi-objective NAS methods, our method achieves a substantial improvement in search resource cost, 1.3 GPU-days, that is 1/6 of that in NSGA-Net(Lu et al., 2019).

5 CONCLUSIONS

This work incorporates the differentiable NAS framework with the capability to handle non-differentiable metrics and aims to reach the trade-off between non-differentiable and differentiable metrics.

Meanwhile, our method reconciles the merits of multi-objective NAS and differentiable NAS, and it is feasible to the applied in real-world NAS scenarios,

e.g., resource-constrained, and platform-specialized. Taking the Parameters for instance, by multi-objective search, we achieve a series of scalable models (S, M, L) that are comparable to the state-of-the-art NAS approaches on CIFAR10/CIFAR100 datasets. There also exist some limitation of our method: First, our representative experiments do not include some other non-differentiable metrics, e.g., Inference latency. Second, while the gaps and non-differentiable metrics issues have been addressed, the employed optimization approach that is based on sampling in discrete space inevitably leads to a greater computational cost than the pure differentiable NAS method.

REFERENCES

- Y. Guo, Y. Luo, Z. He, J. Huang, and J. Chen, "Hierarchical neural architecture search for single image super-resolution," *IEEE Signal Processing Letters*, vol. 27, pp. 1255–1259, 2020.
- D. Stamoulis, R. Ding, D. Wang, D. Lymberopoulos, B. Priyantha, J. Liu, and D. Marculescu, "Single-path mobile autml: Efficient convnet design and nas hyperparameter optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 609–622, 2020.
- X. He, K. Zhao, and X. Chu, "Autml: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.
- B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *International Conference on Learning Representations*, 2017, pp. 1–16.
- M. Tan, B. Chen, R. Pang, V. K. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: platform-aware neural architecture search for mobile," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.
- J.-D. Dong, A.-C. Cheng, D.-C. Juan, W. Wei, and M. Sun, "Dpp-net: Device-aware progressive search for pareto-optimal neural architectures," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 540–555.
- X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1294–1303.
- P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.
- K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

- Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 394–407, 2019.
- Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing cnn architectures using the genetic algorithm for image classification," *IEEE transactions on cybernetics*, vol. 50, no. 9, pp. 3840–3854, 2020.
- E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4780–4789.
- H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *International Conference on Learning Representations*, 2017, pp. 1–13.
- Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "Nsga-net: neural architecture search using multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 419–427.
- T. Han, S. P. Nagesh Rao, D. Filev, K. Redmill, and O'zguner, "An online evolving method for a safe and fast automated vehicle control system," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–13, 2021.
- C. Li, F. Liu, Y. Wang, and M. Buss, "Concurrent learning-based adaptive control of an uncertain robot manipulator with guaranteed safety and performance," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–15, 2021.
- J. Wang, Y. Song, and G. Wei, "Security-based resilient robust model predictive control for polytopic uncertain systems subject to deception attacks and rr protocol," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–12, 2021.
- B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.
- H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International conference on machine learning*. PMLR, 2018, pp. 4095–4104.
- H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *International Conference on Learning Representations*, 2019, pp. 4561–4574.
- Y. Xu, L. Xie, W. Dai, X. Zhang, X. Chen, G.-J. Qi, H. Xiong, and Q. Tian, "Partially-connected neural architecture search for reduced computational redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- (Xie et al., 2021) L. Xie, X. Chen, K. Bi, L. Wei, Y. Xu, L. Wang, Z. Chen, A. Xiao, J. Chang, X. Zhang et al., "Weight-sharing neural architecture search: A battle to shrink the optimization gap," *ACM Computing Surveys (CSUR)*, vol. 54, no. 9, pp. 1–37, 2021.
- R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.
- S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," *arXiv preprint arXiv:1812.09926*, 2018.
- H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *International Conference on Learning Representations*, 2019, pp. 1–13.
- H. Tan, R. Cheng, S. Huang, C. He, C. Qiu, F. Yang, and P. Luo, "Relativenas: Relative neural architecture search via slow-fast learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.