TNASP: A Transformer-based NAS Predictor with a Self-evolution Framework

Shun Lu[1,2], Jixiang Li[3], Jianchao Tan[3], Sen Yang[3], Ji Liu[3]
[1] Research Center for Intelligent Computing Systems, State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences
[3] Kuaishou Technology
lushun19s@ict.ac.cn, {lijixiang,jianchaotan,senyang,jiliu}@kuaishou.com

Speaker: Shun Lu

NEURAL INFORMATION PROCESSING SYSTEMS
NeurIPS 2021

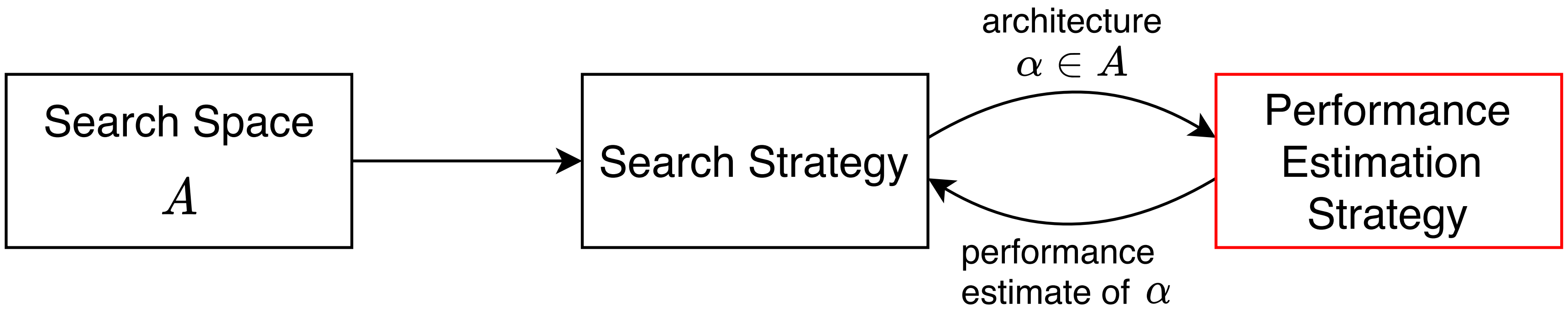
ANS@ICT 快手

1. Background & Motivation

1.1 Neural Architecture Search (NAS)

- Three major elements of NAS:
  - Search Space, Search Strategy, and Performance Estimation Strategy[1]

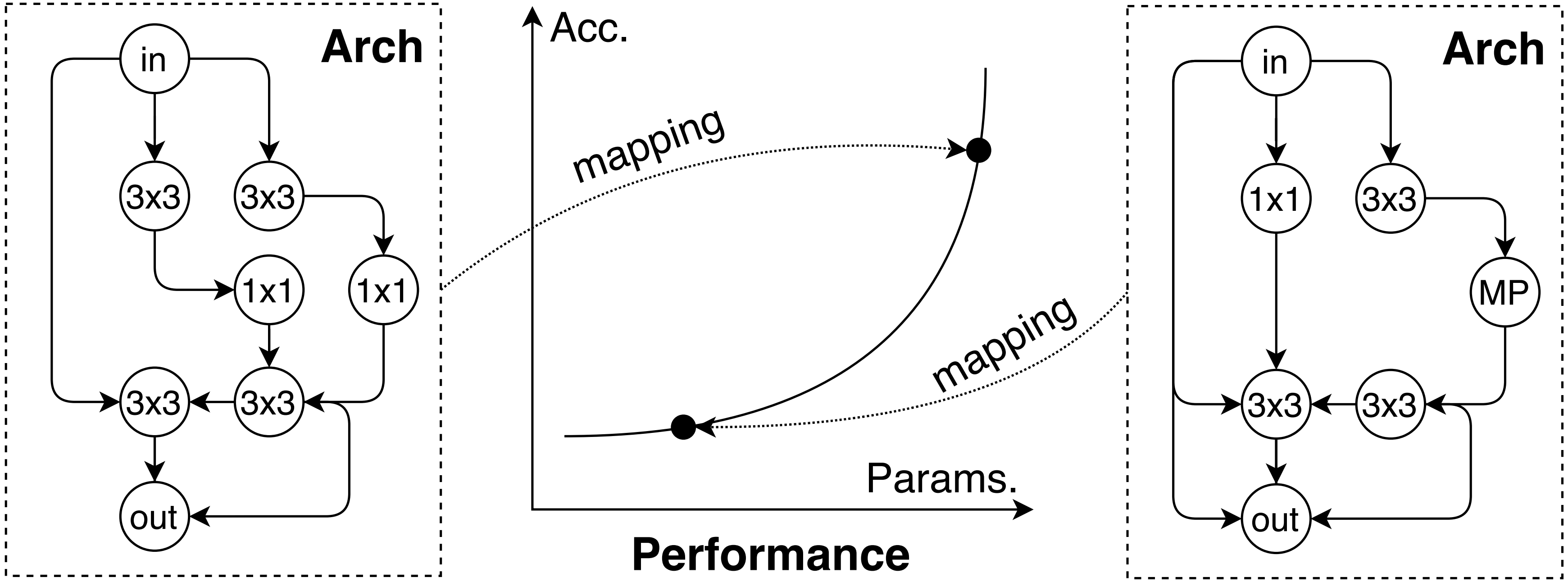
- One major problem :
  - Performance estimation stage of each sub-architecture takes too much time !



1. Background & Motivation

1.2 Predictor-based NAS methods:

- Objective: Learn a mapping relationship between architectures and their real performance
- Advantage: Largely reduce the search cost.



1. Background & Motivation

1.3 Previous works of predictor-based NAS methods:

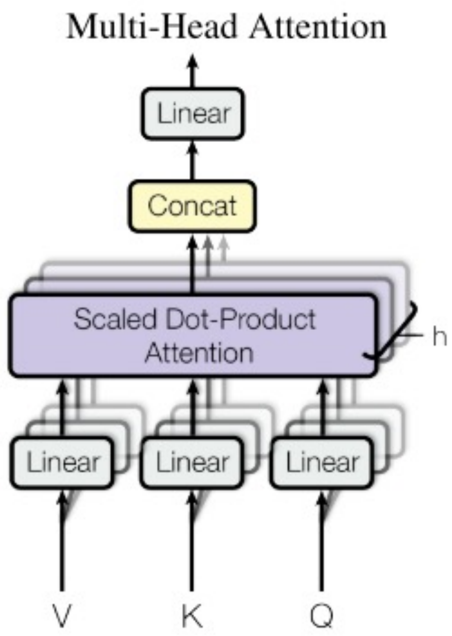
- Training-free: Compute different metrics over graph topology information as feature encodings.

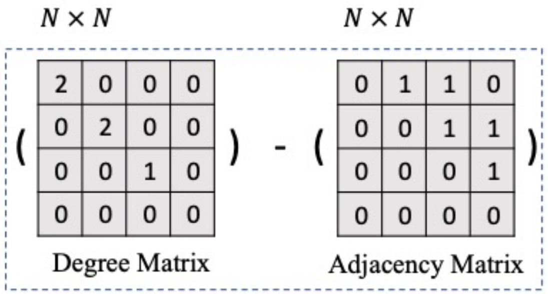
- Training-based (focus): Different DNN backbones for feature encodings → regression.
  - Sequence-based schemes:
    - NAO[2], D-VAE[3], BANANAS[4] and so on.
  - Graph-based methods:
    - BONAS[5], InterpretableNAS[6], CTNAS[7] and so on.

  - Drawbacks:
    - Feature Encodings over graph-structure data are not good enough.
    - Temporal evaluation information is ignored, which however, is useful.

2. Contributions

- A **T**ransformer-based **NAS** performance **P**redictor (**TNASP**) :
  - Use graph Laplacian matrix as the positional encoding.
  - Use multi-head self-attention mechanism to better encode features on graph data
- A generic **S**elf-**E**volution (**SE**) framework :
  - Leverage evaluation score as constraints to optimize.
  - Make full use of temporal evaluation information.
- Achieve state-of-the-art results on **4 benchmark search spaces**.

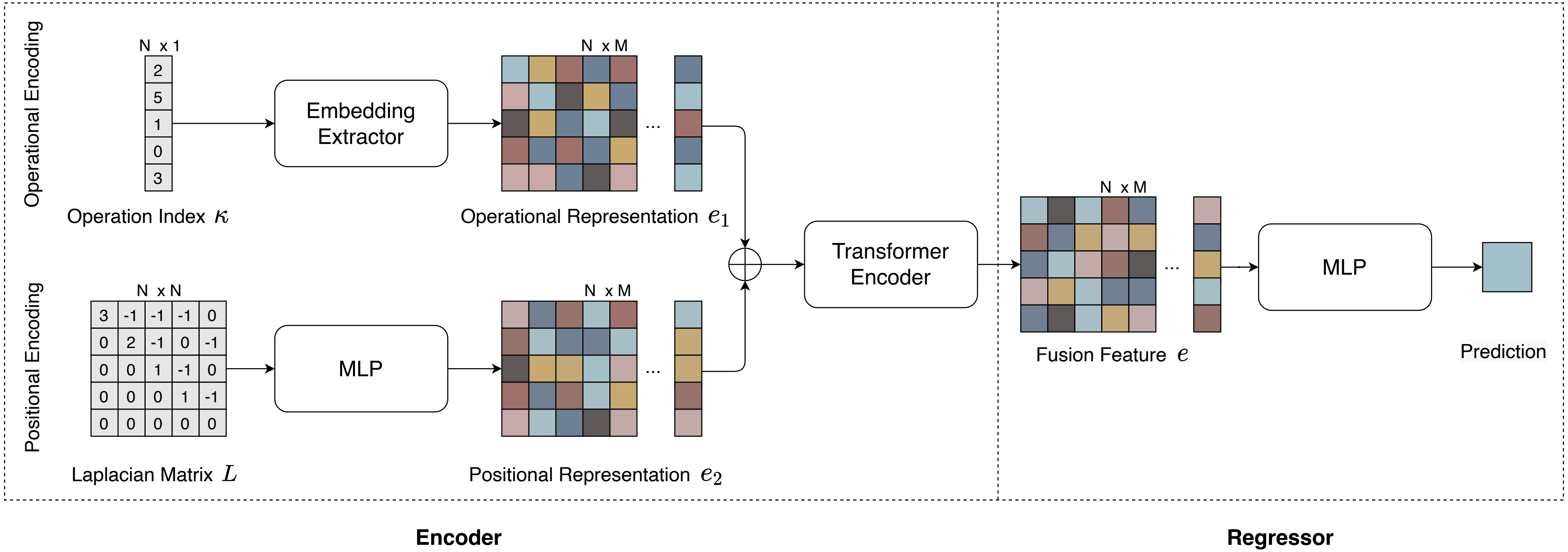




Laplacian Matrix

3. Method

3.1 A Transformer-based NAS Performance Predictor (TNASP) :



- **Components**: an encoder and a regressor
  - ➢ Encoder: 3-layer Transformer Encoder
  - ➢ Regressor: 2-layer MLP

3. Method

3.2 Self-evolution framework:

- Our formulation:

Training Loss   Evaluation Loss

$$\min_{\theta,\bar{y}} \sum_{i=1}^{n} \|f_\theta(x_i) - y_i\|^2 + \alpha \sum_{j=1}^{V} \|f_\theta(v_j) - \boxed{\bar{y}_j}\|^2$$

Auxiliary variables as the proxy of ground truth labels.

$$\text{s.t.} \quad \frac{1}{V}\sum_{j=1}^{V} \|\hat{y}_j^{(t)} - \boxed{\bar{y}_j}\|^2 = e^{(t)}, t = 1, 2, 3, ..., T \qquad (8)$$

Each previous evaluation as each constraint.

- Use Lagrange Multiplier to convert as a minimax optimization problem:

$$L(\theta, \bar{y}, \lambda) = \min_{\theta,\bar{y}} \max_{\lambda} \sum_{i=1}^{n} \|f_\theta(x_i) - y_i\|^2 + \alpha \sum_{j=1}^{V} \|f_\theta(v_j) - \bar{y}_j\|^2 + \frac{1}{T}\sum_{t=1}^{T} \lambda^{(t)} \left(\frac{1}{V}\sum_{j=1}^{V} \|\hat{y}_j^{(t)} - \bar{y}_j\|^2 - e^{(t)}\right)$$

3. Method

3.2 Self-evolution framework:

- Gradient-based iteratively updates:

$$\theta^{k+1} = \theta^k - \eta_\theta \frac{\partial L(\theta, \bar{y}^k, \lambda^k)}{\partial \theta} \tag{11}$$

$$\bar{y}^{k+1} = \bar{y}^k - \eta_{\bar{y}} \frac{\partial L(\theta^k, \bar{y}, \lambda^k)}{\partial \bar{y}} \tag{12}$$

$$\lambda^{k+1} = \lambda^k + \eta_\lambda \frac{\partial L(\theta^k, \bar{y}^k, \lambda)}{\partial \lambda} \tag{13}$$

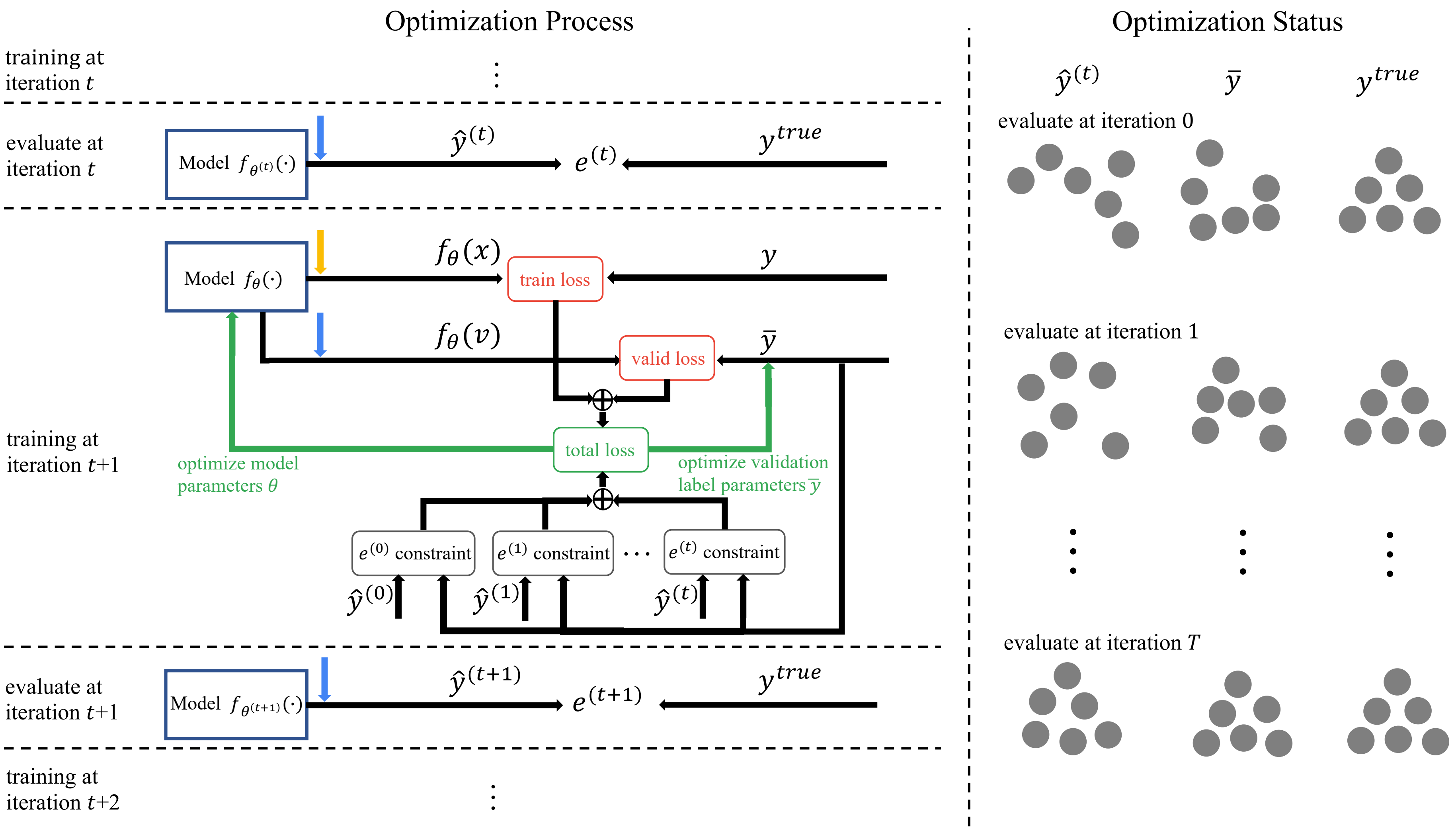
Algorithm 1 Self-evolution Optimization Algorithm

Input: Input training data x, input validation data v, input training target y, neural network f.
Output: Network parameters θ, estimated target $\bar{y}$.

```
Optimize the network parameters θ until convergence using normal training performed on the
training dataset only.
for t = 1 to T do
    Compute e^(t) according to Eq. (9) using predictor's prediction results on validation dataset.
    Add a new constraint: (1/V) Σ_{j=1}^{V} ||ŷ_j^(t) − ȳ_j||^2 = e^(t) into Eq. (8)
    while not converged do
        Update θ according to the Eq. (11)
        Update ȳ according to the Eq. (12)
        Update λ according to the Eq. (13)
    end while
end for
return Network parameters θ and estimated targets ȳ
```

3. Method

3.2 Self-evolution framework:



4. Experiments

4.1 Ranking results on NAS-Bench-101

Training Samples	100 (0.02%)	172 (0.04%)	424 (0.1%)	424 (0.1%)	4236 (1%)
Validation Samples	200	200	200	200	200
Test Samples	all	all	100	all	all
Neural Predictor† [44]	0.391	0.545	0.710	0.679	0.769
SPOS [17]	-	-	0.196⋆	-	-
FairNAS [9]	-	-	-0.232⋆	-	-
NAO‡ [31]	0.501	0.566	0.704	0.666	0.775
ReNAS [47]	-	-	0.634⋆	0.657	0.816
RegressionNAS	-	-	0.430⋆	-	-
CTNAS [8]	-	-	0.751⋆	-	-
TNASP	**0.600**	**0.669**	**0.752**	**0.705**	**0.820**
Neural Predictor† + SE	0.458	0.577	0.713	0.684	0.773
NAO‡ + SE	0.564	0.624	0.732	0.680	0.787
TNASP + SE	**0.613**	**0.671**	**0.754**	**0.722**	**0.820**

Table 1: Comparison with other methods on NAS-Bench-101. We calculate the Kendall's Tau by predicting accuracy of all architectures in NAS-Bench-101. †: re-implemented by ourselves. ‡: implemented based on their released model. ⋆: reported by CTNAS[8].

4. Experiments

4.2 Ranking results on NAS-Bench-201

Training Samples **Validation Samples** **Test Samples**	78(0.05%) 200 all	156(1%) 200 all	469(3%) 200 all	781(5%) 200 all	1563(10%) 200 all
Neural Predictor† [44]	0.343	0.413	0.584	0.634	0.646
NAO‡ [17]	0.467	0.493	0.470	0.522	0.526
TNASP	**0.539**	**0.589**	**0.640**	**0.689**	**0.724**
Neural Predictor† + SE	0.377	0.433	0.602	0.652	0.649
NAO‡ + SE	0.511	0.511	0.514	0.529	0.528
TNASP + SE	**0.565**	**0.594**	**0.642**	**0.690**	**0.726**

Table 2: Comparison with other methods on NAS-Bench-201. We calculate the Kendall's Tau by predicting the accuracy of all architectures in NAS-Bench-201 and comparing them with ground truths. †: re-implemented by ourselves. ‡: implemented based on their released model.

4. Experiments

4.3 Search results on DARTS search space

Architecture	Test Accuracy(%)	#Params.(M)	Search Cost(G·D)
DenseNet-BC [20]	96.54	25.6	-
PyramidNet-BC [18]	96.69	26.0	-
Random search baseline	96.71 ± 0.15	3.2	-
NASNet-A [54] + cutout	97.35	3.3	1,800
NASNet-B [54] + cutout	96.27	**2.6**	1,800
NASNet-C [54] + cutout	96.41	3.1	1,800
AmoebaNet-A [35] + cutout	96.66 ± 0.06	3.2	3,150
SNAS [46]	97.02	2.9	1.5
ENAS [34] + cutout	97.11	4.6	0.5
DARTS [27] + cutout	97.24 ± 0.09	3.4	4
NAONet [31]	97.02	28.6	200
PNAS [26] + cutout	97.17 ± 0.07	3.2	-
GHN [51] + cutout	97.16 ± 0.07	5.7	0.8
D-VAE [52]	94.80	-	-
NGE [23] + cutout	97.40	-	**0.1**
BONAS-A [37] + cutout	97.31	3.45	2.5
CTNAS [8] + cutout	97.41 ± 0.04	3.6	0.3
TNASP + cutout(avg)	$\mathbf{97.43 \pm 0.04}$	3.6 ± 0.1	0.3
TNASP + cutout(best)	**97.48**	3.7	0.3

Table 3: Comparison with other methods in DARTS [27] search space on CIFAR-10. "cutout": evaluate the searched cells using cutout [13] data augmentation. "G·D": GPU days.

NEURAL INFORMATION PROCESSING SYSTEMS
NeurIPS 2021

ANS@ICT 快手

4. Experiments

4.4 Search results on ProxylessNAS search space

Method	**Params(M)**	**FLOPs(M)**	**Top-1(%)**	**Top-5(%)**
FBNet-C [15]	5.5	375	74.9	92.1
Proxyless (GPU) [1]	7.0	457	75.1	92.5
SPOS [6]	5.4	472	74.8	-
RLNAS [17]	5.3	473	75.6	92.6
Neural Predictor [14]	6.4 *	536 *	74.75 ± 0.09	-
NAO [10]	6.5	590	75.5	92.5
TNASP-A	5.0	433	75.1	92.3
TNASP-B	5.1	478	75.5	92.5
TNASP-C	5.3	497	**75.8**	**92.7**

Table 8: Comparison with other methods on ImageNet. *: We compute these information by their released model structure.

Reference:

[1] Elsken T, Metzen J H, Hutter F. Neural architecture search: A survey[J]. The Journal of Machine Learning Research, 2019, 20(1): 1997-2017.

[2] Luo R, Tian F, Qin T, et al. Neural architecture optimization[J]. arXiv preprint arXiv:1808.07233, 2018.

[3] Zhang M, Jiang S, Cui Z, et al. D-vae: A variational autoencoder for directed acyclic graphs[J]. arXiv preprint arXiv:1904.11088, 2019.

[4] White C, Neiswanger W, Savani Y. Bananas: Bayesian optimization with neural architectures for neural architecture search[J]. arXiv preprint arXiv:1910.11858, 2019, 1(2).

[5] Shi H, Pi R, Xu H, et al. Bridging the gap between sample-based and one-shot neural architecture search with bonas[J]. arXiv preprint arXiv:1911.09336, 2019.

[6] Ru B, Wan X, Dong X, et al. Interpretable Neural Architecture Search via Bayesian Optimisation with Weisfeiler-Lehman Kernels[J]. arXiv preprint arXiv:2006.07556, 2020.

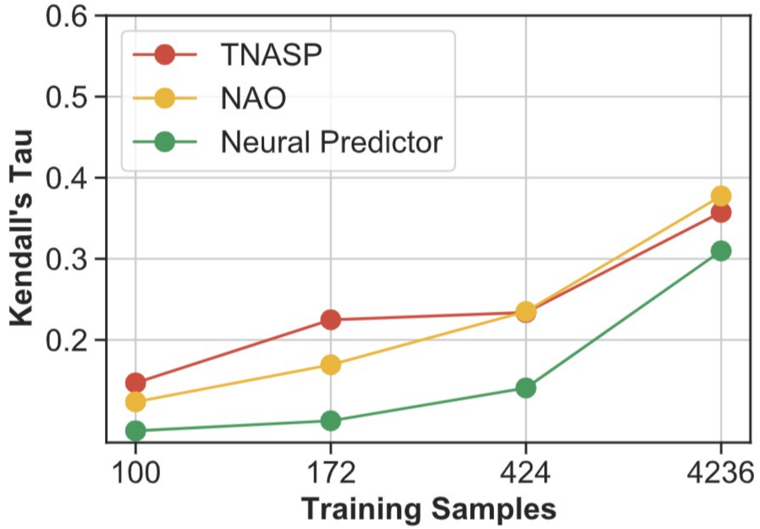
[7] Chen Y, Guo Y, Chen Q, et al. Contrastive Neural Architecture Search with Neural Architecture Comparators[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 9502-9511.

Thank You !

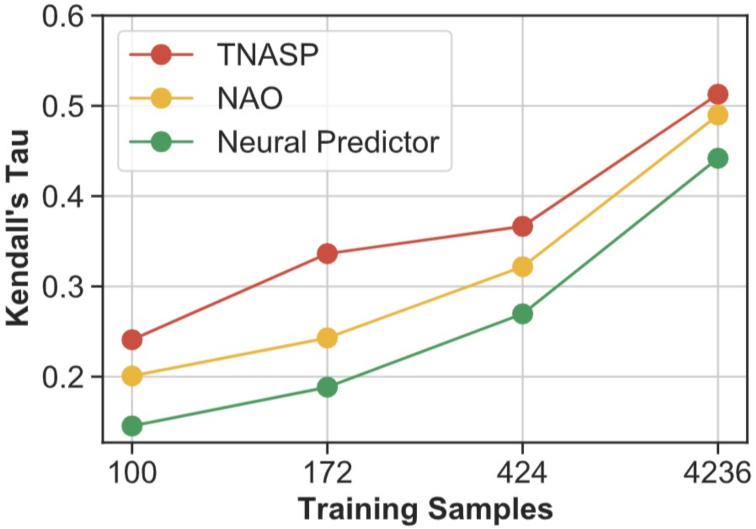
Extra Slides

4. Experiments

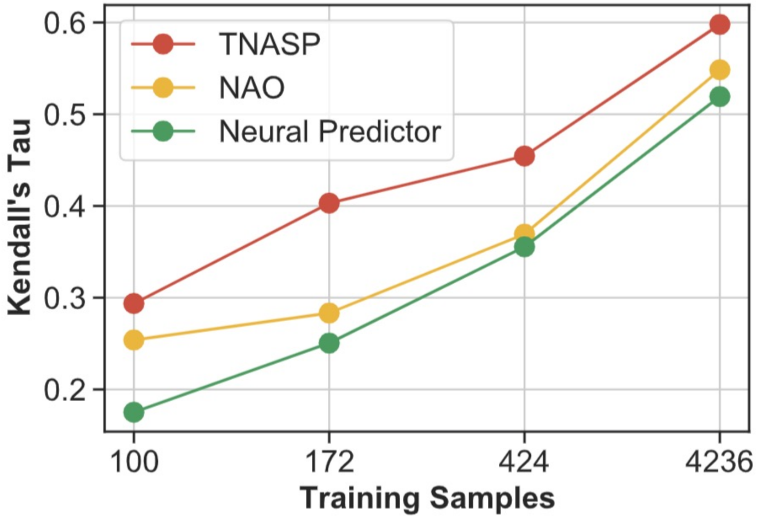
4.1 Ranking results on NAS-Bench-101



(a) Top 10% architectures



(b) Top 20% architectures

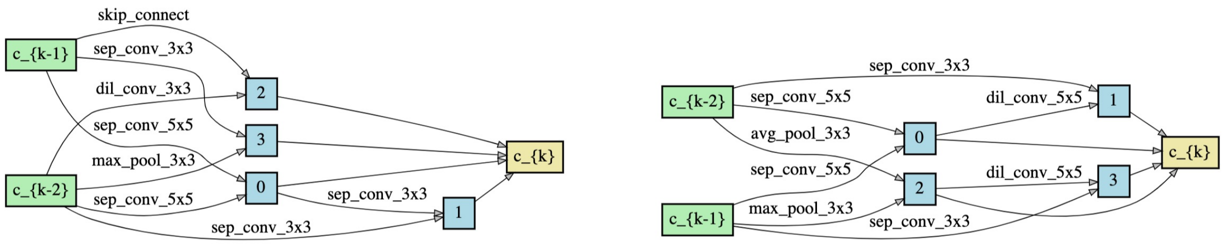


(c) Top 30% architectures

Figure 5: Ranking results over different top portions of good architectures.

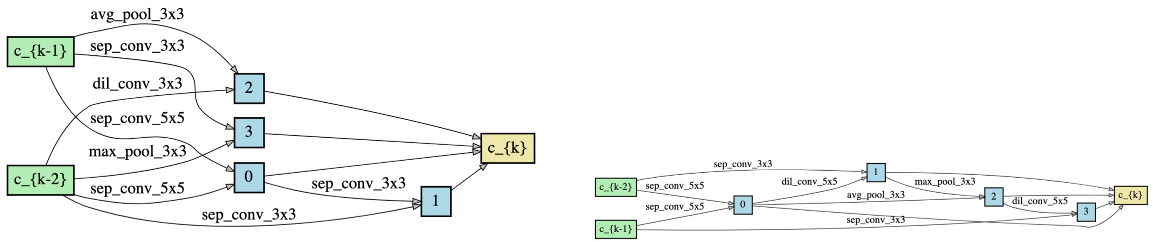
4. Experiments

4.3 Visual results on DARTS search space

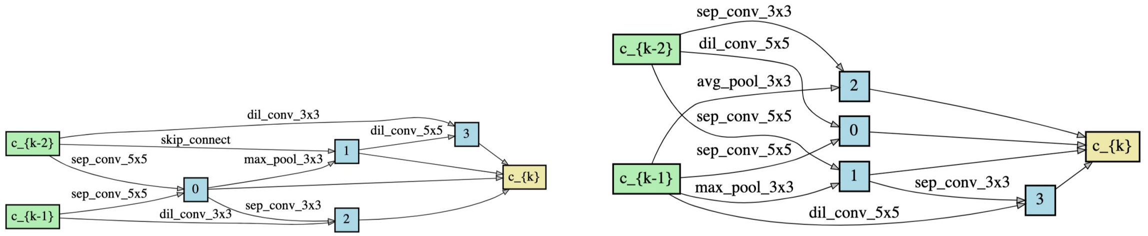


(a) Normal Cell (b) Reduction Cell

Figure 3: Our best searched normal cell and reduction cell.



(a) TNASP-b



(b) TNASP-c

Figure 6: Other searched cells in DARTS search space. Left side are normal cells and right side are reduction cells.

4. Experiments

4.4 Visual results on ProxylessNAS search space

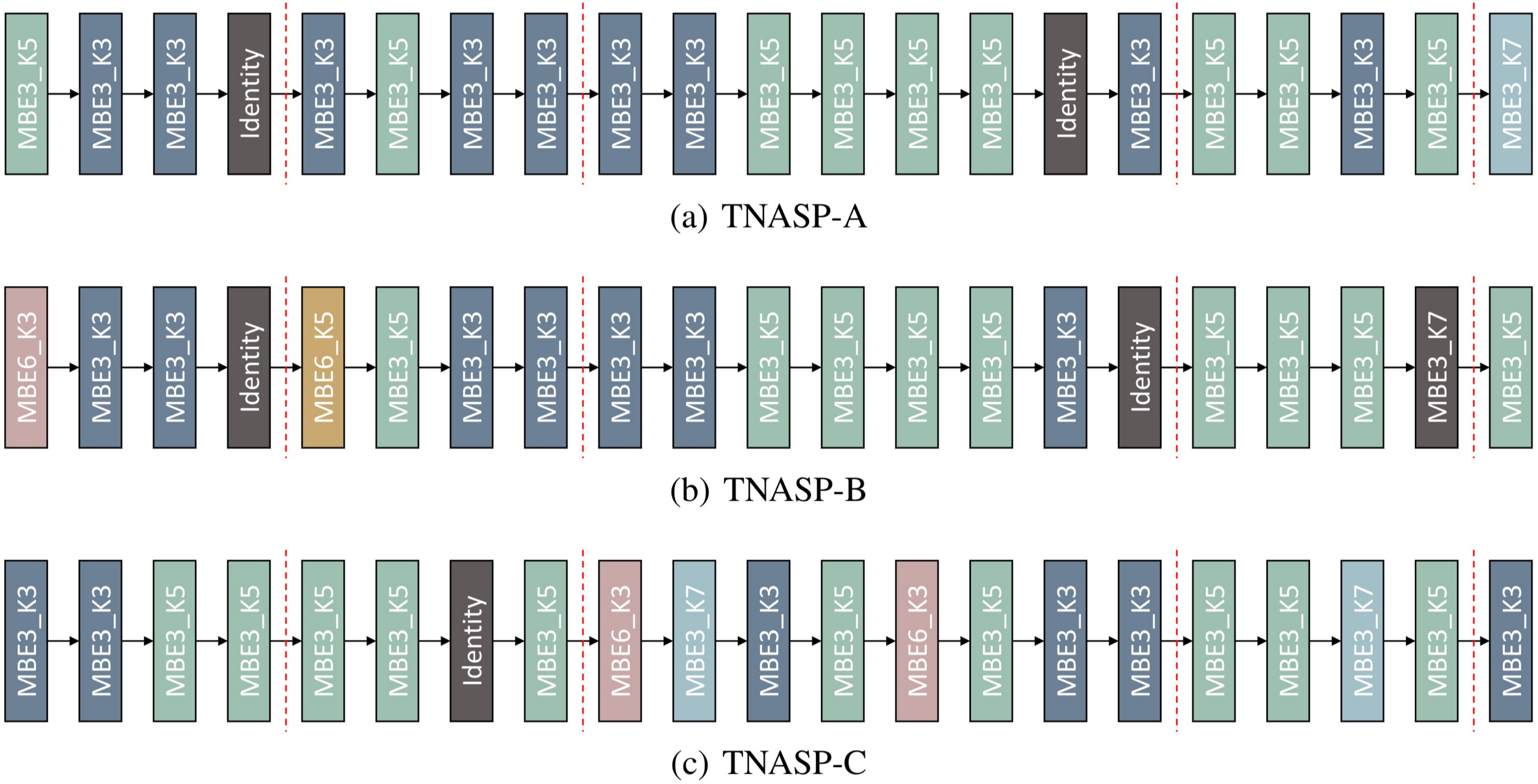


Figure 7: Our searched architectures in MobileNet-like search space.