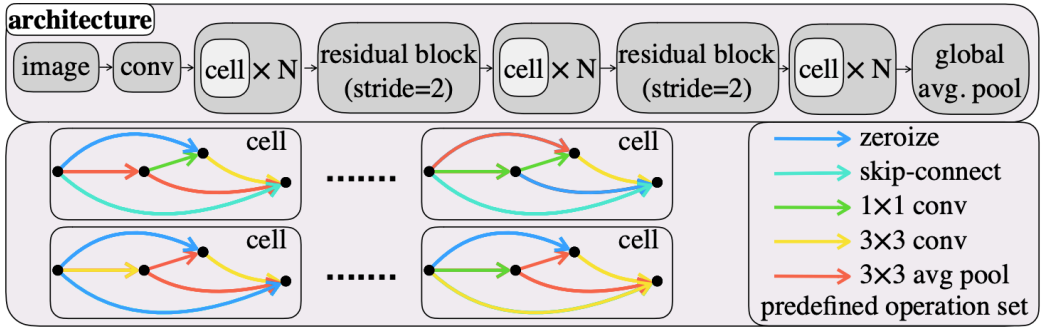
Neural Architecture Search on ImageNet in Four GPU Hours: A Theoretically Inspired Perspective

Wuyang Chen, Xinyu Gong, Zhangyang Wang

UT Austin

https://github.com/VITA-Group/TENAS

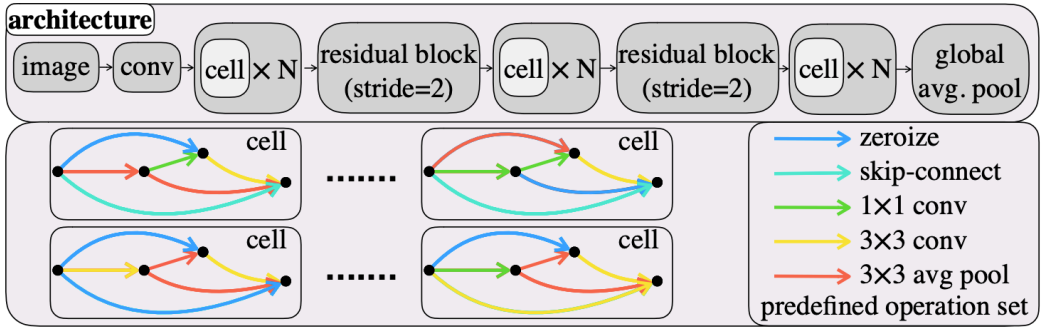
Neural Architecture Search: Definition



- A search space $\mathcal{A}$
  - Different types of layers.
  - Different topologies.
  - Different widths / depths.
- Goal: find an architecture a* from $\mathcal{A}$ s.t. f can be maximized. $a^* = \arg\max_{a \in \mathcal{A}} f(a)$
  $f : \mathcal{A} \subset \mathbb{R}^n \rightarrow \mathbb{R}$
- A search (optimization) algorithm
  - Sample one architecture from the space.
  - Evaluate it's value (accuracy, loss, etc.).
  - Find a next better sampling.
  - Reinforcement Learning, Evolution, Differentiable Search, etc.
- Evaluation: train a* from scratch.

[1] Dong, Xuanyi, and Yi Yang. "Nas-bench-201: Extending the scope of reproducible neural architecture search." *ICLR 2020*.

Neural Architecture Search: Definition



- A search space $\mathcal{A}$
  - Different types of layers.
  - Different topologies.
  - Different widths / depths.
- Goal: find an architecture a* from $\mathcal{A}$ s.t. f can be maximized. $a^* = \arg\max_{a \in \mathcal{A}} f(a)$
  $f : \mathcal{A} \subset \mathbb{R}^n \rightarrow \mathbb{R}$
- A search (optimization) algorithm
  - Sample one architecture from the space.
  - Evaluate it's value (accuracy, loss, etc.).
  - Find a next better sampling.
  - Reinforcement Learning, Evolution, Differentiable Search, etc.
- Train a* from scratch.

[1] Dong, Xuanyi, and Yi Yang. "Nas-bench-201: Extending the scope of reproducible neural architecture search." *ICLR 2020*.

NAS Heavily Counts on Architecture Evaluation

- Training based evaluation
  - High computation cost & slow!
  - Proxy evaluation: early stopping, weight sharing, etc.
  - Train another predictor network to predict architecture's performance.
- ***How to optimize*** NAS at network's initialization w.o. any training?

  ➔ Significantly reduce the NAS search cost.
- Can we define ***how to evaluate*** in NAS by analyzing the trainability & expressivity of architectures?

Analysis of NN's Trainability & Expressivity @ *Init*

- Trainability: can NN be easily optimized by Gradient Descent?
  - Any gradient vanishing/explosion?
  - Any overfitting to training set / easy classes?
- Expressivity: can NN represent complex functions?
- Trainability: ResNet > Vgg
- Expressivity: Vgg vs. ResNet (?)

Trainability via Condition Number of NTK

Strong Correlation w.r.t. Accuracy

Jacobian: $$J_{i\alpha}(\boldsymbol{x}) = \partial_{\theta_\alpha} z_i^L(\boldsymbol{x})$$

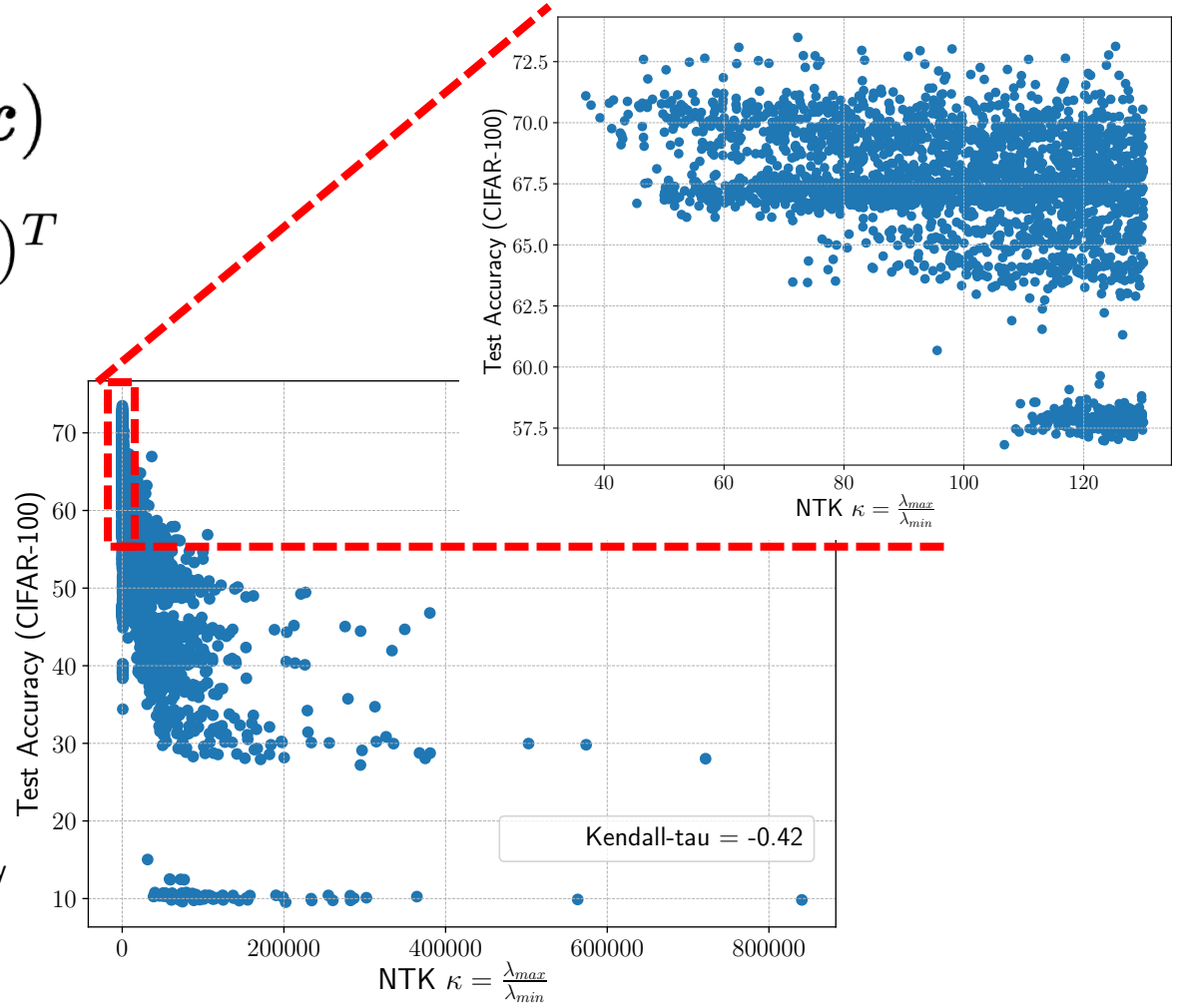
NTK $(\mathbb{S}_{++}^n)$ $$\hat{\Theta}(\boldsymbol{x}, \boldsymbol{x}') = J(\boldsymbol{x})J(\boldsymbol{x}')^T$$

Network training dynamics[1]:

$$\mu_t(\boldsymbol{X}_{\text{train}}) = (\mathbf{I} - e^{-\eta\hat{\Theta}_{\text{train}}t})\boldsymbol{Y}_{\text{train}}$$

$$\mu_t(\boldsymbol{X}_{\text{train}})_i = (\mathbf{I} - e^{-\eta\lambda_i t})\boldsymbol{Y}_{\text{train},i}$$

NTK condition number[2]: $$\kappa_{\mathcal{N}} = \frac{\lambda_0}{\lambda_m}$$



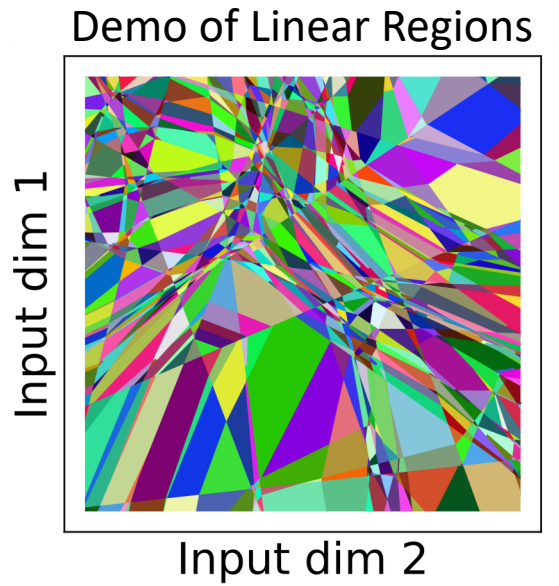
[1] Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Novak, R., Sohl-Dickstein, J., & Pennington, J. "Wide neural networks of any depth evolve as linear models under gradient descent." *Neurips 2019*.
[2] Xiao, L., Pennington, J., & Schoenholz, S. "Disentangling trainability and generalization in deep learning." *ICML 2020*.

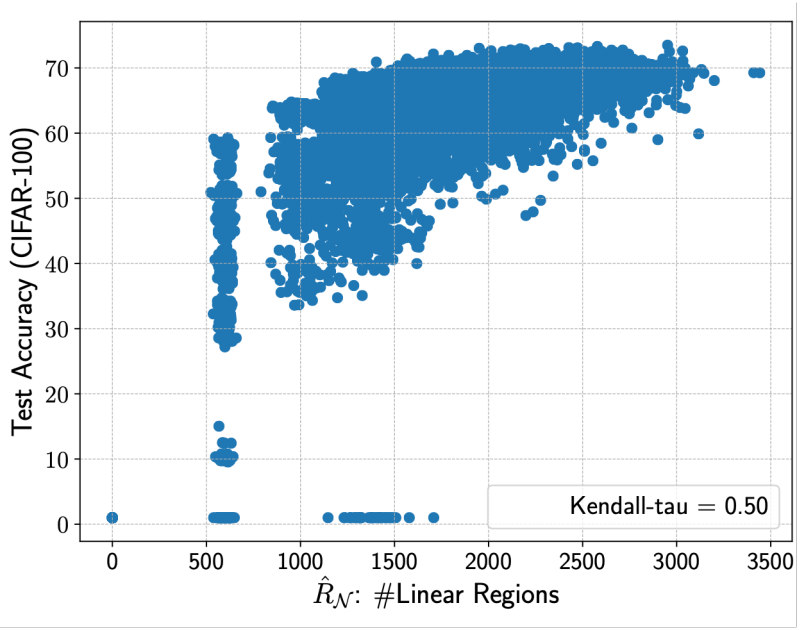
https://github.com/VITA-Group/TENAS

Expressivity via #Linear Regions

Strong Correlation w.r.t. Accuracy

- Number of Linear Regions[1] = Number of **unique ReLU activation patterns** under given input samples[2].

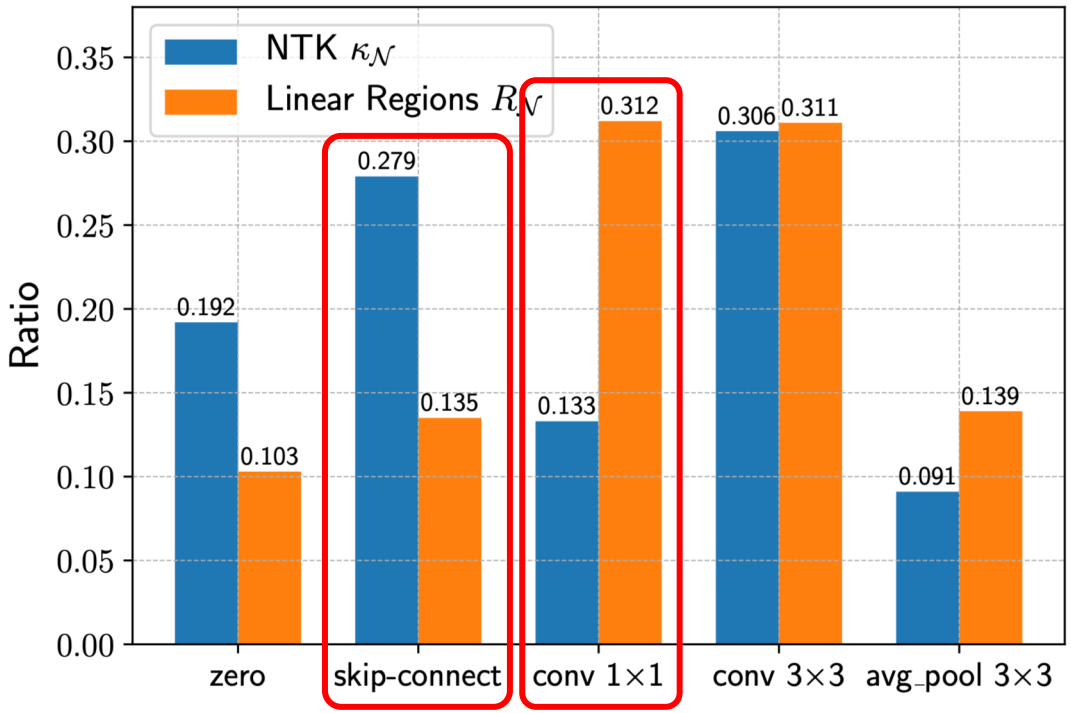




[1] Hanin, B., & Rolnick, D. "Complexity of linear regions in deep networks." *ICML 2019*.
[2] Xiong, H., Huang, L., Yu, M., Liu, L., Zhu, F., & Shao, L. "On the number of linear regions of convolutional neural networks." *ICML 2020*

κ_N v.s. R_N: Different Operator Preference

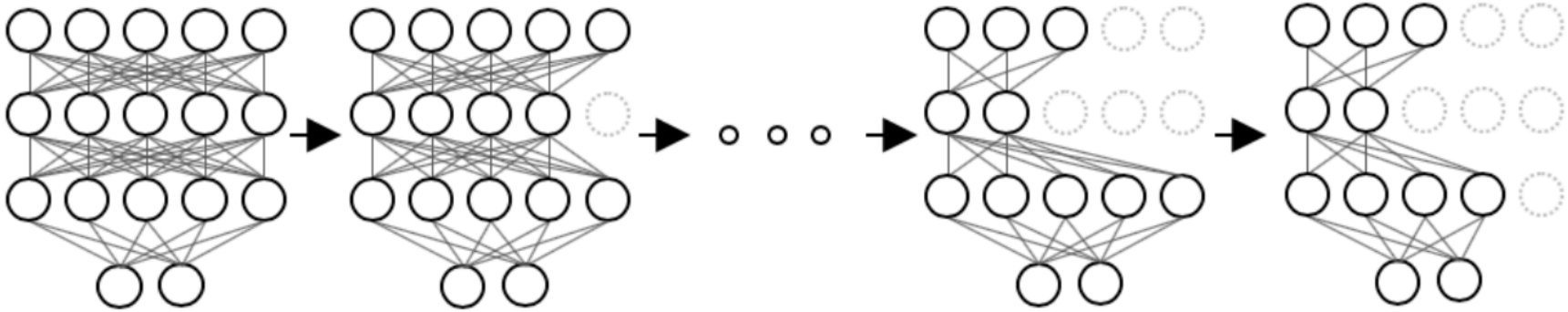
- κ_N: more skip-connect
  ➔ easier to train
- R_N: conv1x1
  ➔ stronger expressivity

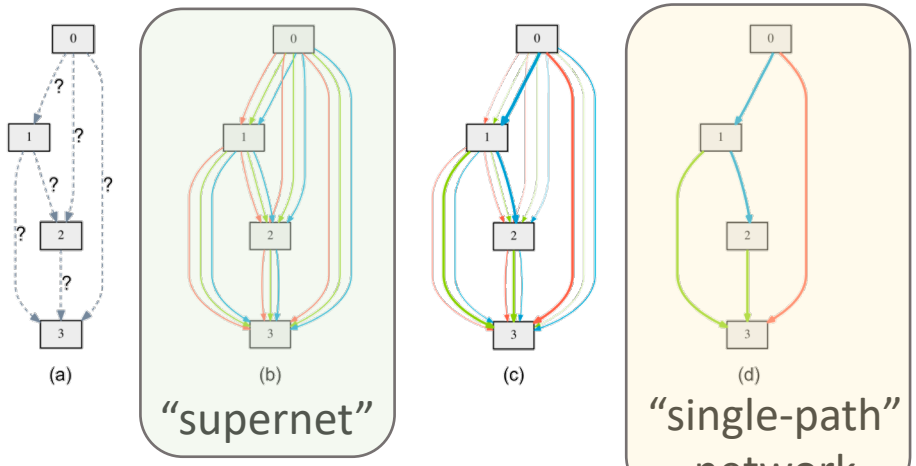


https://github.com/VITA-Group/TENAS

TE-NAS: Training-free & Label-free Efficient NAS

- Prune a supernet by ranking the importance of operators.



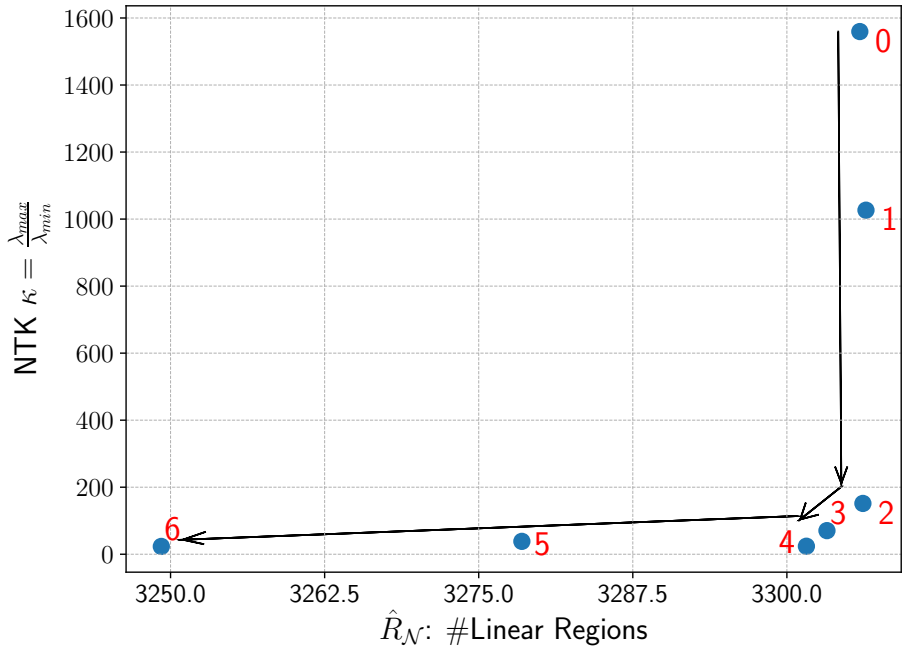


TE-NAS: Training-free & Label-free Efficient NAS

- Prune a supernet by ranking the importance of operators.
- First improve supernet's trainability ➔ then preserve expressivity.

Algorithm 1: TE-NAS: Training-free Pruning-based NAS via Ranking of $\kappa_{\mathcal{N}}$ and $\hat{R}_{\mathcal{N}}$.

Input: supernet $\mathcal{N}_0$ stacked by cells, each cell has E edges, each edge has $|\mathcal{O}|$ operators, step $t = 0$.
while $\mathcal{N}_t$ *is not a single-path network* **do**
  for *each operator* o_j *in* $\mathcal{N}_t$ **do**
    $\Delta\kappa_{t,o_j} = \kappa_{\mathcal{N}_t} - \kappa_{\mathcal{N}_t \backslash o_j}$ ▷ the higher $\Delta\kappa_{t,o_j}$ the more likely we will prune o_j
    $\Delta R_{t,o_j} = R_{\mathcal{N}_t} - R_{\mathcal{N}_t \backslash o_j}$ ▷ the lower $\Delta R_{t,o_j}$ the more likely we will prune o_j
  Get importance by $\kappa_{\mathcal{N}}$: $s_\kappa(o_j) =$ index of o_j in descendingly sorted list $[\Delta\kappa_{t,o_1}, ..., \Delta\kappa_{t,o_{|\mathcal{N}_t|}}]$
  Get importance by $R_{\mathcal{N}}$: $s_R(o_j) =$ index of o_j in ascendingly sorted list $[\Delta R_{t,o_1}, ..., \Delta R_{t,o_{|\mathcal{N}_t|}}]$
  Get importance $s(o_j) = s_\kappa(o_j) + s_R(o_j)$
  $\mathcal{N}_{t+1} = \mathcal{N}_t$
  for *each edge* e_i, $i = 1, ..., E$ **do**
    $j^* = \arg\min_j \{s(o_j) : o_j \in e_i\}$ ▷ find the operator with greatest importance on each edge.
    $\mathcal{N}_{t+1} = \mathcal{N}_{t+1} \backslash o_{j^*}$
  $t = t + 1$
return *Pruned single-path network* $\mathcal{N}_t$.



https://github.com/VITA-Group/TENAS

Fast & Accurate: NAS-Bench-201 & DARTS Space

Table 1: Comparison with state-of-the-art NAS methods on NAS-Bench-201. Test accuracy with mean and deviation are reported. "optimal" indicates the best test accuracy achievable in NAS-Bench-201 search space.

Architecture	CIFAR-10	CIFAR-100	ImageNet-16-120	Search Cost (GPU sec.)	Search Method
ResNet (He et al., 2016)	93.97	70.86	43.63	-	-
RSPS (Li & Talwalkar, 2020)	87.66(1.69)	58.33(4.34)	31.14(3.88)	8007.13	random
ENAS (Pham et al., 2018)	54.30(0.00)	15.61(0.00)	16.32(0.00)	13314.51	RL
DARTS (1st) (Liu et al., 2018b)	54.30(0.00)	15.61(0.00)	16.32(0.00)	10889.87	gradient
DARTS (2nd) (Liu et al., 2018b)	54.30(0.00)	15.61(0.00)	16.32(0.00)	29901.67	gradient
GDAS (Dong & Yang, 2019)	93.61(0.09)	70.70(0.30)	41.84(0.90)	28925.91	gradient
NAS w.o. Training (Mellor et al., 2020)	91.78(1.45)	67.05(2.89)	37.07(6.39)	4.8	training-free
TE-NAS (ours)	**93.9(0.47)**	**71.24(0.56)**	**42.38(0.46)**	1558	training-free
Optimal	94.37	73.51	47.31	-	-

Table 2: Comparison with state-of-the-art NAS methods on CIFAR-10.

Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	Search Method
AmoebaNet-A (Real et al., 2019)	3.34(0.06)	3.2	3150	evolution
PNAS (Liu et al., 2018a)*	3.41(0.09)	3.2	225	SMBO
ENAS (Pham et al., 2018)	2.89	4.6	0.5	RL
NASNet-A (Zoph et al., 2018)	2.65	3.3	2000	RL
DARTS (1st) (Liu et al., 2018b)	3.00(0.14)	3.3	0.4	gradient
DARTS (2nd) (Liu et al., 2018b)	2.76(0.09)	3.3	1.0	gradient
SNAS (Xie et al., 2018)	2.85(0.02)	2.8	1.5	gradient
GDAS (Dong & Yang, 2019)	2.82	2.5	0.17	gradient
BayesNAS (Zhou et al., 2019)	2.81(0.04)	3.4	0.2	gradient
ProxylessNAS (Cai et al., 2018)†	2.08	5.7	4.0	gradient
P-DARTS (Chen et al., 2019)	2.50	3.4	0.3	gradient
PC-DARTS (Xu et al., 2019)	2.57(0.07)	3.6	0.1	gradient
SDARTS-ADV (Chen & Hsieh, 2020)	2.61(0.02)	3.3	1.3	gradient
TE-NAS (ours)	2.63(0.064)	3.8	0.05‡	training-free

* No cutout augmentation.
† Different space: PyramidNet (Han et al., 2017) as the backbone.
‡ Recorded on a single GTX 1080Ti GPU.

Table 3: Comparison with state-of-the-art NAS methods on ImageNet under the mobile setting.

Architecture	Test Error(%) top-1	Test Error(%) top-5	Params (M)	Search Cost (GPU days)	Search Method
NASNet-A (Zoph et al., 2018)	26.0	8.4	5.3	2000	RL
AmoebaNet-C (Real et al., 2019)	24.3	7.6	6.4	3150	evolution
PNAS (Liu et al., 2018a)	25.8	8.1	5.1	225	SMBO
MnasNet-92 (Tan et al., 2019)	25.2	8.0	4.4	-	RL
DARTS (2nd) (Liu et al., 2018b)	26.7	8.7	4.7	4.0	gradient
SNAS (mild) (Xie et al., 2018)	27.3	9.2	4.3	1.5	gradient
GDAS (Dong & Yang, 2019)	26.0	8.5	5.3	0.21	gradient
BayesNAS (Zhou et al., 2019)	26.5	8.9	3.9	0.2	gradient
P-DARTS (CIFAR-10) (Chen et al., 2019)	24.4	7.4	4.9	0.3	gradient
P-DARTS (CIFAR-100) (Chen et al., 2019)	24.7	7.5	5.1	0.3	gradient
PC-DARTS (CIFAR-10) (Xu et al., 2019)	25.1	7.8	5.3	0.1	gradient
TE-NAS (ours)	26.2	8.3	6.3	0.05	training-free
PC-DARTS (ImageNet) (Xu et al., 2019)†	24.2	7.3	5.3	3.8	gradient
ProxylessNAS (GPU) (Cai et al., 2018)†	24.9	7.5	7.1	8.3	gradient
TE-NAS (ours)†	24.5	7.5	5.4	0.17	training-free

† The architecture is searched on ImageNet, otherwise it is searched on CIFAR-10 or CIFAR-100.

https://github.com/VITA-Group/TENAS

Thank you!

Code

Paper

https://github.com/VITA-Group/TENAS