TNAS: Neural Network Architecture Search Sampling Module Based on Variational Autoencoder

Kefan Yan, Shunlong Wang, Yuwei Guan, Ailian Jiang*
Taiyuan University of Technology, Taiyuan, China
**Corresponding Author.*

Abstract

Neural Architecture Search (NAS) searches the architecture of neural network automatically in a large search space. The search space typically contains billions of network architectures, which makes it computationally expensive to search for the best-performing architecture. One-shot and gradient-based NAS methods have achieved good results in various computer vision tasks. Even though they achieved success, the current sampling methods are either fixed or manual, all of which are ineffective. In this paper, we propose a learning sampling module for neural architecture search (NAS) named TNAS, based on variational auto-encoder (VAE). This module can be easily embedded into the existing weight sharing NAS framework such as one-shot approach and gradient-based approach, and significantly improve the performance of search results. In NasNet-like search space, TNAS produced a series of competitive results on CIFAR-10 and ImageNet. In addition, combined with the one-shot method, our method obtains the latest results of ImageNet classification model under 400M FLOPs frequency hopping with a probability of 77.4% in a ShuffleNet-like search space. Finally, we performed an in-depth analysis of TNAS on the NAS-BENCH-201 dataset to verify the effectiveness of our proposed approach.

Keywords: *Neural Architecture Search, deep learning, algorithm*

I. Introduction

Through the design of novel neural structures, deep neural networks have promoted the development of a variety of influential applications [1, 2] significantly. Automatically designing the neural network structures without manual intervention, known as neural structure search (NAS), has been in the spotlight in recent years. It achieves state-of-the-art performance in many areas, such as image recognition [3, 4], object detection [5, 6] and semantic segmentation [7].

In general, the search space for NAS tasks is huge. For example, NasNet [3] proposed a search space that contains 6×109 viable cells. Searching such a huge design space costs 2,400 GPU days. The weight allocation mechanism has been proved to be an effective way to improve the efficiency of NAS search. Recent algorithms for high-efficiency NAS fall into two categories: one-shot approaches [8] and gradient-based approaches [9]. In the one-shot approach, previous studies have focused on the use of fixed sampling strategies [8, 10, 11]. In gradient-based approaches, the search usually does not require a sampling program [9, 12] or manual sampling [13].Despite these NAS methods have succeeded in various benchmarks, these sampling methods do not interactively learn the architectural distribution as the search process progresses, making the sampling process ineffective.

We propose a VAE-based NAS learning and interactive sampling module TNAS and investigated the application of TNAS modules by combining TNAS with two mainstream NAS approaches, namely the one-shot approach [10, 14] and gradient-based approach [9, 15], both of which have achieved the latest performance in neural structure search tasks.

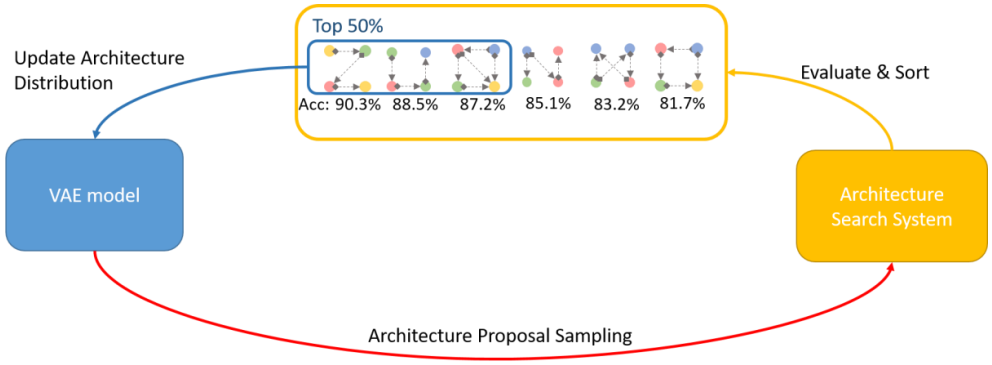

Figure 1: Schematic diagram of TNAS mechanism.

We verify the TNAS in different search spaces for image recognition on CIFAR-10 and ImageNet datasets. In the NasNet-like search space, we apply TNAS to the gradient-based approach and the one-shot approach. Specifically, combined with the one-shot method, TNAS has a test error of 2.26% on CIFAR-10, which is better than the most advanced NAS method. In a Shufflenet-like search space, the combination of TNAS and one-shot achieves a top-1 accuracy of 77.4% with 365M FLOPs in ImageNet classification, which is 1.1% higher than that of the most advanced Efficient-B0, and the computational complexity is reduced by 6.5%. Finally, we carry out analysis deeply on TNAS against the NAS-BENCH-201 benchmark to demonstrate the effectiveness of our proposed approach to architecture sampling.

II. Related Work

Neural Architecture Search (NAS). In recent years, the design of efficient neural networks has mainly shifted from using human knowledge to an automated approach known as neural structure search (NAS). NAS methods in earlier time used reinforcement learning (RL) [3] or evolutionary strategies [4] to search across thousands of individually evaluated networks, which computational cost is large. Recent work can be divided into two categories: One-shot methods [10, 11] and gradient-based approach [9, 15]. These two methods have obtained the latest results on a series of benchmark datasets [5-7].

Sampling. Some previous work has applied sampling methods to the NAS framework. Bender et al. [8] randomly zeroes out a subset of operations in order to perform the one-shot method during the process of training super-network. Guo et al. [10] adopt uniform sampling, while Chu et al. [11] adopts fair and uniform sampling to reduce training bias in a single model. As for the gradient-based approach, Cai et al. [15] proposes a polynomial distributed sampling to alleviate the large memory consumption problem in the traditional gradient NAS approach [9]. Our method provides a new perspective on sampling methods for NAS applications.

III. TNAS

3.1 Motivation

Many previous works of NAS discuss about the architecture distribution $\Gamma(\alpha)$ modeling. Typical $\Gamma(\alpha)$ is the architecture that uses fixed distribution (e.g., uniform distribution [8, 10]), or hand-crafted architecture distributions (e.g., Gumbel Softmax distributions [9, 12]).

The expected structure distribution $\Gamma(\alpha)$ can interactively learn from share the network weights during searching. To this end, we put forward a kind of learning architecture distribution, including the distribution of t moment $\Gamma_t(\alpha)$ learn from new incoming architecture α, the distribution of t - 1 hour earlier architecture is $\Gamma_{t-1}(\alpha)$, we can study the process forms will be distributed to:

$$\Gamma_t(\alpha) \leftarrow T(\Gamma_{t-1}(\alpha), \text{Metric}_{t-1}(\alpha)) \quad (1)$$

In equation 1, the Metric above represents a predefined measure like precision or loss, and T represents a random process. The distribution of architectures at time t is learned by both the prior knowledge on architecture distribution Γ (α) and new architectures data α at t−1 time, as in Figure 1.

3.2 The use of VAE

VAE approximates the marginal probability of the architecture distribution $\log p_\theta(\alpha)$ by maximizing the lower bound of variation:

$$-D_{KL}(q_\varphi(z \mid \alpha) \| p_\theta(z)) + E_{z \sim q_\varphi(z \mid \alpha)}[\log p_\theta(\alpha \mid z)] \quad (2)$$

Specifically, the left term is the Kullback-Leibler divergence between the approximate posteriori (identifying the model) $q_\phi(z|\alpha)$ and the prior p(z). The right item is the expectation of rebuilding loss, which $p_\theta(\alpha|z)$ represents the variational inference. VAE extract latent variable z according to the prior distribution $p_\theta(z)$, and input z into inference model $p_\theta(\alpha|z)$. In general, the prior distribution $P_\theta(z)$ can be set as the standard Gaussian distribution N(0, I).

Let $\alpha=\{o_i\}_i^n$ be the sequence of operations o_i representing an architectural string, where n is the total number of nodes in a network or cell. We use the encoder E to take α as the input and map it to the parameters μ_α and σ_α corresponding to the normal distribution $N(\mu_\alpha, \sigma_\alpha)$. Symmetrically, the decoder D is used to map z_e to the reconstructed architecture $\bar{\alpha}$, where $z_e \sim N(\mu_e, \sigma_e)$. The encoder E corresponds to $q_\phi(z|\alpha)$ and the decoder D maps to $p_\theta(\alpha|z)$. In specific, encoders E and decoder D are implemented by two TRANSFORMER networks.

3.3 The use of transformer

Both encoders and decoders in TNAS are made of multiple transformer [16] components stacked on top of each other. We mark the implicit state after the first layer as $Z^l=\{Z^l_1,Z^l_2,...,Z^l_N\}$. As for $Z^l=T(Z^{l-1})$, T represents a transformer block with nhead's head. The computing formula of the first transformer block is listed bellow:

$$Q_k = H^{l-1} W^l_{qk} \quad (3)$$

$$K_k = H^{l-1} W^l_{kk} \quad (4)$$

$$V_k = H^{l-1} W^l_{vk} \quad (5)$$

$$\hat{H}^l_k = \text{softmax}\left(\frac{Q_k K_k^T}{\sqrt{d_k}}\right) V_k \quad (6)$$

$$\hat{H}^l = \text{concatenate}(\hat{H}^l_1, \hat{H}^l_2, ..., \hat{H}^l_{n_{head}}) \quad (7)$$

$$H^l = \text{ReLU}(\hat{H}^l W_1 + b_1) W_2 + b_2 \quad (8)$$

Among these, Q_k, K_k, V_k respectively represent attention operations "Query", "Key" and "Value" of the kth layer. The usage of transformer architecture can enhance the ability of encoders and decoders to focus on the global and can compute in parallel to boost the computing efficiency. Transformer module can be embedded into VAE modules easily and improve the efficiency of VAE.

IV. TNAS for Neural Architecture Search

4.1 Weight distribution mechanism

The weight sharing method constructs a hypernetwork G (α, W), which contains all the valid structures α and the shared weights W. Any structure α is a subgraph of the hypernetwork and inherits the corresponding weights W_α. Therefore, we can transform the training form of weight allocation mechanism into solving a two-layer optimization problem:

$$W_\alpha^* = \arg\min_{W_\alpha} E_{\alpha\sim\Gamma(\alpha)}[L_{train}(G(\alpha, W_\alpha))] \tag{9}$$

$$\alpha^* = \arg\min_{\alpha\in\alpha} L_{val}(G(\alpha, W_\alpha^*)) \tag{10}$$

4.2 One-shot framework

Review One-Shot Architecture Search. In general, the one-shot method [9,12,15] consists of three steps: 1) training the weight-sharing super network. 2) Rank the architecture according to its performance on the supernetwork. 3) Exporting the final architecture through the search strategy. Previous methods typically used hand-made sampling methods [8, 10, 11].

The one-shot method was combined with TNA. TNAS-OS adopts a one-time framework and optimizes the sampling distribution:

$$\Gamma(\alpha) = (1-\varepsilon) * U(\alpha) + \varepsilon * p_\theta(\alpha \mid z) \tag{11}$$

The U(α) is a random uniform distribution, and $p_\theta(\alpha|z)$ is for variational distribution and mining parameters. In hypernet training, structures are randomly selected from $p_\theta(\alpha|z)$ and U(α) through possibilities ε and $1-\varepsilon$. As the search phase continues, the volume of data increases, so the super network trains more samples from TNAS. Variational distribution $p_\theta(\alpha|z)$ will converge to generate good performance of architecture during training. In the search stage, from $p_\theta(\alpha|z)$ sampling initial seeds, and then use genetic algorithm to search, such as Guo [10].

4.3 Gradient-based framework

Review Gradient-based architecture search. Gradient-based approach uses the independent discrete structural parameters to model the network.

Combine the Gradient-based method with TNAS.

$$\Gamma(\alpha) = p_\theta(\alpha|z) \tag{12}$$

We extract an architecture α_i from $p_\theta(\alpha|z)$, and update the super network weight $W_{\alpha i}$ and multinomial distribution parameter x_i accordingly.

V. Experiments

5.1 NASNET-like search space

Our TNAS is evaluated on gradient-based and one-shot approaches and compared with the latest NAS approaches in the same search space. We use "OS" to represent the one-shot method and "G" to represent the gradient-based

method.

Table 1: Comparison of the most advanced architectures on CIFAR-10 in the Nasnet-like search space.

Architecture	Test Error	Params	Search Method
	(%)	(M)	
NASNet-A* [3]	2.65	3.3	RL
AmoebaNet-B* [4]	2.55±0.05	2.8	Evolution
Hierarchical Evolution [17]	3.75±0.12	15.7	Evolution
PNAS [13]	3.41±0.09	3.2	SMBO
ENAS* [18]	2.89	4.6	RL
NAO-weight-sharing * [19]	3.53	2.5	Gradient
DARTS* [9]	2.76±0.09	3.4	Gradient
SNAS* [12]	2.76±0.09	3.4	Gradient
GraphHypernet* [20]	4.3±0.1	5.1±0.6	Gradient
DSO-share* [21]	2.84±0.07	3.0	Gradient
BayesNAS* [22] + λ = 0.01	2.81±0.04	3.4	Gradient
TNAS-G*	2.40	4.4	Gradient
TNAS-OS*†	2.50	3.4	Evolution
TNAS-OS*	2.26	5.2	Evolution

* indicates that the model is trained with truncation. OS stands for One-Shot method. G represents the gradient-based method. † represents the hard parameter constraint for the search phase.

Table 2: Comparison of the latest schema on ImageNet (200m-400m FLOPs) under Shufflenet-like search space.

Architecture	FLOPs	Params	Top-1 Acc.	Top-5 Acc.
	(M)	(M)	(%)	(%)
ShuffleNet V2 1.5 [23]	300	-	72.6	-
MobileNet V2 1.0 [24]	300	3.4	72.0	91.0
MobileNet V3 Large 1.0 [25]	219	5.4	75.2	92.2
MnasNet-A2 [26]	340	4.8	75.6	92.7
FBNet-B [27]	295	4.5	74.1	-
Proxyless GPU [15]	320	4.0	74.6	92.2
Single-Path NAS [10]	365	4.3	75.0	92.2
FairNAS-A [11]	388	4.6	75.3	92.4
EfficientNet-B0* [28]	390	5.3	76.3	93.2
Baseline	360	6.7	77.1	93.3
TNAS-OS	365	6.7	77.4	93.6

* Represents the results of using the AutoAugment report.

Performance on CIFAR-10. Table 1 indicates the results. Combined with the gradient-based method, the test error of this method can reach 2.4% when the parameter is 4.4M. Combined with the one-shot method, we give two results. One is to restrict the search with 3.5m parameters and obtain 2.5% test error. The other is training without any constrains, achieving 2.26% of 5.2m parameter test error.

5.2 ImageNet classification in mobile environments

Comparison with the latest methods. We choose models with FLOPS in the range of 200M to 400M. As shown in Table 2, it is worth noting that TNAS-OS far outperforms the most advanced models. In particular, TNAS is 1.1%

more accurate than EfficentNet-B0 [28]. To our knowledge, TNAS-OS is the first model to exceed 77% at 400M FLOPs without the use of strong regularization and data enhancement techniques (such as AutoAugment).

5.3 TNA on the NAS-BENCH-201 dataset

NAS-Bench-201 designs a carefully constructed unitary-search space and provides basic fact measures for all 15625 architectures on the CIFAR-10, CIFAR-100, and ImageNet-16-120 datasets.

Versus random sampling. We compare TNAS with random sampling, and the results of the three datasets are shown in Figure 2. Comparing VAE TrainData and RandomTopK, it is noted that the performance of the model sampled by the former is much better than the latter on CIFA-10, CIFA-100 and ImageNet-16-120. In addition, compared to the VAE TrainingData, it shows a comparable and better distribution of VAE Generation, indicating that our VAE training is good and capable of generating promising models.

Comparison with evolutionary search. We compare TNAS with EA equipped with SPOS. The experiments of the three data sets are shown in Figure 3. The performance of TNAS sampling models is comparable to or better than that of EA, while using less computational cost.

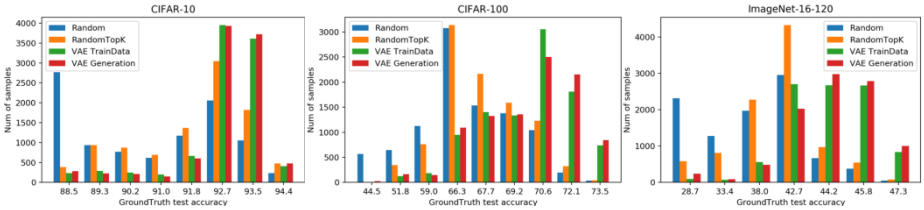

Figure 2: Comparison of TNAS and random sampling capabilities on CIFAR-10, CIFAR-100, and ImageNet-16-120 datasets.

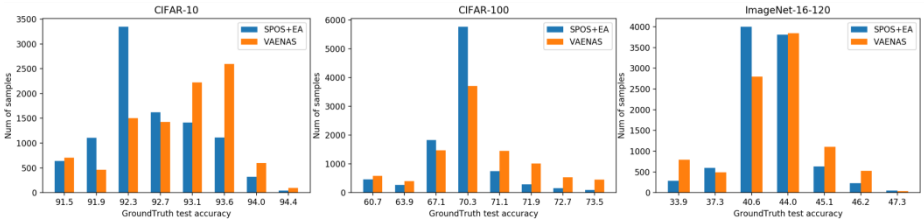

Figure 3: Comparison of TNAS and evolutionary search sampling capabilities across the three datasets.

TNAS performance. We compare TNAS with various methods based on weight allocation, and the results are shown in Table 3. The average performance of TNAS on CIFAR-100 and ImageNet-16-120 is much better than other methods.

In addition, marker ε_{inc} controls the rate of sampling strategy from purely random distribution to absolute VAE generation. Therefore, we run TNAS with different ε_{inc} values to investigate the impact on the generated performance. We run all methods on three datasets separately, without an architectural transformation. We use SPOS+EA as baseline. Note here that we followed [11], running EA only after training SUPNET. In most cases, TNAS outperforms baseline on both measures, especially CIFAR-100 and ImageNet-16-120. Ideally, TNAS of the three datasets are 0.58%, 2.97%, and 2.11% higher than baseline at the maximum measurement, and CIFAR-100 and ImageNet-16-120 are 1.54% and 3.60% higher than baseline at the average accuracy, respectively. We also give a result in Table 5. As can be seen from Table 5, compared to the TNAS in Table 4, TNAS performance in some cases decreases on the maximum measure, while in most cases, the average accuracy of all three datasets improves.

Table 3: Performance of various weight-sharing search methods tested for ground authenticity.

Method	Search (seconds)	CIFAR-10	CIFAR-100	ImageNet-16-120
RSPS	7587.12	87.66±1.69	58.33±4.34	31.14±3.88
DARTS-V1	10889.87	54.30±0.00	15.61±0.00	16.32±0.00
DARTS-V2	29901.67	54.30±0.00	15.61±0.00	16.32±0.00
GDAS	28925.91	93.51±0.13	70.61±0.26	41.84±0.90
SETN	31009.81	86.19±4.63	56.87±7.77	31.90±4.07
ENAS	13314.51	54.30±0.00	15.61±0.00	16.32±0.00
ResNet	—	93.97	70.86	43.63
optimal	—	94.37	73.51	47.31
TNAS†	33325.75	94.22	73.26	46.93
TNAS	34913.27	93.03±0.53	71.86±1.24	45.08±1.52

Table 4: Ground truth test performance of TNAS at different sampling latencies on NAS-BENCH-201.

	CIFAR-10								CIFAR-100								ImageNet-16-120							
Methods	Max accuracy				Mean accuracy				Max accuracy				Mean accuracy				Max accuracy				Mean accuracy			
	10	50	100	300	10	50	100	300	10	50	100	300	10	50	100	300	10	50	100	300	10	50	100	300
SPOS+EA,epoch=300	93.05	93.39	93.52	93.62	92.49	92.39	92.42	92.38	69.18	70.25	70.55	70.77	68.26	68.22	68.20	68.23	44.65	44.92	45.17	45.33	40.63	40.60	40.48	40.48
SPOS+EA,epoch=400	93.33	93.53	93.54	93.64	92.57	92.52	92.50	92.50	69.26	70.51	70.62	71.15	68.24	68.39	68.46	68.50	43.01	44.00	44.31	45.09	40.28	40.15	40.18	40.25
TNAS,$\varepsilon_{inc}=0.0$	93.50	93.91	94.08	94.22	91.35	91.14	91.44	91.52	70.91	71.77	72.36	72.86	66.57	66.94	66.48	66.96	44.31	45.87	45.36	46.94	37.92	39.03	38.74	39.10
TNAS,$\varepsilon_{inc}=0.04$	93.41	93.52	93.51	93.60	92.70	92.52	92.51	92.51	71.56	72.07	72.33	72.54	69.66	69.44	69.37	69.43	43.17	45.29	44.89	45.33	40.29	40.31	40.45	40.55
TNAS,$\varepsilon_{inc}=0.05$	93.10	93.41	93.45	93.63	92.42	92.45	92.45	92.45	71.28	71.74	72.08	72.39	69.80	69.61	69.62	69.56	43.32	45.49	45.33	45.68	40.26	40.70	40.57	40.58
TNAS,$\varepsilon_{inc}=0.0625$	93.20	93.34	93.35	93.52	92.33	92.37	92.37	92.34	71.66	71.93	72.35	72.65	69.37	68.77	68.88	68.74	43.62	45.55	45.86	46.04	40.67	41.15	40.97	41.02
TNAS,$\varepsilon_{inc}=0.1$	93.15	93.30	93.36	93.48	92.36	92.33	92.36	92.35	72.11	72.84	72.99	73.21	69.34	68.91	69.02	68.89	45.85	45.94	46.59	46.59	41.25	41.76	41.69	41.82
TNAS,$\varepsilon_{inc}=0.2$	93.32	93.09	93.23	93.44	92.34	92.17	92.15	92.15	72.23	73.17	72.90	73.22	68.83	68.86	68.59	69.03	44.56	45.31	45.63	45.75	42.51	42.38	42.31	42.45
TNAS,$\varepsilon_{inc}=0.5$	93.15	93.28	93.39	93.42	92.06	92.02	91.95	91.96	70.52	72.17	72.44	72.74	67.14	67.83	67.53	67.75	46.29	47.03	47.00	47.05	43.59	44.20	43.92	44.02

Table 5: Ground truth test performance of TNAS with 500 sample space on NAS-BENCH-201.

	CIFAR-10								CIFAR-100								ImageNet-16-120							
Methods	Max accuracy				Mean accuracy				Max accuracy				Mean accuracy				Max accuracy				Mean accuracy			
	10	50	100	300	10	50	100	300	10	50	100	300	10	50	100	300	10	50	100	300	10	50	100	300
SPOS+EA, epoch=300	93.05	93.39	93.52	93.62	92.49	92.39	92.42	92.38	69.18	70.25	70.55	70.77	68.26	68.22	68.20	68.23	44.65	44.92	45.17	45.33	40.63	40.60	40.48	40.48
SPOS+EA, epoch=400	93.33	93.53	93.54	93.64	92.57	92.52	92.50	92.50	69.26	70.51	70.62	71.15	68.24	68.39	68.46	68.50	43.01	44.00	44.31	45.09	40.28	40.15	40.18	40.25
TNAS,$\varepsilon_{inc}=0.0$	92.89	93.50	93.81	94.21	92.22	92.32	92.41	92.45	69.60	71.15	71.75	72.86	68.21	68.47	68.68	68.61	42.73	44.46	45.27	46.62	40.60	40.58	40.56	40.65
TNAS,$\varepsilon_{inc}=0.04$	92.75	93.39	93.48	93.57	92.63	92.66	92.63	92.54	71.16	72.03	72.07	72.26	70.42	70.15	70.18	70.09	41.84	43.72	44.52	44.94	40.55	40.46	40.55	40.46
TNAS,$\varepsilon_{inc}=0.05$	92.82	93.13	93.34	93.40	92.42	92.33	92.41	92.43	70.48	71.49	71.79	72.14	69.84	70.05	69.89	69.92	41.86	44.76	44.76	45.59	40.05	40.58	40.45	40.67
TNAS,$\varepsilon_{inc}=0.0625$	93.10	93.21	93.36	93.48	92.44	92.41	92.43	92.42	71.01	72.20	72.41	72.55	70.38	70.21	70.22	69.87	43.46	45.19	45.19	45.44	41.41	41.27	41.28	41.21
TNAS,$\varepsilon_{inc}=0.1$	92.95	93.23	93.28	93.42	92.52	92.41	92.37	92.34	72.15	72.30	72.56	73.13	71.13	70.93	70.77	70.05	45.15	45.88	46.22	46.37	43.89	43.81	43.62	43.21
TNAS,$\varepsilon_{inc}=0.2$	92.66	92.77	92.87	92.96	92.20	92.10	92.14	92.13	71.34	72.68	73.09	73.22	70.78	70.66	70.64	70.01	43.93	45.28	45.39	46.11	42.86	43.14	42.94	42.63
TNAS,$\varepsilon_{inc}=0.5$	92.72	92.90	92.98	93.27	92.49	92.36	92.25	92.00	71.39	72.28	72.35	72.57	70.05	69.59	69.50	68.84	45.17	46.00	46.63	47.03	44.36	44.59	44.53	44.60

VI. Conclusion

This paper proposes a learning sampling module (TNAS) based on VAE to improve the efficiency and accuracy of neural network architecture search. The experimental results obtain a series of best results in various data sets and search space. Specifically, the test error of TNAS on CIFAR-10 is 2.26%, and TNAS obtains the latest results of ImageNet under 400M FLOPs frequency in a ShuffleNet-like search space. The probability of finding the optimal structure has increased by 200%. In general, the TNAS method provides a new and practical perspective for sampling methods in neural structure search.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep

convolutional neural networks. In Advances in neural information processing systems, pp. 1097–1105, 2012.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

[3] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8697–8710, 2018.

[4] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pp. 4780–4789, 2019.

[5] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7036–7045, 2019.

[6] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Chunhong Pan, and Jian Sun. Detnas: Neural architecture search on object detection. arXiv preprint arXiv:1903.10979, 2019.

[7] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li FeiFei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 82–92, 2019.

[8] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In International Conference on Machine Learning, pp. 549–558, 2018.

[9] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055, 2018b.

[10] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. arXiv preprint arXiv:1904.00420, 2019.

[11] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. arXiv preprint arXiv:1907.01845, 2019.

[12] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. arXiv preprint arXiv:1812.09926, 2018.

[13] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In Proceedings of the European Conference on Computer Vision (ECCV), pp. 19–34, 2018a.

[14] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. arXiv preprint arXiv:1708.05344, 2017.

[15] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332, 2018.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NIPS, 2017.

[17] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. arXiv preprint arXiv:1711.00436, 2017.

[18] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. arXiv preprint arXiv:1802.03268, 2018.

[19] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In Advances in neural information processing systems, pp. 7816–7827, 2018.

[20] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. arXiv preprint arXiv:1810.05749, 2018a.

[21] Xinbang Zhang, Zehao Huang, and Naiyan Wang. Single shot neural architecture search via direct sparse optimization. 2018b.

[22] Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. Bayesnas: A bayesian approach for neural architecture search. arXiv preprint arXiv:1905.04919, 2019.

[23] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), pp. 116–131, 2018.

[24] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520, 2018.

[25] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. arXiv preprint arXiv:1905.02244, 2019.

[26] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2820–2828, 2019.

[27] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 10734–10742, 2019.

[28] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946, 2019.