Revisiting Training-free NAS Metrics: An Efficient Training-based Method

Taojiannan Yang[1]*, Linjie Yang[2], Xiaojie Jin[2], Chen Chen[1]
[1]Center for Research in Computer Vision, University of Central Florida [2]Bytedance Inc.
taoyang1122@knights.ucf.edu {linjie.yang, jinxiaojie}@bytedance.com
chen.chen@crcv.ucf.edu

Abstract

Recent neural architecture search (NAS) works proposed training-free metrics to rank networks which largely reduced the search cost in NAS. In this paper, we revisit these training-free metrics and find that: (1) the number of parameters (#Param), which is the most straightforward training-free metric, is overlooked in previous works but is surprisingly effective, (2) recent training-free metrics largely rely on the #Param information to rank networks. Our experiments show that the performance of recent training-free metrics drops dramatically when the #Param information is not available. Motivated by these observations, we argue that metrics less correlated with the #Param are desired to provide additional information for NAS. We propose a light-weight training-based metric which has a weak correlation with the #Param while achieving better performance than training-free metrics at a lower search cost. Specifically, on DARTS search space, our method completes searching directly on ImageNet in only 2.6 GPU hours and achieves a top-1/top-5 error rate of 24.1%/7.1%, which is competitive among state-of-the-art NAS methods.

1. Introduction

Neural Architecture Search (NAS) [40, 28, 21, 27, 33, 23, 41] is becoming an important technique in designing efficient and effective deep neural networks. Its effectiveness has been demonstrated in various computer vision tasks such as classification [27, 33, 41], object detection [10, 30] and semantic segmentation [6, 20]. Early NAS methods [40, 28, 29] leverage reinforcement learning or evolutionary algorithm to search networks. But this process is extremely expensive because they need to train thousands of candidate networks. Following works [23, 9, 36] alleviate this problem using differentiable search with candidate networks sampled from a supernet. During training, the network parameters and architecture parameters are optimized alternatively. However, training supernet can still be very slow and the accuracy of sub-networks in the supernet has a poor correlation with their ground truth accuracy [38]. To further reduce the search cost, training-free metrics [24, 7, 2] are proposed to rank the candidate networks without any training process. These metrics are largely inspired by the pruning methods [18, 32, 31] and theoretical findings in deep neural networks [17, 34, 35, 26]. They aim to rank the networks from different aspects of the networks' properties such as trainability and expressivity. These metrics achieve competitive results with previous NAS methods at a much smaller search cost.

However, these works overlooked a straightforward training-free metric, the *number of parameters* (#Param) in a network, which is even faster to compute than those training-free metrics. Our experiments show that #Param is surprisingly good on NAS-Bench-101 [37] and NAS-Bench-201 [12]. We further discover that these training-free metrics have a very high correlation with #Param (details in Sec. 3.1), which indicates that a large portion of their ranking ability may come from the correlation with #Param. To validate our conjecture, we design systematic experiments to remove the impact of #Param. The results show that without the #Param information, recent training-free metrics [7, 24] do not achieve a good performance.

Motivated by the above discovery, our objective is to develop a metric that has a weak correlation with #Param while still being effective so that it can provide additional information on estimating the performance of a network. Intuitively, a network's final performance is indicated by the structure (*e.g.*, #Param, #Layers), weight initialization, and the dynamics during training (*e.g.*, loss, gradients). We believe that metrics arise from the training dynamics should be weakly correlated with #Param. Angle metric is a training dynamic which is first proposed in [5] to indicate the network's generalization ability. It is defined as the angle between the vectorized network weights before and after training. We find that the angle metric at the final fully-connected (FC) layer has a high correlation with the ac-

*Work done during an internship at Bytedance Inc.

curacy but a low correlation with the number of parameters. This indicates that it can provide additional information other than #Param on estimating the network's performance. To reduce the computation for model training, we propose an extremely light-weight training scheme with a small proxy dataset which is thousands times faster than traditional training. Our experiments show that such a short-training scheme already yields effective angle metrics. Besides the angle metric, we also leverage the training loss as a second metric, which achieves better performance without additional computation. To summarize, we make the following contributions.

1. We revisit recent training-free metrics and reveal how they achieve good performance on the evaluated benchmarks. Although training-free metrics claim to rank networks by estimating the model's capacity and convergence speed, our experiments show that they achieve good performance mainly because they have high correlation with #Param, and #Param happens to be a good metric on the evaluated NAS benchmarks. Their functionality is in fact similar to #Param while being unnecessarily complicated.
2. Motivated by our discovery, we propose a training-based metric which provides orthogonal information to #Param on ranking networks. Our method achieves competitive performance with training-free methods on popular NAS benchmarks, and the performance will be significantly better when the #Param information is not helpful. Our search cost is even smaller than training-free metrics.
3. Our findings raise the necessity to design new search spaces where #Param does not dominate the model performance to better evaluate the effectiveness of a NAS metric and understand how it works. Our results also inspire future works to design metrics that provide orthogonal information to #Param because #Param may not be a good metric in many cases (*e.g.*, MLP vs. CNN).

2. Related work

Neural architecture search (NAS). NAS is proposed to search network structures automatically for a given task instead of time-consuming manual design. Early works [40, 28, 29, 22] leverage reinforcement learning or evolutionary algorithms to explore architectures. The controller will generate some networks and the network performance will be used as feedback information to update the controller. However, training a large amount of networks is very expensive, costing thousands of GPU days. Following works accelerate NAS algorithms by weight-sharing in a supernet. ENAS [27] proposes to share the weights among candidate networks so that they can be trained simultaneously. DARTS [23] concatenates all candidate operations into a supernet and each operation is assigned an architecture parameter denoting its importance. During training, the architecture parameters and weight parameters are optimized alternatively. Another kind of weight-sharing method is one-shot NAS [3, 4, 14], where a supernet is trained with sub-networks stochastically sampled in each iteration. However, recent study [38] shows that the network performance via weight-sharing has a poor correlation with its actual performance.

Training-free NAS. To further speedup the search process, recent works [24, 7, 2] propose to predict network performance without training. [2] evaluates the effectivenss of different pruning-at-initialization criteria [32, 31, 18] for NAS. NASWOT [24] leverages the number of linear regions [35] to rank different networks. TE-NAS [7] further combines linear regions with neural tangent kernel (NTK) [17] to rank a network by its expressivity and trainability. However, [25] shows that NTK-based metrics are unstable across different search spaces and initializations. In this work, we further reveal that the effectiveness of training-free metrics (Linear Region and NTK) mainly come from the high correlation with #Param, and #Param is a good metric on the evaluated benchmarks.

3. Methodology

In Sec. 3.1, we first revisit several existing training-free metrics and #Param. We demonstrate that #Param is an effective search metric on NAS-Bench-101 and NAS-Bench-201, and that existing training-free metrics rely on #Param to achieve high performance. Then we introduce our light-weight training-based metric and short-training strategy in Sec. 3.2 and Sec. 3.3, respectively.

3.1. Revisiting training-free metrics

The number of **linear regions (LR)** is used in [24, 7] to rank networks at initialization. Linear region is a well-studied theoretical criteria [35, 26] to indicate the learning capacity of a network. It is defined as how many regions a network could split the input space into. A larger number of linear regions indicates that the network has higher performance. The number of LR is estimated differently in TE-NAS [7] and NASWOT [24]. TE-NAS calculates LR by forwarding a batch of samples to the network and count how many samples have different activation patterns, while NASWOT feeds a batch of samples to the network and compute the Hamming distance between different activation patterns. The Hamming distance between these activation patterns is used to define a kernel matrix $\mathbf{K}$. The ranking metric is defined as the determinant of $\mathbf{K}$. To distinguish these two metrics, we denote the LR estimated by TE-NAS and NASWOT as LR1 and LR2, respectively.

TE-NAS further leverages the **neural tangent kernel (NTK)** to score networks. [17, 34] point out that the network's convergence speed is determined by the condition number of NTK. Intuitively, a faster convergence speed indicates that the network has a higher performance. So the condition number of NTK can be used to rank networks. Note that in [7], NTK is negtively correlated with the accuracy while in this paper we use negative NTK to make it positive.

These theoretical indicators describe a network's property from different perspectives. However, the most naive indicator to describe a network would be the **number of parameters (#Param)**. Intuitively, a larger model tends to have better performance. This makes us wonder whether the number of parameters is a good training-free metric? The answer is yes. In Tab. 1, we show the comparison of #Param and training-free metrics on NAS-Bench-101 [37] and NAS-Bench-201 [12]. We evaluate these metrics based on random search. Specifically, we randomly sample 100 networks from the search space and use the metrics to select the best one. We run each experiment 5 times and report mean accuracy and standard deviation. Surprisingly, the results show that #Param achieves comparable performance with other training-free metrics on different datasets.

The good performance of #Param further motivates us to investigate whether these training-free metrics are correlated with #Param. We compute the Kendall rank correlation coefficient (Kendall's Tau) [16] between different training-free metrics and #Param on NAS-Bench-101 (10000 networks) and NAS-Bench-201 (15625 networks) in Tab. 2. As a reference, the correlation between LR1 and LR2 is 0.56 on NAS-Bench-201. Note that they are the same metric just estimated differently, thus a correlation of 0.56 is high. The results show that all these training-free metrics have high correlations with #Param, especially the two linear region metrics. This is intuitively plausible because the number of linear regions is upper bounded by $2^{\#activations}$, while the number of activation units is highly correlated with the number of parameters. These results imply that the ranking ability of these training-free metrics may mainly come from the high correlation with #Param. In Sec. 4, we validate this conjecture by evaluating training-free metrics on networks of the same number of parameters. Their performance drops dramatically in this situation.

What are the drawbacks of metrics having high correlation with #Param? Firstly, these training-free metrics claim to rank networks by estimating the model's capacity and convergence, but their functionality is in fact similar to #Param while being unnecessarily complicated. Secondly, #Param is not always a good metric. In the scenarios where the #Param is not helpful (*e.g.*, MLP vs. CNN, Residual vs. Plain structure, networks with similar #Param as in Sec. 4), the performance of such metrics will drop dramatically.

Motivated by these observations, we explore a new type of metric in this work, which is weakly correlated with the number of parameters while providing additional information on estimating the performance of the neural networks. Our proposed metric is introduced in the following sections.

Table 1: Comparison of #Param and training-free metrics on NAS-Bench-101 and NAS-Bench-201. Each experiment is repeated 5 times and mean accuracy and standard deviation are reported.

Metrics	NAS-Bench-101	NAS-Bench-201		
	CIFAR-10	CIFAR-10	CIFAR-100	ImageNet16-120
#Param	92.6(1.3)	**93.2(0.5)**	**70.1(0.8)**	41.6(4.1)
LR1	91.6(0.9)	92.3(1.1)	66.2(5.0)	43.1(2.5)
NTK	91.2(0.9)	91.9(1.7)	66.6(4.3)	41.4(4.9)
LR2	**92.8(1.2)**	92.6(0.9)	69.3(1.4)	**43.3(2.9)**

Table 2: Correlation (Kendall's Tau) of different training-free metrics with the number of parameters (#Param).

Correlation with #Param	LR1	NTK	LR2
NAS-Bench-101	0.46	0.36	0.62
NAS-Bench-201	0.39	0.30	0.56

3.2. Angle metric

Since existing training-free metrics all have a high correlation with the number of parameters based on the observations in Sec. 3.1, we shift our attention to the training dynamics. **Angle metric** is a training dynamic which is first proposed in [5] to indicate the generalization ability of a network and later used in [15, 39] as a metric to rank candidate networks in NAS. Considering all the weights of a network as a one-dimensional vector, angle metric is defined as the angle between the weight vectors before and after training. Specifically, let $\boldsymbol{W}_0$ denote the weights of a network $\boldsymbol{N}$ at initialization, and $\boldsymbol{W}_t$ denote the weights after training. Then the angle metric is defined as

$$\theta(\boldsymbol{N}) = \arccos\left(\frac{\boldsymbol{W_0} \cdot \boldsymbol{W_t}}{\|\boldsymbol{W_0}\|_2 \|\boldsymbol{W_t}\|_2}\right), \tag{1}$$

where $\boldsymbol{W_0} \cdot \boldsymbol{W_t}$ is the inner product of $\boldsymbol{W}_0$ and $\boldsymbol{W}_t$. [39] shows that the angle metric is positively correlated with a network's final performance.

However, we find that the angle metric behaves differently at different network stages. Specifically, the angle metric computed with the weights from the feature extraction layers is positively correlated with the network's final accuracy, while the angle metric computed with the weights of the prediction layer (the final fully-connected layer) is negatively correlated with the performance. In most NAS search spaces [37, 12, 23], the feature extraction stage is mainly constructed by a stack of network modules. We denote the angle metric of the feature extraction stage θ_{feat} and the angle metric of the prediction layer θ_{pred} for brevity.

In Tab .3, we demonstrate the impact of model parameters on above two variants of angle metrics through two

Table 3: Comparison of Kendall's Tau of θ_{feat} and θ_{pred} on 50 random networks with different sizes (different #Param) or the same size (same #Param), respectively.

Sampled Networks	θ_{feat}	θ_{pred}
diff. #Param	0.37	-0.50
same #Param	-0.09	-0.25

kinds of network settings. We randomly sample 50 networks with different sizes (setting 1) and the same size (setting 2) from NAS-Bench-201, and fully train them on CIFAR-10. Then we compute the Kendall's Tau of θ_{feat} and θ_{pred} for these two scenarios. In setting 1, it shows that θ_{feat} is positively correlated with the accuracy, which is consistent with [15, 39], but θ_{pred} is negatively correlated and has a higher correlation than θ_{feat}. However, in setting 2, the Kendall's Tau of θ_{feat} degrades dramatically to around 0, which means θ_{feat} fails to rank the networks without the #Param information. But the Kendall's Tau of θ_{pred} degenerates less and is still able to rank the networks of the same number of parameters. Therefore, θ_{pred} is a metric with weak dependency on the number of parameters.

3.3. Short-training scheme

In Sec. 3.2, we show θ_{pred} is a good metric at ranking networks even without the #Param information. However, fully training all candidate networks is too expensive in NAS. To alleviate this problem, we propose an extremely light-weight short-training scheme by using a small proxy dataset from the original target dataset. Specifically, we first randomly sample a sub-set of classes from the target dataset. Then for each sampled class, we randomly sample a small amount of images, generating a highly condensed proxy dataset. We train networks on the proxy dataset for a limited number of iterations. This training procedure is thousands times faster than fully training a network. We find our θ_{pred} metric is effective under such a compact setting in different search spaces and datasets.

Besides θ_{pred}, we also use another training dynamic, the training loss, as an additional metric to evaluate networks. Note that training loss comes for free in our method. In Sec. 4, we show that training loss also has weak correlation with the number of parameters. Combining training loss with θ_{pred} gives richer information on model performance without increasing the computational cost.

Since the scales of θ_{pred} and training loss are different, directly adding their values will cause one dominating the other. To avoid this problem, we first use these two metrics to rank networks respectively. Then we add their ranking index as the final ranking index of each network. Note that both θ_{pred} and training loss are negatively correlated with the model accuracy. For clarity, we take the negative value of the two metrics to make them positive in the following

Algorithm 1 ST-NAS

```
Input: Number of candidate networks $N$. Search space $\mathcal{S}$. Target dataset $\mathcal{D}$. Training iterations $m$.
Output: Model with the highest rank.
▷ Initialization
$\theta_{pred}$ = zeros($N$), $loss$ = zeros($N$)
sampler = RandomSampler()
Sample proxy dataset $\tilde{\mathcal{D}}$ from $\mathcal{D}$
▷ Evaluate candidate networks
for i in $0, 1, ..., N-1$ do
    network = sampler($\mathcal{S}$)
    $W_0$ = network.fc.weights
    Train the network for $m$ iterations with $\tilde{D}$.
    $loss$[i] = - compute_loss(network, $\tilde{\mathcal{D}}$)
    $W_t$ = network.fc.weights
    $\theta_{pred}$[i] = - compute_angle_metric($W_0$, $W_t$)
end for
▷ Combine two metrics
$R_{\theta_{pred}}$ = get_rankings($\theta_{pred}$)
$R_{loss}$ = get_rankings($loss$)
$R$ = $R_{\theta_{pred}}$ + $R_{loss}$
max_idx = model index with the highest rank in $R$
return: $\mathcal{S}$[max_idx]
```

experiments. Since the proposed metric employs a short period of training, we name our NAS method combined with this metric as Short-Training NAS (ST-NAS). A pipeline of ST-NAS based on random search is shown in Algorithm 1.

4. Empirical study

As discussed in Sec. 3, recent training-free metrics are highly correlated with the number of parameters, which implies their effectiveness comes from the high correlation with number of parameters. To further validate our claim, we thoroughly evaluate different training-free metrics and our metric on curated search spaces with the same number of parameters. This prevents metrics from leveraging the parameter information to evaluate networks. In the following sections, $Angle$ denotes searching with θ_{pred}, $Loss$ denotes searching with training loss and $AngleLoss$ denotes searching with the combination of the two metrics.

We craft several search spaces based on NAS-Bench-201 [12]. NAS-Bench-201 defines a cell-based search space. Each cell is represented as a densely-connected directed acyclic graph (DAG). Each cell has 4 nodes and 6 edges, where each edge represents an operation. There are 5 candidate operations, including *zeroize, skip-connect,* 1×1 *conv,* 3×3 *conv, and* 3×3 *avg pooling*. Different models may have the same number of parameters but with different structures and performances. We choose 8 groups of models, and models in the same group has the same number of parameters, i.e. $\{0.37, 0.40, 0.59, 0.62, 0.64, 0.83, 0.86, 1.05\}$ M, respectively. The number of networks in

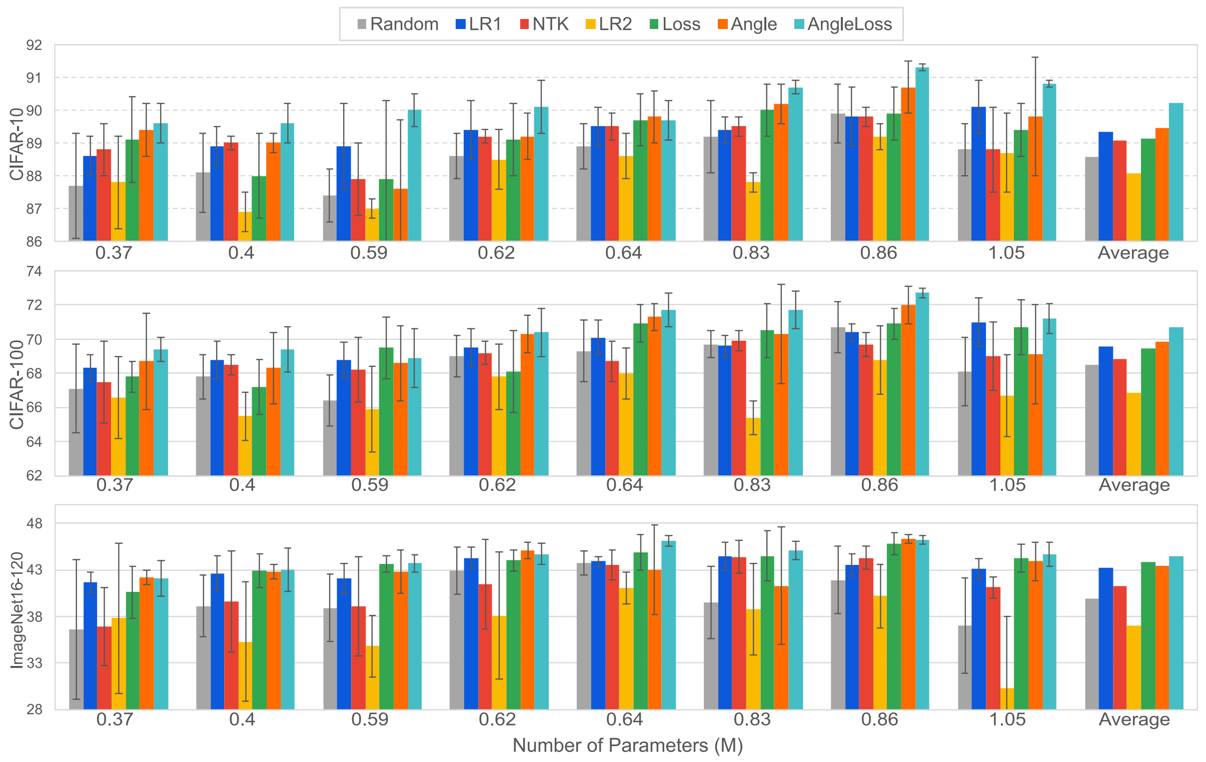

Figure 1: Test accuracy (%) of different metrics when evaluated on networks of the same number of parameters. X-axis is the number of parameters (M) in each network group. Each experiment is repeated 5 times and the mean accuracy and standard deviation are reported.

Table 4: Kendall's Tau between our metrics and #Param.

Metrics	Angle	Loss	AngleLoss
Correlation	0.20	-0.11	0.07

each group is $\{1602, 540, 1602, 810, 180, 540, 180, 135\}$, respectively. We evaluate the effectiveness of different metrics on each of these network groups. We compute the training-free metrics using the settings in the original papers [7, 24]. Our training scheme is detailed in Sec. 3.3. We randomly sample 10 classes and 10 images from each class. The network is trained for 50 iterations with a fixed learning rate of 0.2. Other settings follow those in NAS-Bench-201 [12]. Note this is the default setting throughout our experiments if not specified.

We compare the performance of previous training-free metrics and our metrics using random search. We randomly sample 100 networks from each network group and select the best-performing network per the metric. We also add a baseline which randomly selects a network from candidate networks. Each experiment is repeated 5 times and the mean accuracy and standard deviation are reported. As shown in Fig. 1, LR2, which has the highest correlation with #Param in Tab. 2 and the best performance in Tab. 1, performs the worst in this scenario. It is even worse than the random baseline. Our $AngleLoss$ metric consistently outperforms training-free metrics on all the network groups on three datasets. In most cases, $AngleLoss$ is higher than training-free metrics by more than 1%. We also show our metrics' Kendall's Tau with #Param in Tab. 4. As can be seen, the correlations are much lower than that of the training-free metrics in Tab. 2. Above experiment evidences that training-free metrics largely rely on the parameter information to rank networks, and that our metric is advantageous by having weak correlation with the number of parameters, providing additional useful information to estimate a network's performance.

5. Experiments

In Sec. 5.1, we first show the comparisons of training-free metrics and our metric on NAS-Bench-101 and NAS-Bench-201. We apply metrics to both random search method and pruning-based search method. Then we compare our metric with other methods on DARTS search space in Sec. 5.2. Finally, we conduct ablation studies to show the impact of short-training hyperparameters.

5.1. Results on NAS-Bench-101/201

Random search. We first evaluate different metrics based on random search. We randomly sample 100 networks from the search space and use different metrics to

Table 5: Comparison of the test accuracy of different metrics on NAS-Bench-101 and NAS-Bench-201 based on random search ($N = 100$). Each experiment is repeated 5 times to compute its mean and standard deviation.

Metrics	Search Cost (s)	NAS-Bench-101	NAS-Bench-201		
		CIFAR-10	CIFAR-10	CIFAR-100	ImageNet16-120
#Param	3	92.58(1.26)	93.21(0.49)	70.15(0.83)	41.58(4.07)
LR1	60	91.98(1.31)	92.30(1.07)	66.23(4.96)	43.12(2.52)
LR1+#Param	60	92.52(1.37)	92.96(0.55)	69.83(0.43)	43.71(2.20)
NTK	181	91.23(1.11)	91.94(1.70)	66.63(4.29)	41.38(4.88)
NTK+#Param	181	91.48(1.52)	93.12(0.48)	69.82(0.73)	42.39(1.61)
LR2	48	91.95(1.16)	92.65(0.93)	69.28(1.40)	43.33(2.91)
LR2+#Param	48	92.58(1.39)	93.33(0.13)	70.10(1.22)	42.83(1.49)
AngleLoss	437	92.86(0.77)	84.65(5.88)	58.06(0.40)	28.08(0.31)
AngleLoss+#Param	437	**93.60(0.46)**	**93.46(0.59)**	**70.58(0.82)**	**43.74(1.48)**
AngleLoss+LR2	462	93.47(0.47)	93.08(0.66)	69.62(0.59)	43.43(1.62)

select the best one. We follow the default settings in [24, 7] to compute training-free metrics LR1, LR2, and NTK. Our training settings are the same as in Sec. 4. We run each experiment 5 times and report the mean accuracy and standard deviation. The search cost is measured on a single GTX-1080Ti GPU.

The results are shown in Tab. 5. We add #Param as a baseline metric in Tab. 5. It is shown that #Param performs well on both NAS-Bench-101 and NAS-Bench-201. It is even slightly better than training-free metrics on CIFAR-10 and CIFAR-100. Note that #Param is very easy to compute, with a search cost of only 3 seconds on 100 networks. The linear region based metrics (LR1 and LR2) are better and more stable than NTK. The performance of NTK is low and has a very large variance. Although both LR1 and LR2 are based on linear regions, LR2 is slightly better and more stable. Note the effectiveness of training-free metrics could be attributed to their high correlation with #Param.

Surprisingly, our metric $AngleLoss$ does not perform well on the overall search space of NAS-Bench-201, although we have demonstrated in Sec. 4 that it is significantly better than other training-free metrics in different network groups. By visualizing the searched network structures, we find that our $Angle$ metric could collapse to some trivial structures, where most of the connections are *zeroize, skip-connect* or *avg_pooling*. Our conjecture is that in these trivial structures, the feature extraction layers are not learning anything meaningful, and the prediction layer is optimized towards random directions in each training iteration. So the weight vector of the prediction layer almost does not change after training, which means $Angle$ metric will give a high score to these structures. However, this problem could be easily resolved if we combine our metric with #Param to avoid the structures with a small number of parameters. It can also be avoided when we use a pruning-based search method. In Tab. 5, we see that our metric is significantly boosted by around 10% when combined with #Param, and it achieves higher performance than other training-free metrics. On NAS-Bench-101, we don't have the collapse problem because there are fewer trivial structures. We achieve significantly better performance than training-free metrics.

We also combine training-free metrics with #Param. It shows that these training-free metrics can also slightly benefit from #Param, but the improvement is marginal. Taking #Param as the baseline, combined with training-free metrics will even degrade its performance on NAS-Bench-201 CIFAR-10 and CIFAR-100. However, our metric achieves consistent improvements upon #Param on three datasets. We also show that when combined with LR2, AngleLoss+LR2 improves upon LR2 on all datasets. These experiments demonstrate that our metric provides orthogonal information to #Param and training-free metrics. They can be combined together to achieve better performance.

Pruning-based search. We also apply our metric to pruning-based search used in TE-NAS [7]. All the settings are the same as in Sec. 4, except that we train the supernet for 100 iterations because it takes longer for the supernet to converge. Each experiment is repeated 5 times and the mean and standard deviation are reported.

We compare our method with TE-NAS in Tab. 6. The performances of some other NAS methods are cited from [12] for reference. We report two results for TE-NAS, one is reported in the original paper [7] and the other is reproduced by us using the official codes [1] since we cannot reproduce the results in the original paper using the default setting. The reproduced performance is lower while the search cost is also cheaper (we evaluate it on a 1080Ti GPU, which is the same as in TE-NAS). In Tab. 6, we can see that our short-training method is even faster than TE-NAS. This is because TE-NAS needs to compute two metrics (LR1 and NTK), and for each metric it repeats 3 times and takes the average value to have a better and stable performance. However, we only compute our metric once with an extremely short training scheme.

Table 6: Comparison of the test accuracy on NAS-Bench-201 based on pruning-based search in [7]. † indicates the results are reproduced by us using the official released codes [1]. The search cost of our method and TE-NAS is measured on 1080Ti GPU while LGA is measured on Tesla A40 GPU. The **best** and <u>second best</u> results are bold and underlined, respectively.

Method	Search Cost (s)	CIFAR-10	CIFAR-100	ImageNet16-120
RSPS [19]	8007	87.66(1.69)	58.33(4.34)	31.14(3.88)
DARTS (1st) [23]	10889	54.30(0.00)	15.61(0.00)	16.32(0.00)
GDAS [13]	28925	93.61(0.09)	70.70(0.30)	41.84(0.90)
LGA [25]	5400	**93.94(N/A)**	**72.42(N/A)**	**45.17(N/A)**
TE-NAS [7]	1558	<u>93.90(0.47)</u>	<u>71.24(0.56)</u>	42.38(0.46)
TE-NAS† [7]	<u>682</u>	93.20(0.29)	70.44(1.34)	42.34(0.63)
AngleLoss	**508**	93.16(0.37)	70.48(1.04)	43.04(1.82)
AngleLoss+#Param	**508**	93.36(0.26)	70.87(0.41)	<u>43.77(1.33)</u>

Table 7: Comparison with state-of-the-art on DARTS CIFAR-10. The **best** and <u>second best</u> results are bold and underlined, respectively.

Method	Search Cost (GPU days)	Params (M)	Top-1 Acc (%)	Search Method
NASNet-A [41]	2000	3.3	97.35	RL
ENAS [27]	0.5	4.6	97.11	RL
AmoebaNet-A [28]	3150	3.2	96.66	evolution
Random baseline [23]	4	3.2	96.71	random
DARTS (1st) [23]	0.4	3.3	97.00	gradient
DARTS (2nd) [23]	1.0	3.3	97.24	gradient
GDAS [13]	0.17	2.5	97.18	gradient
P-DARTS [9]	0.3	3.4	**97.50**	gradient
PC-DARTS [36]	0.1	3.6	97.43	gradient
SDARTS-ADV [8]	1.3	3.3	97.39	gradient
TE-NAS [7]	**0.05**	3.8	97.37	training-free
AngleLoss	<u>0.09</u>	3.2	97.37	short-training
AngleLoss+#Param	<u>0.09</u>	3.2	<u>97.44</u>	short-training

Under the pruning-based search, our metric does not show the collapse problem as in random search. This is because pruning-based method starts from a supernet, which is definitely non-trivial. With a limited number of pruning steps, the network almost never reach a trivial structure with large numbers of empty operations. As shown in Tab. 6, the original results of TE-NAS are better than ours on CIFAR-10 and CIFAR-100, but the search cost is 3$\times$ of ours. Our performance is comparable with the reproduced results of TE-NAS at a lower search cost. On ImageNet16-120, our metric is better than TE-NAS in both cases. We also combine our metric with #Param with negligible additional search cost. It further improves our performance on all three datasets by $0.2\% - 0.7\%$.

5.2. Results on DARTS search space

We apply our metric to the pruning-based search method used in TE-NAS [7] for the following experiments.

Results on CIFAR-10. We first compare our metric with other methods on CIFAR-10 dataset. As shown in Tab. 7, our metric completes the search process in 0.09 days (*i.e.*, 2.16 hours) on a single 1080Ti GPU. Different from the results on NAS-Bench-201, our search cost is higher than TE-NAS in this case. This is because TE-NAS uses a smaller batch-size to compute NTK on DARTS CIFAR-10, resulting in less computation. Nevertheless, our search cost is still much lower than other NAS methods. Our metric also achieves comparable performance with TE-NAS, but the searched network size is much smaller. When combined with #Param, our metric again achieves a lower test error of 2.56%, which is competitive with state-of-the-art methods.

Results on ImageNet-1K. We compare our metric with state-of-the-art NAS methods on ImageNet-1K [11] in Tab. 8. Our short-training setting is the same as in CIFAR-10. For evaluation, we follow [7] to stack the network with 14 cells and the initial number of channel is 48. In the top half of Tab. 8, the networks are searched on CIFAR-10 and then evaluated on ImageNet-1K. We can see that our metric is competitive with state-of-the-art NAS methods with a much lower search cost. Compared to TE-NAS, our performance is significantly better and the network size is much smaller. The bottom half of Tab. 8 shows the results with different methods searched directly on ImageNet-1K. Pruning-based search with our metric completes in only 0.11 GPU days

Table 8: Comparison with state-of-the-art NAS methods on DARTS search space ImageNet-1K dataset.

Method	Search Cost (GPU days)	Params (M)	Top-1 (%)	Top-5 (%)	Search Method	Search Dataset
NASNet-A [41]	2000	5.3	74.0	91.6	RL	CIFAR-10
AmoebaNet-C [28]	3150	6.4	75.7	92.4	evolution	
DARTS (2nd) [23]	4.0	4.7	73.3	91.3	gradient	
GDAS [13]	0.21	5.3	74.0	91.5	gradient	
P-DARTS [9]	0.3	4.9	75.6	92.6	gradient	
PC-DARTS [36]	0.1	5.3	74.9	92.2	gradient	
TE-NAS [7]	0.05	6.3	73.8	91.7	training-free	
AngleLoss	0.09	4.7	75.3	92.5	short-training	
AngleLoss+#Param	0.09	4.7	74.8	92.3	short-training	
ProxylessNAS [21]	8.3	7.1	75.1	92.5	gradient	ImageNet-1K
PC-DARTS [36]	3.8	5.3	74.8	92.7	gradient	
TE-NAS [7]	0.17	5.4	75.5	92.5	training-free	
AngleLoss	**0.11**	4.8	74.5	91.9	short-training	
AngleLoss+#Param	**0.11**	5.9	**75.9**	**92.9**	short-training	

Table 9: Ablation study of different training hyper-parameters on NAS-Bench-201 CIFAR-100.

(a) Number of training iterations.

#Iters	10	25	50	75
Cost (s)	99	230	437	673
Acc (%)	70.22(1.08)	70.33(0.91)	70.58(0.82)	70.37(0.57)

(b) Number of sampled classes.

#Classes	5	10	20
Cost (s)	332	437	641
Acc (%)	70.02(0.74)	70.58(0.82)	70.30(0.74)

(c) Network initialization.

Init.	Kaiming_uniform	Kaiming_normal	Xavier_uniform
Cost (s)	437	437	437
Acc (%)	70.58(0.82)	70.40(0.70)	70.25(1.00)

(d) Number of sampled images.

#Images	5	10	20
Cost (s)	347	437	627
Acc (%)	70.26(1.08)	70.58(0.82)	70.28(0.97)

(*i.e.*, 2.64 GPU hours), which is even faster than TE-NAS. Our metric is more than 30$\times$ faster than the other NAS methods. The performance of our metric alone is slightly lower than other methods but with a smaller model size. When combined with #Param, the performance of our metric is largely improved and reaches a competitive top-1/top-5 error rate of 24.1%/7.1%, outperforming listed differentiable and training-free methods. Note that our search cost is also significantly lower than other methods.

5.3. Ablation Study

Here we study the impact of different hyper-parameters in our short-training scheme, including number of training iterations, sampled classes, images per class and weight initialization methods. We conduct experiments on CIFAR-100. The results of different settings are shown in Tab. 9. We use the random search method in Tab. 5.1 as the baseline. We can see that longer training iterations tend to achieve better performance. This is because longer training iterations allow the network to converge better, which yields more informative angle metric and training loss. But even only 10 training iterations can achieve a decent performance. Increasing the number of classes does not always improve the performance. We speculate that although more classes could provide more information about the target dataset, it also makes the proxy dataset harder, which makes the network harder to converge in the limited iterations and yields less informative angle metric and training loss. Similarly, increasing the number of images does not guarantee better performance either. To achieve the optimal accuracy-efficiency trade-off, one may need to tune the training hyper-parameters. But the performance is not very sensitive to the hyper-parameters and it is feasible to tune hyper-parameters because our method is highly efficient.

6. Conclusion

We conduct a systematic study to explore the relationship between recent training-free metrics and #Param. Our empirical study shows that recent training-free metrics works similarly to #Param while being unnecessarily complicated. Motivated by this discovery, we propose a light-weight training-based metric which provides orthogonal information than #Param on estimating model performance. Our method achieves competitive performance with state-of-the-art NAS methods, while being even faster than training-free metrics. On the search spaces where the #Param information is not useful, the performance of training-free metrics drops dramatically while our method significantly outperforms them on different datasets. We hope our work could inspire future works to design new metrics which provide more parameter-independent information on estimating the network's performance.

References

[1] Official implementation of te-nas. `https://github.com/VITA-Group/TENAS`.

[2] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas Donald Lane. Zero-cost proxies for lightweight nas. In *International Conference on Learning Representations*, 2020.

[3] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 550–559. PMLR, 10–15 Jul 2018.

[4] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*, 2017.

[5] Simon Carbonnelle and Christophe De Vleeschouwer. Layer rotation: a surprisingly powerful indicator of generalization in deep networks? *arXiv preprint arXiv:1806.01603*, 2018.

[6] Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. Fasterseg: Searching for faster real-time semantic segmentation. In *International Conference on Learning Representations*, 2019.

[7] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *International Conference on Learning Representations*, 2020.

[8] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *International Conference on Machine Learning*, pages 1554–1565. PMLR, 2020.

[9] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1294–1303, 2019.

[10] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. *Advances in Neural Information Processing Systems*, 32:6642–6652, 2019.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[12] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2019.

[13] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019.

[14] 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. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020.

[15] Yiming Hu, Yuding Liang, Zichao Guo, Ruosi Wan, Xiangyu Zhang, Yichen Wei, Qingyi Gu, and Jian Sun. Angle-based search space shrinking for neural architecture search. In *European Conference on Computer Vision*, pages 119–134. Springer, 2020.

[16] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[17] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32:8572–8583, 2019.

[18] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2018.

[19] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, pages 367–377. PMLR, 2020.

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

[21] 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)*, pages 19–34, 2018.

[22] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations*, 2018.

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

[24] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR, 2021.

[25] Jisoo Mok, Byunggook Na, Ji-Hoon Kim, Dongyoon Han, and Sungroh Yoon. Demystifying the neural tangent kernel from a practical perspective: Can it be trusted for neural architecture search without training? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11861–11870, 2022.

[26] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *Advances in Neural Information Processing Systems*, 27:2924–2932, 2014.

[27] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104. PMLR, 2018.

[28] 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, pages 4780–4789, 2019.

[29] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[30] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[31] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33, 2020.

[32] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2019.

[33] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[34] Lechao Xiao, Jeffrey Pennington, and Samuel Schoenholz. Disentangling trainability and generalization in deep neural networks. In *International Conference on Machine Learning*, pages 10462–10472. PMLR, 2020.

[35] Huan Xiong, Lei Huang, Mengyang Yu, Li Liu, Fan Zhu, and Ling Shao. On the number of linear regions of convolutional neural networks. In *International Conference on Machine Learning*, pages 10514–10523. PMLR, 2020.

[36] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2019.

[37] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114. PMLR, 2019.

[38] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations*, 2019.

[39] Xuanyang Zhang, Pengfei Hou, Xiangyu Zhang, and Jian Sun. Neural architecture search with random labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10907–10916, 2021.

[40] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[41] 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*, pages 8697–8710, 2018.