Evaluating Efficient Performance Estimators of Neural Architectures

Anonymous Author(s)
Affiliation
Address
email

Abstract

Conducting efficient performance estimations of neural architectures is a major challenge in neural architecture search (NAS). To reduce the architecture training costs in NAS, **one-shot** estimators (OSEs) amortize the architecture training costs by sharing the parameters of one "supernet" between all architectures. Recently, **zero-shot** estimators (ZSEs) that involve no training are proposed to further reduce the architecture evaluation cost. Despite the high efficiency of these estimators, the quality of such estimations has not been thoroughly studied. In this paper, we conduct an extensive and organized assessment of OSEs and ZSEs on three NAS benchmarks: NAS-Bench-101/201/301. Specifically, we employ a set of NAS-oriented criteria to study the behavior of OSEs and ZSEs and reveal that they have certain biases and variances. After analyzing *how and why* the OSE estimations are unsatisfying, we explore *how to mitigate* the correlation gap of OSEs from several perspectives. For ZSEs, we find that current ZSEs are not satisfying enough in these benchmark search spaces, and analyze their biases. Through our analysis, we give out suggestions for future application and development of efficient architecture performance estimators. Furthermore, the analysis framework proposed in our work could be utilized in future research to give a more comprehensive understanding of newly designed architecture performance estimators.

1 Introduction

Neural architecture search (NAS) can automatically discover architectures that outperform the hand-crafted ones for various applications [34, 9, 10]. Early NAS methods [34, 20] suffer from an extremely heavy computational burden, and can take tens of thousands of GPU hours to run. One of the major reasons for the computational challenge of NAS is that evaluating each candidate architecture is slow, which includes a full training and testing process. In the past years, studies [2, 19, 3, 5, 7, 32, 16, 1] have been focusing on developing more efficient performance estimators of neural architectures.

One-shot Estimator (OSE) Traditional NAS methods [34, 20, 2] conduct a costly separate training process to acquire the suitable parameters to evaluate each candidate architecture. To make NAS computationally tractable, ENAS [19] proposes the parameter-sharing technique to accelerate the architecture evaluation. Specifically, ENAS constructs an over-parametrized super network (i.e. "supernet") such that all architectures in the search space are its sub-architectures. During the search process, candidate architectures are evaluated on the validation data by using the corresponding subset of weights, without undergoing a separate training process. Following this work, the parameter sharing technique is widely used for architecture search in different search spaces [27, 14] or incorporated with different search strategies [15, 14, 18, 28]. We refer to the parameter-sharing estimations as the "one-shot" estimations since it requires the training cost of *one* supernets.[1]

[1] See more discussion on this terminology in Sec. 2.1.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

How well the one-shot estimations are correlated with the standalone architecture performances is essential for the efficacy of NAS methods. Despite the widespread use of OSEs, studies [30] have revealed that the one-shot estimations might fail to reflect the true ranking of architectures. However, their experiments are conducted in a toy search space with only 32 architectures. In this paper, we conduct a more comprehensive study on OSEs in three different benchmarking search spaces (i.e. NAS-Bench-101 [29], NAS-Bench-201 [8], and the recent NAS-Bench-301 [21]). We further analyze *how and why* OSE estimations have bias & variance. Moreover, we explore *how to improve* the OSEs.

Zero-shot Estimator (ZSE) More recently, in order to further reduce the architecture evaluation cost, several studies [16, 1] introduce "zero-shot" estimators that involve no training. In this work, we compare these ZSEs on several benchmarks. And by inspecting and analyzing how and why these ZSEs are biased, we find that existing ZSEs adapted from the pruning literature [1] have clear biases when used for ranking architectures.

Knowledge Our work reveals pieces of knowledge on OSEs and ZSEs. First of all, the quality of OSE and ZSE estimations depends on the characteristics of the search space. And these estimators are relatively better at distinguishing bad architectures than distinguishing good ones (Sec. 4).

For OSEs, some of the knowledge revealed by our work include 1) OSEs bias towards architectures with lower complexity in the early training phase. And this bias can be alleviated to various extents with sufficient training in different spaces (Sec. 5.1). 2) OSEs have variance and can be mitigated to some extent (Sec. 5.3, Sec. 6.1). 3) Reducing the sharing extent of OSEs can potentially improve the ranking quality, we experiment with multiple ways of sharing extent reduction and give out some insights (Sec. 6.3). For ZSEs, we reveal that 1) ZSEs are still performing worse than OSEs in their current stage, despite their better efficiency. They also cannot benefit from one-shot training. (Sec. 4.2). 2) Existing ZSEs adapted from pruning literature only reflect the inter-parameter sensitivity in an architecture, and cannot provide reliable rankings for architectures (Sec. 5.2).

Suggestions Based on our experiments and analyses, we give out some suggestions for future OSE applications: 1) Longer training makes one-shot estimations better (Sec. 4.1); 2) Using one-shot loss instead of accuracy helps in large search spaces with smaller inter-architecture differences, such as the DARTS space [14] (Sec. 4.1); 3) One should use enough validation data for OSEs, instead of merely several batches as in ZSEs (Sec. 4.1); 4) Using temporal ensemble helps reduce the ranking instability, and brings non-negative improvements on the ranking quality in different search spaces (Sec. 6.1); 5) In search space with isomorphic architectures, augmenting the sampling strategy to improve the sampling fairness is essential to avoid overestimating simple architectures (Sec. 6.2); 6) Affine operation should not be used in batch normalization (BN) during supernet training (Sec. 6.3).

Analysis Framework Our analysis framework of efficient architecture performance estimators is organized as follows. We first introduce the evaluation criteria for estimator quality in Sec. 3. And Sec. 4 presents the quality evaluation of multiple OSEs and ZSEs. Then, we conduct an organized analysis on *how and why* the OSE and ZSE estimations have biases and variances in Sec. 5. Specifically, their complexity-level, operation-level, and architecture-level biases are demonstrated and analyzed. And the stability of OSE accuracy and ranking along the training process are analyzed. And in Sec. 6, based on our analysis framework, we present several case studies on improving OSEs from three perspectives: i.e. reducing the variance, bias, and parameter sharing extent.

2 Related Work

2.1 Efficient Performance Estimators of Neural Architectures

One-shot Estimators The vanilla NAS method [34] trains each candidate architecture for 50 epochs to acquire its suitable parameters, which makes the overall NAS process prohibitively costly. As a remedy, ENAS [19] proposes to amortize the separate training costs by sharing the parameters among different architectures. Specifically, ENAS constructs an over-parametrized supernet such that all architectures can be evaluated using a parameter subset of it. Throughout the search process, the shared supernet parameters are updated on the training set. And an RNN controller is updated alternatively on the validation set. There are two types of parameter-sharing methods: 1) One-shot NAS methods [3, 12] that first train a supernet and then conduct architecture search without further supernet tuning. 2) Non-one-shot methods [19, 14, 28] that conduct supernet training and architecture search (i.e. controller update) jointly. This work focuses on evaluating the estimations of the "one-

shot" supernet, since it is the cleaner case without the complexity of varying controller settings and possible controller-supernet co-adaption. And we use "one-shot estimations" indistinguishably for parameter-sharing estimations w/ or w/o a jointly-trained controller.

Correlation of One-shot Estimators There exist some studies that carried out correlation evaluation for one-shot estimators. Zhang et al. [32] conducted a correlation comparison between the GHN hypernetwork and OSE. However, the correlation is evaluated using 100 architectures randomly sampled from a large search space, which is not a convincing and consistent benchmark metric. Yu et al. [30] conduct parameter sharing NAS in a toy RNN search space with only 32 architectures in total, and discover that the parameter sharing rankings do not correlate with the true rankings of architectures. Zela et al. [31] also report that the correlation of parameter-sharing estimations is not satisfying with a Spearman correlation coefficient. between -0.25 and 0.3 on a larger search space (NAS-Bench-1shot1 sub search space 3) with 24k architectures.[2]

In this paper, we conduct a more comprehensive study of OSE behaviors across three search spaces, and further investigate how and why the OSE estimations are not satisfying. We also propose and compare several techniques to mitigate the OSE correlation gap.

Zero-shot Estimators More recently, in order to further reduce the architecture evaluation cost, several researches [16, 1] propose "zero-shot" estimators that conduct no training and utilize random initialized models to estimate architecture performances. Based on the observation that good architectures have distinct local jacobian on different images, Mellor et al. [16] propose ta indicator based on input jacobian correlation. Abdelfattah et al. [1] adapt several ZSEs from recent pruning literature, and claim that these adapted estimators can perform well in NAS on NAS-Bench-201.

2.2 NAS Benchmarks

NAS benchmarks are proposed to enable researchers to verify the effectiveness of NAS methods efficiently. NAS-Bench-101 (NB101) [29] provides the performances of the 423k valid architectures in a cell-based search space. OSE cannot be easily applied for the whole NB101 search space due to its specific channel number rule. To reuse NB101 for benchmarking OSE, NAS-Bench-1shot1 (NB1shot) [31] picks out three sub-spaces of NB101, and a supernet can be easily constructed for these sub-spaces. In this work, we use the largest sub-space in NB1shot: NB1shot-3, and use the name "NB101" to refer to it. Another benchmark, NAS-Bench-201 (NB201) [8], provides the performances of all the 15625 architectures in a single-cell search space.

Previous tabular benchmarks exhaustively train all architectures in a search space much smaller than commonly-used ones (e.g. DARTS [14] with size over 10^{18}). Recently, NAS-Bench-301 (NB301) [21] is proposed as a benchmark in the DARTS space. It adopts a surrogate-based methodology that predicts architecture performances with the performances of about 60k anchor architectures.

3 Evaluation Criteria

This section introduces the major evaluation criteria used in our analysis framework. We denote the search space size as M, the true (ground-truth, GT) performances and approximated estimated scores of architectures $\{a_i\}_{i=1,\cdots,M}$ as $\{y_i\}_{i=1,\cdots,M}$ and $\{s_i\}_{i=1,\cdots,M}$, respectively, and the ranking of the true and estimated score y_i, s_i as $r_i, n_i \in \{1, \cdots, M\}$ ($r_i = 1$ indicates that a_i is the best architecture). The correlation criteria are

- Pearson coefficient of linear correlation (LC): $\text{corr}(y, s)/\sqrt{\text{corr}(y, y)\text{corr}(s, s)}$.
- Kendall's Tau ranking correlation (KD τ): The relative difference of concordant pairs and discordant pairs $\sum_{i<j} \text{sgn}(y_i - y_j)\text{sgn}(s_i - s_j)/\binom{M}{2}$.
- Spearman's ranking correlation (SpearmanR): The pearson correlation coefficient between the rank variables $\text{corr}(r, n)/\sqrt{\text{corr}(r, r)\text{corr}(n, n)}$.

Since the ability of differentiating between good architectures matters more than differentiating between bad ones, criteria that emphasize more on the relative order of architectures with good

[2] Zela et al. [31] report that the size of search space 3 is 24066, however, there is a small bug in the code that includes wrong architectures during the evaluation. The actual search space size is 14580.

performances are desired. Denoting $A_K = \{a_i | n_i < KM\}$ as the set of architectures whose estimated scores s is among the top K portion of the search space, we use two set of criteira

- Precision@K (P@topK) $\in (0, 1] = \frac{\#\{i | r_i < KM \wedge n_i < KM\}}{KM}$: The proportion of true top-K proportion architectures in the top-K architectures according to the scores.
- BestRanking@K (BR@K) $\in (0, 1] = \arg\min_{i \in A_K} r_i$: The best normalized ranking among the top K proportion of architectures according to the scores (Lower is better).

Corresponding to P@topK, we also compare P@bottomK $= \frac{\#\{i | r_i > (1-K)M \wedge n_i > (1-K)M\}}{KM}$ to reveal how the worst architectures are distinguished. And corresponding to BR@K, we inspect WorstRanking@K (WR@K) $= \arg\max_{i \in A_K} r_i$ to reveal how the supernet is likely to regard a bad architecture to be good (Lower is better). Note that the rankings and architecture numbers are all relative numbers normalized by the search space size M.

4 Evaluating Efficient Performance Estimators

4.1 Evaluation of One-shot Estimators

Trend of Different Criteria We inspect how the proposed criteria evolve during the training process. At each training step, S architectures are randomly sampled to calculate the update gradients. Unless otherwise noted, S=1 is used for most demonstrations. As shown in Fig. 1, **the convergence speeds of criteria are different.** And almost all criteria show a rising trend as the training goes on, which means that **OSE gives more reliable rankings with more sufficient training. And on larger search spaces (e.g. NB301), the training time should be longer.**

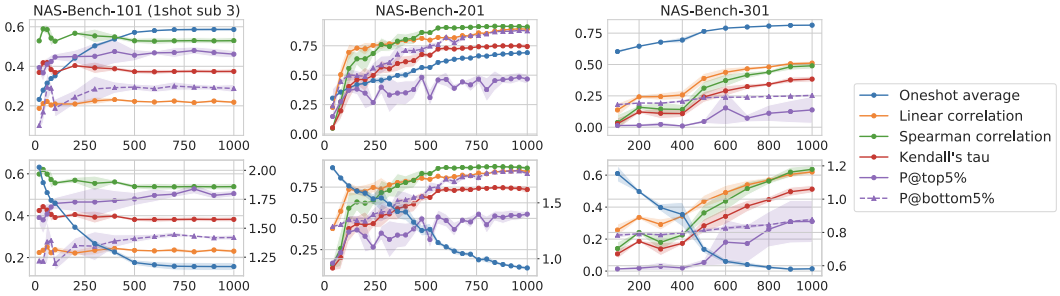


Figure 1: Top: Criteria of using OS accuracy as the estimations. Bottom: Criteria of using OS loss as the estimations (left Y axis: OS loss value).

NB201 is relatively easy, on which the supernet can give good estimations. In comparison, NB101 and NB301 are harder. On NB201, P@topK converges in around 250 epochs and then even gradually decreases. Meanwhile, LC, KD τ, and SpearmanR are still growing till 500 epochs, while the parameter sharing accuracy grows during the whole 1000 epochs. This suggests that differently-ranked models learn at different speeds, and the top-ranked models stand out faster. Another fact is that **one-shot (OS) estimations are better at distinguishing bad architectures (higher P@bottom5%) than distinguishing good ones (lower P@top5%).** Also, as shown in the appendix figure, the chance of regarding bad architectures as good (WR@5%) is relatively low, and the best ranking in top-predicted architecture (BR@5%) converges very quickly.

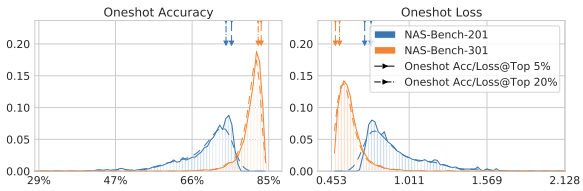


Figure 2: One-shot accuracy/loss distribution.

On the NB301 (DARTS) space with small inter-architecture differences, OS loss gives significantly better estimations than OS accuracy. For example, at epoch 1000, the KD τ of OS acc and loss are 0.381 and 0.512, respectively, while their P@top5% are 13.8% and 31.0%. Fig. 2 shows distributions of OS accuracy and loss, and we can see that the accuracy distribution is more concentrated than loss, which indicates the loss carries more information about prediction confidence.

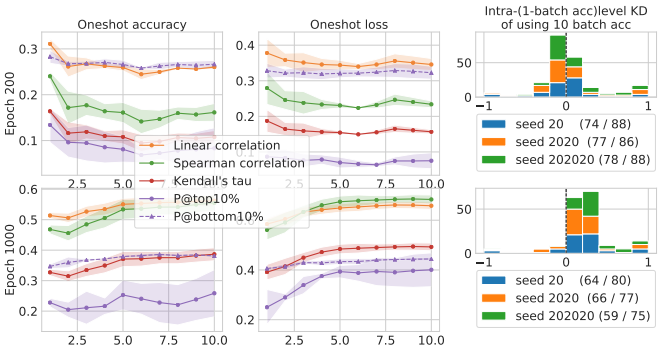


Figure 3: Criteria vary on NB301 as the batch number (X axis) changes. Right: The histogram of intra-level KDs using 10-batch OS acc, the "levels" are partitioned according to 1-batch OS acc. The legend gives out the number of levels in 1-batch evaluation with format "#acc levels with #arch>1/#total".

Effect of Validation Data Size We inspect the OSE quality of using different numbers of validation data batches to evaluate the OS scores, and find that on both NB201/NB301, **using more data improves the estimation quality.** For example, as shown by NB301 results in Fig. 3, criteria get better when the batch number increases from 1 to 10 at epoch 1000. Interestingly, at early training stages (Upper, epoch 200), criteria decrease with more batches (especially those of the OS acc). This is because that using only 1 batch (batch size=128) results in fewer levels (smaller resolution) of OS accuracy and gives many ties. And as shown in the intra-level KD histogram in Fig. 3(upper right), when the supernet is under-trained, its ability in distinguishing the intra-level architectures is weak, thus using more data brings negative effects.

4.2 Evaluation of Zero-shot Estimators

The ZSEs evaluated in our paper include *grad_norm*, *plain* [17], *snip* [13], *grasp* [25], *fisher* [23, 24], *synflow* [22], *jacob_cov* [16], and the assembled indicator *vote* [1]. Besides *jacob_cov*, other ZSEs are designed for network pruning by measuring the approximate loss change when certain parameters or activations are pruned. Denoting the parameters/activations/loss as θ, z and $\mathcal{L}$, the formula of these parameter-wise sensitivity indicators can be written as

$$\begin{aligned} &\textit{plain} : \mathcal{S}(\theta) = \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta; \quad \textit{snip} : \mathcal{S}(\theta) = |\frac{\partial \mathcal{L}}{\partial \theta} \odot \theta|; \quad \textit{grasp} : \mathcal{S}(\theta) = -(H\frac{\partial \mathcal{L}}{\partial \theta}) \odot \theta; \\ &\textit{synflow} : \mathcal{R} = \mathbb{1}^T(\prod_{\theta_i \in \theta} |\theta_i|)\mathbb{1}, \mathcal{S}(\theta) = \frac{\partial \mathcal{R}}{\partial \theta} \odot \theta; \quad \textit{fisher} : \mathcal{S}(z) = \sum_{z_i \in z} (\frac{\partial \mathcal{L}}{\partial z} z)^2. \end{aligned} \tag{1}$$

A recent work [1] proposes to sum up all parameter-wise sensitivities to evaluate the architecture. *jacob_cov* is a recently designed ZSE for architecture performances, it utilizes the correlation of input jacobian to indicate whether an architecture can differentiate between different inputs. And *vote* conducts a majority vote between various metrics as a comparator to rank two architectures.

Fig. 4 shows that on both search spaces, **the best ZSE is worse than that of OSE, even worse than the GT-Param correlation. And the relative effectiveness of ZSEs also varies between search spaces.** On NB301, *plain* achieves the best result among ZSEs (0.39 KD, 6.8% P@top5%), while the OSE achieves 0.52 KD and 31% P@top5%. While on NB201, *plain* performs worst among all ZSEs. And *jacob_cov* and *synflow* give relatively good estimations with KD of 0.61 / 0.57, which are nearly good as the Param-GT KD (0.61), but they do not perform well on NB301 (KDs are 0.23 / 0.2). For *vote*, we use the best-performing single ZSE as the voting experts (*jacob_cov*/*snip*/*synflow* on NB201, *plain*/*grasp*/*synflow* on NB301) and present the vote results too. Nevertheless, we find that *vote* does not bring improvements over the best one in the three consisting ZSEs. More results can be found in the appendix.

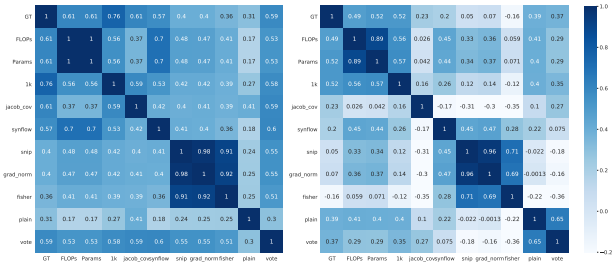


Figure 4: Kendall's Tau between GT, FLOPs/Params, OSEs (1k epoch) and ZSEs. Left: NB201; Right: NB301.

It is a natural idea to apply ZSEs on trained networks, thus we explore whether ZSEs can benefit from one-shot training. According to the results shown in the appendix, the estimation quality of ZSEs

decreases as the training process goes on. A possible explanation is that the parameter gradients are relatively small on trained supernet so that the ability of these gradient-based ZSEs decreases.

5 How & Why the Estimations Are Not Satisfying

5.1 Bias of One-shot Estimators

Complexity-level Bias To identify which architectures are under- or overestimated, we investigate the relationship of the true-estimated Ranking Difference (RD) $r_i - n_i; i = 1, \cdots, M$ and the architecture complexity (i.e. Params/FLOPs). RD serves as an indicator of overestimation for arch i: A positive RD indicates that this architecture is overestimated, otherwise, it is underestimated.

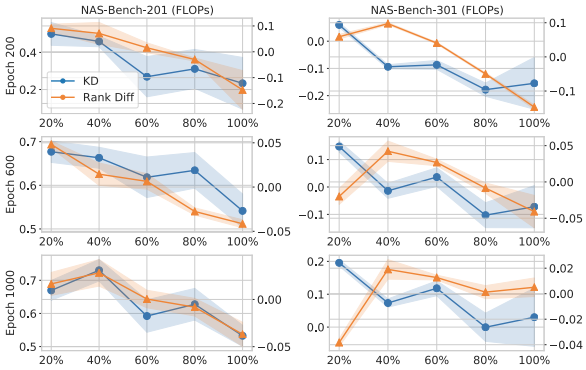


Figure 5: Complexity-level bias. Y axis left/right: KD τ / Average RD within the complexity group.

Sub-architectures have different amounts of calculation and might converge with a different speed. Thus, we conduct the complexity-level bias analysis. In, Fig. 5, we divide the architectures into five groups according to the amount of calculation (FLOPs), and show the KD & average RD in each group (the group with the smallest FLOPs is at the leftmost). In the early training stages, the average RD shows a decreasing trend, which means that the larger the model, the easier it is to be underestimated. This is because larger models converge slower. As the training goes on, the absolute average RD decreases, indicating that the issue of underestimating larger models gets gradually alleviated. Also, on both search spaces, the decreasing intra-group KD τ indicates that it is harder for the OSEs to compare larger models (which usually have better performances) than comparing smaller models.

Op-level Bias We inspect the changes of GT and OS accuracy when one operation is mutated to another (edit distance=1). On NB301, we examine 23476 mutation pairs and find that the OSE estimations overestimate the effects brought by dilation (Dil) convolutions (Convs): All mutation types from other operations *to* DilConvs witness a higher OS increase ratio than the GT one. And the skip_connect operation is underestimated: All mutation pairs *from* skip_connect cause the OS increase ratio to be higher than the GT one. For example, when mutating one skip_connect operation to dil_conv_3x3, only 39.0% out of 2336 pairs get GT increases, while 94.9% get OS increases. This phenomenon is more remarkable when we only consider mutation pairs within the largest complexity group (grouped by Param): Only 15.3% of 569 pairs get GT increases, while 92.3% get OS increases. On NB201, based on a similar inspection of the mutation pairs, we find that OSE estimations slightly overestimate avgpool3x3 and underestimate conv3x3. Generally speaking, the op-level bias on NB201 is not as large as that on NB301. See the appendix for the figures and more results.

5.2 Bias of Zero-shot Estimators

Arch-level Bias By inspecting the best and worst architectures indicated by ZSEs, we find that some existing ZSEs are not reliable for evaluating architectures. Fig. 6 shows that *synflow* has an excessive preference for large architectures. Actually, this preference is immediately evident from its formula. *snip*, *grad_norm* and *fisher* give similar rankings of architectures (see Fig. 4), and show improper preference on architectures with gradient explosion: On NB201, they show a clear preference for architectures without skip connections, which are far from optimal. The detailed analysis is presented in the appendix. To summarize, the estimators adapted from the fine-grained pruning literature are not yet suitable for evaluating architectures, since they are designed to reflect the (intra-model) relative *parameter-wise* sensitivity. For inter-model comparison, due to their sensitivity to scales and gradient explosion, a simple form of adding up the parameter-wise sensitivity provides improperly biased estimations for architecture performances, even coarse-level sensitivity (layer-wise).

Efficient ZSEs have the potential to compare arbitrary architectures (OSEs require parameter sharing,

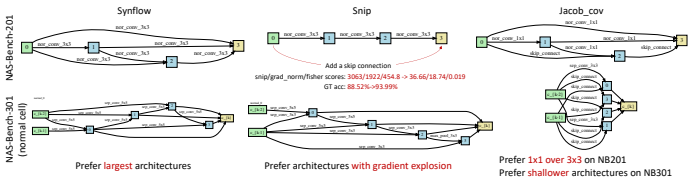
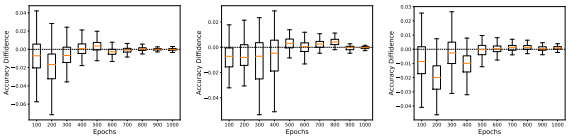


Figure 6: Best architectures ranked by zero-shot estimators.

and cannot support inter-space comparison easily). However, our analyses show that more engineering is needed to improve ZSEs. *jacob_cov* is a reasonable attempt. Since instead of adding up parameter-wise sensitivity, it conducts analysis at the architecture level, thus has the potential to reveal architecture performance. However, its performance on the harder NB301 space is not satisfying: The values are distributed in a very small range and shallow architectures are preferred.

5.3 Variance of One-shot Estimators

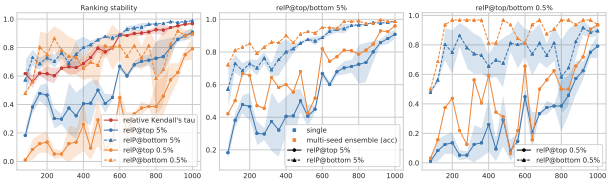
Accuracy Forgetting Due to the parameter sharing and the random sample training scheme, the training of subsequent architectures overwrites the weights of previous ones, thus degrades their OS accuracy. This "multi-model forgetting" phenomenon [4, 33] accounts for the variance of OS accuracies. Fig. 7 verifies the existence of the forgetting phenomenon. For each architecture in one epoch, we define its forgetting value (FV) as $acc_2 - acc_1$, where acc_1 refers to its valid accuracy right after its training, and acc_2 refers to its accuracy after all the architectures in this epoch have been trained. Fig. 7 shows that the forgetting phenomenon exists in the early training stages, where the FVs are negative. As training progresses, the variance of the FVs decreases, which is natural due to the learning rate decay. Also, the mean FV becomes positive, indicating that training other architectures can have positive transferring effects on previous architectures instead of negative ones (i.e. forgetting). This observation can be explained by the increasing trend of inter-architecture gradient similarity in Fig. 9.



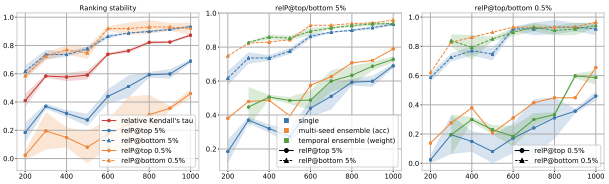
(a) NasBench-101 (b) NasBench-201 (c) NasBench-301

Figure 7: Multi-model forgetting phenomenon.

Ranking Stability We demonstrate the ranking stability in Fig. 8, since it plays an important role that influences the NAS process more directly than the accuracy stability. The criteria in this figure (i.e. relative KD, relative P@top/bottomK) are calculated with two sets of adjacent OS estimations, while the estimations of the latter checkpoint are taken as the GT one. We can see that the ranking stability increases with sufficient training and the OS rankings of bad architectures are relatively stable (relP@bottomK). On NB301, even with rather sufficient training (1k epoch) where the mean OS accuracy already saturates (Fig. 1), the ranking stability of top architectures is still not high (relP@top 0.5%$\sim$0.46). This is reasonable due to the small inter-architecture difference in the DARTS space. And as expected, averaging the OS accuracy of multiple supernets stabilizes OSE estimations. Also, the temporal weight ensemble of multiple checkpoints can stabilize the estimations (Sec. 6.1).



(a) NasBench-201 (measured every 40 epoch)



(b) NasBench-301 (measured every 100 epoch)

Figure 8: Ranking stability of OSEs.

6 How to Improve One-shot Estimations

Since different architectures require different values for supernet parameters, as the side effect of acceleration, parameter sharing serves as the intrinsic reason for the OSE correlation gap.

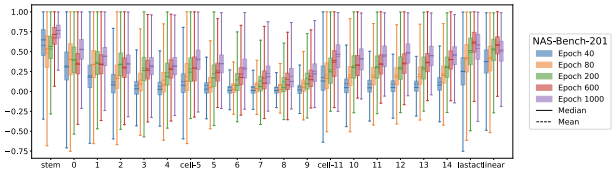


Figure 9: Gradient (cosine) similarity between NB201 architecture pairs of different layers at different epochs.

Fig. 9 shows the gradient similarity distribution between architecture pairs on NB201. We can see that the inter-architecture gradient similarities vary in a large range, and one common phenomenon on NB201 and NB301 is that the mean similarity between architecture pairs is lower in the middle-stage layers and the architectures' gradients in the very first and last layers are more similar. Another slightly counterintuitive fact is that the gradient directions become more similar as the training goes on, especially on NB201. This can explain the positive transferring effect in the latter training stages (Fig. 7).

Due to the parameter sharing, the random sample training scheme of the supernet causes the estimation variances. On the other hand, improper sampling distribution leads to estimation biases. There are two types of reasons for the bias: 1) Some architectures (e.g. with larger complexity) might need higher sampling probability to match their relative performance in standalone training. 2) Architecture are sampled from an unfair distribution, i.e., some architectures have undesirable higher probabilities.

In this section, echoing the above analysis, we conduct case studies to improve the OSE estimations from 3 perspectives, i.e. reducing the variance, bias, and parameter sharing extent. Sec. 6.1 experiments with 2 techniques that can reduce the OS estimation variance.And in Sec. 6.2, we demonstrate that using de-isomorphic sampling in space with isomorphic architectures (NB201) helps improve the sampling fairness, thus reduce the estimation bias.

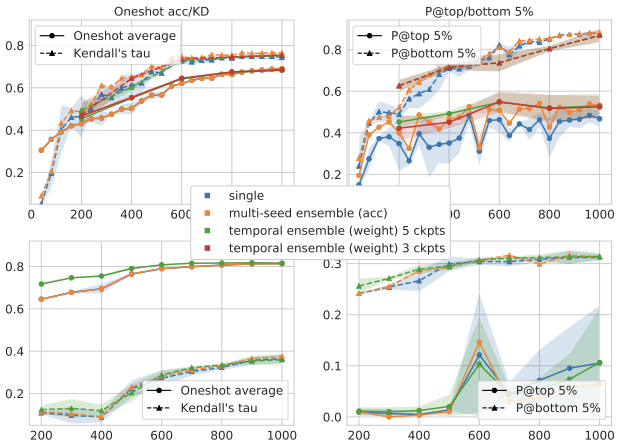


Figure 10: Effect of ensemble techniques on OSEs. Top: NB201; Bottom: NB301.

6.1 Variance Reduction

Ensemble to Reduce Temporal Variance Sec. 5.3 shows that averaging OS scores of several supernets stabilizes the estimations, but this is not practical due to the linearly enlarged consumption. Guo et al. [11] propose to stabilize OS estimations by temporally averaging weights of supernet checkpoints. Besides the variance reduction effect shown in Fig. 8, Fig. 10 shows whether ensembling techniques can bring other ranking quality improvements. We can see that temporally ensembling 3/5 checkpoints brings improvements on NB201, but brings no bias improvements on NB301.

Sampling Variance Reduction We compare the results of using different MC sample numbers S in supernet training. We also adapt Fair-NAS [6] sampling strategy to the NB201/301 spaces (a special case of MC sample 5/7 for 201/301). Using multiple MC architecture samples has different influences in different spaces. Using multiple MC samples on NB301 is beneficial for the estimation quality, while the estimation quality on NB201 decreases slightly as the MC sample number increases. See the appendix for detailed results and discussions.

6.2 Sampling Fairness Improvement

The NB201 search space contains many isomorphic architectures with different representations. There are 6466 unique structures (out of 15625) after de-isomorphism. And we find that even with rather sufficient training, the supernet still overestimates some simple architectures significantly. Fig. 11(left) shows the top-2 ranked architectures by the average of 3 supernet's OS scores at epoch 1000. With vanilla sampling (Iso), OS estimations bias towards simple architectures (a single Conv)

with many isomorphic counterparts (Iso group size=31). This is because their equivalent sampling probability is higher and the parameters are trained towards their desired directions.

We compare the results of sampling with/without isomorphic architectures in Fig. 11(right). If de-isomorphism (deiso) sampling is not used, the supernet performs much worse on good architectures in that BR@0.5% and P@5% are significantly worse (1.9% V.S. 0.23%, 21.3% V.S. 46.7%). We also experiment with a post-de-isomorphism (post-deiso) technique, in which the estimations of architectures in an isomorphic group are averaged during testing, while no changes are made during training. Post-deiso achieves slight improvements over "no post-deiso", which might owe to the decreased estimation variances. Actually, the deiso sampling strategy is to find a de-isomorphic representation space and conduct uniform sampling in it, and our case study provides another evidence for the statement made by [26] that representations can be critical for NAS methods.

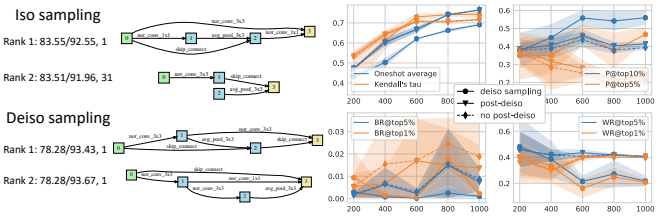


Figure 11: Comparison of Iso / Deiso sampling strategy. Left: Top-2 ranked architectures with Iso/Deiso sampling, the legend's format is "OS/GT acc (%), Iso group size". Right: Criteria comparison along the training process.

6.3 Sharing Extent Reduction

Operation Pruning We remove one or two operations in the search space (SS) and conduct supernet training on the resulting sub-SS. After training the supernet, we compare the OS estimations on the sub-SS provided by the supernet trained on full SS and the sub-SS. The detailed results and analyses can be found in the appendix. And the conclusion is: Sharing extent reduction by removing operations can bring improvements to the average OS scores of the remaining architectures in the sub-SS, especially in the early training stages. However, **whether the improved absolute OS scores can bring ranking quality improvements is questionable, and the results vary across SSes**.

One-shot Pruning We conduct SS pruning on NB201 by selecting the top 10%, 25%, 50% architectures ranked by the OS scores of supernet (epoch 600), and continue to finetune the supernet to 1000 epoch with these architectures. The good news is that on NB201, OS pruning brings improvements on both the average OS score and ranking quality in the sub-SS: 2.2%/1.3%/0.1% average OS score increases and 0.189/0.046/0.086 KD increases when the sub-SS contains 10%/25%/50% architectures, respectively. The results reveal **the potential of dynamic SS pruning for improving the OSE quality, especially for good architectures**. However, this per-architecture hard pruning scheme is not practical since it needs an exhaustive test of the full search space. To explore practical dynamic SS pruning methods, we conduct a case study on per-architecture soft pruning with a jointly-trained controller, where the controller gives higher sampling probability to the architectures with higher OS scores. The results and analyses are shown in the appendix.

Remove the Affine Operation in BN Through gradient visualization, we find that the minimum gradient similarity of BN scales between architecture pairs is low. Thus we compare using or not using BN affine operations. Check the appendix for the gradient visualization and comparison results. And the suggestion is that **one should not use BN affine operations in the search process**.

7 Conclusion

We present an analysis framework of efficient architecture performance estimators in NAS, which contains a set of carefully developed criteria, and well-organized analyses. Within the framework, we conduct an in-depth analysis of multiple one-shot (OSEs) and zero-shot estimators (ZSEs), and interpret their behaviors on three different benchmarking search spaces. Our work reveals the properties, weaknesses (variance and bias) of current neural architecture estimators. For OSEs, we also experiment with several corresponding mitigations of their weakness. Through our study, suggestions are made to guide future NAS applications, and directions are pointed out to mitigate the current OSE and ZSE weaknesses. Besides the take-away knowledge, our analysis framework could be utilized in future research to diagnose efficient architecture performance estimators.

References

[1] Mohamed S. Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D. Lane. Zero-Cost Proxies for Lightweight NAS. In *International Conference on Learning Representations*, 2021.

[2] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating neural architecture search using performance prediction. In *International Conference on Learning Representations Workshop*, 2018.

[3] 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*, pages 550–559, 2018.

[4] Yassine Benyahia, Kaicheng Yu, Kamil Bennani Smires, Martin Jaggi, Anthony C Davison, Mathieu Salzmann, and Claudiu Musat. Overcoming multi-model forgetting. In *International Conference on Machine Learning*, pages 594–603. PMLR, 2019.

[5] Andrew Brock, Theodore Lim, James Millar Ritchie, and Nicholas J Weston. Smash: One-shot model architecture search through hypernetworks. In *6th International Conference on Learning Representations*, 2018.

[6] 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.

[7] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3681–3690, 2019.

[8] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020.

[9] Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21, 2019.

[10] 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*, pages 7036–7045, 2019.

[11] Ronghao Guo, Chen Lin, Chuming Li, Keyu Tian, Ming Sun, Lu Sheng, and Junjie Yan. Powering one-shot topological nas with stabilized share-parameter proxy. In *European Conference on Computer Vision*, pages 625–641. Springer, 2020.

[12] 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.

[13] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

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

[15] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Advances in Neural Information Processing Systems 31*, pages 7816–7827. 2018.

[16] Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J. Crowley. Neural architecture search without training. *arXiv preprint arXiv:2006.04647*, 2021.

[17] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in neural information processing systems*, pages 107–115, 1989.

[18] Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik. Xnas: Neural architecture search with expert advice. *Advances in Neural Information Processing Systems*, 32:1977–1987, 2019.

[19] 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.

[20] 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.

[21] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 2020.

[22] Hidenori Tanaka, Daniel Kunin, Daniel LK Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*, 2020.

[23] Lucas Theis, Iryna Korshunova, Alykhan Tejani, and Ferenc Huszár. Faster gaze prediction with dense networks and fisher pruning. *arXiv preprint arXiv:1801.05787*, 2018.

[24] Jack Turner, Elliot J Crowley, Michael O'Boyle, Amos Storkey, and Gavin Gray. Block-swap: Fisher-guided block substitution for network compression on a budget. *arXiv preprint arXiv:1906.04113*, 2019.

[25] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

[26] Colin White, Willie Neiswanger, Sam Nolen, and Yash Savani. A study on encodings for neural architecture search. *Advances in Neural Information Processing Systems*, 33, 2020.

[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*, pages 10734–10742, 2019.

[28] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. Cars: Continuous evolution for efficient neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1829–1838, 2020.

[29] 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.

[30] 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*, 2020.

[31] Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*, 2020.

[32] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *International Conference on Learning Representations*, 2019.

[33] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, and Steven Su. Overcoming multi-model forgetting in one-shot nas with diversity maximization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[34] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Although we spend GPU hours on analyzing and benchmarking the efficient architecture estimators. But the information given by our work can help reduce trial-and-error in future NAS application, which means large positive environmental impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We give the related information in the appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use the data provided by NAS-Bench-101/201/301, and the codes provided by a previous paper on zero cost indicators for NAS [1]. We've cited all these papers.
   (b) Did you mention the license of the assets? [Yes] These data/codes all have open-source licenses.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include our codes and some of the assets in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] Since these data/codes are all open sourced data/codes from previous papers with open-source licenses.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] These data do not contain personally identifiable information and offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]