Landmark Regularization: Ranking Guided Super-Net Training in Neural Architecture Search

Kaicheng Yu*
CVLab, EPFL
kaicheng.yu.yt@gmail.com

René Ranftl
Intelligent Systems Lab, Intel
rene.ranftl@intel.com

Mathieu Salzmann
CVLab, EPFL
mathieu.salzmann@epfl.ch

Abstract

Weight sharing has become a de facto standard in neural architecture search because it enables the search to be done on commodity hardware. However, recent works have empirically shown a ranking disorder between the performance of stand-alone architectures and that of the corresponding shared-weight networks. This violates the main assumption of weight-sharing NAS algorithms, thus limiting their effectiveness. We tackle this issue by proposing a regularization term that aims to maximize the correlation between the performance rankings of the shared-weight network and that of the standalone architectures using a small set of landmark architectures. We incorporate our regularization term into three different NAS algorithms and show that it consistently improves performance across algorithms, search-spaces, and tasks.

1. Introduction

Modern algorithms for neural architecture search (NAS) can now find architectures that outperform the human-designed ones for many computer vision tasks [21, 42, 7, 34]. A driving factor behind this progress was the introduction of parameter sharing [27], which reduces the search time from thousands of GPU hours to just a few and has thus become the backbone of most state-of-the-art NAS frameworks [3, 49, 5, 23]. At the heart of all these methods is a shared network, a.k.a. super-net, that encompasses all architectures within the search space.

To train the super-net, NAS algorithms essentially sample individual architectures from the super-net and train them for one or a few steps. The sampling can be done explicitly, with strategies such as reinforcement learning [27, 6], evolutionary algorithms [15, 39], or random sampling [52, 18], or implicitly, by relying on a differentiable parameterization of the architecture space [19, 22, 5, 42, 53]. Whether explicit or implicit, the underlying assumption of these methods is that the relative performance of the individual architectures in the super-net is highly correlated with the performance of the same architectures when they are trained in a stand-alone fashion. If this were the case, one could then safely choose the best individual architecture from the super-net after the search and use it for evaluation. However, this assumption was disproved in [52, 54], who showed a correlation close to zero between the two rankings on complex search spaces [47, 22]. The major reason behind this is fairly intuitive: To be optimal, different individual architectures should have different parameter values, which they cannot because the parameters are shared. Super-net training will thus *not* produce the same results as stand-alone training. More importantly, there is no guarantee that even the relative ranking of the architectures will be maintained. While for simple, linear search spaces the ranking can be improved by using a carefully crafted sampling strategy [8, 9], addressing the ranking disorder for more realistic, complex search spaces remains an open problem [50].

In this paper, we propose to explicitly encourage archi-

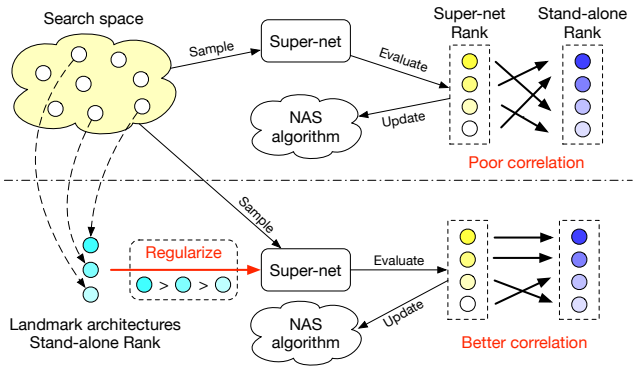


Figure 1: Traditional super-net training leads to poor correlation between relative stand-alone performance and super-net performance (top). We sample landmark architectures and use their relative performance to guide training towards an improved ranking and show that this improves the search performance (bottom).

*This work was partially done during an internship at Intel, and supported in part by the Swiss National Science Foundation.

tectures represented by the super-net to have a similar ranking to their counterparts trained in a stand-alone fashion. As illustrated by Figure 1, we leverage a set of landmark architectures, that is, architectures with known stand-alone performance, to define a regularization term that guides super-net training towards this goal. We show that a small set of landmark architectures suffices to significantly improve the global ranking correlation, so that the overall search procedure, including the independent training of the landmark architectures, remains tractable.

Our regularization term is general and does not make assumptions about the specific sampling algorithm used for super-net training. As such, it can easily be combined with many popular weight-sharing NAS algorithms. We demonstrate this by integrating it into three different algorithms [15, 24, 11] that are representative of three different categories of weight-sharing NAS algorithms: i) Algorithms that sample architectures from the super-net in an unbiased manner throughout the super-net training [18, 52, 2, 15, 8]; ii) approaches that employ learning-based samplers, which are updated during the training based on the performance of the partially-trained super-net [27, 19, 24, 40, 55]; and iii) algorithms that rely on differentiable architecture search [22, 6, 44, 25, 45].

Our extensive experiments on CIFAR-10 and ImageNet show that landmark regularization significantly reduces the ranking disorder that occurs in these algorithms and that they are consequently able to consistently find better-performing architectures. To further showcase the effectiveness and generality of our approach, we study its use in the context of architecture search for monocular depth estimation. To the best of our knowledge, this is the first attempt at performing NAS for this task. We, therefore, construct a dedicated search space and show that a landmark-regularized NAS algorithm can find novel architectures that improve upon the state of the art in this field.

2. Related work

Different from manual designing convolutional neural networks, which have been shown successful in many computer vision tasks [16, 14, 41, 51], neural architecture search (NAS) methods automate the design process and can be categorized into conventional approaches, that obtain architecture performance via stand-alone training [56, 57, 36, 40, 31, 32, 39], and weight sharing NAS, where the performance is obtained from one or a few super-nets that encompass all architectures within the search space [27, 24, 6, 55, 26, 48]. Motivated by the success of early NAS works, the literature has now branched into several research directions, such as using multi-objective optimization to discover architectures under resource constraints for mobile devices [37, 36, 42, 6, 15, 3], applying NAS to other computer vision tasks than image recognition [21, 7, 20, 34], and using knowledge distillation to eliminate the performance gap between super-net and stand-alone training for linear search spaces based on MobileNet [5, 49].

In contrast to the diversity of these research directions, super-net training in weight-sharing NAS has remained virtually unchanged since its first appearance in [18, 15, 2]. At its core, it consists of sampling one or few architectures at each training step, and updating the parameters encompassed by these architectures with a small batch of data. This approach has been challenged in many ways [4, 18, 52, 46], particular thanks to the introduction of the NASBench series [28, 47, 12, 35, 54, 10] of NAS benchmarks, which provide stand-alone performance of a substantial number or architectures and thus facilitate the analysis of the behavior of NAS algorithms. A critical issue that has been identified recently is the inability of most modern NAS algorithms to surpass simple random search under a fair comparison. In [52], this was traced back to to the low ranking correlation between stand-alone performance and the corresponding super-net estimates. While recent works [15, 8] have shown that the ranking correlation is high on a MobileNet-based search space, where one only searches for the convolutional operations and the number of channels, others [50, 54] have revealed that the correlation remains low on cell-based NASNet-like search spaces [22, 24, 56], even when carefully tuning the design of the super-net.

In this paper, we introduce a simple, differentiable regularization term to improve the ranking correlation in weight-sharing NAS algorithms. We show that the regularization term can be used in a variety of weight-sharing NAS algorithms, and that it leads to a consistent improvement in terms of ranking correlation and final search performance.

Our regularization leverages the stand-alone performance of a few architectures. While some contemporary works also use ground-truth architecture performance, our approach differs fundamentally from theirs. Specifically, these methods aim to train a performance predictor, based on an auto-encoder in [23] or on a graphical neural network in [38], and are thus only applicable to weight-sharing NAS strategies that exploit such a performance predictor. By contrast, we add a regularizer to the super-net training loss, which allows our method to be applied to most weight-sharing NAS search strategies. Furthermore, our approach requires an order of magnitude fewer architectures with associated stand-alone performance; in our experiments, we use 30 instead of 300 in [23] and 1000 in [38].

3. Preliminaries

We first revisit the basics of super-net training and highlight the ranking disorder problem.

Let Ω be a search space, defined as a set of N neural network architectures $a_i, i \in [1, N]$. stand-alone training

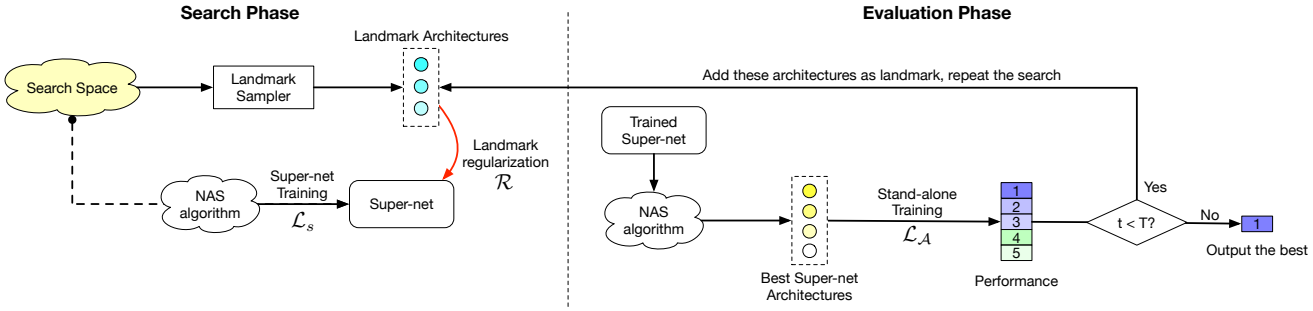


Figure 2: Overview of our approach. Left: During the search phase, we first sample a set of landmark architectures and obtain their stand-alone performance. We train the super-net with our regularization term such that the landmark ranking is preserved. Right: After a round of training, we sample the best architectures given the current super-net performance and evaluate their stand-alone performance. We add these architectures to the set of landmarks and repeat the process for a few iterations.

optimizes the parameters θ_{a_j} of architecture a_j independently from the other architectures by minimizing a loss function $\mathcal{L}(x, \theta_{a_j})$, thus yielding the optimal parameters $\theta^*_{a_j}$ for the given training data x_{train}. Without weight sharing, NAS then aims to train a search algorithm S to sample architectures $S(\Omega) = \{a_k\}$ whose stand-alone performance outperforms that of other architectures, that is, $\mathcal{L}(x, \theta^*_{a_k}) < \mathcal{L}(x, \theta^*_{a_j}),\ \forall a_j \notin S(\Omega)$. In its simplest form, i.e., random search, there is no search algorithm to train per se, and one just samples a set of architectures, trains them in a stand-alone fashion, and ranks them to choose the ones with lowest loss.

With a proper search algorithm, however, NAS without weight sharing requires training and evaluating an impractically large amount of stand-alone architectures [56, 57, 39, 36, 37] . To circumvent this, weight-sharing NAS strategies construct a super-net θ^s that encompasses all architectures in the search space. The relative performance of individual architectures sampled from the super-net then acts as an estimate of their relative stand-alone performance. Training is typically formulated as minimizing a joint loss over all architectures that are represented by the super-net:

$$\mathcal{L}_s(\theta^s) = \sum_{i=1}^{N} \mathcal{L}(x, \theta^s_{a_i}), \tag{1}$$

where the optimization is made tractable by randomly sampling terms from this sum for each update.

In contrast to stand-alone training, parameters overlap between different architectures and in general we have that $\theta^s_{a_i} \cap \theta^s_{a_j} \neq \varnothing$. Since the parameters shared by a_i and a_j would typically not have the same optimal values in the stand-alone training [4], the optimal solution of super-net training is *not* the same as that of stand-alone training, and neither is the ranking of the architectures.

4. Landmark regularization

We address the issue of low ranking correlation by introducing a simple yet effective approach to regularize super-net training with prior knowledge about the relative performance of individual architectures. To this end, we sample $M << N$ architectures to form a set of landmark architectures, $\Omega_L = \{a_i\}_{i=1}^M$, and obtain their stand-alone performance $\mathcal{L}_{valid}(x_{valid}, \theta^*_{a_i})$ on validation data, where $\theta^*_{a_i} = \arg\min \mathcal{L}(x_{train}, \theta_{a_i})$. We shorten this to $\mathcal{L}_\mathcal{A}(x, \theta^*_{a_i})$ to simplify the notation. To ensure that the trained super-net is predictive of the performance of the stand-alone architectures, we aim to preserve the relative performance of these landmark architectures in the super-net. Formally, we want to enforce that

$$\mathcal{L}_\mathcal{A}(x, \theta^*_{a_i}) \leq \mathcal{L}_\mathcal{A}(x, \theta^*_{a_j}) \Rightarrow \mathcal{L}(x, \theta^s_{a_i}) \leq \mathcal{L}(x, \theta^s_{a_j}), \tag{2}$$

for all pairs of architectures $i, j \in [1, M]$.

To achieve this, we propose a differentiable regularization term that can readily be integrated into standard super-net training procedures. We first sort the landmark architectures in ascending order based on their ground-truth loss:

$$\forall i, j \in [1, M], \quad i < j \Leftrightarrow \mathcal{L}_\mathcal{A}(x, \theta^*_{a_i}) \leq \mathcal{L}_\mathcal{A}(x, \theta^*_{a_j}), \tag{3}$$

and use this ordering to define a regularization term

$$\mathcal{R}(\theta^s) = \sum_{i=1}^{M} \sum_{j=i+1}^{M} \max(0,\ \mathcal{L}(x, \theta^s_{a_i}) - \mathcal{L}(x, \theta^s_{a_j}))\ , \tag{4}$$

which penalizes deviations from the ordering induced by (2). Since all operations involved in the proposed regularizer are differentiable, it is straight-forward to implement in existing deep learning frameworks. Note that the several alternative formulations of this loss are possible. We discuss them in the supplementary material.

The landmark-regularized training loss is then given by

$$\mathcal{L}(\theta^s) = \mathcal{L}_s(\theta^s) + \lambda \mathcal{R}(\theta^s)\ , \tag{5}$$

where $\lambda > 0$ is a hyper-parameter. To avoid leakage of validation data into the super-net training, we follow [22, 24, 6] and split the training set into two parts. We use one part to evaluate the super-net loss $\mathcal{L}_s$ and the other part to evaluate the regularization $\mathcal{R}$ during training.

Computational cost. The complexity of evaluating the regularizer discussed above is $O(M^2)$. This factor can have a significant impact on the training time, as the regularizer has to be evaluated at every training iteration. To reduce this computational burden while still encouraging the super-net to encode a correct architecture ranking, we propose to randomly sample m pairs of landmark architectures i, j at each iteration and evaluate their ranking:

$$\mathcal{R}(\theta^s) = \sum_{i,j}^{m} \max(0, \mathcal{L}(x, \theta^s_{a_i}) - \mathcal{L}(x, \theta^s_{a_j})) \; . \qquad (6)$$

This reduces the time complexity from $O(M^2)$ to $O(m)$. We will show empirically that even evaluation with a single pair introduces virtually no degradation of the resulting architecture ranking.

Landmark selection. The choice of the landmark architectures has an impact on the effectiveness of our regularizer. In particular, we would like to use landmarks that cover the complete search space. To promote this, we introduce the diverse landmark sampling strategy described by Algorithm 1. We start by randomly sampling a root architecture from the search space. We then generate $M-1$ diverse architectures by mutating the root architecture such that the Hamming distance is larger than a threshold τ. For example, in the DARTS search space [22], one architecture is encoded as a sequence, where each element represents selecting either a previous node or an operation. Mutating an architecture is then achieved by randomly altering one element, and the Hamming distance between two architectures is computed as the number of unequal elements.

Regularization schedule. For all practical applications, the number of landmark architectures will be several orders of magnitude smaller than the total number of architectures in the search space. The regularization term thus needs to have high weight to be effective and have a noticeable effect on the training. However, too much regularization can negatively impact the training dynamics, especially in the early stages. To alleviate this issue, we propose to enable regularization after a warm-up phase and to gradually increase its influence using a cosine schedule. Specifically, we set the regularization weight at epoch t to

$$\lambda_t = \mathbf{1}_{t>t_w} \cdot \frac{1}{2} \left(1 + \cos \frac{\pi(t - t_w)}{t_{total}} \right) \lambda_{\max}, \qquad (7)$$

where t_w denotes the number of warm-up epochs, t_{total} denotes the total number of epochs, and λ_{max} denotes the final value for the regularization parameter.

Algorithm 1: Landmark-regularized training.

Input : Search space Ω, NAS algorithm $\mathcal{S}$, super-net and stand-alone losses $\mathcal{L}_s$, $\mathcal{L}_\mathcal{A}$, distance threshold τ.

initialize super-net parameters θ^s
initialize an empty landmark set Ω_L
while *step* $t < T$ **do**
  Obtain landmark architectures
  $a_0 \leftarrow \texttt{RandomSample}(\Omega, 1)$
  while $|\Omega_L| < M \times T$ **do**
    $a_t \leftarrow \texttt{Mutate}(a_0)$
    if $d_{Hamming}(a_t, a_0) > \tau$ **then**
      add a_t to Ω_L
    end
  end
  foreach *training step* **do**
    Train super-net $\mathcal{L}$ while sampling m pairs
    $\{(a_i, a_j)\} \leftarrow \texttt{RandomSample}\ (\Omega_L \times \Omega_L, m)$
    $\mathcal{L} = \mathcal{L}_s(\theta^s) + \lambda \mathcal{R}(\theta^s)$
    Train sampler $\mathcal{S}$ *if necessary*
  end
  Sample architectures to get stand-alone performance
  $\forall a_j \in \Omega_t \leftarrow \mathcal{S}(\Omega)$, obtain $\mathcal{L}_\mathcal{A}(x, \theta_{a_j})$
  $\Omega_L \leftarrow \Omega_L \cup \Omega_t$
end
Output: Model $a_t \leftarrow \arg\min_{a \in \Omega_t} \mathcal{L}_{GT}(x, \theta_a)$

Application to existing NAS methods. The proposed regularization term is independent of the search algorithm, and thus widely applicable to many different weight-sharing NAS algorithms. We discuss its use in three different classes of NAS strategies, specific instances of which will act as baselines in our experiments.

We categorize weight sharing NAS algorithms into three broad categories according to their interaction with the super-net: i) unbiased architecture sampling algorithms [18, 52, 2, 15, 8] that sample one or a few paths uniformly at random, ii) learning based sampling that favors the most promising architectures given the performance of the current, partially trained super-net [27, 19, 24, 40, 55], and iii) differentiable architecture search that parametrizes the architecture sampling probability as part of the super-net [22, 6, 44, 25, 45].

For the first two categories, our method can be directly incorporated into the super-net training to improve its quality, and hence to improve the final search results. For algorithms in the last category, the algorithm is usually composed of two distinct phases that are executed alternatingly. In the the first phase the parameters that define the architecture are fixed and only the weights are updated, whereas in the second phase the weights are fixed and the architecture parameters are updated. Our regularization term can directly be integrated into the first phase when using discrete architectures in the forward pass as in works that employ the Gumbel-Softmax [11, 42] or binary gates [6, 44]. When

using continuous architecture specifications in the forward pass [22, 45], we do not have a discrete sub-path to sample and evaluate the regularization term, so additional care has to be taken to incorporate it. We discuss various ways to do this in supplementary material.

Multi-iteration pipeline. Figure 2 depicts the complete training pipeline. We first sample landmark architectures using our landmark selection strategy and obtain their stand-alone performance. We then train the NAS algorithm with landmark regularization. After a round of training, we sample the top M architectures using the trained NAS algorithm, obtain their stand-alone performances, and add these architectures to the set of landmarks. We proceed training of the super-net with the expanded set of landmarks and iterate this process.

In our experiments on different tasks and algorithms, we observed a stable improvement after sampling 3 sets of $M = 10$ architectures for a total of 30 landmarks, which is computationally feasible. Additionally, our algorithm can leverage previously trained models to improve the search by simply adding them to the landmark set. Considering that search spaces usually encompass billions of architectures, the number of landmarks is negligible.

5. Experiments

To validate the landmark regularization we incorporate it into three popular weight-sharing algorithms and evaluate them on the task of image classification using the CIFAR-10 [17] and ImageNet [33] datasets. We then discuss a new search space for monocular depth estimation architectures and show that our approach also applies to this new task. Finally, we ablate the key components and hyperparameter choices of our landmark regularization.

We used PyTorch for all our experiments and follow the evaluation framework defined in [52] to ensure a fair comparison with the baseline methods. Following [28, 29], we shorten the training time from 600 to 100 epochs on CIFAR-10 and from 250 to 50 on ImageNet, which still yields a good prediction quality. We release our code at https://github.com/kcyu2014/nas-landmarkreg.

Baselines. We select single-path one-shot (SPOS) as a representative unbiased architecture sampling algorithm. We use SPOS to train the super-net, followed by an evolutionary search to select the best models based on the super-net performance [15]. Among the learning-based architecture sampling methods, we select neural architecture optimization (NAO) [24], which trains an explicit auto-encoder-based performance predictor. Finally, for differentiable architecture search, we select the gradient-based search using a differentiable architecture sampler (GDAS) [11], which has been widely used in other works [44, 42, 6, 19]. See the supplementary material for more details and hyper-parameter settings of the baseline algorithms.

	NASBench-101				
Model	S-KdT	Mean Acc.	Best Rank	Best Acc.	Cost
SPOS	0.267 ± 0.02	91.02 ± 0.52	38953	92.82	11.89
SPOS+Ours	0.347 ± 0.03	92.48 ± 0.51	27697	92.99	14.92
NAO	0.329 ± 0.11	90.56 ± 0.88	131969	91.60	15.46
NAO+Ours	0.457 ± 0.03	92.23 ± 1.32	9313	93.34	21.04
	NASBench-201				
Model	S-KdT	Mean Acc.	Best Rank	Best Acc.	Cost
SPOS	0.771 ± 0.04	87.66 ± 4.95	3383	92.30	6.26
SPOS+Ours	0.802 ± 0.02	92.08 ± 0.37	2557	92.53	7.23
GDAS	0.691 ± 0.01	93.58 ± 0.12	463	93.48	14.42
GDAS+Ours	0.755 ± 0.01	93.98 ± 0.09	109	93.84	16.28
NAO	0.653 ± 0.05	91.75 ± 1.52	649	93.35	3.51
NAO+Ours	0.758 ± 0.05	92.84 ± 0.71	179	93.75	4.20

Table 1: Results on NASBench-101 and NASBench-201. We report the S-KdT at the end of training, the mean stand-alone accuracy of the searched architectures, the best rank, and the best accuracy. Each method was run 3 times. We also report the search cost in hours on Tesla-V100 (32Gb).

Hyperparameters. We sample $M = 10$ landmark architectures at each iteration, and perform $T = 3$ iterations. We sample $m = 1$ pairs for each training step and set $\lambda_{max} = 10$ in all of our experiments unless otherwise specified. The Hamming distance threshold τ is set according to the configuration of each search space. We train the baselines for the same total number of epochs, to ensure that any performance improvement cannot be attributed to our approach sampling more architectures.

Metrics. We follow [15, 50] and report the ranking correlation in terms of the sparse Kendall-Tau (S-KdT). We sample 200 architectures randomly to compute this metric for the CIFAR-10 experiments, 90 for the ImageNet experiments, and 20 for the monocular depth estimation experiments. Note that we exclude the landmark architectures from this set to avoid reporting overly optimistic numbers for our approach. Furthermore, following [52, 12], we report the mean and best stand-alone performance of the best architectures found over three independent runs.

5.1. Image classification on CIFAR-10

Since the inception of NAS, CIFAR-10 has acted as one of the main datasets to benchmark NAS performance [57, 27, 24, 15, 45, 48]. We utilize two search spaces, NASBench-101 and NASBench-201, for which the stand-alone performance of many architectures is known.

NASBench-101. NASBench-101 [47] is a cell search space that contains 423,624 architectures with known stand-alone accuracy on CIFAR-10. It is the largest tabular benchmark search space to date. We use the implementation of [52] to benchmark the performance of SPOS and NAO. We do not report the results of GDAS on this search space, as the

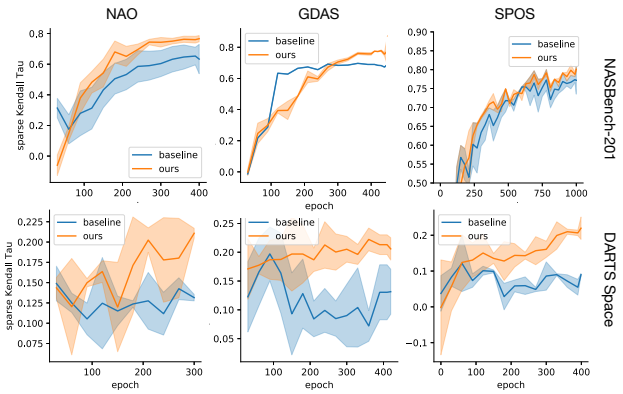


Figure 3: Evolution of the S-KdT of three NAS algorithms on two search spaces. Landmark regularization significantly improves the ranking correlation of the super-net in all cases.

matrix-based configuration used in NASBench-101 is not amenable to differentiable approaches [47, 54]. We set the Hamming distance threshold to $\tau = 5$.

As shown in Table 1 (top), landmark regularization improves the ranking correlation (S-KdT) from 0.267 to 0.347 for SPOS, and from 0.329 to 0.457 for NAO. This translates to a 1-2% improvement in terms of mean stand-alone accuracy over three runs. The best architecture discovered on this search space, thanks to our regularizer, ranks 9313-th which corresponds to the top 2% of architectures. Note that NAO without landmark regularization consistently got trapped in local minima, leading to its best architecture being only in the top 30%.

NASBench-201. We report results on the NASBench-201 cell search space [12], where each cell is a fully connected graph with 4 nodes. Each edge contains 5 searchable operations. This yields a total of 15,625 architectures. We set $\tau = 5$ and benchmark all three baseline algorithms on this space. As shown in Table 1 (bottom), landmark regularization consistently improves the ranking correlation (S-KdT) across all three methods. We also observe an improvement in mean accuracy of more than 4% with SPOS. The best architecture is obtained by GDAS with landmark regularization and ranks 109-th. This corresponds to the top 0.7% architectures across the search space. In Figure 3 (top), we plot the mean S-KdT as super-net training progresses. We can see that the regularization improves the ranking correlation by a significant margin, especially towards the end of training. Note that we extend our experiments to CIFAR-100 and ImageNet16-120 of NASBench-201 in supplementary material.

DARTS search space. The NASBench search spaces are relatively small. To evaluate our approach in a more realistic scenario, we make use of the DARTS search space [22, 44, 45, 25] which spans 3.3×10^{13} architectures and is commonly used to evaluate real-world NAS performance. In contrast to NASBench, for which we could query the existing stand-alone performances, here, we need to train the discovered architecture from scratch. To compute the ranking correlation, we rely on the 5,000 pre-trained models of [28] from which we randomly sample 200 as before.

Model	S-KdT	Mean Acc.	Params	Best Acc.	Cost
SPOS	0.058 ± 0.010	92.80 ± 0.03	5.082M	92.88	12.53
SPOS+Ours	0.206 ± 0.018	93.41 ± 0.43	2.181M	93.84	15.96
GDAS	0.176 ± 0.014	90.48 ± 2.95	3.418M	93.43	9.27
GDAS+Ours	0.209 ± 0.001	94.32 ± 0.28	2.540M	94.60	13.24
NAO	0.102 ± 0.018	92.93 ± 0.87	5.080M	93.03	19.72
NAO+Ours	0.231 ± 0.012	93.53 ± 0.43	2.184M	93.78	28.18

Table 2: Results on the DARTS search space on CIFAR-10. Our best model (GDAS+Ours) surpasses the state-of-the-art model of [45] (94.02% accuracy with 3.62M parameters) with 30% fewer parameters.

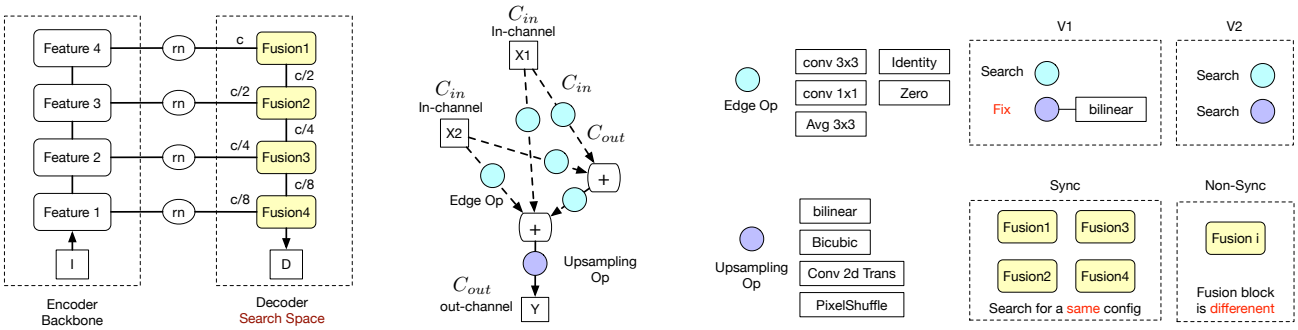
Model	S-KdT	Mean Top-1	Params	Best Top-1 (50/250)	Cost
SPOS	0.210 ± 0.010	64.57 ± 3.30	4.579M	67.88 / 73.69	7.29
SPOS+Ours	0.267 ± 0.018	67.38 ± 0.92	4.766M	68.61 / 74.58	9.39
GDAS	0.247 ± 0.012	67.50 ± 0.26	5.076M	67.26 / 74.03	9.13
GDAS+Ours	0.272 ± 0.023	68.82 ± 0.24	5.073M	68.36 / 74.82	10.39
NAO	0.253 ± 0.006	67.70 ± 0.43	4.675M	68.21 / 73.71	8.39
NAO+Ours	0.279 ± 0.003	68.89 ± 0.58	4.488M	69.58 / 74.92	11.21

Table 3: Results on ImageNet. We report mean top-1 accuracy over 3 runs after 50 epochs and best top-1 accuracy after 50 and 250 epochs, respectively.

We report our results in Table 2. We observe a clear improvement in terms of both S-KdT and mean accuracy over three independent searches across all three algorithms. Interestingly, the best model obtained using our ranking loss can surpass the baseline models by almost 1% with only around 50% of the parameters. The best model from GDAS with our regularizer surpasses the state-of-the-art model on this space [45] by 0.58% with 30% fewer parameters.

5.2. Image classification on ImageNet

To further evidence the effectiveness of our method, we move to ImageNet classification. For evaluation, we pick the best model of three independent runs predicted by each NAS algorithm and train them from scratch on the entire ImageNet training set for 50 epochs. We follow the setup of [45] and use stochastic gradient descent with a linear learning rate scheduler, which linearly increases from 0.1 to 0.5 in the first five epochs, and decreases to 0 over the remaining 45 epochs. We use a weight decay of 3e-4 and a label smoothing coefficient of 0.1 for all models. For this dataset, we use the popular DARTS search space, which provides 120 architectures trained for 50 epochs [28]. We split these architectures into 90 to report the sparse Kendall-Tau evaluation metric, and 30 for landmark sampling. We



(a) Overview of MiDAS Net with searchable decoder (b) Decoder Fusion block as a search cell (c) Operation choices (d) Search space configuration

Figure 4: Monocular depth estimation search space. (a) We modify MiDaS [30] to construct the search space. We keep the backbone unchanged and search for fusion blocks in the decoder branches. (b) To define a fusion block, we model the input from the backbone and from the preceding fusion block as two nodes. We add two feature nodes, which sum up all previous inputs. The nodes are connected by edges, which represent the searchable operations. The final output node Y takes the output of the last feature node and applies a potentially searchable upsampling operation. (c) Each edge, except the output edge, represents an *Edge Op (blue)* that contains five operations to choose from, while an upsampling edge *(purple)* contains four. (d) We propose four sub-space configurations: 'V1' indicates that we only search edge operations and fix the upsampling operator to bilinear upsampling. 'V2' includes a search over the upsampling operators. 'Sync' indicates that all fusion blocks share the same configuration, while 'Non-sync' allows them to differ.

train the super-net with only 15% of the training dataset as in [45]. As the test data is not public, we report the top-1 validation accuracy as a metric for stand-alone training.

Results. Table 3 shows that landmark regularization consistently improves the three baselines. Overall, the best model is found by NAO with landmark regularization and achieves 74.92% top-1 accuracy, which outperforms the best baseline model by more than 1%. This further evidences the effectiveness of our regularization and its robustness across datasets.

5.3. Monocular depth estimation

To showcase the generality of our approach, we apply our landmark-regularized NAS to the task of monocular depth estimation. Monocular depth estimation aims to predict pixel-wise depth from a single RGB image. Different paradigms have emerged on how to train single-image depth predictors, ranging from fully supervised training [1, 43, 30], to self-supervised approaches [13]. We follow the supervised paradigm and use the loss function proposed by Ranftl *et al*. [30] to search for an architecture on the ReDWeb [43] dataset.

Search space. Figure 4 gives a detailed overview of the structure of our search space. Figure 4 (a) shows the structure of a traditional monocular depth estimation network. It is composed of a backbone network that acts as a feature extractor, typically pre-trained on ImageNet, followed by decoder fusion blocks that aggregate multi-scale information into a final prediction. Since using a pre-trained high-capacity network has been shown to be of high importance for final performance [1, 30], its architecture is fixed. We thus propose to search for the fusion blocks that define the decoder.

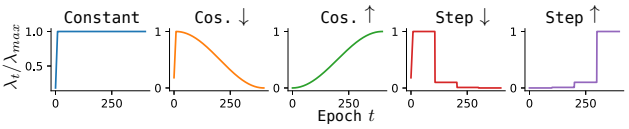
Search space	Method	Sync		Non-sync	
		s-KdT	Best Val. loss	S-KdT	Best Val. loss
V1		$\lvert\Omega\rvert = 3,125$		$\lvert\Omega\rvert = 9.5 \times 10^{13}$	
	SPOS	0.751 ± 0.003	0.0960 ± 0.001	0.732 ± 0.008	0.0973 ± 0.002
	SPOS+*Ours*	0.781 ± 0.002	0.0958 ± 0.001	0.867 ± 0.044	0.0974 ± 0.001
V2		$\lvert\Omega\rvert = 12,500$		$\lvert\Omega\rvert = 2.4 \times 10^{16}$	
	SPOS	0.401 ± 0.010	0.0957 ± 0.001	0.611 ± 0.004	0.0973 ± 0.001
	SPOS+*Ours*	0.555 ± 0.026	$\mathbf{0.0936 \pm 0.000}$	0.681 ± 0.002	0.0964 ± 0.001

Table 4: Results on the RedWeb validation set. The performance achieved by [30] is 0.0942 (lower is better).

As depicted in Figure 4 (b), each fusion block is a search cell and takes the output of its preceding fusion block and the features from the backbone network as input. As the first fusion block does not have a predecessor, we simply duplicate the features from the encoder. Figure 4 (c) shows the possible searchable operations, whereas (d) illustrates four possible configurations of our search space. We report the total number of architectures $|\Omega|$ for each search space in Table 4. While the search space is relatively simple, it is large enough to exhibit the problem of ranking disorder.

Results. We use the single-path one-shot algorithm to benchmark the influence of our regularization term on this task. We run $T = 2$ iterations, and sample 10 architectures in the first iteration. After the first round of training, we pick the three top models, obtain their stand-alone performance, and add them to the set of landmarks before we perform training for a second iteration. In addition to S-KdT, we report the scale- and shift-invariant loss [30] on the validation set as the performance metric.

Our results in Table 4 indicate that our method consistently yields an improvement in terms of S-KdT for all four configurations of the search space. Our best model



Schedule						λ_{max}		
	Const	Cos ↓	Cos ↑	Step ↓	Step ↑	1	10	100
S-KdT	0.753	0.712	0.803	0.709	0.785	0.789	0.803	0.793
Mean Acc.	91.13	91.38	92.08	91.02	91.31	91.89	92.08	91.97

Table 5: Influence of the regularization parameter λ. Left: Different schedules (*c.f.* top plots) to modify the strength of regularization throughout training. Right: Influence of λ_{max} with the increasing cosine schedule.

	SPOS			SPOS+*Ours*		
Iteration	T=1	T=3	T=10	T=1	T=3	T=10
S-KdT	0.763	0.771	0.758	0.760	0.802	0.811
Mean Acc.	91.13	91.48	91.78	91.29	92.08	92.17

Table 6: Influence of the number of iterations T on NASBench-201.

m Pair(s)	0	1	2	10	20	50	100
S-KdT	0.771	0.803	0.801	0.805	0.812	0.807	0.791
Mean Acc.	87.66	92.08	92.12	92.13	92.10	92.11	92.00

Table 7: Influence of the stochastic approximation of Eq. 4. We randomly sample m pairs from 30 landmarks during each training step .

improves upon the state-of-the-art handcrafted architecture of [30] in terms of the final performance.

5.4. Ablation studies

We finally provide an analysis of different aspects of our approach. We first evaluate the influence of the hyper-parameter λ and different regularization schedules. We further study the robustness of our method to the landmark sampling distance τ and the number of iterations T. We conduct the ablations under the experimental setting described in Section 5.1 and use SPOS as baseline.

Loss coefficient λ and scheduler. In Table 5, we ablate five different coefficient schedulers: constant regularization throughout training, two schedulers that gradually decrease the regularization, and two schedulers that gradually increase regularization (*c.f.* Table 5 (top)). For the constant and decreasing schedulers, we gradually increase the loss from 0 to λ_{max} linearly in the first 10 epochs to avoid an abrupt change of the loss. As shown in Table 5, the cosine increasing scheduler, with $\lambda_{max} = 10$, yields the best results. We used this strategy in all our experiments.

Iterations. We investigate the impact of the number of iterations T in Table 6. We first pre-train the super-net for 250 epochs and then continue training for another 150 epochs per iteration T both with and without landmark regularization. Table 6 indicates that our method improves the results when increasing the number of iterations, while the performance of the baseline does not increase. We selected $T = 3$ for our experiments as it strikes a balance between efficiency and accuracy.

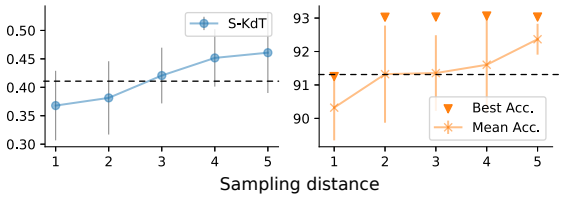


Figure 5: Comparison of different sampling distances τ. The black, dashed line indicates the baseline performance.

Sampling in loss computation. In Table 7, we show that sampling a single pair of architectures per iteration to compute the regularization term is sufficient. Using more pairs does not improve the ranking correlation. We hypothesize this to be due to the fact that, overall, super-net training undergoes thousands of steps, thus providing a good coverage of all possible combinations of landmark architectures when the landmark set is small.

Influence of the sampling distance. We finally evaluate the importance of the distance threshold τ in our landmark sampler. We first pre-train the super-net with SPOS on NASBench-201 for 150 epochs, then train for 50 epochs with landmark regularization, where we sample landmarks with varying distances τ. We repeated this experiment 3 times and report the average S-KdT as well as the mean accuracy of the discovered architectures. Figure 5 shows that the performance degrades if τ is chosen too small. Performance gradually improves as τ increases. This highlights the importance of a diverse set of landmarks. Note that our sampling strategy does not require knowledge about the stand-alone performance and thus is applicable to new NAS search spaces.

6. Conclusion

We have presented a simple yet effective approach to leverage a few landmark architectures to guide the super-net training of weight-sharing NAS algorithms towards a better correlation with stand-alone performance. Our strategy is applicable to most NAS algorithms and our experiments have shown that it consistently improves both the ranking correlation between the super-net and stand-alone performance as well as the final performance across three different search algorithms and three different tasks. Additionally, our approach can leverage the information from previously trained stand-alone models to improve NAS performance. In the future, we will focus on developing a more advanced landmark sampling strategy.

References

[1] Ibraheem Alhashim and Peter Wonka. High quality monocular depth estimation via transfer learning. *arXiv e-prints*, abs/1812.11941, 2018. 7

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *ICML*, pages 549–558, 2018. 2, 4

[3] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V. Le. Can weight sharing outperform random architecture search? an investigation with tunas. In *CVPR*, 2020. 1, 2

[4] Yassine Benyahia, Kaicheng Yu, Kamil Bennani-Smires, Martin Jaggi, Anthony C. Davison, Mathieu Salzmann, and Claudiu Musat. Overcoming multi-model forgetting. *ICML*, 2019. 2, 3

[5] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *ICLR*, 2020. 1, 2

[6] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *ICLR*, 2019. 1, 2, 4, 5

[7] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Chunhong Pan, and Jian Sun. DetNAS: Neural Architecture Search on Object Detection. *NeurIPS*, 2019. 1, 2

[8] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. FairNAS: Rethinking Evaluation Fairness of Weight Sharing Neural Architecture Search. *arXiv:*, 2019. 1, 2, 4

[9] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: Eliminating unfair advantages in differentiable architecture search. *ECCV*, 2020. 1

[10] Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. Nats-bench: Benchmarking nas algorithms for architecture topology and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1, 2021. 2

[11] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *CVPR*, 2019. 2, 4, 5

[12] Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *ICLR*, 2020. 2, 5, 6

[13] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, 2019. 7

[14] Shuxuan Guo, Jose M. Alvarez, and Mathieu Salzmann. Expandnets: Linear over-parameterization to train compact convolutional networks. In *NeurIPS*, volume 33, 2020. 2

[15] 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. *ECCV*, 2019. 1, 2, 4, 5

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, June 2016. 2

[17] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 (canadian institute for advanced research). 2009. 5

[18] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *UAI*, 2019. 1, 2, 4

[19] Xiang Li, Chen Lin, Chuming Li, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Improving one-shot NAS by suppressing the posterior fading. *CVPR*, 2020. 1, 2, 4, 5

[20] Yanxi Li, Zhaohui Yang, Yunhe Wang, and Chang Xu. Adapting neural architectures between domains. *CVPR*, 2020. 2

[21] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan Yuille, and Li Fei-Fei. Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. *CVPR*, 2019. 1, 2

[22] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. *ICLR*, 2019. 1, 2, 4, 5, 6

[23] Renqian Luo, Xu Tan, Rui Wang, Tao Qin, Enhong Chen, and Tie-Yan Liu. Semi-Supervised Neural Architecture Search. *NeurIPS*, 2020. 1, 2

[24] Renqian Luo, Fei Tian, Tao Qin, En-Hong Chen, and Tie-Yan Liu. Neural architecture optimization. In *NeurIPS*, 2018. 2, 4, 5

[25] Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik. Xnas: Neural architecture search with expert advice. In *NeurIPS*, 2019. 2, 4, 6

[26] Houwen Peng, Hao Du, Hongyuan Yu, Qi Li, Jing Liao, and Jianlong Fu. Cream of the crop: Distilling prioritized paths for one-shot neural architecture search. *CVPR*, 2020. 2

[27] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *ICML*, 2018. 1, 2, 4, 5

[28] Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On Network Design Spaces for Visual Recognition. In *ICCV*, 2019. 2, 5, 6

[29] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. *CVPR*, 2020. 5

[30] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 7, 8

[31] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *AAAI*, 2019. 2

[32] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. *ICML*, 2017. 2

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 5

[34] Michael S. Ryoo, AJ Piergiovanni, Mingxing Tan, and Anelia Angelova. Assemblenet: Searching for multi-stream neural connectivity in video architectures. In *ICLR*, 2020. 1, 2

[35] 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. 2

[36] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. *CVPR*, 2018. 2, 3

[37] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *ICML*, 2019. 2, 3

[38] Yehui Tang, Yunhe Wang, Yixing Xu, Hanting Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. A Semi-Supervised Assessor of Neural Architectures. In *CVPR*, 2020. 2

[39] Linnan Wang, Saining Xie, Teng Li, Rodrigo Fonseca, and Yuandong Tian. Neural architecture search by learning action space for monte carlo tree search. *AAAI*, 2020. 1, 2, 3

[40] Linnan Wang, Yiyang Zhao, Yuu Jinnai, Yuandong Tian, and Rodrigo Fonseca. AlphaX: eXploring Neural Architectures with Deep Neural Networks and Monte Carlo Tree Search. *AAAI*, 2020. 2, 4

[41] Wei Wang, Kaicheng Yu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Recurrent u-net for resource-constrained segmentation. In *ICCV*, October 2019. 2

[42] 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. *CVPR*, 2019. 1, 2, 4, 5

[43] Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, Yang Xiao, Ruibo Li, and Zhenbo Luo. Monocular relative depth perception with web stereo data supervision. In *CVPR*, 2018. 7

[44] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic neural architecture search. *ICLR*, 2019. 2, 4, 5, 6

[45] 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 *ICLR*, 2020. 2, 4, 5, 6, 7

[46] Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. NAS evaluation is frustratingly hard. In *ICLR*, 2020. 2

[47] Chris Ying, Aaron Klein, Esteban Real, Eric Christiansen, Kevin Murphy, and Frank Hutter. NAS-Bench-101: Towards reproducible neural architecture search. *ICLR*, 2019. 1, 2, 5, 6

[48] Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. GreedyNAS: Towards fast one-shot NAS with greedy supernet. *CVPR*, 2020. 2, 5

[49] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. BigNAS: Scaling Up Neural Architecture Search with Big Single-Stage Models. *ECCV*, 2020. 1, 2

[50] Kaicheng Yu, Rene Ranftl, and Mathieu Salzmann. How to Train Your Super-Net: An Analysis of Training Heuristics in Weight-Sharing NAS. *arXiv*, 2020. 1, 2, 5

[51] Kaicheng Yu and Mathieu Salzmann. Statistically-motivated second-order pooling. In *ECCV*, September 2018. 2

[52] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *ICLR*, 2020. 1, 2, 4, 5

[53] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020. 1

[54] Arber Zela, Julien Siems, and Frank Hutter. NAS-Bench-1Shot1: Benchmarking and dissecting one-shot neural architecture search. In *ICLR*, 2020. 1, 2, 6

[55] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. Few-shot neural architecture search. *arXiv*, 2020. 2, 4

[56] Barret Zoph and Quoc V. Le. Neural Architecture Search with Reinforcement Learning. *ICLR*, 2017. 2, 3

[57] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 2, 3, 5