Debiased One-shot NAS via Density-aware Sampling

Supplementary Material

1. Search Spaces

1.1. Pooling Benchmark

This search space introduced by Roshtkhari et al. [11] is built using Resnet [5] blocks. The search involves changing the position of the pooling layers within this architecture. It considers architectures with 2 pooling operations with 9 available locations within Resnet20 architecture. Other architecture components, such as channels, types of convolutions *etc*. are fixed.

1.2. NAS-Bench-Macro

This benchmark presented by Su et al. [15] is based on MobileNetV2 [13] blocks. It offers 3 operational choices $\{ID, MB3\_K3, MB6\_K5\}$ for 8 layers, resulting in 6,561 architectures.

1.3. NAS-Bench-201

NAS-Bench-201 [3] is a cell-based search space. Each cell contains 4 nodes with 5 possible operations per node, resulting in $15,626$ cells/architectures in total. The macro skeleton of the network is fixed, and the cell structure is repeated to form the architecture. The structure has a 3×3 convolution with 16 channels as the stem. The architecture has three stacks of cells, each containing 5 cells with 16, 32, and 64 channels. Between each stack, a basic residual block [5] of stride 2 is used to halve the feature map size.

1.4. MobileNet Search Space

The search space used for the ImageNet experiments is a macro-structure based on the MobileNetV2 architecture (Tab. 1). This design choice was made for fair comparison with other one-shot NAS methods. The architecture consists of a supernet with 21 searchable blocks, where each block is a MobileNetV2 inverted bottleneck. For each of these 21 blocks, one operation must be selected from a set of 13 candidates.

The combination of these choices results in a total search space size of 13^{21} possible architectures. The full list of operation candidates is detailed in Table 2.

2. Implementation Details

Training details and hyperparameters For pooling search space [11], we used the publicly available code. We used a batch size of 512 and trained for 300 epochs. We used 50 epochs for warm-up for uniform sampling. For the rest of the hyperparameters we use the same setting as the original paper: we used a learning rate of 0.1 with cosine annealing and weight decay 1e-3.

Table 1. The macro-structure of MobileNetV2 search space. Two stem layers are followed by stacks of choice operations (Op.) from Tab. 2. "No." determines the number of stacked layers, and "Input" determines the input feature map size.

No.	Op.	Input	Channels	Stride
1	3×3 conv	224	32	2
1	MB1_K3	112	16	1
4	choice	112	32	2
4	choice	56	40	2
4	choice	28	80	2
4	choice	14	96	1
4	choice	14	192	2
1	choice	7	320	1
1	1×1 conv	7	1280	1
1	avg. pool	7	-	-
1	fc	-	-	-

Table 2. The operation choices for the MobileNetV2 search space. ID indicates an identity mapping. Operations have the option for Squeeze-Excitation (SE) modules.

Block Type	Exp. Ratio	Kernel Size	SE
MB1_K3	1	3	-
ID	-	-	-
MB3_K3	3	3	yes
MB3_K5	3	5	yes
MB3_K7	3	7	yes
MB6_K3	6	3	yes
MB6_K5	6	5	yes
MB6_K7	6	7	yes

For NAS-Bench-Macro, we implemented one-shot NAS based on the benchmark from the original paper [15]. We followed similar hyperparameter settings for our training: training for 20/50 epochs for warm-up/training with batch size 512 and SGD with an initial learning rate of 0.1.

For ImageNet experiments, we performed our experiments on an and A100 GPU. For EA with shift [17], PA&DA [9], and FairNas [2] methods, we used the available implementations of the papers. We used FFCV [8] and mixed-precision in our runs to accelerate the experiments. We use a batch size of 512, the model is trained using an SGD optimizer with 0.9 momentum. A cosine annealing strategy is used with an initial learning rate of 0.1, which decays over

120 epochs.

Boltzmann Softmax Exploration (BSE) Boltzmann sampling is a simple exploration strategy. The probability of sampling architecture a is defined as:

$$P(a) = Softmax(R(a)/T), \tag{1}$$

where $R(a)$ is the reward and T is the temperature that controls exploration and exploitation trade-off.

3. More Details on Architectures Similarity/Distances

More Related Work Distance (or similarity) of architectures has been used for NAS, most notably for EA NAS methods. They are mainly used to maintain diversity (exploration), and/or guide the search towards promising candidates (exploitation) or a combination of both. These objectives are complementary in how they promote sampling, as increasing diversity requires maximizing a distance measure, while guiding the search requires sampling similar to a desirable architecture.

Distances have been used to maintain population diversity in EA [14, 18] in early stages and avoid premature convergence to suboptimal solutions. Furthermore, similarity measures are often used to guide the search towards local regions with promising architectures [1].

To calculate similarity, the architectures are often represented as strings with distances calculated based on edit distance. This is the computationally cheapest method; however, it generally ignores the relationship among operations/layers [1], giving the same importance to vastly different operational changes (particularly when skip connections are present), and does not use any knowledge gained from training.

Functional and representational similarity are two complementary measures to quantify neural network similarity [6]. Functional (semantic) similarity is based on output vectors and has been used for clustering architectures [12], and for selecting parent individuals in EA [16] in NAS.

Representation (activation) based similarity compares the pattern of activations in hidden layer feature maps for a given input. Measures such as Centered Kernel Alignment (CKA) [7] have been used to guide training of individual architectures towards a well-performing reference architecture by adding the term to the objective loss function [19].

Representational distances Centered Kernel Alignment (CKA) [7] is a technique used to measure the similarity between the learned representations of two neural network layers. Considering centered samples X_1 and X_2, the normalized Hilbert-Schmidt Independence Criterion (HSIC) [4, 20] is defined as:

Table 3. Comparison of correlation coefficients for Pooling and Macro-bench-NAS methods under different sampling strategies.

Sampling Strategy	Kendall's τ	Spearman
Pooling Benchmark		
Uniform	0.16	0.21
Debiased	0.34	0.55
NAS-Bench-Macro		
Uniform	0.72	0.87
Debiased	0.74	0.91

Table 4. Comparison of density calculation method for KDE and inverse of mean distance used in the main text.

Density Calculation	Kendall's τ
Inverse mean distance (eq. 5 + eq. 6)	0.34
KDE (Eq. (3))	0.32

$$CKA(X_1, X_2) = \frac{||X_2^T X_1||_F^2}{||X_1^T X_1||_F ||X_2^T X_2||_F}, \tag{2}$$

where $||\cdot||_F$ is the Frobenius norm. We use this similarity to estimate representational distances.

Encoding distances A very common encoding used is vector or string encoding. For an architecture a with L layers is represented as vector $v_a = [a^{(1)}, a^{(2)}, ..., a^{(L)}]$. Each a^i denotes an operation (*e.g.* convolution, pooling). To calculate distances, we used $d(i,j) = \sum_{k=1}^{L} \mathbb{I}\{v_i^{(k)} \neq v_j^{(k)}\}$, with function $\mathbb{I}$ counting the number of different items in two architectures.

4. Additional Results

In this section, we provide additional experimental results, ablations, and analysis.

Rank Correlation For our benchmarks, we calculate the rank correlation with independent training in table 3. Debiased supernet training shows improvement in rank correlation compared to uniform sampling on both benchmarks. In figure 1, we show that the debiased method improves correlation, particularly among high-performing architectures.

Kernel Density Estimation In eq. 5 and 6, we used average cosine distances to estimate density. Kernel Density Estimation (KDE) [10] can also be used to estimate the density:

$$\alpha_a = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} K_h(f(a), f(i)) \tag{3}$$

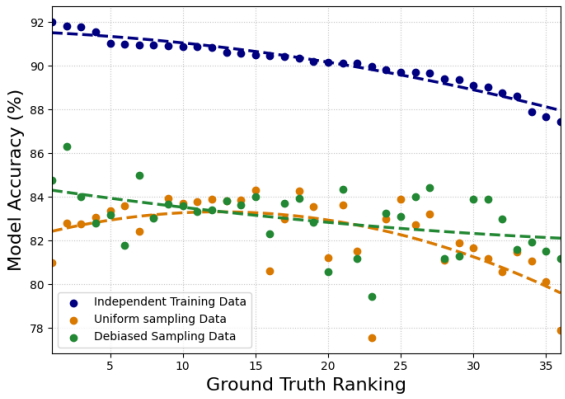

Figure 1. **Uniform and debiased sampling.** Debiased sampling corrects the negative correlation present in uniform sampling for high rank architectures (low Ground truth ranks). This results in better evaluations of these architectures.

Table 5. Rank correlation for debiased training with different temperatures. We compare Kendall tau on NAS-Bench-Macro for debiasing with different temperatures.

Temperature	**Kendall's τ**
50	0.72
7	0.74
0.5	0.67
0.1	0.65

with K_h, the Gaussian kernel function with a bandwidth h, a hyperparameter that determines the smoothness of the density estimation. Table 4 shows that both approaches provide similar performances. Therefore, KDE introduces an additional hyperparameter h with no clear benefit to performance.

Additional temperature ablations We show additional ablations on debiasing temperature τ in eq. 8. In table 5, we show rank correlation for different temperature values on NAS-Bench-Macro. Similar to the Pooling benchmark (figure 5 in the Experiments section), low temperature $\tau < 1$ seems to negatively affect the correlation.

Warm-up for density estimation We compare different epochs of warm-up with uniform sampling to estimate density in Tab. 6.

Gradient directions during training In figure 2, we show the average gradient cosine distances during supernet training. Deeper convolutional layers show greater distance due to more architectural variations in those layers, and at initial iterations of supernet training, early layers show more diverse gradient directions. Since gradients in those layers show greater indiscriminateness, we chose those layers to calculate gradient density.

Table 6. We compare different warm-up iterations to estimate density for the Pooling benchmark. We estimate correlation to density estimation at convergence.

Warm-up	**Spearman**
0	0.00
5	0.76
10	0.88
50	0.95
100	0.96
200	1.00
300	1.00

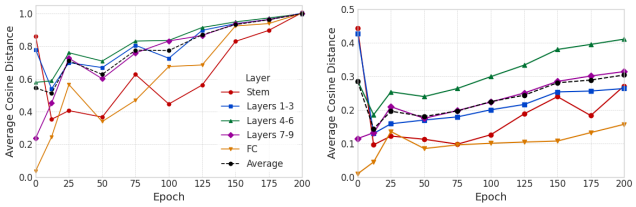

Figure 2. **Gradient cosine distances during training using uniform sampling.** Average gradient distances show that after an initial decrease, gradients become increasingly orthogonal. (left) Shows the gradient distances during training, while (right) is calculated using the same validation batch to isolate the effect of architectural differences.

References

[1] Yaofo Chen, Yong Guo, Daihai Liao, Fanbing Lv, Hengjie Song, James Tin-Yau Kwok, and Mingkui Tan. Automated dominative subspace mining for efficient neural architecture search. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(10):9281–9297, 2024. 2

[2] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on computer vision*, pages 12239–12248, 2021. 1

[3] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020. 1

[4] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005. 2

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[6] Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of neural network models: A

survey of functional and representational measures. *ACM Computing Surveys*, 57(9):1–52, 2025. 2

[7] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMlR, 2019. 2

[8] Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Madry. Ffcv: Accelerating training by removing data bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12011–12020, 2023. 1

[9] Shun Lu, Yu Hu, Longxing Yang, Zihao Sun, Jilin Mei, Jianchao Tan, and Chengru Song. Pa&da: Jointly sampling path and data for consistent nas. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11940–11949, 2023. 1

[10] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3): 1065–1076, 1962. 2

[11] Mehraveh Javan Roshtkhari, Matthew Toews, and Marco Pedersoli. Balanced mixture of supernets for learning the cnn pooling architecture. In *International Conference on Automated Machine Learning*, pages 8–1. PMLR, 2023. 1

[12] Mehraveh Javan Roshtkhari, Matthew Toews, and Marco Pedersoli. Neural architecture search by learning a hierarchical search space. *arXiv preprint arXiv:2503.21061*, 2025. 2

[13] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1

[14] Nilotpal Sinha and Kuan-Wen Chen. Novelty driven evolutionary neural architecture search. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 671–674, 2022. 2

[15] Xiu Su, Tao Huang, Yanxi Li, Shan You, Fei Wang, Chen Qian, Changshui Zhang, and Chang Xu. Prioritized architecture sampling with monto-carlo tree search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10968–10977, 2021. 1

[16] Yu Xue, Jiajie Zha, Mohamed Wahib, Tinghui Ouyang, and Xiao Wang. Neural architecture search via similarity adaptive guidance. *Applied Soft Computing*, 162:111821, 2024. 2

[17] Beichen Zhang, Xiaoxing Wang, Xiaohan Qin, and Junchi Yan. Boosting order-preserving and transferability for neural architecture search: a joint architecture refined search and fine-tuning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5662–5671, 2024. 1

[18] Miao Zhang, Huiqi Li, Shirui Pan, Taoping Liu, and Steven Su. One-shot neural architecture search via novelty driven sampling. In *29th International Joint Conference on Artificial Intelligence*. International Joint Conference on Artificial Intelligence (IJCAI), 2020. 2

[19] Xiawu Zheng, Xiang Fei, Lei Zhang, Chenglin Wu, Fei Chao, Jianzhuang Liu, Wei Zeng, Yonghong Tian, and Rongrong Ji. Neural architecture search with representation mutual information. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11912–11921, 2022. 2

[20] Xiawu Zheng, Yuexiao Ma, Teng Xi, Gang Zhang, Errui Ding, Yuchao Li, Jie Chen, Yonghong Tian, and Rongrong Ji. An information theory-inspired strategy for automated network pruning. *International Journal of Computer Vision*, pages 1–28, 2025. 2