Approximate Neural Architecture Search via Operation Distribution Learning

- 作者：Xingchen Wan，Binxin Ru，Pedro M. Esperança et.al.
- 机构：Huawei Noah’s Ark Lab, London, UK；Machine Learning Research Group,牛津大学

- 会议：arxiv
- 地址：https://arxiv.org/abs/2111.04670

- 代码：暂无

论文主要内容

摘要

标准的NAS范式是去搜索一个最优的确定性的架构（连接方式+op）。本文提出搜索最优的operation **distribution**（带有随机性，能够根据分布采样得到任意长度的架构）。本文提出：给定一个架构cell，它的performance很大程度依赖于使用的operation的比例而不是任何一个具体的连接模式（在典型的search space上）。通过在4个数据集、4种NAS技术上实验验证：1.op分布足以用来鉴别好的solution；2.op分布比传统encoding更易优化。能够在no-cost的情况下提高速度。

研究内容

Motivation

对比每一个架构是intractable 和 unnecessary

- intracable：
  - 搜索空间的绝对大小过大
- Unnecessary:
  - 差别小的架构最后的结果也差别小
  - 实验：抽取随机200个ANASOD-encoding，在每个encoding上再采样5个架构，一共1000个架构

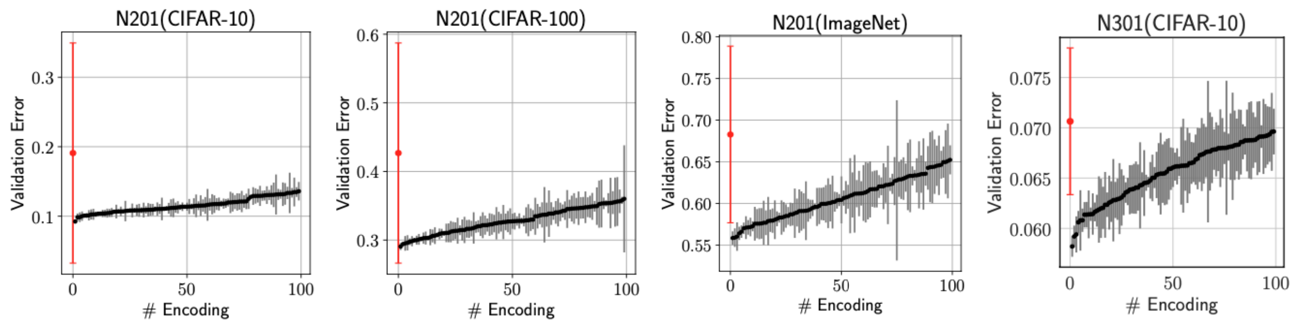

Figure 2: To what extent does the ANASOD encoding determine performance? We randomly draw 200 ANASOD encodings in 4 tasks. Within each, we draw 5 architectures *for each encoding* and show the mean $\pm$ 1 standard deviation (black and gray, respectively) of the top-100 encodings vs those of *all 1,000 sampled architectures* (red). Architectures sampled from the same encoding usually perform similarly and encodings that on average perform better also have smaller variability.

使用ANASOD编码，同编码不同架构的标准差比较小（适合作为编码）

Table 1: Overall standard deviation (SD), median SD of *different* architectures sampled from the *same encoding* and the median SD of the *same* architecture trained with *different seeds*. All SD are w.r.t validation error in percentage.

Benchmark	NB201			NB301
Task	C10	C100	ImageNet16	C10
Overall	9.5	14	10	0.77
Median (same encoding)				
- All encodings	1.2	2.4	2.7	0.25
- Top 50%-performing	0.84	1.4	1.9	0.22
Median (different seeds)	0.19	0.35	0.36	0.17

同编码（可以有不同架构）和同架构在噪声大小上接近，且比全空间所有架构的变差小很多

所以不去找最优的那一个架构而是通过找最优的op分布来获得近似解

- 无视具体的cell类型，op分布就是一个有效、有信息的表示

方法

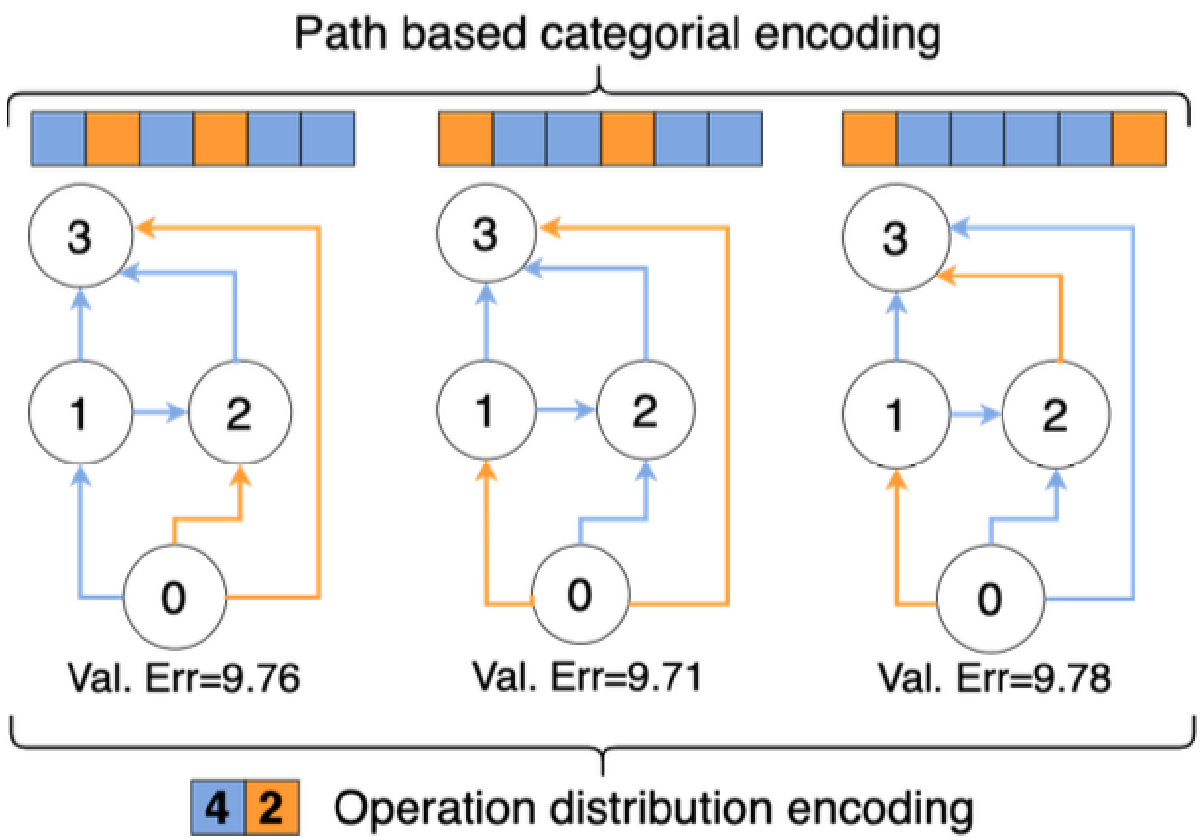

Figure 1: Three architecture cells from NAS-Bench-201 (NB201) [11] giving very similar validation performance. Each edge represents an operation: either conv1×1 or conv3×3. From the viewpoint of existing encodings, they represent three distinct architectures which all need to be evaluated. The ANASOD framework assumes that they all belong to the same operation distribution (4×conv3×3 and 2×conv1×1 in 6 operations) and does not repeatedly re-evaluate each one.

ANASOD（Approximate NAS via Operation Distribution - encoding）

- encoding：
  - $\{n_1, \ldots, n_N\}$ 表示一个Cell里面有N个op
  - $\{o_1, \ldots, o_k\}$ 表示这个cell里共有k个op可以选择，对于bench201，N=6，k=5
  - ANASOD编码 $\tilde{\mathbf{p}}$ 是一个k-dim向量（定义在一个k维的概率单纯形上）
    - $\left\{\tilde{\mathbf{p}} \in \mathbb{R}^k, \sum_{i=1}^{k} \tilde{p}_i = 1, \tilde{p}_i \geq 0 \forall i = \{1, \ldots, k\}\right\}$

        - $\tilde{p}_i = \frac{n(o_i)}{N}$ 。 $n(o_i)$ 是cell内操作 o_i 出现的次数
        - unnormalised encoding： $\mathbf{p} = N\tilde{\mathbf{p}}$
- decoding： $p(\alpha \mid \tilde{\mathbf{p}})$
    - 两种方法（当cell的大小N很大的时候两者等价）
        i. 将 $\mathbf{p}$ 的第k个分量直接作为cell内操作 o_k 的出现次数，随机shuffle顺序和连接方式可以获得架构 $\{\alpha_1, \alpha_2, \ldots\}$
            - 由于这样会使得 $\tilde{\mathbf{p}}$ 定义在一个k-单纯形的regluar grid中，使用标准的连续优化方法会有问题。所以利用round规则，把空间中任意一个点 $\mathbf{m}$ snap到合法的单纯形 Δ^k 上：
            - $\mathbf{m}$ 的分数部分 $\mathbf{s}(\mathbf{m}) = [m_i - \lfloor m_i \rfloor]_{i \in [1,k]}$
            - 记所有分数部分之和 $g(\mathbf{m}) := \sum_{i=1}^{k} s(m_i) = N - \sum_{i=1}^{k} \lfloor m_i \rfloor \in [0, k-1]$

non-negative integer. We then round $g(\mathbf{m})$ largest elements of $s(m_i)$ to 1 and the rest to 0 to obtain a rounded integer vector $\mathbf{m}_r$:

$$\mathbf{m_r} := \left[\lfloor m_1 \rfloor + 1, \ldots, \lfloor m_{g(\mathbf{m})} \rfloor + 1, \lfloor m_{g(\mathbf{m})+1} \rfloor, \ldots, \lfloor m_k \rfloor\right]^\top \quad (1)$$

            - $\mathbf{m}_r$ 就是在单纯形 Δ_k 上离 $\mathbf{m}$ 最近的合法点
        ii. 使用概率，每次op的选择都有概率分布采样得到
            - $n_j \sim \operatorname{Cat}(\tilde{p}_1, \ldots, \tilde{p}_k) \, \forall j \in [1, N]$
            - 该方法不限制 $\tilde{\mathbf{p}}$ 在一个grid上，可以直接采用连续的优化手段

优势

1. 编码方式不需要learning过程、复杂的计算，但可以巨大的压缩search sapce（仍然足以区分架构的性能）
    a. 从原始空间的 k^N 个架构到 $\binom{N+k-1}{k-1}$ 数量的encoding
    b. 如darts从10^12到10^5
2. decoding过程只需要简单的生成样本，所以可以用two-stage手段，先找最优encoding，再找最优架构，从而近似优化整个架构空间
3. ANASOD定义在一个单纯形空间具备well-defined距离度量、具有适当的维度。能够在此基础上有效的建立model-based方法（如基于高斯过程的贝叶斯优化）

ANASOD编码的应用

应用在不同方法上

Random Search(RS)

实现

- 标准版RS(vanilla RS)：从均匀Dirichlet分布采样编码 $\tilde{\mathbf{p}}_i \sim \operatorname{Dir}(1, \ldots, 1)$
- 增加exploit的RS（*biased RS*）： $\tilde{\mathbf{p}}_{t+1} \sim \operatorname{Dir}\left(\alpha_1, \ldots, \alpha_k\right)$ where $\alpha_i = k\beta_t\tilde{p}_i^* + 1 \forall i$
  - β_t 是温度参数，从0逐渐增加到 β_T ，trade-off 探索开发
  - 逐渐偏向当前best encoding的局部周围
  - No addition computing cost

Differentiable NAS (DNAS)

实现

与标准的DNAS不同

- DNAS搜每条edge都有k-dim的向量，这里搜的是一个cell内**一个**k-dim向量
  - 正则化了架构的学习 也避免了 catastrophic collapse、过拟合
- DNAS搜索最后会argmax cat-分布（离散后的架构与连续松弛架构的rank disorder），这里最后会依据ANASOD编码采样架构cell

Algorithm 1 ANASOD-DNAS. Key differences from existing DNAS algorithms marked blue.

1: Create a mixed operation $\bar{o}^j \sim \operatorname{Cat}(\tilde{\mathbf{p}})$ for each operation block. Note that the parameters of the categorical distribution are shared.
2: **while** not converged **do**
3: Update the encoding $\tilde{\mathbf{p}}$ by descending $\nabla_{\tilde{\mathbf{p}}}\left(\mathcal{L}_{\text{val}}(\tilde{\mathbf{p}}, w)\right)$, and keep w constant
4: Enforce the simplex constraint $\tilde{\mathbf{p}} \leftarrow \frac{\tilde{\mathbf{p}}}{\sum_i^k \tilde{p}_i}$ (i.e. mirror descent)
5: Update the supernet weights w by descending $\nabla_w \mathcal{L}_{\text{train}}(\tilde{\mathbf{p}}, w)$, and keep $\tilde{\mathbf{p}}$ constant.
6: **end while**
7: Sample cells from the optimised encoding $\alpha \sim p(\alpha|\tilde{\mathbf{p}}^*)$ and stack them into a final neural architecture.

Local Search(LS)

探索和开发的一个极端：purely exploitative

Sequential Model-based Optimisation (SMBO)

正常来说SMBO在NAS上受离散搜索空间、高维encoding、well-defined距离影响

ANASOD解决了这些问题

实现

Algorithm 2 ANASOD-BO. Key differences from conventional BO are marked blue.

1: **Input:** Objective function (default: validation error) y, number of initialising random samples n_{init}
2: Initialise the *encoding generating distribution* to the uniform Dirichlet distribution $p(\tilde{\mathbf{p}}) = \mathrm{Dir}(1, ..., 1)$
3: Sample n_{init} random encodings $\tilde{\mathbf{p}}_{[1:n_{\text{init}}]} \sim p(\tilde{\mathbf{p}})$ and evaluate to obtain $y(\tilde{\mathbf{p}})$ to initialise the surrogate GP.
4: **for** i=n_{init},,,.T **do**
5: Sample a pool of B candidate encodings from $p(\tilde{\mathbf{p}})$
6: Select the next query point(s) by identifying the encoding that maximises the acquisition function $\tilde{\mathbf{p}}_i = \arg\max\big(\mathrm{acq}(\tilde{\mathbf{p}})\big)$.
7: Evaluate a single architecture α_i from the encoding $\tilde{\mathbf{p}}_i$ to approximate the performance of all architectures parameterised by $\tilde{\mathbf{p}}_i$.
8: Augment the surrogate GP with new encoding-observation pair(s) $\mathcal{D}_i \leftarrow \mathcal{D}_{i-1} \cup \{\tilde{\mathbf{p}}_i, y(\tilde{\mathbf{p}}_i)\}$ and optimise the GP hyperparameters via the marginal log-likelihood maximisation.
9: Update the encoding generating distribution $p(\tilde{\mathbf{p}})$.
10: **end for**

- 第9行中不是像RS里那样用温度的退火过程，而是借用了TRuBO的思想，连续成功则减半 β ,连续失败则加倍 β

实验结果

Results

Baseline对比

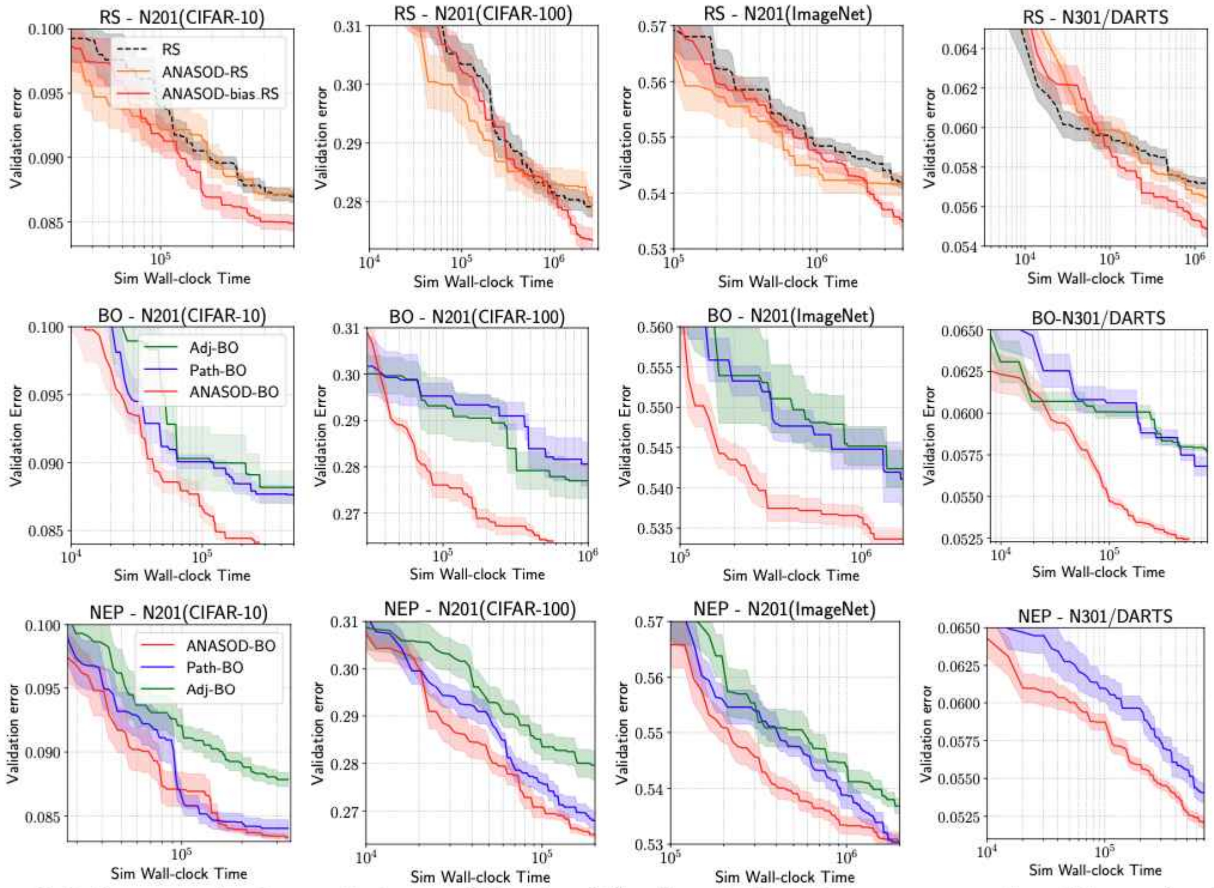

Figure 3: Performance of various methods on NB201 and NB301 with and without the ANASOD encoding: (**Top row**) random search, RS; (**Middle row**) GP-BO; (**Bottom row**) NEP-BO. Note the x-axis which shows the (simulated) GPU-seconds is in log-scale. For RS we set a budget of maximum 300 architecture queries; for BO, we set a more stringent budget of 150 queries for ANASOD-BO but allow the baselines to run for longer (to observe the amount of speedup). Adj-BO and Path-BO are the variants of BO that are otherwise identical to ANASOD-BO as outlined in Algorithm 2, but with the ANASOD encoding replaced by the adjacency [55, 50] and path [44, 42] encoding, respectively. Lines and shades denote mean ± 1 standard error, across 10 different random trials.

- 第1行
  - RS和ANASOD-RS表现一致，因为两者都是均匀从全空间中采样，说明ANASOD编码没有bias
  - 带非均匀的Dirichlet分布采样，使得RS有了exploit，搜索效果提升
- 第二行
  - 相比于其他编码方式，ANASOD使用BO搜索效果显著
- 第三行
  - path-BO使用了截断trick（原始论文提出的）

SOTA对比

Table 2: Performance on NAS-Bench datasets. Unless otherwise specified, we report mean ± 1 standard error of the validation error across 10 random trials. For fair comparison, in RS and LS experiments, the numbers shown denote the best validation error seen after **300** architecture queries; in BO experiments, we show the best validation error seen for each method after **150** architecture queries (which is the budget we set for ANASOD-BO).

Benchmark Dataset	 CIFAR-10	NB201 CIFAR-100	 ImageNet16	NB301 CIFAR-10
RS	$8.67_{\pm 0.03}$	$27.91_{\pm 0.17}$	$54.17_{\pm 0.10}$	$5.71_{\pm 0.02}$
ANASOD-RS	$8.71_{\pm 0.05}$	$27.95_{\pm 0.20}$	$53.85_{\pm 0.12}$	$5.65_{\pm 0.04}$
ANASOD-biasedRS	$\mathbf{8.48}_{\pm 0.06}$	$\mathbf{27.35}_{\pm 0.20}$	$\mathbf{53.30}_{\pm 0.10}$	$\mathbf{5.47}_{\pm 0.04}$
RL [55]	$8.91_{\pm 0.05}$	$28.15_{\pm 0.18}$	$54.45_{\pm 0.12}$	*
RE [33]	$8.86_{\pm 0.05}$	$28.40_{\pm 0.14}$	$54.28_{\pm 0.10}$	$5.62_{\pm 0.03}$
SMAC [18]	$8.89_{\pm 0.05}$	$27.80_{\pm 0.20}$	$53.64_{\pm 0.13}$	$5.45_{\pm 0.03}$
TPE [3]	$8.57_{\pm 0.04}$	$27.28_{\pm 0.14}$	$53.54_{\pm 0.14}$	$5.51_{\pm 0.02}$
GCNBO [37]	$8.84_{\pm 0.01}$	$27.93_{\pm 0.03}$	$53.46_{\pm 0.06}$	$5.54_{\pm 0.04}$
BANANAS [44]	$8.51_{\pm 0.08}$	$26.53_{\pm 0.02}$	$53.41_{\pm 0.04}$	$5.36_{\pm 0.05}$
NAS-BOWL [35]	$8.50_{\pm 0.09}$	$26.51_{\pm 0.00}$	$\mathbf{53.36}_{\pm 0.04}$	$5.31_{\pm 0.06}$
ANASOD-BO	$\mathbf{8.41}_{\pm 0.05}$	$\mathbf{26.41}_{\pm 0.02}$	$\mathbf{53.36}_{\pm 0.10}$	$\mathbf{5.24}_{\pm 0.02}$

*: The original repo does not support the NB301 search space.

对比one-shot，CIFAR-10上搜架构

Table 3: Comparison of one-shot NAS methods on NB201. To reflect real-world applications, we search on CIFAR-10 and transfer the search result to the other datasets.

Benchmark Dataset	Search epoch	NB201 CIFAR-10	CIFAR-100	ImageNet16
Optimal	-	5.70	26.50	52.70
RSPS* [25]	50	$12.34_{\pm 1.7}$	$41.67_{\pm 4.3}$	$68.86_{\pm 3.9}$
DARTS† [28]	50	$45.70_{\pm 0.0}$	$84.39_{\pm 0.0}$	$83.68_{\pm 0.0}$
SETN* [10]	50	$12.36_{\pm 0.0}$	$41.95_{\pm 0.2}$	$67.48_{\pm 0.2}$
ENAS† [32]	50	$45.70_{\pm 0.0}$	$84.97_{\pm 0.0}$	$83.68_{\pm 0.0}$
GAEA-DARTS* [24]	25	$8.36_{\pm 2.6}$	$31.61_{\pm 4.5}$	$58.41_{\pm 4.2}$
ANASOD-DNAS	20	$7.75_{\pm 1.2}$	$31.33_{\pm 1.7}$	$58.00_{\pm 2.9}$

*: Results taken from [24]; †: Results taken from [11].

mator (TPE [3]), graph convolutional network based BO (GCNBO [37]), NEP-BO with path encodings (BANANAS [44]) and Weisfeiler–Lehman kernel-based BO (NAS-BOWL [35]). Similarly, we compare ANASOD-DNAS with a number of existing DNAS algorithms in Table 3. All additional details about the experimental setup can be found in App. A.

NASNET-style search space

Table 4: Performance on CIFAR-10 in the NASNET-styl
search space. ANASOD-BO experiment is conducted on 4:
NVIDIA Tesla V100 GPUs using 0.6 wall-clock days.

Algorithm	Val. Err	#Params(M)	Method
Random-WS [46]	$2.85_{\pm 0.08}$	4.3	RS
NASNet-A [56]	2.65	3.3	RL
LaNAS [41]	$2.53_{\pm 0.05}$	3.2	MCTS
DARTS [28]	$2.76_{\pm 0.09}$	3.3	GD
DARTS+† [27]	$2.37_{\pm 0.13}$	4.3	GD
P-DARTS [8]	2.50	3.4	GD
DropNAS [15]	$2.58_{\pm 0.14}$	4.1	GD
DropNAS† [15]	1.88	4.1	GD
BANANAS [44]	2.64	-	BO
BOGCN [37]	2.61	3.5	BO
NAS-BOWL [35]	$2.61_{\pm 0.08}$	3.7	BO
ANASOD-BO (Mixup)	2.63	3.5	BO
ANASOD-BO (CutMix)	2.41	3.5	BO
ANASOD-BO+	1.86	3.5	BO

†: Training protocol comparable to ANASOD-BO+
MCTS: Monte Carlo tree search; GD: Gradient descent.

Table 5: Performance on CIFAR-100 in the NASNET-style search space. The ANASOD-BO result is transferred from CIFAR-10.

Algorithm	Val. Err	#Params(M)	Method
DARTS [28]	17.76	3.3	GD
DARTS+† [27]	14.87	3.9	GD
P-DARTS* [8]	16.55	3.4	GD
DropNAS [15]	16.39	4.4	GD
DropNAS+† [15]	14.10	4.4	GD
ANASOD-BO* (CutMix)	16.33	3.5	BO
ANASOD-BO+*	13.76	3.5	BO

*: Transferred from the CIFAR-10 search.
†: Training protocol comparable to ANASOD-BO+.