# BPNAS: Bayesian Progressive Neural Architecture Search

**Hyunwoong Chang** [*1] **Anirban Samaddar** [*2] **Sandeep Madireddy** [2]

## Abstract

In the performance landscape of the multiple NAS benchmarks, only a few operations contribute to higher performance while others have detrimental effects. This motivates tailoring a posterior distribution by imposing a higher prior quantity on a sparser supernetwork to progressively prune unimportant operations. Moreover, the Bayesian scheme enables the straightforward generation of architecture samples when provided with an estimated architecture from any NAS method. To that end, we propose `BPNAS`, a Bayesian progressive neural architecture search (NAS) method under the differentiable NAS framework that combines recent advances in the differentiable NAS framework with Bayesian inference adopting sparse prior on network architecture for faster convergence and uncertainty quantification in architecture search. With numerical experiments on the popular NAS search space, we show that `BPNAS` improves the accuracy and convergence speed compared to state-of-the-art NAS approaches on benchmark datasets.

## 1. Introduction

Neural Architecture Search (NAS) has emerged as a critical field in modern deep learning, focusing on automating the design of neural network architectures. This automation addresses the challenges of manual design, which is laborious, resource-intensive, and requires significant expertise. Rapid advancements in NAS, evidenced by a surge in research publications [1]–[3], have introduced methods such as neuro-evolutionary algorithms, reinforcement learning-based algorithms, and differentiable architecture search approaches. These innovations have proven to be effective in diverse applications, such as computer vision [4] and natural language processing [5]. Traditional NAS methods based on neuro-evolutionary and reinforcement learning, characterized by their exhaustive and combinatorial search strategies, often face significant computational challenges. Differentiable NAS, however, marks a significant paradigm shift in this field. By leveraging gradient-based optimization, it substantially reduces computational load and enhances the efficiency and effectiveness of exploring the architectural space. Pioneered notably by [6], this framework converts the combinatorial optimization problem of NAS into a continuous one through weight sharing and continuous relaxation, a transformation that is crucial to applying gradient-based methods. This streamlined approach in the architectural search process not only accelerates the discovery but also facilitates the uncovering of more effective and efficient neural network architectures, showcasing the innovative potential of Differentiable NAS in the realm of machine learning.

In current NAS research, differentiable NAS methods show promise but struggle with instability and generalization [7], [8]. Addressing these issues, probabilistic modeling, including Bayesian approaches and deep ensembling, offers robustness and better uncertainty quantification [9]–[11]. However, their integration in differentiable NAS, as seen in a few recent works [12]–[14], remains limited to learning point estimates of architecture and weight parameters. Recently, Neural Ensemble Search (NES) has emerged, focusing on optimal ensembles of diverse networks rather than single architectures, as explored in NESBS [15] and UraeNAS [16]. This approach improves accuracy and robustness but faces challenges in selecting architectures that effectively curate a collection of architectures that function cohesively as an ensemble. Random selection risks forming less than optimal ensembles, undermining this approach's intended benefits.

The contributions of this paper are as follows:

- We propose a unified framework, **B**ayesian **P**rogressive **N**eural **A**rchitecture **S**earch (`BPNAS`) for architecture search (NAS) and ensemble (NES) under the differentiable search schemes. The key concept involves incorporating a sparse prior on the network structure, progressively guiding any supernetwork structure towards a single architecture.

- Our Bayesian framework naturally leads to operation pruning algorithms for NAS and NES, similar to the

[*]Equal contribution [1]Department of Statistics, Texas A&M University, College Station, USA [2]Argonne National Lab, IL, USA. Correspondence to: Sandeep Madireddy <smadireddy@anl.gov>.

*Published at the $2^{nd}$ Differentiable Almost Everything Workshop at the $41^{st}$ International Conference on Machine Learning*, Vienna, Austria. July 2024. Copyright 2024 by the author(s).

backward elimination procedure in linear regression. Our algorithms ensure efficient model selection from the high-dimensional search space of architectures.

- We compare our framework against state-of-the-art NAS and NES approaches on the popular NAS-Bench-201 search space [17]. We empirically show that `BPNAS` is more accurate with faster convergence than these approaches on CIFAR-10, CIFAR-100, and ImageNet-16-120 datasets.

## 2. Background

### 2.1. Landscape of the search space

Analyzing how performance varies among different architectural structures is crucial for developing a better algorithm. However, the analysis is exceedingly difficult due to the complexity, for example, given the potential existence of as many as $10^{18}$ candidates in the DARTS space [6]. As described in Section B in the appendix, NAS-Bench-201 [17] enables us to glimpse the performance landscape by evaluating the performance of each architecture on three different datasets and providing some diagnostic information. It utilizes a cell-based search space, where a cell is represented by a densely connected directed acyclic graph (DAG) $(V = [N], E)$ by assigning a direction from the $i$-th node to the $j$-th node, if $i < j$ for $i, j \in [N]$, where $[N] = \{1, \dots, N\}$ and every edge $(i, j) \in E$ is associated with one of the elements $o^{(i,j)}$ of a predefined set of operations $\mathcal{O}$. In NAS-Bench-201, the number of edges in $E$ is 6 and the operation set contains 5 operations.

### 2.2. Differentiable architecture search for architecture ensemble

As there are infinitely many possible architectures, it is necessary to restrict the search space, opting not to search for candidates from scratch. The prevalent approach involves utilizing a cell-based search space defined in Section 2.1. The whole architecture is then composed by stacking these cells sequentially multiple times. The cell-based design, as introduced by [18], markedly diminishes the complexity of search spaces and has become widely adopted [1].

Given the inherent difficulty in searching within a discrete space, differential NAS has become the most widely adopted method to address this challenge. DARTS [6] enables the efficient architecture search using a gradient-based optimizer by employing a supernetwork of operations within each edge. To be more specific, let $\Delta^k = \{z \in \mathbb{R}^k : \sum_{j=1}^k z_j = 1, z_j \geq 0 \text{ for } j \in [k]\}$ denote the $k$-simplex space. we can define the output of $j$-th node $x_j = \sum_{i<j} \hat{o}^{i,j}(x_i)$ where $\hat{o}^{i,j}(x) = \sum_{k \in [|\mathcal{O}|]} \theta_k o_k^{(i,j)}(x)$ for $\theta \in \Delta^{|\mathcal{O}|}$ where $|\cdot|$ denotes the cardinality of a set. This framework transforms

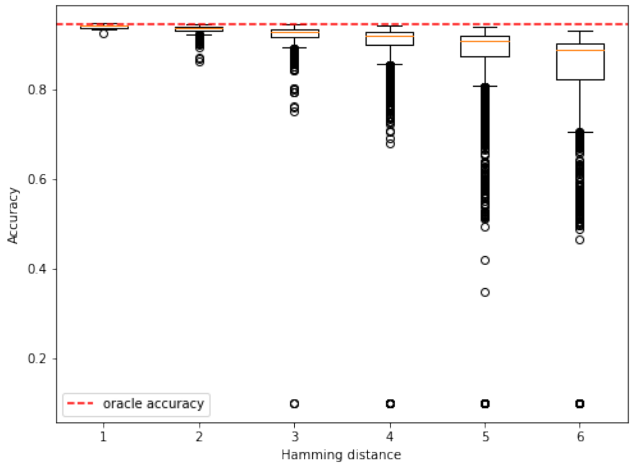


*Figure 1.* The figure shows box plots illustrating the distribution of the accuracy for different architecture groups based on the Hamming distance from the highest accuracy architecture (the best architecture) from NAS-Bench-201 dataset (CIFAR10). Each box plot depicts the accuracy of architectures with a $k$-Hamming distance from the best architecture, for $k = 1, \dots, 6$. It indicates that greater structural dissimilarity from the best architecture generally results in lower accuracy.

the problem of selecting categorical operations into learning continuous architecture mixing weights $\theta$. The parameters subject to optimization are two-fold: the network weight $W = (W^o_{i,j})$ where $W^o_{i,j}$ is the associated network weight parameter for an operation $o$ in an edge $e \in E$, and the latent architecture parameter $\alpha = (\alpha_e)$ where $\alpha_e$ is a parameter to define mixing weights $\theta_e = \theta_e(\alpha_e) \in \Delta^{|\mathcal{O}|}$. There are variations for the function form, such as $\theta_e = \alpha_e$ ([19]), $\theta_e = \text{Softmax}(\alpha_e)$ ([6]), or $\theta_e = \mathbb{E}_{\text{Dirichlet}(\alpha_e)}(\eta)$ ([12]). The bi-level objective $\mathcal{L}(\alpha, W)$ is to find the solution $(\alpha^*, W^*)$ satisfying

$$\min_{\alpha} \mathcal{L}_{\text{val}}(\alpha, W^*) \text{ s.t. } W^* = \arg\min_{W} \mathcal{L}_{\text{train}}(\alpha, W). \quad (1)$$

## 3. The proposed approach

We propose a unified Bayesian framework `BPNAS` for neural architecture search and its ensemble. The framework not only facilitates the discovery of a single maximum a posteriori (MAP) architecture, but also enables uncertainty quantification at the architecture structure level. Additionally, it provides an efficient neural ensemble search (NES) scheme to sample a set of architectures, naturally enhancing performance through model averaging.

Let $\mathcal{O}$ the set of operations with $|\mathcal{O}| = n_{\text{op}}$. For simplicity, we assume that the candidate architectures contain only one operation per edge. Let $|E| = n_{\text{edge}}$, then we have total $n_{\text{op}}^{n_{\text{edge}}}$ models in the search space. We define the structure of (super)network as $\Gamma = (\gamma_1, \dots, \gamma_{n_{\text{edge}}})$, where the $n_{\text{op}}$ vector $\gamma_m = (\gamma_{1m}, \ldots, \gamma_{n_{\text{op}}m})^{\text{T}}$ be the inclusion vector of operations for each $m \in [n_{\text{edge}}]$, i.e., for $\ell \in [n_{\text{op}}]$, $\gamma_{\ell m} = 1$ means the supernetwork contains $\ell$-th operation in $m$-th edge, otherwise, $\gamma_{\ell m} = 0$. Let $\mathcal{M} = \{\Gamma : \|\gamma_m\|_0 = 1 \text{ for all } m \in [n_{\text{edge}}]\}$ be the set of the candidate architectures where $\|\cdot\|_0$ denotes the $\text{L}_0$ norm. We define $d_{\text{HD}}(\Gamma, \Gamma') = \sum_{m=1}^{n_{\text{edge}}} \mathbb{1}(\gamma_m \neq \gamma'_m)$ for $\Gamma, \Gamma' \in \mathcal{M}$ as the Hamming distance, which indicates the number of different edges between $\Gamma$ and $\Gamma'$. We define $\mathbf{1}_{n,n'}$ (resp. $\mathbf{1}_n$) as the $n \times n'$ matrix (resp. $n$ vector) whose elements are all one, and let $\mathbb{1}$ be an indicator function.

We can rewrite the bi-level objective function (1) as

$$\mathcal{L}(\alpha, W) = \mathcal{L}(\alpha(\Gamma), W(\Gamma) | \Gamma = \mathbf{1}_{n_{\text{edge}}, n_{\text{op}}}),$$

since the supernetwork structure remains intact throughout the training. By using (1), given the above form of $\mathcal{L}$ , we define the conditional posterior distribution on $(\alpha, W)$

$$\pi(\alpha, W | \Gamma) \propto e^{-\mathcal{L}(\alpha(\Gamma), W(\Gamma)|\Gamma)}, \tag{2}$$

as a Gibbs distribution. Putting the prior on

$$\pi(\Gamma) \propto \prod_{m=1}^{n_{\text{edge}}} \pi(\gamma_m) = \prod_{m=1}^{n_{\text{edge}}} e^{-c\|\gamma_m\|_0} \mathbb{1}_{\{|\gamma_m|>0\}}(\gamma_m),$$

the posterior distribution is defined as

$$\pi(\alpha, W, \Gamma) \propto \pi(\alpha, W | \Gamma) \pi(\Gamma). \tag{3}$$

Note that the prior enforces the sparsity in the supernetwork to eventually find one of the candidate architectures $\Gamma \in \mathcal{M}$ as one of the modes with one operation per edge. This can be done by choosing $c$ sufficiently large, $\pi(\alpha, W, \Gamma) \leq \pi(\alpha', W', \Gamma')$ if $\|\Gamma\|_0 > \|\Gamma'\|_0$ , so that modes of the distribution can be formed at $\mathcal{M}$. Therefore, the objective is to find a single architecture $\hat{\Gamma}^{\text{MAP}} \in \mathcal{M}$ where

$$(\hat{\alpha}^{\text{MAP}}, \hat{W}^{\text{MAP}}, \hat{\Gamma}^{\text{MAP}}) = \arg\max_{\alpha, W, \Gamma} \pi(\alpha, W, \Gamma).$$

Finding a solution from $\pi(\alpha, W, \Gamma)$ is hard since the posterior distribution is a joint distribution of discrete and continuous variables, and the continuous parameters $\alpha, W$ depend on the discrete parameter $\Gamma$. It is necessary to devise efficient movement between different dimensional spaces. Note that this setup is commonly associated with reversible jump Markov chain Monte Carlo (MCMC) algorithms [20], which are commonly applied to Bayesian model selection problems, although we are only interested in finding a MAP estimate of $\Gamma$. Among many ways to construct these jumps and automate the process, we propose an efficient algorithm that prunes less important operations progressively after a sufficient amount of time for training the continuous variables $\alpha(\Gamma)$ and $W(\Gamma)$ until we obtain a single network in $\mathcal{M}$.

**Pruning criterion** For $m$-th edge with $\|\gamma_m\|_0 > 1$, we prune an operation if

$$\theta(\alpha_m) \notin \mathcal{B}_\delta(\mathbf{c}),$$

where $\mathcal{B}_\delta(\mathbf{c}) = \{\mathbf{r} \in \mathbb{R}^{\|\gamma_m\|_0} : \|\mathbf{r} - \mathbf{c}\|_2 \leq \delta\}$, $\|\cdot\|_2$ denotes the $\text{L}_2$ norm, and $\mathbf{c} = \mathbf{1}_{\|\gamma_m\|_0} / \|\gamma_m\|_0$ is the center of $\Delta^{\|\gamma_m\|_0}$. In our experiments, we have considered $\theta(\alpha_m)$ as a random sample drawn from $\text{Dirichlet}(\alpha_m)$. Intuitively, the criterion implies pruning if the current architecture mixing weights of the supernetwork deviate too far away (controlled by $\delta$) from a uniform distribution. This is analogous to the backward elimination procedure [22], where the algorithm starts with all operations on all edges and then decides which operation to keep on each edge for the architecture by sequentially excluding the others.

**Algorithm 1** Bayesian Progressive Architecture search (BPNAS)

```
Input: An initial supernetwork $\Gamma$ with architecture parameters $\alpha$ and weights $W$, posterior distribution $\pi(W, \alpha, \Gamma)$, and a threshold $\delta$
while $\|\gamma_m\|_0 > 1$ for some $m$ do
    Update $W, \alpha$ by stochastic gradient descent.
    if $\|\theta(\alpha_m) - \frac{\mathbf{1}_{\|\gamma_m\|_0}}{\|\gamma_m\|_0}\|_2 \geq \delta$ then
        Select $(\ell, m) = \arg\min_{\{(\ell', m'):\Gamma_{\ell' m'}=1\}} \alpha_{\ell', m'}$ and set $\Gamma_{\ell m} = 0$.
        Reset $\alpha$.
Output: An architecture $\Gamma$ with a single operation for each edge.
```

**Sampling architectures for ensemble** One prominent advantage of this framework is an efficient method to sample a set of architectures given one architecture $\hat{\Gamma} \in \mathcal{M}$ from any NAS algorithms. For a predefined number $q \in [n_{\text{edge}}]$, we let $S_q$ be a set of $q$ numbers randomly sampled from $[n_{\text{edge}}]$ and define the reconstructed supernetwork $\hat{\Gamma}^{\text{recon}}$ as

$$\hat{\gamma}_m^{\text{recon}} = \begin{cases} \mathbf{1}_{n_{\text{op}}}, & \text{if } m \in S_q, \\ \hat{\gamma}_m, & \text{otherwise,} \end{cases}$$

which reconstructs supernetworks partially for some edges. We run Algorithm 1 with $\hat{\Gamma}^{\text{recon}}$ as the initial supernetwork. The final output $\tilde{\Gamma}$ from the procedure is at most $q$-Hamming distance away from the given architecture $\hat{\Gamma}$. We empirically find that the performance of the architecture is similar to that of the given architecture, probably due to its structural similarity (Figure 1). We repeat this procedure $N$ times to curate an ensemble of well-performing architectures. Note that, since each repetition initializes only a partial supernetwork, this procedure is less memory intensive and can be efficiently parallelized.

| Methods | CIFAR-10 | | CIFAR-100 | | | ImageNet-16-120 | | |
|---|---|---|---|---|---|---|---|---|
| | test | epochs | validation | test | epochs | validation | test | epochs |
| ResNet [21] | 93.97 | 100 | 70.42 | 70.86 | 100 | 44.53 | 43.63 | 100 |
| DrNAS [12] | **94.36** $\pm$ 0.00 | 100 | 70.25 $\pm$ 1.53 | 71.00 $\pm$ 1.27 | 100 | **46.37** $\pm$ 0.00 | **46.34** $\pm$ 0.00 | 100 |
| BPNAS | 94.26 $\pm$ 0.22 | **70** | **73.38** $\pm$ 0.23 | **73.40** $\pm$ 0.19 | **50** | **46.37** $\pm$ 0.00 | **46.34** $\pm$ 0.00 | **26** |
| Optimal | 94.37 | - | 73.49 | 73.51 | - | 46.77 | 47.31 | - |

*Table 1.* Performance compared to the state-of-the-art methods on NAS-Bench-201.On the CIFAR-10 and ImageNet, BPNAS performs similarly to DrNAS; however, it outperforms DrNAS on the CIFAR-100. On all datasets, BPNAS significantly reduces the computational cost of the architecture search.

| Methods | Cifar-10 | CIFAR-100 | ImageNet-16-120 |
|---|---|---|---|
| NES-RS [9] | 94.17 $\pm$ 0.33 | 74.42 $\pm$ 0.84 | 45.66 $\pm$ 1.67 |
| NESBS [15] | 94.08 $\pm$ 0.07 | 75.00 $\pm$ 0.17 | 47.32 $\pm$ 0.35 |
| BPNAS | **95.10** $\pm$ 0.07 | **77.47** $\pm$ 1.04 | **50.27** $\pm$ 0.45 |

*Table 2.* Comparison of performances of the state-of-the-art NES methods on NAS-Bench-201. For all datasets, BPNAS outperforms all other methods.

## 4. Experiments

In this section, we describe the experiments conducted to study both the efficiency and effectiveness of BPNAS in the popular NAS search space: the NAS-Bench-201 [17]. In this search space, we compare the BPNAS architecture search algorithm (Algorithm 1) with the state-of-the-art NAS algorithms [12], and the BPNAS ensemble algorithm is compared with the state-of-the-art NES algorithms [15]. We compare these methods on CIFAR-10, CIFAR-100, and ImageNet-16-120 datasets.

### 4.1. Architecture search results

Table 1 presents the mean accuracies of the NAS methods across 10 seeds on the three datasets, along with the standard deviations. We also report the accuracies of a baseline ResNet network [21] and the highest accuracy attained among all architectures from the NAS-Bench-201 database. In all the datasets, We observe that BPNAS can find architectures that perform similarly to the optimal architecture in NAS-Bench-201, with a gap of less than 1%. As DrNAS [12] has been reported as the state-of-the-art method, we compare its performance with BPNAS. While BPNAS performs similarly to DrNAS on CIFAR-10 and ImageNet, it outperforms DrNAS on CIFAR-100 by more than 2% in average accuracy.

One of the key properties of BPNAS is the pruning criterion described in the previous section which prunes the less important operations from the edges of the cell as the training progresses. The amount of pruning applied is controlled through the parameter $\delta$ which we treat as a hyperparameter. The setting of $\delta$ is done by maximizing the accuracy and speed of convergence of Algorithm 1 on the validation set. More details are presented in Appendix C.

In Table 1, we report the average number of epochs it took for BPNAS to arrive at a final architecture on each data set across replications. For the other methods, the default number of epochs was set at 100 across all datasets. We observe that across all datasets, BPNAS significantly reduces computational time without any significant detrimental effect on classification performance, notably requiring 70% fewer epochs on ImageNet-16-120.

### 4.2. Ensemble results

In Table 2, we compare BPNAS with state-of-the-art NES methods on the three datasets. We present the mean test accuracies and standard deviations of the BPNAS algorithm with an ensemble size of 3, as well as those of the two most recent NES methods [15]. For each BPNAS ensemble, we set the number of perturbed edges $q = 1$ for the initial supernetwork.

BPNAS ensembling achieves the highest test accuracy across all datasets, significantly outperforming the other NES methods. By setting $q$ small, the algorithm quickly converges to a candidate architecture for the ensemble. In addition, the BPNAS ensembling approach builds upon an architecture generated by any NAS algorithm; in our case, we use Algorithm 1, making it efficient in finding a set of candidates for the ensemble.

## 5. Discussion

In this paper, we propose BPNAS, a Bayesian framework for fast exploration and uncertainty quantification of the neural architecture search space. A pruning procedure is proposed to sample well-performing architectures from the posterior distribution. The experiments demonstrate the efficacy of BPNAS, revealing improved efficiency and accuracy compared to the state-of-the-art NAS and NES methods on benchmark datasets. It would be interesting to extend our framework by incorporating uncertainty in edge weights and biases, potentially further improving accuracy.

## References

[1] C. White, M. Safari, R. Sukthanker, B. Ru, T. Elsken, A. Zela, D. Dey, and F. Hutter, “Neural architecture search: Insights from 1000 papers,” *arXiv preprint arXiv:2301.08727*, 2023.

[2] D. Baymurzina, E. Golikov, and M. Burtsev, “A review of neural architecture search,” *Neurocomputing*, vol. 474, pp. 82–93, 2022.

[3] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, “A comprehensive survey of neural architecture search: Challenges and solutions,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 4, pp. 1–34, 2021.

[4] J.-S. Kang, J. Kang, J.-J. Kim, K.-W. Jeon, H.-J. Chung, and B.-H. Park, “Neural architecture search survey: A computer vision perspective,” *Sensors*, vol. 23, no. 3, p. 1713, 2023.

[5] N. Klyuchnikov, I. Trofimov, E. Artemova, M. Salnikov, M. Fedorov, A. Filippov, and E. Burnaev, “Nas-bench-nlp: Neural architecture search benchmark for natural language processing,” *IEEE Access*, vol. 10, pp. 45 736–45 747, 2022.

[6] H. Liu, K. Simonyan, and Y. Yang, “Darts: Differentiable architecture search,” *arXiv preprint arXiv:1806.09055*, 2018.

[7] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, “Understanding and robustifying differentiable architecture search,” *arXiv preprint arXiv:1909.09656*, 2019.

[8] X. Chen and C.-J. Hsieh, “Stabilizing differentiable architecture search via perturbation-based regularization,” in *International conference on machine learning*, PMLR, 2020, pp. 1554–1565.

[9] S. Zaidi, A. Zela, T. Elsken, C. C. Holmes, F. Hutter, and Y. Teh, “Neural ensemble search for uncertainty estimation and dataset shift,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 7898–7911, 2021.

[10] M. Chen, J. Fu, and H. Ling, “One-shot neural ensemble architecture search by diversity-guided search space shrinking,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 530–16 539.

[11] A. R. Narayanan, A. Zela, T. Saikia, T. Brox, and F. Hutter, “Multi-headed neural ensemble search,” *arXiv preprint arXiv:2107.04369*, 2021.

[12] X. Chen, R. Wang, M. Cheng, X. Tang, and C.-J. Hsieh, “Drnas: Dirichlet neural architecture search,” *arXiv preprint arXiv:2006.10355*, 2020.

[13] H. Zhou, M. Yang, J. Wang, and W. Pan, “Bayesnas: A bayesian approach for neural architecture search,” in *International conference on machine learning*, PMLR, 2019, pp. 7603–7613.

[14] M. Zhang, S. Pan, X. Chang, S. Su, J. Hu, G. R. Haffari, and B. Yang, “Balenas: Differentiable architecture search via the bayesian learning rule,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 871–11 880.

[15] Y. Shu, Y. Chen, Z. Dai, and B. K. H. Low, “Neural ensemble search via bayesian sampling,” in *Uncertainty in Artificial Intelligence*, PMLR, 2022, pp. 1803–1812.

[16] S. Premchandar, S. Madireddy, S. Jantre, and P. Balaprakash, “Unified probabilistic neural architecture and weight ensembling improves model robustness,” *arXiv preprint arXiv:2210.04083*, 2022.

[17] X. Dong and Y. Yang, “Nas-bench-201: Extending the scope of reproducible neural architecture search,” *arXiv preprint arXiv:2001.00326*, 2020.

[18] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[19] L. Li, M. Khodak, M.-F. Balcan, and A. Talwalkar, “Geometry-aware gradient algorithms for neural architecture search,” *arXiv preprint arXiv:2004.07802*, 2020.

[20] P. J. Green, “Reversible jump markov chain monte carlo computation and bayesian model determination,” *Biometrika*, vol. 82, no. 4, pp. 711–732, 1995.

[21] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[22] M. A. Efroymson, “Multiple regression analysis,” *Mathematical methods for digital computers*, pp. 191–203, 1960.

[23] X. Chu, T. Zhou, B. Zhang, and J. Li, “Fair darts: Eliminating unfair advantages in differentiable architecture search,” in *European conference on computer vision*, Springer, 2020, pp. 465–480.

[24] G. Li, G. Qian, I. C. Delgadillo, M. Muller, A. Thabet, and B. Ghanem, “Sgas: Sequential greedy architecture search,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1620–1630.

[25] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, and Z. Li, “Darts+: Improved differentiable architecture search with early stopping,” *arXiv preprint arXiv:1909.06035*, 2019.

[26] X. Chen, L. Xie, J. Wu, and Q. Tian, “Progressive differentiable architecture search: Bridging the depth gap between search and evaluation,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1294–1303.

[27] S. Yan, Y. Zheng, W. Ao, X. Zeng, and M. Zhang, “Does unsupervised architecture representation learning help neural architecture search?” *Advances in neural information processing systems*, vol. 33, pp. 12 486–12 498, 2020.

[28] R. Ardywibowo, S. Boluki, X. Gong, Z. Wang, and X. Qian, “Nads: Neural architecture distribution search for uncertainty awareness,” in *International Conference on Machine Learning*, PMLR, 2020, pp. 356–366.

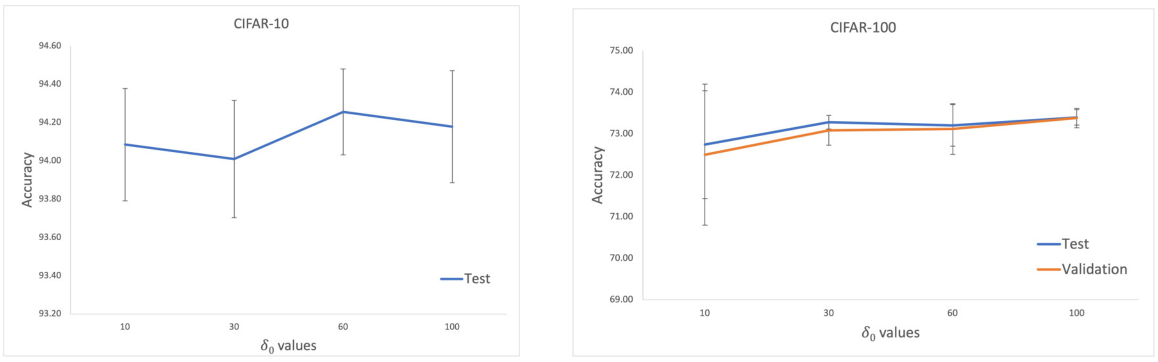
## A. Related Works

**Differentiable NAS** In the landscape of Differentiable NAS, recent efforts have concentrated on overcoming the collapse problem in DARTS [6], with various methodologies emerging. FairDARTS [23] mitigates exclusive competition using the sigmoid function, while SGAS [24] adopts a greedy strategy. DARTS+ [25] and Progressive DARTS [26] employ early stopping to regulate identity operations. While approaches like FairDARTS, SGAS, DARTS+, and Progressive DARTS offer some remedies, they often require significant human intervention and struggle with flexibility and task transferability. Innovations such as Hessian eigenvalues as indicators [7] and unsupervised representation learning [27] address these issues to some extent. However, distribution-based methods [12]–[14] present more effective solutions. DrNAS [12], for instance, formulates differentiable NAS as a distribution learning problem using Dirichlet distribution, optimizing it with pathwise derivative estimators and a distance regularizer for stability and generalization. BaLeNAS [14] tackles the instability with Bayesian Learning and NGVI, enhancing exploration and stability. BayesNAS [13] adopts a Bayesian approach, using hierarchical automatic relevance determination priors to ensure sparsity and interconnectedness in the pruned network structure. These methods collectively signify a shift towards more stable, explorative, and effective architecture search techniques in NAS. However, these works only learn a point estimate of the architecture and weights, so additional properties of distributional formulation such as uncertainty quantification, ensembling, or robustness have not been explored.

**Neural Ensemble Search** In the realm of Neural Ensemble Search (NES), various methodologies have been developed to optimize ensembles of neural networks. NES-RS [9] introduced a technique that uses random search and regularized evolution, achieving enhanced accuracy and robustness, albeit with considerable computational demands. MH-NES [11] proposed a Multi-headed NES approach that applies differentiable NAS to multi-headed networks, optimizing ensemble loss and promoting diversity among predictions, thereby reducing computational intensity. NESBS approach [15] utilized a supernetwork to estimate ensemble performance, incorporating Bayesian sampling for efficient selection. Similarly, NADS [28] employed an evolutionary search with a focus on improving uncertainty estimates and out-of-distribution detection, demonstrating the growing versatility and sophistication of NES strategies.

## B. NAS-Bench-201 space

The NAS-Bench-201 database contains 15,625 pre-trained models with varying cell architectures on CIFAR-10, CIFAR-100, and ImageNet-16-120 datasets. Given an architecture, we can query this database to get the final accuracy and thus save time for running the expensive evaluation stage for any NAS method. Therefore, this database enables us to separately compare the architecture search phase and the evaluation phase of the NAS algorithms.



(a) (b)

*Figure 2.* The effect of $\delta_0$ on the accuracy of the architecture selected by the BPNAS algorithm 1 on NAS-BENCH-201 search space for CIFAR-10 and CIFAR-100. It is observed that lower $\delta_0$ leads to lower accuracy and higher instability in the results.

## C. On hyperparameter $\delta$

In Algorithm 1, the pruning criterion depends on the hyperparameter $\delta$. This hyperparameter controls how fast the algorithm proposes a candidate architecture.In our implementation, we considered $\delta$ as the 95-th percentile value of the distribution of the random variable $\|\text{Dirichlet}(\delta_0 \mathbf{1}_{\|\gamma_m\|_0}) - \frac{\mathbf{1}_{\|\gamma_m\|_0}}{\|\gamma_m\|_0}\|_2$ for some $m$. Therefore, increasing $\delta$ or equivalently $\delta_0$ results in more aggressive pruning as the Dirichlet(.) distribution becomes increasingly dense around its mean $\frac{\mathbf{1}_{\|\gamma_m\|_0}}{\|\gamma_m\|_0}$. In Section 4, we presented the experimental results for a single $\delta_0$ value. In this section, we explore the sensitivity of the results with changing $\delta_0$.

In Fig. 2, we plot the mean accuracy across 10 seeds (with the standard deviations) against four $\delta_0$ values on CIFAR-10 and CIFAR-100. We observe that lower $\delta_0$ on these datasets resulted in higher standard deviation and lower accuracy on both the test and validation set. In the experiments, we found the accuracy on ImageNet-16-120 to be insensitive to the hyperparameter $\delta_0$.