BaLeNAS: Differentiable Architecture Search via the Bayesian Learning Rule

Anonymous CVPR submission

Paper ID 5389

Abstract

Differentiable Architecture Search (DARTS) has received massive attention in recent years, mainly because it significantly reduces the computational cost through weight sharing and continuous relaxation. However, more recent works find that existing differentiable NAS techniques struggle to outperform naive baselines, yielding deteriorative architectures as the search proceeds. Rather than directly optimizing the architecture parameters, this paper formulates the neural architecture search as a distribution learning problem through relaxing the architecture weights into Gaussian distributions. By leveraging the natural-gradient variational inference (NGVI), the architecture distribution can be easily optimized based on existing codebases without incurring more memory and computational consumption. We demonstrate how the differentiable NAS benefits from Bayesian principles, enhancing exploration and improving stability. The experimental results on NAS-Bench-201 and NAS-Bench-1shot1 benchmark datasets confirm the significant improvements the proposed framework can make. In addition, instead of simply applying the argmax on the learned parameters, we further leverage the recently-proposed training-free proxies in NAS to select the optimal architecture from a group architectures drawn from the optimized distribution, where we achieve state-of-the-art results on the NAS-Bench-201 and NAS-Bench-1shot1 benchmarks. Our best architecture in the DARTS search space also obtains competitive test errors with 2.37%, 15.72%, and 24.2% on CIFAR-10, CIFAR-100, and ImageNet datasets, respectively.

1. Introduction

Neural Architecture Search (NAS) [12, 25–27, 38] is attaining increasing attention in the deep learning community by automating the labor-intensive and time-consuming neural network design process. More recently, NAS has achieved the state-of-the-art results on various deep learning applications, including image classification [41], object detection [11], stereo matching [13]. Although NAS has the potential to find high-performing architectures without human intervention, the early NAS methods have extremely-high computational requirements [19, 37, 54]. For example, in [37, 54], NAS costs thousands of GPU days to obtain a promising architecture through reinforcement learning (RL) or evolutionary algorithm (EA). This high computational requirement in NAS is unaffordable for most researchers and practitioners. Since then, more researchers shift to improve the efficiency of NAS methods [20, 28, 36]. Weight sharing NAS, also called One-Shot NAS [2, 36], defines the search space as a supernet, and only the supernet is trained for once during the architecture search. The architecture evaluation is based on inheriting weights from the supernet without retraining, thus significantly reducing the computational cost. *Differentiable architecture search* (DARTS) [31], which is one of the most representative works, further relaxes the discrete search space into continuous space and jointly optimize supernet weights and architecture parameters with gradient descent, to further improve efficiency. Through employing two techniques, weight sharing [2, 36] and continuous relaxation [6, 16, 31, 45], DARTS reformulates the discrete operation selection problem in NAS as a continuous magnitude optimization problem, which reduces the computational cost significantly and completes the architecture search process within several hours on a single GPU.

Despite notable benefits on computational efficiency from differentiable NAS, more recent works find it is still unreliable [8, 49] to directly optimize the architecture magnitudes. For example, DARTS is unable to stably obtain excellent solutions and yields deteriorative architectures during the search proceeds, performing even worse than random search in some cases [48]. This critical weakness is termed as *instability* in differentiable NAS [49]. Zela *et al*. [49] empirically point out that the instability of DARTS is highly correlated with the dominant eigenvalue of the Hessian of the validation loss with respect to the architectural parameters, while this dominant eigenvalue increases during the architecture search. Accordingly, they proposed a simple early-stopping criterion based on this dominant eigenvalue to robustify DARTS. In addition, Wang *et al*. [44] observe that the instability in DARTS's final discretization process of architecture selection, where the optimized magnitude could hardly indicate the importance of operations. On the other hand, several works [9, 29, 39, 51] state that directly optimizing the architecture parameters without exploration easily entails the rich-gets-richer problem, leading to those architectures that converge faster at the beginning while achieve poor performance at the end of training, e.g. architectures with intensive *skip-connections* [14, 30].

Unlike most existing works that directly optimize the architecture parameters, we investigate differentiable NAS from a distribution learning perspective, and introduce the **Ba**yesian **Le**arning rule [22, 23, 33, 35] to the architecture optimization in differentiable **NAS** with considering natural-gradient variational inference (NGVI) methods to optimize the architecture distribution, which we call **BaLeNAS**. We theoretically demonstrate how the framework naturally enhance the exploration for differentiable NAS and improves the stability, and the experimental results confirm that our framework enhances the performance for differentiable NAS. Rather than simply applying *argmax* on the mean to get a discrete architecture, we for the first time leverage the training free proxies [1, 7, 32] to select a more competitive architecture from the optimized distribution, without incurring any additional training costs. Specifically, our approach achieves state-of-the-art performance on NAS-Bench-201 [17] and improves the performance on NAS-Bench-1shot1 [50] by large margins, and obtains competitive results on CIFAR-10, CIFAR-100, and ImageNet datasets in the DARTS [31] search space, with test error 2.37%, 15.72%, and 24.2%, respectively. Our contributions are summarized as follows.

- Firstly, this paper formulates the neural architecture search as a distribution learning problem and builds a generalized Bayesian framework for architecture optimization in differentiable NAS. We demonstrate that the proposed Bayesian framework is a practical solution to enhance exploration for differentiable NAS and improve stability as a by-product via implicitly regularizing the Hessian norm.
- Secondly, instead of directly applying the *argmax* on the learned parameters to get architectures, we for the first time leverage zero-cost proxies to select competitive architectures from the optimized distributions. As these proxies are calculated without any training, the architecture selection phase can be finished extremely efficiently.
- Thirdly, the proposed framework is built based on DARTS and is also comfortable to be extended to other differentiable NAS methods with minimal modifications through leveraging the natural-gradient variational inference (NGVI). Experiments show that our framework consistently improves the baselines with obtaining more competitive architectures in various search spaces.

2. Preliminaries

2.1. Differentiable Architecture Search

Differentiable architecture search (DARTS) is built on weight-sharing NAS [2, 36], where the supernet is trained for once per the architecture search cycle. Rather than using the heuristic methods [36, 51] to search for the promising architecture in the discrete architecture space $\mathcal{A}$, DARTS [31] proposes the differentiable NAS framework by applying a continuous relaxation (usually a *softmax*) to the discrete architecture space and enabling gradient descent for architecture optimization. Therefore, architecture parameters α_θ and supernet weights w could be jointly optimized during the supernet training, and the promising architecture parameters α_θ^* are searched from the continuous search space $\mathcal{A}_\theta$ once the supernet is trained. The bilevel optimization formulation is usually adopted to alternatively learn α_θ and w:

$$\min_{\alpha_\theta \in \mathcal{A}_\theta} \mathcal{L}_{\texttt{val}}\Big(\underset{w}{\texttt{argmin}}\, \mathcal{L}_{\texttt{train}}(w(\alpha_\theta), \alpha_\theta)\Big), \tag{1}$$

and the best discrete architecture α^* is obtained after applying *argmax* on α_θ^*.

Despite notable benefits on computational efficiency from DARTS, more recent works find it is still unreliable [8, 49] that directly optimizes the architecture magnitudes, where DARTS usually observes a performance collapses with search progresses. This phenomenon is also called the instability of differentiable NAS [8]. Zela *et al.* [49] observed that the there is a strong correlation between the dominant eigenvalue of the Hessian of the validation loss and the architecture's generalization error in DARTS, and keeping the the Hessian matrix's norm in a low level plays a key role in robustifying the performance of differentiable NAS [8]. In addition, as described above, the differentiable NAS first relaxes the discrete architectures into continuous representations to enable the gradient descent optimization, and projects the continuous architecture representation α_θ into discrete architecture α after the differentiable architecture optimization. However, more recent works [44] cast doubts on the robustness of this discretization process in DARTS that the magnitude of architecture parameter α_θ^* could hardly indicate the importance of operations with *argmax*. Taking the DARTS as example, the searched architecture parameters α_θ are continuous, while α is represented with $\{0, 1\}$ after *argmax*. DARTS assumes that the $\mathcal{L}_{val}(w^*, \alpha_\theta^*)$ is a good indicator to the validation performance of α, $\mathcal{L}_{val}(w^*, \alpha^*)$. However, when we conduct the Taylor expansion on the local optimal α_θ^* [8, 9], we have:

$$\begin{aligned}\mathcal{L}_{val}(w^*, \alpha^*) &= \mathcal{L}_{val}(w^*, \alpha_\theta^*) + \nabla_{\alpha_\theta}\mathcal{L}_{val}(w^*, \alpha_\theta^*)^T(\alpha^* - \alpha_\theta^*) \\ &\quad + \frac{1}{2}(\alpha^* - \alpha_\theta^*)^T\mathcal{H}(\alpha^* - \alpha_\theta^*) \\ &= \mathcal{L}_{val}(w^*, \alpha_\theta^*) + \frac{1}{2}(\alpha^* - \alpha_\theta^*)^T\mathcal{H}(\alpha^* - \alpha_\theta^*)\end{aligned} \tag{2}$$

where $\nabla_{\alpha_\theta}\mathcal{L}_{val} = 0$ due to the local optimality condition, and $\mathcal{H}$ is the Hessian matrix of $\mathcal{L}_{val}(w^*, \alpha_\theta)$. We can see that the incongruence of the final continuous architecture representation and the final discrete architecture relates to the Hessian matrix's norm. However, as demonstrated by the empirical results in [49], the eigenvalue of this Hessian matrix increases during the architecture search, incurring more incongruence.

2.2. Bayesian Deep Learning

Given a dataset $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_1, ..., \mathcal{D}_N\}$ and a deep neural network with parameters θ, the most popular method to learn θ with $\mathcal{D}$ is Empricial Risk Minimization (ERM):

$$\min \bar{\ell}(\theta) := \sum_{i=1}^{N} \ell_i(\theta) + \eta\mathcal{R}(\theta), \tag{3}$$

where ℓ_i is a loss function, e.g., $\ell_i = -\log p(\mathcal{D}_i \mid \theta)$ for classification and $\mathcal{R}$ is the regularization term.

In contrast, the **Bayesian deep learning** estimate the posterior distribution of θ, $p(\theta \mid \mathcal{D}) := p(\mathcal{D} \mid \theta)p(\theta)/p(\mathcal{D})$, where $p(\theta)$ is the prior distribution. However, the normalization constant $p(\mathcal{D}) = \int p(\mathcal{D} \mid \theta)p(\theta)d\theta$ is difficult to compute for large DNNs. The variational inference (VI) [18] resolves this issue in Bayesian deep learning by approximating $p(\theta \mid \mathcal{D})$ with a new distribution $q(\theta)$, and minimizes the Kullback-Leibler (KL) divergence between $p(\theta \mid \mathcal{D})$ and $q(\theta)$,

$$\text{argmin}_\theta \text{KL}(q(\theta) \parallel p(\theta \mid \mathcal{D})). \tag{4}$$

When considering both $p(\theta)$ and $q(\theta)$ as Gaussian distributions with diagonal covariances:

$$p(\theta) := \mathcal{N}(\theta \mid \mathbf{0}, \mathbf{I}/\delta), \;\; q(\theta) := \mathcal{N}(\theta \mid \mu, \text{diag}(\sigma^2)), \tag{5}$$

where δ is a known precision parameter with $\delta > 0$, the mean μ and deviation σ^2 of q can be estimated by minimizing the negative of evidence lower bound (ELBO) [3]:

$$\begin{aligned} \mathcal{L}(\mu, \sigma) : &= -\sum_{i=1}^{N} \mathbb{E}_q \left[\log p(\mathcal{D}_i \mid \theta)\right] + \text{KL}(q(\theta) \parallel p(\theta)) \\ &= -\mathbb{E}_q \sum_{i=1}^{N} \log p(\mathcal{D}_i \mid \theta) + \mathbb{E}_q \left[\log \frac{q(\theta)}{p(\theta)}\right] \end{aligned} \tag{6}$$

A straightforward approach is using the stochastic gradient descent to learn μ and σ^2 along with minimizing $\mathcal{L}$, called as the Bayes by Backprob (BBB) [4]:

$$\mu_{t+1} = \mu_t - \varsigma_t \hat{\nabla}_\mu \mathcal{L}_t, \;\; \sigma_{t+1} = \sigma_t - \varphi_t \hat{\nabla}_\sigma \mathcal{L}_t, \tag{7}$$

where ς_t and φ_t are the learning rates, and $\hat{\nabla}_\mu \mathcal{L}_t$ and $\hat{\nabla}_\sigma \mathcal{L}_t$ are the unbiased stochastic gradient estimates of $\mathcal{L}$ at μ_t and σ_t. However, VI remains to be impractical for learning large deep networks. The obvious issue is that VI introduces more parameters to learn, as it needs to replace all neural networks weights with random variables and simultaneously optimize two vectors μ and σ to estimate the distribution of θ, so the memory requirement is also doubled, leading a lot of modifications when fitting existing differentiable NAS codebases with the variational inference.

2.3. Training Free Proxies for NAS

Training Free NAS tries to identify promising architectures at initialization without incurring training. Mellor *et al*. [32] empirically find that the correlation between sample-wise input-output Jacobian can indicate the architecture's test performance, and propose using the Jacobian to score a set of randomly sampled models with randomly initialized weights, which greedily chooses the model with the highest score. TE-NAS [7] utilizes the spectrum of NTKs and the number of linear regions to analyzing the trainability and expressivity of architectures. Rather than evaluating the whole architecture, TE-NAS uses the perturbation-based architecture selection as [44], to measure the importance of each operation for the supernet prune.

Zero-cost NAS [1] extends the saliency metrics in the network pruning at initialization to score an architecture, through summing scores of all parameters θ in the architecture. There are three popular saliency metrics, SNIP [24], GraSP [43], and Synflow [42]:

$$\mathcal{S}_{snip}(\theta) = \left|\frac{\partial \mathcal{L}}{\partial \theta} \odot \theta\right|, \; \mathcal{S}_{grasp}(-\theta) = -(H\frac{\partial \mathcal{L}}{\partial \theta}) \odot \theta, \; \mathcal{S}_{\text{SF}}(\theta) = \frac{\partial \mathcal{R}_{\text{SF}}}{\partial \theta} \odot \theta, \tag{8}$$

where $\mathcal{L}$ is the common loss based on initialized weights, H is the Hessian matrix, and $\mathcal{R}_{\text{SF}}$ is defined as $\mathcal{R}_{\text{SF}} = \mathbf{1}^T \left(\prod_{l=1}^{L} \left|\theta^{[l]}\right|\right) \mathbf{1}$ that makes SynFlow data-agnostic. Since these scores can be obtained without any training, zero-cost NAS utilizes these zero-cost proxies to assist NAS by *warmup* different search algorithms, e.g., initializing population or controller for aging evolution NAS and RL based NAS, respectively. Different from zero-cost NAS that leverages proxies before the search, we utilize these zero-cost proxies for the architecture selection after search, to select more competitive architectures from the optimized distributions.

3. The Proposed Method: BaLeNAS

3.1. Formulating NAS as Distribution Learning

Differentiable NAS normally considers the architecture parameters α_θ as learnable parameters and directly conducts optimization in this space. Most previous differentiable NAS methods first optimize the architecture parameters based on the gradient of the performance, then update the supernet weights based on the updated architecture parameters. Since architectures with updated supernet weights are supposed to have higher performance, architectures with better performance in the early stage have a higher probability of being selected for the supernet training. The supernet training again improves these architectures' performance. This is to say, directly optimizing α_θ without exploration easily entails the *rich-get-richer problem* [29, 51], leading to suboptimal paths in the search space that converges faster at the beginning but plateaued quickly [9, 39]. In contrast, formulating the differentiable NAS as a distribution learning problem by relaxing architecture parameters can naturally introduce **stochasticity** and encourage **exploration** to resolve this problem [8, 9].

In this paper, we formulate the architecture search as a distribution learning problem, that for the first time consider the more general Gaussian distributions for the architecture parameters to optimize the posterior distribution $p(\alpha_\theta \mid \mathcal{D})$ rather than α_θ. Considering both $p(\theta)$ and $q(\theta)$ as Gaussian distributions as Eq.(5), the bilevel optimization problem in Eq.(1) could be reformulated as the distribution learning based NAS:

$$\begin{aligned} &\min_{\mu,\sigma} \mathbb{E}_{q(\alpha_\theta|\mu,\sigma)}\mathcal{L}_{\texttt{val}}(w^*(\alpha_\theta),\alpha_\theta), \\ &\text{s.t. } w^*(\alpha_\theta) = \operatorname*{argmin}_w \mathcal{L}_{\texttt{train}}(w(\alpha_\theta),\alpha_\theta), \end{aligned} \tag{9}$$

where μ and σ are the two learnable parameters for the distribution $q(\alpha_\theta \mid \mu,\sigma) := \mathcal{N}(\alpha_\theta \mid \mu, \text{diag}(\sigma^2))$. Considering the variational inference and Bayesian deep learning, based on Eq.(4)-(6), the loss function for the outer-loop architecture distribution optimization problem could be defined as:

$$\mathbb{E}_q\left[\mathcal{L}_{\texttt{val}}\right] := -\mathbb{E}_q \sum_{i=1}^{N} \log p(\mathcal{D}_i \mid \alpha_\theta) + \mathbb{E}_q\left[\log \frac{q(\alpha_\theta)}{p(\alpha_\theta)}\right]. \tag{10}$$

Since the architecture parameters α_θ are random variables sampled from the Gaussian distribution $q(\alpha_\theta \mid \mu,\sigma)$, the distribution learning-based method naturally encourages exploration during the architecture search.

3.2. Natural-Gradient VI for NAS

As describe in Sec.2.2, the traditional variational inference has double memory requirement and needs to re-design the object function, making it difficult to fit with the differentiable NAS. Thus, this paper considers natural-gradient variational inference (NGVI) methods [22, 35] to optimize the architecture distribution $p(\alpha_\theta \mid \mathcal{D})$ in a natural parameter space, which requires the same number of parameters as the traditional learning method. By leveraging NGVI, the architecture parameter distribution could be learned by only updating a natural parameter λ during the search.

NGVI parameterizes the distribution $q(\alpha_\theta)$ with a natural parameter λ, considering $q(\alpha_\theta \mid \lambda)$ in a class of minimal exponential family with natural parameter λ [21]:

$$q(\alpha_\theta \mid \lambda) := h(\alpha_\theta)\exp\left[\lambda^T\phi(\alpha_\theta) - A(\lambda)\right], \tag{11}$$

where $h(\alpha_\theta)$ is the base measure, $\phi(\alpha_\theta)$ is a vector containing sufficient statistics, and $A(\lambda)$ is the log-partition function.

When $h(\alpha_\theta) \equiv 1$, the distribution $q(\alpha_\theta \mid \lambda)$ could be learned by only updating λ during the training [22, 23], and λ could be learned in the natural-parameter space by:

$$\lambda_{t+1} = (1-\rho_t)\lambda_t - \rho_t \nabla_\mu \mathbb{E}_{q_t}\left[\bar{\ell}(\alpha_\theta)\right], \tag{12}$$

where ρ_t is the learning rate, $\bar{\ell}$ is in the form of Eq.(3), and the derivative $\nabla_\mu \mathbb{E}_{q_t(\alpha_\theta)}\left[\bar{\ell}(\alpha_\theta)\right]$ is taken at $\mu = \mu_t$ which is the expectation parameter with Markov Chain Monte Carlo (MCMC) sampling. And q_t is the $q(\alpha_\theta \mid \lambda)$ parameterized by λ_t, $\mu = \mu(\lambda)$ is the expectation parameter of $q(\alpha_\theta \mid \lambda)$. This is also called as the Bayesian learning rule [23].

When we consider Gaussin mean-field VI that $p(\alpha_\theta)$ and $q(\alpha_\theta)$ are in the form of Eq.(5), the Variational Online-Newton (VON) method proposed by Khan et. al. [22] shows that the NGVI update could be written with the following update:

$$\mu_{t+1} = \mu_t - \beta_t(\hat{\mathbf{g}}(\theta_t) + \tilde{\delta}\mu_t)/(\mathbf{s}_{t+1} + \tilde{\delta}), \tag{13}$$

$$\mathbf{s}_{t+1} = (1-\beta_t)\mathbf{s}_t + \beta_t \, \text{diag}[\hat{\nabla}^2 \bar{\ell}(\theta_t)], \tag{14}$$

where β_t is the learning rate, $\theta_t \sim \mathcal{N}(\alpha_\theta \mid \mu_t, \sigma_t^2)$ with $\sigma_t^2 = 1/[N(\mathrm{s}_t + \tilde{\delta})]$ and $\tilde{\delta} = \delta/N$. $\hat{\mathbf{g}}$ is the stochastic estimate with respect to q through MCMC sampling that, $\hat{\mathbf{g}}(\theta_t) = \frac{1}{M}\sum_{i\in\mathcal{M}} \nabla_{\alpha_\theta}\bar{\ell}_i(\alpha_\theta)$, and the minibatch $\mathcal{M}$ contains M samples. More details are in [22]. Variational RMSprop (Vprop) [22] further uses gradient magnitude (GM) [5] approximation to reformulate Eq.(14) as:

$$\mathbf{s}_{t+1} = (1-\beta_t)\mathbf{s}_t + \beta_t[\hat{\mathbf{g}}(\theta_t) \circ \hat{\mathbf{g}}(\theta_t)], \tag{15}$$

with $\hat{\nabla}^2_{j,j}\bar{\ell}(\theta_t) \approx \left[\frac{1}{M}\sum_{i\in\mathcal{M}_t} g_i(\alpha_\theta^j)\right]^2 = [\hat{g}(\theta_t^j)]^2$ [5]. The most important benefit of VON and Vprop is that they only need to calculate one parameter's gradient to update posterior distribution. In this way, this learning paradigm requires the same number of parameters as traditional learning methods and easy to fit with existing codebases.

We implement the proposed BaLeNAS based on the DARTS [31] framework, the most popular differentiable NAS baseline. Similar to DARTS, BaLeNAS also considers an Adam-like optimizer for the architecture optimization, updating the natural parameter λ of $p(\theta \mid \mathcal{D})$ as:

$$\lambda_{t+1} = \lambda_t - \rho_t \nabla_\lambda \mathcal{L}_t + \gamma_t(\lambda_t - \lambda_{t-1}), \tag{16}$$

where the last term is the momentum. Based on the Vprop in Eq.(13) and (15), the update of μ and σ for the Adam-like optimizer with NGVI, also called as Variational Adam (VAdam), could be defined as following:

$$\begin{aligned} \mu_{t+1} =& \mu_t - \beta_t(\hat{\mathbf{g}}(\theta_t) + \tilde{\delta}\mu_t) \circ \frac{1}{(\mathbf{s}_{t+1} + \tilde{\delta})} \\ &+ \gamma_t \left[\frac{\mathbf{s}_t + \tilde{\delta}}{\mathbf{s}_{t+1} + \tilde{\delta}}\right] \circ (\mu_t - \mu_{t-1}), \end{aligned} \tag{17}$$

Algorithm 1 BaLeNAS

Initialize a supernet with supernet weights w and architecture parameters α_θ

while *not converged* **do**

  Update μ and σ^2 for $q(\alpha_\theta \mid \mu, \sigma^2)$ based on Eq.(17) and Eq.(18), with VAdam optimizer.

  Update supernet weights w based on cross-entropy loss with the common SGD optimizer.

end while

Obtain discrete architecture α^* through *argmax* on μ; or sample a set of α_θ from $q(\alpha_\theta^* \mid \mu, \sigma^2)$, and utilize the training free proxies for selection.

$$\mathbf{s}_{t+1} = (1-\beta_t)\mathbf{s}_t + \beta_t[\hat{\mathbf{g}}(\theta_t) \circ \hat{\mathbf{g}}(\theta_t)]. \tag{18}$$

where "$\circ$" stands for element-wise product, $\theta_t \sim \mathcal{N}(\alpha_\theta \mid \mu_t, \sigma_t^2)$ with $\sigma_t^2 = 1/[N(\mathrm{s}_t + \tilde{\delta})]$. As pointed out in Sec. 2.2 and shown in Eq.(17) and Eq.(18), the distribution $q(\alpha_\theta) = \mathcal{N}(\alpha_\theta \mid \mu, \sigma^2)$ is now optimized, needing to calculate the gradient of only one parameter.

Implicit Regularization from MCMC Sampling: Several recent works [8,9,49] empirically and theoretically show that the performance of differentiable NAS is highly related to the norm of $\mathcal{H}$, the Hessian matrix of $\mathcal{L}_{val}(w^*, \alpha_\theta)$, and keeping this norm in a low level plays a key role in robustifying differentiable NAS. As described before, we know the loss $\mathbb{E}_{q_t(\alpha_\theta)}\left[\bar{\ell}(\alpha_\theta)\right]$ of architecture optimization in BaLeNAS is calculated based on MCMC sampling, showing the naturality of enhancing exploration. Besides, $\mathbb{E}_{q_t(\alpha_\theta)}\left[\bar{\ell}(\alpha_\theta)\right]$ also has the naturality to enhance the stability in differentiable NAS as SDARTS [8]. When conducting the Taylor expansion, the loss function for the architecture parameters update $\mathbb{E}_{q_t(\alpha_\theta)}\left[\bar{\ell}(\alpha_\theta)\right]$ could be described as:

$$\begin{aligned}
&\mathbb{E}_{q_t(\alpha_\theta)}\left[\bar{\ell}(\alpha_\theta)\right] \\
=&\mathbb{E}_{q(\alpha_\theta|\mu,\sigma)}\mathcal{L}_{val}(w,\alpha_\theta) = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)}\mathcal{L}_{val}(w,\mu+\epsilon) \\
=&\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)}[\mathcal{L}_{val}(w,\mu) + \nabla_\mu\mathcal{L}_{val}(w,\mu)^T\epsilon + \frac{1}{2}\epsilon^T\mathcal{H}\epsilon] \\
=&\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)}\left[\mathcal{L}_{val}(w,\mu) + \frac{1}{2}\epsilon^T\mathcal{H}\epsilon\right] \\
=&\mathcal{L}_{val}(w,\mu) + \frac{\sigma^2}{2}\mathrm{Tr}\left\{\mathcal{H}\right\},
\end{aligned} \tag{19}$$

where the line 4 in Eq.(19) is obtained since $\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)}[\nabla_\mu\mathcal{L}_{val}(w,\alpha_\theta)^T\epsilon] = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)}[\epsilon] * \nabla_\mu\mathcal{L}_{val}(w,\alpha_\theta) = 0$, as $\epsilon \sim \mathcal{N}(0,\sigma^2)$ is a Gaussian distribution with zero mean, and $\mathbb{E}(\epsilon^2) = \sigma^2$. μ is the expectation parameter of $q(\alpha_\theta \mid \mu, \sigma^2)$, and $\mathcal{H}$ is the Hessian matrix of $\mathcal{L}_{val}(w,\mu)$. We can find the loss function that could implicitly control the trace norm of $\mathcal{H}$ similar as [8,9], helping **stabilizing** differentiable NAS.

3.3. Architecture Selecting from the Distribution

After the optimization of BaLeNAS, we learns an optimized Gaussian distribution for the architecture parameters $q(\alpha_\theta^* \mid \mu, \sigma^2)$, which is used to get the optimal architecture α^*. In this paper, we consider two methods to get the discrete architecture α^*. The first one is a simple and direct method, which utilizes the expectation of α_θ^* to select the best operation for each edge through the *argmax* as DARTS, where the expectation term is simply the mean μ [9]. However, as we described in Sec. 2.1, this method may result in instability and incongruence. The second one is more general, which samples a set of α from the distribution $q(\alpha_\theta^* \mid \mu, \sigma^2)$ for architecture selection. However, in the neural architecture search, evaluating a set of architectures will incur unaffordable computational costs. In this paper, instead of utilizing training-free proxies to assist NAS by *warmup* before search as [1], we leverage these proxies, including SNIP [24], GraSP [43], and Synflow [42], to score the sampled architectures for selection after search.

Algorithm 1 gives a simple implementation of BaLeNAS, where only the red part is different from DARTS. As shown, in our BaLeNAS, only architecture parameter optimization is different from DARTS which uses the VAdam optimizer, making it easy to be implemented. Furthermore, as most existing differentiable NAS methods are built based on DARTS codebase, our BaLeNAS is also comfortable to be adapted to them with minimal modifications.

4. Experiments and Results

In this section, we consider three different search spaces to analyze the proposed BaLeNAS framework. The first two are NAS benchmark datasets, NAS-Bench-201 [17] and NAS-Bench-1shot1 [50]. The ground-truth for all candidate architectures in the two benchmark datasets is known. The NAS methods could be evaluated without retraining the searched architectures based on these benchmark datasets, thus greatly relieving the computational burden. The third one is the commonly-used CNN search space in DARTS [31]. We first analyze our proposed BaLeNAS in the two benchmark datasets, then compare BaLeNAS with state-of-the-art NAS methods in the DARTS search space.

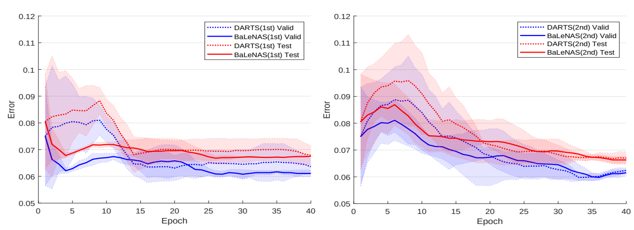
4.1. Experiments on Benchmark Datasets

The NAS-Bench-201 [17] has a unified cell-based search space, where the cell structure is densely-connected, containing four nodes with five candidate operations applied on each node, resulting in 15,625 architectures. NAS-Bench-201 reports the CIFAR-10, CIFAR-100, and Imagenet performance for all architecture in this search space. The NAS-Bench-1shot1 [50] is built from the NAS-Bench-101 benchmark dataset [47], through dividing all architectures in NAS-Bench-101 into 3 different unified cell-based search spaces,

Table 1. Comparison results with state-of-the-art NAS approaches on NAS-Bench-201.

Method	CIFAR-10		CIFAR-100		ImageNet-16-120	
	Valid(%)	Test(%)	Valid(%)	Test(%)	Valid(%)	Test(%)
Random baseline	83.20±13.28	86.61±13.46	60.70±12.55	60.83±12.58	33.34±9.39	33.13±9.66
ENAS [36]	37.51±3.19	53.89±0.58	13.37±2.35	13.96±2.33	15.06±1.95	14.84±2.10
RandomNAS [28]	85.63±0.44	88.58±0.21	60.99±2.79	61.45±2.24	31.63±2.15	31.37±2.51
SETN [15]	84.04±0.28	87.64±0.00	58.86±0.06	59.05±0.24	33.06±0.02	32.52±0.21
GDAS [16]	90.00±0.21	93.51±0.13	71.14±0.27	70.61±0.26	41.70±1.26	41.84±0.90
DrNAS [9]	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
DARTS (1st) [31]	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
DARTS (2nd) [31]	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
Zero-cost NAS [1]	90.19±0.66	93.45±0.28	70.55±1.61	70.73±1.36	43.24±2.52	43.64±2.42
BaLeNAS (1st)	91.03±0.15	93.62±0.12	70.88±0.60	70.98±0.41	45.19±0.75	45.25±0.86
BaLeNAS (2nd)	91.32±0.09	94.02±0.14	71.53±0.08	71.93±0.27	45.39±0.17	45.48±0.39
BaLeNAS-TF	91.52±0.04	94.33±0.03	72.67±0.41	72.95±0.28	46.14±0.23	46.54±0.36
optimal	91.61	94.37	74.49	73.51	46.77	47.31

The best single run of BaLeNAS-TF achieves **94.37%**, **73.22%**, and **46.71%** test accuracy on three datasets, respectively. Our BaLeNAS-TF considers the Synflow based proxy for architecture selection in this experiment.



(a) First order approximation (b) Second order approximation

Figure 1. Validation and test error of BaLeNAS and DARTS on the search space 3 of NAS-Bench-1shot1.

Table 2. Ablation study on the architecture selection.

Method (size)	Test Accuracy CIFAR-10	CIFAR-100	ImageNet
Zero-cost NAS(10)	92.12±1.25	68.1±2.49	40.07±1.86
Zero-cost NAS(50)	92.52±0.05	70.27±0.25	42.92±0.95
Zero-cost NAS(100)	93.45±0.16	69.87±0.35	44.43±0.75
BaLeNAS-TF(10)	94.08±0.13	72.55±0.42	45.82±0.30
BaLeNAS-TF(50)	**94.33±0.03**	**72.95±0.28**	**46.54±0.36**
BaLeNAS-TF(100)	**94.33±0.03**	**72.95±0.28**	**46.54±0.36**

containing 6,240, 29,160, and 363,648 architectures, respectively, and the CIFAR-10 performance for all architectures are reported. The architectures in each search space have the same number of nodes and connections, making the differentiable NAS could be directly applied to each space.

4.1.1 Reproducible Comparison on NAS Benchmarks

Table 1 summarizes the performance of BaleNAS on NAS-Bench-201 compared with differentiable NAS baselines, where the statistical results are obtained from 4 independent search experiments with four different *random seeds*. In our BaLeNAS, we consider the expectation of α_θ with *argmax* to get the valid architecture, while BaLeNAS-TF consider the training-free proxies for the architecture selection, with the sample size is set as 100. As shown in Table 1, BaLeNAS achieves the best results on the NAS-Bench-201 benchmark and greatly outperforms other baselines on all three datasets. As described in Sec. 3, BaLeNAS is built based on the DARTS framework, with only modeling the architecture parameters into distributions and introducing Bayesian learning rule for optimization. As shown in Table 1, BaLeNAS with first and second-order approximations both outperform DARTS by large margins, verifying the effectiveness of our method. More interesting, combining with the training-free proxies, BaLeNAS-TF can achieve better results, showing that apart from *warmup*, these proxies could also assist differentiable NAS at architecture selection. The best single run of our BaLeNAS-TF achieves **94.37%**, **73.22%**, and **46.71%** test accuracy on three datasets, respectively, which are state-of-the-art on this benchmark dataset.

We also conduct a comparison study on the NAS-Bench-1shot1 dataset to further verify the effectiveness of our BaLeNAS which reformulates architecture search as a distribution learning problem. We have compared BaLeNAS with the baseline DARTS on the three search spaces of NAS-Bench-1shot1 with tracking the validation and test performance of the search architectures in every iteration. As shown in Fig. 1, our BaLeNAS, without training-free proxies based architecture selection, generally outperforms DARTS during the architecture search in terms of validation and test error in the most complicated search space 3, both with first and second-order approximation. More specifically, our BaLeNAS significantly outperforms the baseline in the early stage, demonstrating our BaLeNAS could quickly find the superior architectures and is more stable. The results on both NAS-Bench-201 and NAS-Bench-1shot1 verify that, by formulating the architecture search as a distribution learning

Table 3. Comparison results with state-of-the-art weight-sharing NAS approaches.

Method	Test Error (%) CIFAR-10	CIFAR-100	ImageNet	Param (M)	FLOPs (M)	Search Cost	Architecture Optimization
RandomNAS [28]	2.85±0.08	17.63	27.1	4.3	595	2.7	random
SNAS [45]	2.85±0.02	20.09	27.3 / 9.2	2.8	467	1.5	gradient
BayesNAS [53]	2.81±0.04	-	26.5 / 8.9	3.40	-	0.2	gradient
MdeNAS [52]	2.55	17.61	25.5 / 7.9	3.61	500	0.16	gradient
GDAS [16]	2.93	18.38	26.0 / 8.5	3.4	538	0.21	gradient
XNAS [34]	2.57±0.09	16.34	24.7 / 7.5	3.7	590	0.3	gradient
PDARTS [10]	2.50	16.63	24.4 / 7.4	3.4	543	0.3	gradient
PC-DARTS [46]	2.57±0.07	17.11	25.1 / 7.8	3.6	571	0.3	gradient
DrNAS [9]	2.54±0.03	16.30	**24.2 / 7.3**	4.0	644	0.4	gradient
DARTS+ [30]	2.50±0.11	16.28	-	3.7	-	0.4	gradient
DARTS (1st) [31]	2.94	-	-	2.9	505	1.5	gradient
DARTS (2nd) [31]	2.76±0.09	17.54	26.9 / 8.7	3.4	530	4	gradient
BaLeNAS	2.50±0.07	16.84	25.0 / 7.7	3.82	593	0.6	gradient
BaLeNAS-TF	**2.43±0.08**	**15.72**	**24.2 / 7.3**	3.86	597	0.6	gradient

problem and introducing the Bayesian learning rule to optimize the posterior distribution, BaLeNAS can relieve the instability and naturally enhance exploration to avoid local optimum for differentiable NAS.

4.1.2 Ablation Study on the Architecture Selection

As described, our BaLeNAS-TF samples several architectures from the optimized distribution and leverages the training-free proxies for architecture selection, rather than simply applying *argmax* on the mean. In this subsection, we conduct ablation study to investigate the benefits of our training-free based architecture selection. We considered 3 different training-free proxies as described in Sec. 2.3, including SNIP, GraSP, and Synflow. We find that Synflow is the most reliable proxies in the architecture selection, as it achieves better performance than the remaining two proxies for both zero-cost NAS and BaLeNAS, and also consistently enhances the performance with the increase of sample size. More detailed comparison can be found in the Appendix. Zero-cost NAS [1] randomly generates samples and calculates the scores based on the proxies for architecture selection, while our BaLeNAS-TF generates samples based on the optimized distribution $(\alpha_\theta^* \mid \mu, \sigma^2)$.

Table 2 compared zero-cost NAS and BaLeNAS-TF with different sample sizes in the architecture selection. As shown, the Synflow proxy can assist NAS as zero-cost NAS with different sample sizes achieve much better results than the Random baseline in Table 1, and these proxies also enhance our BaLeNAS, where our BaLeNAS-TF achieve higher accuracy. These results again verified that the architecture selection with train-free proxies can further improve the performance for distribution learning based NAS. More interesting, Table 2 also showed that our BaLeNAS-TF outperformed zero-cost NAS by a large margin, suggesting that our BaLeNAS can converge to a competitive distribution.

4.2. Experiments on DARTS Search Space

To compare with the state-of-the-art differentiable NAS methods, we applied BaLeNAS to the typical DARTS search space [16,28,31] for convolutional architecture search, where all experiment settings are following DARTS [31] for fair comparisons as the same as the most recent works. Our BaLeNAS-TF also considers the Synflow proxy in this experiment. The architecture search in DARTS space generally contains three stages: The differentiable NAS first searches for micro-cell structures on CIFAR-10, and then stack more cells to form the full structure for the architecture evaluation. The best-found cell on CIFAR-10 is finally transferred to larger datasets to evaluate its transferability.

4.2.1 Search Results on CIFAR-10

The comparison results with the state-of-the-art NAS methods are presented in Table 3. The best architecture searched by our BaLeNAS-TF achieves a 2.37% test error on CIFAR-10, which outperforms state-of-the-art NAS methods. We can also see that both BaLeNAS-TF and BaLeNAS outperform DARTS by a large margin, demonstrating the effectiveness of the proposed method. Besides, although BaLeNAS introduced MCMC during architecture optimization, it is still efficient in the sense that the whole architecture search phase in BaLeNAS (2nd) only took 0.6 GPU days.

4.2.2 Transferability Results Analysis

Following DARTS experimental setting, the best-searched architectures on CIFAR-10 are then transferred to CIFAR-100

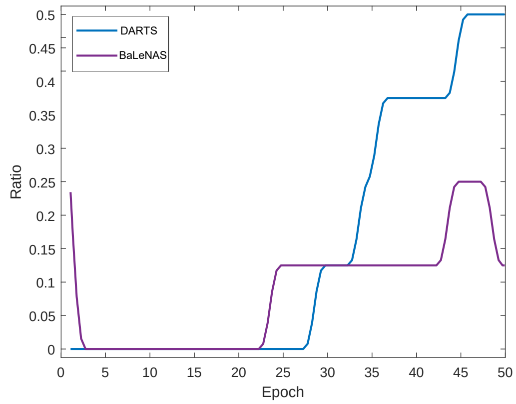


Figure 2. The ratio of skip-connection the searched normal cells during the architecture search in the DARTS space.

and ImageNet to evaluate the transferability. The comparison results with state-of-the-art differentiable NAS approaches on CIFAR-100 and ImageNet are demonstrated in Table 3. As shown in Table2, BaLeNAS-TF achieves a 15.72% test error on the CIFAR-100 dataset, which is a state-of-the-art performance and outperforms peer algorithms by a large margin. On the ImageNet dataset, the best-discovered architecture by our BaLeNAS-TF also achieved a competitive result with 24.2 / 7.3 % top1 / top5 test error, outperforming or on par with all peer algorithms.

4.2.3 Analysis on the Effect of Exploration

Several recent works [9, 39, 51] point out that directly optimizing architecture parameters without exploration easily entails the rich-gets-richer problem, leading to those architectures that converge faster at the beginning while achieve poor performance at the end of training, e.g. architectures with intensive *skip-connections* [14, 30]. However, when the number of *skip-connections* is larger than 3, the architecture's retraining accuracy is usually extremely low [30, 49]. To relieve this issue, BaLeNAS formulates the differentiable neural architecture search as a distribution learning problem, and this experiment verifies how the proposed formulation naturally enhance the exploration to relieve this issue. Fig. 2 plots the ratio of *skip-connection* in the searched normal cell for BaLeNAS and DARTS (the total number of operations in a cell is 8). As shown, DARTS is likely to select more than 3 *skip-connection* in the normal cell during the search. In contrast, in the proposed BaLeNAS, the number of *skip-connections* is generally less than 2 in the normal cell during the search for BaLeNAS.

4.2.4 Tracking of the Hessian norm

As described in Section 2.1, a large Hessian norm deteriorate the robustness of DARTS, and the incongruence between $\mathcal{L}_{val}(w^*, \alpha_\theta^*)$ and $\mathcal{L}_{val}(w^*, \alpha^*)$ is not negligible if we could not maintain the maintains the Hessian norm at a low level. The analysis in Sec. 3.2 and Eq. (19) shows that the loss function of the proposed BaLeNAS implicitly controls the trace norm of $\mathcal{H}$ similar as [8, 9], helping stabilizing differentiable NAS. We plot the trajectory of the Hessian norm of BaLeNAS compared with the vanilla DARTS in Fig. 3. As show, the Hessian norm in our BaLeNAS is always kept in a low level. Although the Hessian norm of BaLeNAS also increases with the supernet training similar as DARTS, BaLeNAS's largest Hessian norm is still smaller than DARTS in the early stage, showing the effectiveness of implicit regularization of our BaLeNAS as described in Sec. 3.2.

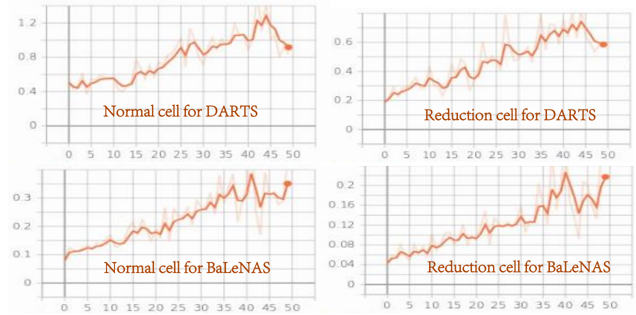


Figure 3. Trajectory of the Hessian norm in DARTS space.

5. Conclusion

In this paper, we have formulated the architecture optimization in the differentiable NAS as a distribution learning problem and introduced a Bayesian learning rule to optimize the architecture parameters posterior distributions. We have theoretically demonstrated that the proposed framework can enhance the exploration for differentiable NAS and implicitly impose regularization on the Hessian norm to improve the stability. The above properties show that reformulating differentiable NAS as distribution learning is a promising direction. In addition, with leveraging the training-free proxies, our BaLeNAS can select more competitive architectures from the optimized distributions instead of applying *argmax* on the mean to get the the discrete architecture, so that alleviate the discretization instability and enhance the performance. We operationalize the framework based on the common differentiable NAS baseline, DARTS, and experimental results on NAS benchmark datasets and the common DARTS search space have verified the proposed framework's effectiveness.

Although BaLeNAS improves the differentiable NAS baseline by large margins, it computational consumption and memory consumption are similar with DARTS where our BaLeNAS is built on. Further questions include how to further decrease the computational and memory cost and also eliminate the *depth gap* existing between architecture search and evaluation in differentiable NAS [10].

References

[1] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. In *ICLR*, 2021. 2, 3, 5, 6, 7

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 549–558, 2018. 1, 2

[3] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017. 3

[4] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622, 2015. 3

[5] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018. 4

[6] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019. 1

[7] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *ICLR*, 2021. 2, 3

[8] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *ICML*, 2020. 1, 2, 4, 5, 8

[9] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Drnas: Dirichlet neural architecture search. In *ICLR*, 2021. 2, 4, 5, 6, 7, 8

[10] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1294–1303, 2019. 7, 8

[11] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In *Advances in Neural Information Processing Systems*, pages 6642–6652, 2019. 1

[12] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yunchao Dai, Xiaojun Chang, Tom Drummond, Hongdong Li, and Zongyuan Ge. Hierarchical Neural Architecture Search for Deep Stereo Matching. In *NeurIPS*, 2020. 1

[13] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Tom Drummond, Hongdong Li, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. In *NeurIPS*, 2020. 1

[14] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019. 2, 8

[15] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3681–3690, 2019. 6

[16] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2019. 1, 6, 7

[17] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *ICLR*, 2020. 2, 5, 11

[18] Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011. 3

[19] Minghao Guo, Zhao Zhong, Wei Wu, Dahua Lin, and Junjie Yan. Irlas: Inverse reinforcement learning for architecture search. In *CVPR*, 2019. 1

[20] 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. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019. 1

[21] Mohammad Khan and Wu Lin. Conjugate-computation variational inference: Converting variational inference in non-conjugate models to inferences in conjugate models. In *Artificial Intelligence and Statistics*, pages 878–887. PMLR, 2017. 4

[22] Mohammad Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam. In *International Conference on Machine Learning*, pages 2611–2620, 2018. 2, 4

[23] Mohammad Emtiyaz Khan and Haavard Rue. Learning-algorithms from bayesian principles. *arXiv preprint arXiv:2002.10778*, 2020. 2, 4

[24] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019. 3, 5

[25] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 1986–1995, 2020. 1

[26] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. *CoRR*, abs/2103.12424, 2021. 1

[27] Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 8607–8617. Computer Vision Foundation / IEEE, 2021. 1

[28] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *UAI*, 2019. 1, 6, 7

[29] Xiang Li, Chen Lin, Chuming Li, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Improving one-shot nas by suppressing the posterior fading. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, pages 13836–13845, 2020. 2, 4

[30] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*, 2019. 2, 7, 8

[31] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*, 2019. 1, 2, 4, 5, 6, 7

[32] Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. *arXiv preprint arXiv:2006.04647*, 2020. 2, 3

[33] Xiangming Meng, Roman Bachmann, and Mohammad Emtiyaz Khan. Training binary neural networks using the bayesian learning rule. *arXiv preprint arXiv:2002.10778*, 2020. 2

[34] Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik. Xnas: Neural architecture search with expert advice. In *Advances in Neural Information Processing Systems*, pages 1975–1985, 2019. 7

[35] Kazuki Osawa, Siddharth Swaroop, Mohammad Emtiyaz E Khan, Anirudh Jain, Runa Eschenhagen, Richard E Turner, and Rio Yokota. Practical deep learning with bayesian principles. In *Advances in neural information processing systems*, pages 4287–4299, 2019. 2, 4

[36] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, pages 4092–4101, 2018. 1, 2, 6

[37] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *AAAI*, 2019. 1

[38] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *arXiv preprint arXiv:2006.02903*, 2020. 1

[39] Yao Shu, Wei Wang, and Shaofeng Cai. Understanding architectures learnt by cell-based neural architecture search. In *International Conference on Learning Representations*, 2020. 2, 4, 8

[40] 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. 11

[41] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019. 1

[42] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33, 2020. 3, 5

[43] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020. 3, 5

[44] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. In *ICLR*, 2021. 1, 2, 3

[45] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *ICLR*, 2019. 1, 7

[46] 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. 7

[47] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *ICML*, pages 7105–7114, 2019. 5, 11

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

[49] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020. 1, 2, 3, 5, 8

[50] Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *ICLR*, 2020. 2, 5, 11

[51] Miao Zhang, Huiqi Li, Shirui Pan, Taoping Liu, and Steven Su. One-shot neural architecture search via novelty driven sampling. In *International Joint Conference on Artificial Intelligence*, 2020. 2, 4, 8

[52] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial distribution learning for effective neural architecture search. In *International Conference on Computer Vision (ICCV)*, 2019. 7

[53] Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. Bayesnas: A bayesian approach for neural architecture search. In *International Conference on Machine Learning*, pages 7603–7613, 2019. 7

[54] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 1