DRNAS: DIRICHLET NEURAL ARCHITECTURE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper proposes a novel differentiable architecture search method by formulating it into a distribution learning problem. We treat the continuously relaxed architecture mixing weight as random variables, modeled by Dirichlet distribution. With recently developed pathwise derivatives, the Dirichlet parameters can be easily optimized with gradient-based optimizer in an end-to-end manner. This formulation improves the generalization ability and induces stochasticity that naturally encourages exploration in the search space. Furthermore, to alleviate the large memory consumption of differentiable NAS, we propose a simple yet effective progressive learning scheme that enables searching directly on large-scale tasks, eliminating the gap between search and evaluation phases. Extensive experiments demonstrate the effectiveness of our method. Specifically, we obtain a test error of 2.46% for CIFAR-10, 23.7% for ImageNet under the mobile setting. On NAS-Bench-201, we also achieve state-of-the-art results on all three datasets and provide insights for the effective design of neural architecture search algorithms.

1 INTRODUCTION

Recently, Neural Architecture Search (NAS) has attracted lots of attentions for its potential to democratize deep learning. For a practical end-to-end deep learning platform, NAS plays a crucial role in discovering task-specific architecture depending on users' configurations (e.g. dataset, evaluation metric, etc.). Pioneers in this field develop prototypes based on reinforcement learning (Zoph & Le, 2017), evolutionary algorithms (Real et al., 2019) and Bayesian optimization (Liu et al., 2018). These works usually incur large computation overheads, which make them impractical to use. More recent algorithms significantly reduce the search cost including one-shot methods (Pham et al., 2018; Bender et al., 2018), a continuous relaxation of the space (Liu et al., 2019) and network morphisms (Cai et al., 2018). In particular, Liu et al. (2019) proposes a differentiable NAS framework - DARTS, converting the categorical operation selection problem into learning a continuous architecture mixing weight. They formulate a bi-level optimization objective allowing the architecture search to be efficiently performed by a gradient-based optimizer.

While current differentiable NAS methods achieve encouraging results, they still have shortcomings that hinder their real-world applications. Firstly, several works have cast doubt on the stability and generalization of these differentiable NAS methods (Chen & Hsieh, 2020; Zela et al., 2020a). They discover that directly optimizing the architecture mixing weight is prone to overfitting the validation set and often leads to distorted structures, e.g. searched architectures dominated by parameter-free operations. Secondly, there exist disparities between the search and evaluation phases, where proxy tasks are usually employed during search with smaller datasets or shallower and narrower networks, due to the large memory consumption of differentiable NAS.

In this paper, we propose an effective approach that addresses the aforementioned shortcomings named Dirichlet Neural Architecture Search (DrNAS). Inspired by the fact that directly optimizing the architecture mixing weight is equivalent to performing point estimation (MLE/MAP) from a probabilistic perspective, we formulate the differentiable NAS as a distribution learning problem instead, which naturally induces stochasticity and encourages exploration. Making use of the probability simplex property of the Dirichlet samples, DrNAS models the architecture mixing weight as random variables sampled from a parameterized Dirichlet distribution. Optimizing the Dirichlet objective can thus be done efficiently in an end-to-end fashion, by employing the pathwise derivative

estimators to compute the gradient of the distribution (Martin Jankowiak, 2018). A straightforward optimization, however, turns out to be problematic due to the uncontrolled variance of the Dirichlet, i.e. too much variance leads to training instability and too little variance suffers from insufficient exploration. In light of that, we apply an additional distance regularizer directly on the Dirichlet concentration parameter to strike a balance between the exploration and the exploitation. We further derive a theoretical bound showing that the constrained distributional objective promotes stability and generalization of architecture search by implicitly controlling the Hessian of the validation error.

Furthermore, to enable a direct search on large-scale tasks, we propose a progressive architecture learning scheme, eliminating the gap between the search and evaluation phases. Based on partial channel connection (Xu et al., 2020), we maintain a task-specific super-network of the same depth and number of channels as the evaluation phase throughout searching. To prevent loss of information and instability induced by partial connection, we divide the search phase into multiple stages and progressively increase the channel fraction via network transformation (Chen et al., 2016). Meanwhile, we prune the operation space according to the learnt distribution to maintain the memory efficiency.

We conduct extensive experiments on different datasets and search spaces to demonstrate DrNAS's effectiveness. Based on the DARTS search space (Liu et al., 2019), we achieve an average error rate of 2.46% on CIFAR-10, which ranks top amongst NAS methods. Furthermore, DrNAS achieves superior performance on large-scale tasks such as ImageNet. It obtains a top-1/5 error of 23.7%/7.1%, surpassing the previous state-of-the-art (24.0%/7.3%) under the mobile setting. On NAS-Bench-201 (Dong & Yang, 2020), we also set new state-of-the-art results on all three datasets with low variance.

2 The Proposed Approach

In this section, we first briefly review differentiable NAS setups and generalize the formulation to motivate distribution learning. We then layout our proposed DrNAS and describe its optimization in section 2.2. In section 2.3, we provide a generalization result by showing that our method implicitly regularizes the Hessian norm over the architecture parameter. The progressive architecture learning method that enables direct search is then described in section 2.4.

2.1 Preliminaries: Differentiable Architecture Search

Cell-Based Search Space The cell-based search space is constructed by replications of normal and reduction cells (Zoph et al., 2018; Liu et al., 2019). A normal cell keeps the spatial resolution while a reduction cell halves it but doubles the number of channels. Every cell is represented by a DAG with N nodes and E edges, where every node represents a latent representation $\mathbf{x}^i$ and every edge (i,j) is associated with an operations $o^{(i,j)}$ (e.g. *max pooling* or *convolution*) selected from a predefined candidate space $\mathcal{O}$. The output of a node is a summation of all input flows, i.e. $\mathbf{x}^j = \sum_{i<j} o^{(i,j)}(\mathbf{x}^i)$, and a concatenation of intermediate node outputs, i.e. $concat(\mathbf{x}^2, ..., \mathbf{x}^{N-1})$, composes the cell output, where the first two input nodes $\mathbf{x}^0$ and $\mathbf{x}^1$ are fixed to be the outputs of previous two cells.

Gradient-Based Search via Continuous Relaxation To enable gradient-based optimization, Liu et al. (2019) apply a continuous relaxation to the discrete space. Concretely, the information passed from node i to node j is computed by a weighted sum of all operations alone the edge, forming a mixed-operation $\hat{o}^{(i,j)}(x) = \sum_{o\in\mathcal{O}} \theta_o^{(i,j)} o(x)$. The operation mixing weight $\theta^{(i,j)}$ is defined over the probability simplex and its magnitude represents the strength of each operation. Therefore, the architecture search can be cast as selecting the operation associated with the highest mixing weight for each edge. To prevent abuse of terminology, we refer to θ as the architecture/operation mixing weight, and concentration parameter β in DrNAS as the architecture parameter throughout the paper.

Bilevel-Optimization with Simplex Constraints With continuous relaxation, the network weight w and operation mixing weight θ can be jointly optimized by solving a constraint bi-level optimization problem:

$$\min_{\theta} \ \mathcal{L}_{val}(w^*, \theta) \ \text{ s.t. } \ w^* = \arg\min_{w} \ \mathcal{L}_{train}(w, \theta), \quad \sum_{o=1}^{|\mathcal{O}|} \theta_o^{(i,j)} = 1, \ \forall\,(i,j), \ i<j, \tag{1}$$

where the simplex constraint $\sum_{o=1}^{|\mathcal{O}|} \theta_o^{(i,j)} = 1$ can be either solved explicitly via Lagrangian function (Li et al., 2020), or eliminated by substitution method (e.g. $\theta = Softmax(\alpha), \alpha \in \mathcal{R}^{|\mathcal{O}| \times |E|}$) (Liu et al., 2019). In the next section we describe how this generalized formulation motivates our method.

2.2 Differentiable Architecture Search as Distribution Learning

Learning a Distribution over Operation Mixing Weight Previous differentiable architecture search methods view the operation mixing weight θ as learnable parameters that can be directly optimized (Liu et al., 2019; Xu et al., 2020; Li et al., 2020). This has been shown to cause θ to overfit the validation set and thus induce large generalization error (Zela et al., 2020a;b; Chen & Hsieh, 2020). We recognize that this treatment is equivalent to performing point estimation (e.g. MLE/MAP) of θ in probabilistic view, which is inherently prone to overfitting (Bishop, 2016; Gelman et al., 2004). Furthermore, directly optimizing θ lacks sufficient exploration in the search space, and thus cause the search algorithm to commit to suboptimal paths in the DAG that converges faster at the beginning but plateaus quickly (Shu et al., 2020).

Based on these insights, we formulate the differentiable architecture search as a distribution learning problem. The operation mixing weight θ is treated as random variables sampled from a learnable distribution. Formally, let $q(\theta|\beta)$ denotes the distribution of θ parameterized by β. The bi-level objective is then given by:

$$\min_{\beta} E_{q(\theta|\beta)}\left[\mathcal{L}_{val}(w^*, \theta)\right] + \lambda d(\beta, \hat{\beta}) \ \text{ s.t. } \ w^* = \arg\min_{w} \ \mathcal{L}_{train}(w, \theta). \tag{2}$$

Since θ lies on the probability simplex, we select Dirichlet distribution to model its behavior, i.e. $q(\theta|\beta) \sim Dir(\beta)$, where β represents the Dirichlet concentration parameter. Dirichlet distribution is a widely used distribution over the probability simplex (Joo et al., 2019; David M. Blei, 2003), and it enjoys nice properties that enables gradient-based training (Martin Jankowiak, 2018).

The concentration parameter β controls the sampling behavior of Dirichlet distribution and is crucial in balancing the exploration and exploitation during the search phase. When $\beta_o \ll 1$ for most $o = 1 \sim |\mathcal{O}|$, Dirichlet tends to produce sparse samples with high variance, reducing the training stability; when $\beta_o \gg 1$ for most $o = 1 \sim |\mathcal{O}|$, the samples will be dense with low variance, leading to insufficient exploration. Therefore, we add a penalty term in the objective (2) to regularize the distance between β and the anchor $\hat{\beta} = 1$, which corresponds to the symmetric Dirichlet. In section 2.3, we also derive a theoretical bound showing that our formulation additionally promotes stability and generalization of the architecture search by implicitly regularizing the Hessian of validation loss w.r.t. architecture parameters.

Learning Dirichlet Parameters via Pathwise Derivative Estimator Optimizing objective (2) with gradient-based methods requires back-propagation through stochastic nodes of Dirichlet samples. The commonly used reparameterization trick does not apply to Dirichlet distribution, therefore we approximate the gradient of Dirichlet samples via pathwise derivative estimators (Martin Jankowiak, 2018)

$$\frac{d\theta_i}{d\beta_j} = -\frac{\frac{\partial F_{Beta}}{\partial \beta_j}(\theta_j|\beta_j, \beta_{tot} - \beta_j)}{f_{Beta}(\theta_j|\beta_j, \beta_{tot} - \beta_j)} \times \left(\frac{\delta_{ij} - \theta_i}{1 - \theta_j}\right) i, j = 1, ..., |\mathcal{O}|, \tag{3}$$

where F_{Beta} and f_{Beta} denote the CDF and PDF of beta distribution respectively, δ_{ij} is the indicator function, and β_{tot} is the sum of concentrations. F_{Beta} is the iregularised incomplete beta function, for which its gradient can be computed by simple numerical approximation. We refer to (Martin Jankowiak, 2018) for the complete derivations.

Joint Optimization of Model Weight and Architecture Parameter With pathwise derivative estimator, the model weight w and concentration β can be jointly optimized with gradient descent. Concretely, we draw a sample $\theta \sim Dir(\beta)$ for every forward pass, and the gradients can be obtained easily through backpropagation. Following DARTS (Liu et al., 2019), we approximate w^* in the lower level objective of equation 2 with one step of gradient descent, and run alternative updates between w^* and β.

Selecting the Best Architecture At the end of the search phase, a learnt distribution of operation mixing weight is obtained. We then select the best operation for each edge by the most likely operation in expectation:

$$o^{(i,j)} = \arg\max_{o \in \mathcal{O}} E_{q(\theta_o^{(i,j)}|\beta^{(i,j)})}\left[\theta_o^{(i,j)}\right]. \tag{4}$$

In the Dirichlet case, the expectation term is simply the Dirichlet mean $\frac{\beta_o^{(i,j)}}{\sum_{o'} \beta_{o'}^{(i,j)}}$. Note that under the distribution learning framework, we are able to sample a wide range of architectures from the learnt distribution. This property alone has many potentials. For example, in practical settings where both accuracy and latency are concerned, the learnt distribution can be used to find architectures under resource restrictions in a post search phase. We leave these extensions to future work.

2.3 Implicit Regularization on Hessian

It has been observed that the generalization error of differentiable NAS is highly related to the dominant eigenvalue of the Hessian of validation loss w.r.t. architecture parameter. Several recent works report that the large dominant eigenvalue of $\nabla_\alpha^2 \tilde{\mathcal{L}}_{val}(w, \alpha)$ in DARTS results in poor generalization performance (Zela et al., 2020a; Chen & Hsieh, 2020). Our objective (2) is the Lagrangian function of the following constraint objective:

$$\min_{\beta} E_{q(\theta|\beta)}\left[\mathcal{L}_{val}(w^*, \theta)\right] \text{ s.t. } w^* = \arg\min_{w} \mathcal{L}_{train}(w, \theta)\ ,\ d(\beta, \hat{\beta}) \leq \delta, \tag{5}$$

Here we derive an approximated lower bound based on (5), which demonstrates that our method implicitly controls this Hessian matrix.

Proposition 1 *Let $d(\beta, \hat{\beta}) = \|\beta - \hat{\beta}\|_2 \leq \delta$ and $\hat{\beta} = 1$ in the bi-level formulation (5). If $\nabla_\mu^2 \tilde{\mathcal{L}}_{val}(w^*, \mu)$ is Positive Semi-definite, the upper-level objective can be approximated bounded by:*

$$E_{q(\theta|\beta)}(\mathcal{L}_{val}(w, \theta)) \gtrsim \tilde{\mathcal{L}}_{val}(w^*, \mu) + \frac{1}{2}\left(\frac{1}{1+\delta}\left(1 - \frac{2}{|\mathcal{O}|}\right) + \frac{1}{|\mathcal{O}|}\frac{1}{1+\delta}\right) tr\left(\nabla_\mu^2 \tilde{\mathcal{L}}_{val}(w^*, \mu)\right) \tag{6}$$

with:

$$\tilde{\mathcal{L}}_{val}(w^*, \mu) = \mathcal{L}_{val}(w^*, Softmax(\mu)),\ \ \mu_o = \log \beta_o - \frac{1}{|\mathcal{O}|}\sum_{o'} \log \beta_{o'},\ \ o = 1, \ldots, |\mathcal{O}|.$$

This proposition is driven by the Laplacian approximation to the Dirichlet distribution (MacKay, 1998; Akash Srivastava, 2017). The lower bound (6) indicates that minimizing the expected validation loss controls the trace norm of the Hessian matrix. Empirically, we observe that DrNAS always maintains the dominant eigenvalue of Hessian at a low level (Appendix A.2). The detailed proof can be found in Appendix A.1.

2.4 Progressive Architecture Learning

The GPU memory consumption of differentiable NAS methods grows linearly with the size of operation candidate space. Therefore, they usually use a easier proxy task such as training with a smaller dataset, or searching with fewer layers and number of channels (Cai et al., 2019). For instance, the architecture search is performed on 8 cells and 16 initial channels in DARTS (Liu et al., 2019). But during evaluation, the network has 20 cells and 36 initial channels. Such gap makes it hard to derive an optimal architecture for the target task (Cai et al., 2019).

PC-DARTS (Xu et al., 2020) proposes a partial channel connection to reduce the memory overheads of differentiable NAS, where they only send a random subset of channels to the mixed-operation while directly bypassing the rest channels in a shortcut. However, their method causes loss of information and makes the selection of operation unstable since the sampled subsets may vary widely across iterations. This drawback is amplified when combining with the proposed method since we learn the architecture distribution from Dirichlet samples, which already injects certain stochasticity. As shown in Table 1, when directly applying partial channel connection with distribution learning, the

test accuracy of the searched architecture decreases over 3% and 18% on CIFAR-10 and CIFAR-100 respectively if we send only 1/8 channels to the mixed-operation.

To alleviate such information loss and instability problem while being memory-efficient, we propose a progressive learning scheme which gradually increases the fraction of channels that are forwarded to the mixed-operation and meanwhile prunes the operation space based on the learnt distribution. We split the search process into consecutive stages and construct a task-specific super-network with the same depth and number of channels as the evaluation phase at the initial stage. Then after each stage, we increase the partial channel fraction, which means that the super-network in the next stage will be wider, i.e. have more convolution channels, and in turn preserve more information. This is achieved by enlarging every convolution weight with a random mapping function similar to Net2Net (Chen et al., 2016). The mapping function $g : \{1, 2, \dots, q\} \to \{1, 2, \dots, n\}$ with $q > n$ is defined as

$$g(j) = \begin{cases} j & j \leq n \\ \text{random sample from } \{1, 2, \dots, n\} & j > n \end{cases} \tag{7}$$

To widen layer l, we replace its convolution weight $\mathbf{W}^{(l)} \in \mathbb{R}^{Out \times In \times H \times W}$ with a new weight $\mathbf{U}^{(l)}$.

$$\mathbf{U}^{(l)}_{o,i,h,w} = \mathbf{W}^{(l)}_{g(o),g(i),h,w}, \tag{8}$$

where Out, In, H, W denote the number of output and input channels, filter height and width respectively. Intuitively, we copy $\mathbf{W}^{(l)}$ directly into $\mathbf{U}^{(l)}$ and fulfill the rest part by choosing randomly as defined in g. Unlike Net2Net, we do not divide $\mathbf{U}^{(l)}$ by a replication factor here because the information flow on each edge has the same scale no matter the partial fraction is. After widening the super-network, we reduce the operation space by pruning out less important operations according to the Dirichlet concentration parameter β learnt from the previous stage, maintaining a consistent memory consumption. As illustrated in Table 1, the proposed progressive architecture learning scheme effectively discovers high accuracy architectures and retains a low GPU memory overhead.

3 Discussions and Relationship to Prior Work

Early methods in NAS usually include a full training and evaluation procedure every iteration as the inner loop to guide the consecutive search (Zoph & Le, 2017; Zoph et al., 2018; Real et al., 2019). Consequently, their computational overheads are beyond acceptance for practical usage, especially on large-scale tasks.

Table 1: Test accuracy of the derived architectures when searching on NAS-Bench-201 with different partial channel fraction, where $1/K$ channels are sent to the mixed-operation.

CIFAR-10		
K	**Test Accuracy (%)**	**GPU Memory (MB)**
1	94.36 ± 0.00	2437
2	93.49 ± 0.28	1583
4	92.85 ± 0.35	1159
8	91.06 ± 0.00	949
Ours	94.36 ± 0.00	949
CIFAR-100		
K	**Test Accuracy (%)**	**GPU Memory (MB)**
1	73.51 ± 0.00	2439
2	68.48 ± 0.41	1583
4	66.68 ± 3.22	1161
8	55.11 ± 13.78	949
Ours	73.51 ± 0.00	949

Differentiable NAS Recently, many works are proposed to improve the efficiency of NAS (Pham et al., 2018; Cai et al., 2018; Liu et al., 2019; Bender et al., 2018). Amongst them, DARTS (Liu et al., 2019) proposes a differentiable NAS framework, which introduces a continuous architecture parameter that relaxes the discrete search space. Despite being efficient, DARTS only optimizes a single point on the simplex every search epoch, which has no guarantee to generalize well after the discretization during evaluation. So its stability and generalization have been widely challenged (Li & Talwalkar, 2019; Zela et al., 2020a; Chen & Hsieh, 2020). Following DARTS, SNAS (Xie et al., 2019) and GDAS (Dong & Yang, 2019) leverage the gumbel-softmax trick to learn the exact architecture parameter. However, their reparameterization is motivated from reinforcement learning perspective, which is an approximation with softmax rather than an architecture distribution. Besides, their methods require tuning of temperature schedule (Yan et al., 2017; Caglar Gulcehre, 2017). GDAS linearly decreases the temperature from 10 to 1 while SNAS anneals it from 1 to 0.03. In comparison, the proposed method can automatically learn the architecture distribution without the requirement of handcrafted scheduling. BayesNAS (Zhou et al., 2019) applies Bayesian Learning in NAS. Specifically, they cast NAS as model compression problem and use Bayes Neural Network as the super-network, which is difficult to optimize and requires oversimplified approximation. While our method considers the stochasticity in architecture mixing weight, as it is directly related to the generalization of differentiable NAS algorithms (Zela et al., 2020a; Chen & Hsieh, 2020).

Memory overhead When dealing with the large memory consumption of differentiable NAS, previous works mainly restrain the number of paths sampled during the search phase. For instance, ProxylessNAS (Cai et al., 2019) employs binary gates and samples two paths every search epoch. PARSEC (Casale et al., 2019) samples discrete architectures according to a categorical distribution to save memory. Similarly, GDAS (Dong & Yang, 2019) and DSNAS (Hu et al., 2020) both enforce a discrete constraint after the gumbel-softmax reparametrization. However, such discretization manifests premature convergence and cause search instability (Zela et al., 2020b; Zhang et al., 2020). Our experiments in section 4.3 also empirically demonstrate this phenomenon. As an alternative, PC-DARTS (Xu et al., 2020) proposes a partial channel connection, where only a portion of channels is sent to the mixed-operation. However, partial connection can cause loss of information as shown in section 2.4 and PC-DARTS searches on a shallower network with less channels, suffering the search and evaluation gap. Our solution, by progressively pruning the operation space and meanwhile widening the network, searches in a task-specific manner and achieves superior accuracy on challenging datasets like ImageNet (+2.8% over BayesNAS, +2.3% over GDAS, +2.3% over PARSEC, +2.0% over DSNAS, +1.2% over ProxylessNAS, and +0.5% over PC-DARTS).

4 Experiments

4.1 Results on CIFAR-10

Architecture Space For both search and evaluation phases, we stack 20 cells to compose the network and set the initial channel number as 36. We place the reduction cells at the 1/3 and 2/3 of the network and each cell consists of $N = 6$ nodes.

Search Settings We equally divide the 50K training images into two parts, one is used for optimizing the network weights by momentum SGD and the other for learning the Dirichlet architecture distribution by an Adam optimizer. Since Dirichlet concentration β must be positive, we apply the shifted exponential linear mapping $\beta = \text{ELU}(\eta) + 1$ and optimize over η instead. We use l_2 norm to constrain the distance between η and the anchor $\hat{\eta} = 0$. The η is initialized by standard Gaussian with scale 0.001, and λ in (2) is set to 0.001. The ablation study in Appendix A.3 reveals the effectiveness of our anchor regularizer, and DrNAS is insensitive to a wide range of λ. These settings are consistent for all experiments. For progressive architecture learning, the whole search process consists of 2 stages, each with 25 iterations. In the first stage, we set the partial channel parameter K as 6 to fit the super-network into a single GTX 1080Ti GPU with 11GB memory, i.e. only 1/6 features are sampled on each edge. For the second stage, we prune half candidates and meanwhile widen the network twice, i.e. the operation space size reduces from 8 to 4 and K becomes 3.

Retrain Settings The evaluation phase uses the entire 50K training set to train the network from scratch for 600 epochs. The network weight is optimized by an SGD optimizer with a cosine annealing learning rate initialized as 0.025, a momentum of 0.9, and a weight decay of 3×10^{-4}. To allow a fair comparison with previous work, we also employ cutout regularization with length 16, drop-path (Zoph et al., 2018) with probability 0.3 and an auxiliary tower of weight 0.4.

Results Table 2 summarizes the performance of DrNAS compared with other popular NAS methods, and we also visualize the searched cells in Appendix A.2. DrNAS achieves a test error of 2.46%, ranking among the top of recent NAS results. ProxylessNAS is the only method that achieves lower test error than us, but it searches on a different space with a much longer search time and has larger model size. We also perform experiments to assign proper credit to the two parts of our proposed algorithm, i.e. Dirichlet architecture distribution and progressive learning scheme. When searching on a proxy task with 8 stacked cells and 16 initial channels as the convention (Liu et al., 2019; Xu et al., 2020), we achieve a test error of 2.54% that surpasses most baselines. Our progressive learning algorithm eliminates the gap between the proxy and target tasks, which further reduces the test error. Consequently, both of the two parts contribute a lot to our performance gains.

4.2 Results on ImageNet

Architecture Space The network architecture for ImageNet is slightly different from that for CIFAR-10 in that we stack 14 cells and set the initial channel number as 48. We also first downscale

Table 2: Comparison with state-of-the-art image classifiers on CIFAR-10.

Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	Search Method
DenseNet-BC (Huang et al., 2017)*	3.46	25.6	-	manual
NASNet-A (Zoph et al., 2018)	2.65	3.3	2000	RL
AmoebaNet-A (Real et al., 2019)	3.34 ± 0.06	3.2	3150	evolution
AmoebaNet-B (Real et al., 2019)	2.55 ± 0.05	2.8	3150	evolution
PNAS (Liu et al., 2018)*	3.41 ± 0.09	3.2	225	SMBO
ENAS (Pham et al., 2018)	2.89	4.6	0.5	RL
DARTS (1st) (Liu et al., 2019)	3.00 ± 0.14	3.3	0.4	gradient
DARTS (2nd) (Liu et al., 2019)	2.76 ± 0.09	3.3	1.0	gradient
SNAS (moderate) (Xie et al., 2019)	2.85 ± 0.02	2.8	1.5	gradient
GDAS (Dong & Yang, 2019)	2.93	3.4	0.3	gradient
BayesNAS (Zhou et al., 2019)	2.81 ± 0.04	3.4	0.2	gradient
ProxylessNAS (Cai et al., 2019)†	2.08	5.7	4.0	gradient
PARSEC (Casale et al., 2019)	2.81 ± 0.03	3.7	1	gradient
P-DARTS (Chen et al., 2019)	2.50	3.4	0.3	gradient
PC-DARTS (Xu et al., 2020)	2.57 ± 0.07	3.6	0.1	gradient
SDARTS-ADV (Chen & Hsieh, 2020)	2.61 ± 0.02	3.3	1.3	gradient
GAEA + PC-DARTS (Li et al., 2020)	2.50 ± 0.06	3.7	0.1	gradient
DrNAS (without progressive learning)	2.54 ± 0.03	4.0	0.4‡	gradient
DrNAS	2.46 ± 0.03	4.1	0.6‡	gradient

* Obtained without cutout augmentation.
† Obtained on a different space with PyramidNet (Han et al., 2017) as the backbone.
‡ Recorded on a single GTX 1080Ti GPU.

the spatial resolution from 224×224 to 28×28 with three convolution layers of stride 2 following previous works (Xu et al., 2020; Chen et al., 2019). The other settings are the same with section 4.1.

Search Settings Following PC-DARTS (Xu et al., 2020), we randomly sample 10% and 2.5% images from the 1.3M training set to alternatively learn network weight and Dirichlet architecture distribution by a momentum SGD and an Adam optimizer respectively. We use 8 RTX 2080 Ti GPUs for both search and evaluation, and the setup of progressive pruning is the same with that on CIFAR-10, i.e. 2 stages with operation space size shrinking from 8 to 4, and the partial channel K reduces from 6 to 3.

Retrain Settings For architecture evaluation, we train the network for 250 epochs by an SGD optimizer with a momentum of 0.9, a weight decay of 3×10^{-5}, and a linearly decayed learning rate initialized as 0.5. We also use label smoothing and an auxiliary tower of weight 0.4 during training. The learning rate warm-up is employed for the first 5 epochs following previous works (Chen et al., 2019; Xu et al., 2020).

Results As shown in Table 3, we achieve a top-1/5 test error of 23.7%/7.1%, outperforming all compared baselines and achieving state-of-the-art performance in the ImageNet mobile setting. The searched cells are visualized in Appendix A.2. Similar to section 4.1, we also report the result achieved with 8 cells and 16 initial channels, which is a common setup for the proxy task on ImageNet (Xu et al., 2020). The obtained 24.2% top-1 accuracy is already highly competitive, which demonstrates the effectiveness of the architecture distribution learning on large-scale tasks. Then our progressive learning scheme further increases the top-1/5 accuracy for 0.5%/0.2%. Therefore, learning in a task-specific manner is essential to discover better architectures.

4.3 Results on NAS-Bench-201

Recently, some researchers doubt that the expert knowledge applied to the evaluation protocol plays an important role in the impressive results achieved by leading NAS methods (Yang et al., 2020; Li & Talwalkar, 2019). So to further verify the effectiveness of DrNAS, we perform experiments on NAS-Bench-201 (Dong & Yang, 2020), where architecture performance can be directly obtained by querying in the database. NAS-Bench-201 provides support for 3 dataset (CIFAR-10, CIFAR-100, ImageNet-16-120 (Chrabaszcz et al., 2017)) and has a unified cell-based search space containing 15,625 architectures. We refer to their paper (Dong & Yang, 2020) for details of the space. Our experiments are performed in a task-specific manner, i.e. the search and evaluation are based on the same dataset. The hyperparameters for all compared methods are set as their default and for DrNAS,

Table 3: Comparison with state-of-the-art image classifiers on ImageNet in the mobile setting.

Architecture	**Test Error(%)**		**Params**	**Search Cost**	**Search**
	top-1	top-5	**(M)**	**(GPU days)**	**Method**
Inception-v1 (Szegedy et al., 2015)	30.1	10.1	6.6	-	manual
MobileNet (Howard et al., 2017)	29.4	10.5	4.2	-	manual
ShuffleNet 2× (v1) (Zhang et al., 2018)	26.4	10.2	~ 5	-	manual
ShuffleNet 2× (v2) (Ma et al., 2018)	25.1	-	~ 5	-	manual
NASNet-A (Zoph et al., 2018)	26.0	8.4	5.3	2000	RL
AmoebaNet-C (Real et al., 2019)	24.3	7.6	6.4	3150	evolution
PNAS (Liu et al., 2018)	25.8	8.1	5.1	225	SMBO
MnasNet-92 (Tan et al., 2019)	25.2	8.0	4.4	-	RL
DARTS (2nd) (Liu et al., 2019)	26.7	8.7	4.7	4.0	gradient
SNAS (mild) (Xie et al., 2019)	27.3	9.2	4.3	1.5	gradient
GDAS (Dong & Yang, 2019)	26.0	8.5	5.3	0.3	gradient
BayesNAS (Zhou et al., 2019)	26.5	8.9	3.9	0.2	gradient
DSNAS (Hu et al., 2020)†	25.7	8.1	-	-	gradient
ProxylessNAS (GPU) (Cai et al., 2019)†	24.9	7.5	7.1	8.3	gradient
PARSEC (Casale et al., 2019)	26.0	8.4	5.6	1	gradient
P-DARTS (CIFAR-10) (Chen et al., 2019)	24.4	7.4	4.9	0.3	gradient
P-DARTS (CIFAR-100) (Chen et al., 2019)	24.7	7.5	5.1	0.3	gradient
PC-DARTS (CIFAR-10) (Xu et al., 2020)	25.1	7.8	5.3	0.1	gradient
PC-DARTS (ImageNet) (Xu et al., 2020)†	24.2	7.3	5.3	3.8	gradient
GAEA + PC-DARTS (Li et al., 2020)†	24.0	7.3	5.6	3.8	gradient
DrNAS (without progressive learning)†	24.2	7.3	5.2	3.9	gradient
DrNAS†	23.7	7.1	5.7	4.6	gradient

† The architecture is searched on ImageNet, otherwise it is searched on CIFAR-10 or CIFAR-100.

we use the same search settings with section 4.1. We run every method 4 independent times with different random seeds and report the mean and standard deviation in Table 4.

As shown, we achieve the best accuracy on all 3 datasets. On CIFAR-100, we even achieve the global optimal. Specifically, DrNAS outperforms DARTS, GDAS, DSNAS, PC-DARTS, and SNAS by 103.8%, 35.9%, 30.4%, 6.4%, and 4.3% on average. We notice that the two methods (GDAS and DSNAS) that enforce a discrete constraint, i.e. only sample a single path every search iteration, perform undesirable especially on CIFAR-100. In comparison, SNAS, employing a similar Gumbel-softmax trick but without the discretization, performs much better. Consequently, a discrete constraint during search can reduce the GPU memory consumption but empirically suffers instability. In comparison, we develop the progressive learning scheme on top of the architecture distribution learning, enjoying both memory efficiency and strong search performance.

Table 4: Comparison with state-of-the-art NAS methods on NAS-Bench-201.

Method	**CIFAR-10**		**CIFAR-100**		**ImageNet-16-120**	
	validation	test	validation	test	validation	test
ResNet (He et al., 2016)	90.83	93.97	70.42	70.86	44.53	43.63
Random (baseline)	90.93 ± 0.36	93.70 ± 0.36	70.60 ± 1.37	70.65 ± 1.38	42.92 ± 2.00	42.96 ± 2.15
RSPS (Li & Talwalkar, 2019)	84.16 ± 1.69	87.66 ± 1.69	45.78 ± 6.33	46.60 ± 6.57	31.09 ± 5.65	30.78 ± 6.12
Reinforce (Zoph et al., 2018)	91.09 ± 0.37	93.85 ± 0.37	70.05 ± 1.67	70.17 ± 1.61	43.04 ± 2.18	43.16 ± 2.28
ENAS (Pham et al., 2018)	39.77 ± 0.00	54.30 ± 0.00	10.23 ± 0.12	10.62 ± 0.27	16.43 ± 0.00	16.32 ± 0.00
DARTS (1st) (Liu et al., 2019)	39.77 ± 0.00	54.30 ± 0.00	38.57 ± 0.00	38.97 ± 0.00	18.87 ± 0.00	18.41 ± 0.00
DARTS (2nd) (Liu et al., 2019)	39.77 ± 0.00	54.30 ± 0.00	38.57 ± 0.00	38.97 ± 0.00	18.87 ± 0.00	18.41 ± 0.00
GDAS (Dong & Yang, 2019)	90.01 ± 0.46	93.23 ± 0.23	24.05 ± 8.12	24.20 ± 8.08	40.66 ± 0.00	41.02 ± 0.00
SNAS (Xie et al., 2019)	90.10 ± 1.04	92.77 ± 0.83	69.69 ± 2.39	69.34 ± 1.98	42.84 ± 1.79	43.16 ± 2.64
DSNAS (Hu et al., 2020)	89.66 ± 0.29	93.08 ± 0.13	30.87 ± 16.40	31.01 ± 16.38	40.61 ± 0.09	41.07 ± 0.09
PC-DARTS (Xu et al., 2020)	89.96 ± 0.15	93.41 ± 0.30	67.12 ± 0.39	67.48 ± 0.89	40.83 ± 0.08	41.31 ± 0.22
DrNAS	$\mathbf{91.55 \pm 0.00}$	$\mathbf{94.36 \pm 0.00}$	$\mathbf{73.49 \pm 0.00}$	$\mathbf{73.51 \pm 0.00}$	$\mathbf{46.37 \pm 0.00}$	$\mathbf{46.34 \pm 0.00}$
optimal	91.61	94.37	73.49	73.51	46.77	47.31

5 Conclusion

In this paper, we propose Dirichlet Neural Architecture Search (DrNAS). We formulate the differentiable NAS as a constraint distribution learning problem, which explicitly models the stochasticity in the architecture mixing weight and balances exploration and exploitation in the search space. The proposed method can be optimized efficiently via gradient-based algorithm, and possesses theoretical benefit to improve the generalization ability. Furthermore, we propose a progressive learning scheme to eliminate the search and evaluation gap. DrNAS consistently achieves strong performance across several image classification tasks, which reveals its potential to play a crucial role in future end-to-end deep learning platform.

REFERENCES

Charles Sutton Akash Srivastava. Autoencoding variational inference for topic models. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1703.01488.

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 550–559, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/bender18a.html.

Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer New York, 2016.

Yoshua Bengio Caglar Gulcehre, Sarath Chandar. Memory augmented neural networks with wormhole connections, 2017.

Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *AAAI*, 2018.

Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HylVB3AqYm.

Francesco Paolo Casale, Jonathan Gordon, and Nicolo Fusi. Probabilistic neural architecture search, 2019.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*, 2016. URL http://arxiv.org/abs/1511.05641.

Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization, 2020.

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*, pp. 1294–1303, 2019.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets, 2017.

Michael I. Jordan David M. Blei, Andrew Y. Ng. Latent dirichlet allocation. *The Journal of Machine Learning Research*, Mar 2003. ISSN 1532-4435. doi: 10.1162/jmlr.2003.3.4-5.993. URL http://dx.doi.org/10.1109/CVPR.2017.243.

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1761–1770, 2019.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=HJxyZkBKDr.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2nd ed. edition, 2004.

Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. doi: 10.1109/cvpr.2017.668. URL http://dx.doi.org/10.1109/CVPR.2017.668.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pp. 770–778, 06 2016. doi: 10.1109/CVPR.2016.90.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. Dsnas: Direct neural architecture search without parameter retraining, 2020.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. doi: 10.1109/cvpr.2017.243. URL http://dx.doi.org/10.1109/CVPR.2017.243.

Weonyoung Joo, Wonsung Lee, Sungrae Park, , and Il-Chul Moon. Dirichlet variational autoencoder, 2019. URL https://openreview.net/forum?id=rkgsvoA9K7.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search, 2019.

Liam Li, Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Geometry-aware gradient algorithms for neural architecture search, 2020.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *Lecture Notes in Computer Science*, pp. 19–35, 2018. ISSN 1611-3349. doi: 10.1007/978-3-030-01246-5_2. URL http://dx.doi.org/10.1007/978-3-030-01246-5_2.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *The European Conference on Computer Vision (ECCV)*, September 2018.

David J. C. MacKay. Choice of basis for laplace approximation. *Machine Language*, October 1998. doi: 10.1023/A:1007558615313. URL https://link.springer.com/article/10.1023/A:1007558615313.

Fritz Obermeyer Martin Jankowiak. Pathwise derivatives beyond the reparameterization trick. In *International Conference on Machine Learning*, 2018. URL https://arxiv.org/abs/1806.01851.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4095–4104, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/pham18a.html.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:4780–4789, Jul 2019. ISSN 2159-5399. doi: 10.1609/aaai.v33i01.33014780. URL http://dx.doi.org/10.1609/aaai.v33i01.33014780.

Yao Shu, Wei Wang, and Shaofeng Cai. Understanding architectures learnt by cell-based neural architecture search. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJxH22EKPS.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. URL http://arxiv.org/abs/1409.4842.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rylqooRqK7.

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 *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJlS634tPr.

Shiyang Yan, Jeremy S. Smith, Wenjin Lu, and Bailing Zhang. Hierarchical multi-scale attention networks for action recognition, 2017.

Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. Nas evaluation is frustratingly hard. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HygrdpVKvr.

Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020a. URL https://openreview.net/forum?id=H1gDNyrKDS.

Arber Zela, Julien Siems, and Frank Hutter. NAS-BENCH-1SHOT1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=SJx9ngStPH.

M. Zhang, H. Li, S. Pan, X. Chang, and S. Su. Overcoming multi-model forgetting in one-shot nas with diversity maximization. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7806–7815, 2020.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. Bayesnas: A bayesian approach for neural architecture search. In *ICML*, pp. 7603–7613, 2019. URL http://proceedings.mlr.press/v97/zhou19e.html.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. 2017. URL https://arxiv.org/abs/1611.01578.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. doi: 10.1109/cvpr.2018.00907. URL http://dx.doi.org/10.1109/CVPR.2018.00907.