IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 30, 2022, accepted June 13, 2022, date of publication June 21, 2022, date of current version June 24, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3185095

RARTS: An Efficient First-Order Relaxed Architecture Search Method

FANGHUI XUE, YINGYONG QI, AND JACK XIN
Department of Mathematics, University of California at Irvine, Irvine, CA 92697, USA
Corresponding author: Fanghui Xue (fanghuix@uci.edu)

This work was supported in part by NSF under Grant DMS-1854434 and Grant DMS-1952644, and in part by the Qualcomm Faculty Award.

ABSTRACT Differentiable architecture search (DARTS) is an effective method for data-driven neural network design based on solving a bilevel optimization problem. Despite its success in many architecture search tasks, there are still some concerns about the accuracy of first-order DARTS and the efficiency of the second-order DARTS. In this paper, we formulate a single level alternative and a relaxed architecture search (RARTS) method that utilizes the whole dataset in architecture learning via both data and network splitting, without involving mixed second derivatives of the corresponding loss functions like DARTS. In our formulation of network splitting, two networks with different but related weights cooperate in search of a shared architecture. The advantage of RARTS over DARTS is justified by a convergence theorem and an analytically solvable model. Moreover, RARTS outperforms DARTS and its variants in accuracy and search efficiency, as shown in adequate experimental results. For the task of searching topological architecture, i.e., the edges and the operations, RARTS obtains a higher accuracy and 60% reduction of computational cost than second-order DARTS on CIFAR-10. RARTS continues to out-perform DARTS upon transfer to ImageNet and is on par with recent variants of DARTS even though our innovation is purely on the training algorithm without modifying search space. For the task of searching width, i.e., the number of channels in convolutional layers, RARTS also outperforms the traditional network pruning benchmarks. Further experiments on the public architecture search benchmark like NATS-Bench also support the preeminence of RARTS.

INDEX TERMS Convolutional neural networks, neural architecture search, differentiable architecture search, network compression.

I. INTRODUCTION

Neural Architecture Search (NAS) is an automated machine learning technique to design an optimal neural network architecture by searching its building blocks of deep neural networks from a collection of candidate structures and operations. Although NAS has achieved many successes in several computer vision tasks [1]–[6], the search process demands huge computational resources. The current search times have come down considerably from as many as 2000 GPU days in early NAS [2], thanks to subsequent studies [7]–[13] among others. Differentiable Architecture Search (DARTS) [14] is an appealing method that avoids searching over all possible combinations by relaxing the categorical architecture indicators to continuous parameters. The higher level architecture can be learned along with lower level weights via stochastic gradient descent by approximately solving a bilevel optimization problem. DARTS can be further sorted into first-order DARTS and second-order DARTS, in line with whether a mixed second derivative estimation of loss function is used or not.

Despite its search efficiency obtained from continuous relaxation, DARTS can still have some problems experimentally and theoretically. They are the efficiency problem with second-order DARTS, the convergence problem with first-order DARTS, and the *architecture collapse* problem (i.e., the selected architecture contains too many skip-connections) with both DARTS. Second-order DARTS takes much longer search time than first-order DARTS as it involves the mixed second derivatives. It has also been pointed out that second-order DARTS can have superposition effect [15], which means the approximation of the gradient of α is based on the approximation of the weight w one step ahead. This is believed to cause gradient errors and

The associate editor coordinating the review of this manuscript and approving it for publication was Joey Tianyi Zhou.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

failures in finding optimal architectures. Therefore, it is used less often in practice than first-order DARTS [15], [16]. However, first-order DARTS learns the architecture using half of the data only. Evidences are provided to show that it can result in incorrect limits and worse performance [14]. The experimental results also show that first-order DARTS (3.00% error) is less accurate than second-order DARTS (2.76% error) on the CIFAR-10 dataset [14], [17]. For the architecture collapse problem, typically such a bias in operation selection degrades the model performance. This problem has been observed by a few researchers [18], [19], who have tried to solve it by replacing some operations of the architecture.

In addition to the search for topological architectures, i.e., edges and operations of cells (building blocks) in some early NAS works and DARTS [2], [14], many NAS style methods have been developed to search for the width of a model, i.e., the number of channels in convolutional layers [16], [20]. Searching for width is supposed to be a way of channel pruning, which is a common tool for *network compression*, i.e., constructing slim networks from redundant ones [21]. Specifically, channel pruning can be formulated as an architecture search problem, via the setup of learnable channel scoring parameters [21]–[23] as architecture parameters. This is an elegant approach for compression without relying on channel magnitude (group ℓ_1 norm), which is used in previous regularization methods [24]. The previous way of setting up channel scoring parameters [21] utilizes the scale parameters of the batch normalization layers, yet they are not contained in many modern networks [25], [26]. Another challenge remains to be solved is to replace its plain gradient descent by the more accurate DARTS style algorithms.

Apart from the bilevel formulation of DARTS, a single level approach (SNAS) based on a differentiable loss and sampling has been proposed [27]. On CIFAR-10, SNAS is more accurate than the first-order DARTS yet with 50% more search time than the second-order DARTS. This inspires us to formulate a new single level method which is more efficient and accurate. Our *main contribution* is to introduce a novel *Relaxed Architecture Search (RARTS)* method based on single level optimization, and the computation of only the first-order partial derivatives of loss functions, for both topology and width search of architectures. Through both data and network splitting, the training objective (a relaxed Lagrangian function) of RARTS allows two networks with different but related weights to cooperate in the search of a shared architecture.

We have carried out both analytical and experimental studies below to show that RARTS achieves *better performance than first and second-order DARTS, with higher search efficiency than second-order DARTS* consistently:

- Compare RARTS with DARTS directly on the analytical model with quadratic loss functions, where the RARTS iterations approach the true global minimal point missed by the first-order DARTS, in a robust fashion. A convergence theorem is proved for RARTS based on descent of its Lagrangian function, and equilibrium equations are discovered for the limits.
- On the CIFAR-10 based search of topological architecture, the model found by RARTS obtains smaller size and higher test accuracy than that by the second-order DARTS with 65% search time saving. A hardware-aware search option via a latency penalty in the Lagrangian function helps control the model size. Upon transfer to ImageNet [28], [29], the model found by RARTS achieves better performance as well, compared with DARTS and its variants. Apart from the standard search space used in the DARTS paper, RARTS also beats DARTS on the public NAS benchmark of search spaces like NATS-Bench [30].
- For channel pruning of ResNet-164 [31] on CIFAR-10 and CIFAR-100 [17] with fixed pruning ratio (percentage of pruned channels), RARTS outperforms the differentiable pruning benchmarks: Network Slimming [21] and TAS [20]. Comparisons between DARTS and RARTS have also been made in a ℓ_1 regularized (unfixed ratio) pruning task, where RARTS achieves a high sparsity of 70% and exceeds DARTS in accuracy.

II. RELATED WORK

A. DIFFERENTIABLE ARCHITECTURE SEARCH

DARTS training relies on an iterative algorithm to solve a bilevel optimization problem [14], [32] which involves two loss functions computed via *data splitting* (splitting the dataset into two halves, i.e. training data and validation data):

$$\begin{aligned} &\min_{\alpha} L_{val}(w^*(\alpha), \alpha), \\ &\text{where} \quad w^*(\alpha) = \arg\min_{w} L_{train}(w, \alpha). \end{aligned} \tag{1}$$

Here w denotes the network weights, α is the architecture parameter, L_{train} and L_{val} are the loss functions computed on the training data D_{train} and the validation data D_{val}. Since many common datasets like CIFAR do not include the validation data, D_{train} and D_{val} are usually two non-overlapping halves of the original training data. We denote L_{train} and L_{val} by L_t and L_v to avoid any confusions with the meaning of the subscripts. D_t and D_v are defined similarly. DARTS has adopted data splitting because it is believed that joint training of both α and w via gradient descent on the whole dataset by minimizing the overall loss function:

$$L(w, \alpha) = L_t(w, \alpha) + L_v(w, \alpha) \tag{2}$$

can lead to overfitting [14], [15]. Therefore, DARTS searches for the architectures through a two-step differentiable algorithm which updates the network weights and the architecture parameters in an alternating way:

- update weight w by descending along $\nabla_w L_t(w, \alpha)$
- update architecture parameter α by descending along:

$$\nabla_\alpha L_v(w - \xi \nabla_w L_t(w, \alpha), \alpha)$$

where $\xi = 0$ ($\xi > 0$) gives the first or second-order approximation. The bilevel optimization problem also

arises in hyperparameter optimization and meta-learning, where a second-order algorithm and a convergence theorem on minimizers have been proposed in previous work [33] (Theorem 3.2), under the assumption that the α-minimization is solved exactly, and $w^t(\alpha)$ converges uniformly to $w(\alpha)$. However, the α-minimization of DARTS is approximated by gradient methods only, and hence the convergence of DARTS algorithm remains unknown theoretically.

We are aware of the fact that the first-order DARTS updates the architecture parameters on D_v by descending along $\nabla_\alpha L_v(w, \alpha)$, which means it merely uses half of the data to train α and might cause some convergence issues (see Fig. 1). MiLeNAS has developed a mixed-level solution, where the architecture parameters can be learned on $D = D_t \cup D_v$ via a first-order descending algorithm [15]:

$$\begin{aligned} w^{t+1} &= w^t - \eta_w^t \nabla_w L_t(w^t, \alpha^t) \\ \alpha^{t+1} &= \alpha^t - \eta_\alpha^t \left(\nabla_\alpha L_t(w^{t+1}, \alpha^t) + \lambda \nabla_\alpha L_v(w^{t+1}, \alpha^t)\right), \end{aligned} \quad (3)$$

We shall see that MiLeNAS is actually a constrained case of RARTS when our two network splits become identical. However, we point out that computing L_v using an identical network makes MiLeNAS still suffer from the same convergence issue in a later example (Section III-D). The second-order DARTS is observed to approximate the optimum better than first-order DARTS in a solvable model and through experiments, yet it requires computing the mixed derivative $\nabla^2_{\alpha,w} L_t$, at a considerable overhead. Searching by DARTS can also lead to the *architecture collapse* issue, meaning the selected architecture contains too many skip-connections. Typically such a bias in operation selection degrades the model performance. SNAS [27], FBNet [12], and GDAS [18] use differentiable Gumbel-Softmax to mimic one-hot encoding which implies exclusive competition and risks of unfair advantages [19]. This unfair dominance of skip-connections in DARTS has also been noted by Fair-DARTS [19], which has proposed a collaborative competition approach by making the architecture parameters independent, through replacing the softmax with sigmoid. They have further penalized the operations in the search space with probability close to $\frac{1}{2}$, i.e. a neutral and ambiguous selection. As these methods focus on replacing some operations or the loss function, it would be worthwhile to explore other solutions such as replacing the gradient-based DARTS search algorithm.

In addition to DARTS, many other differentiable methods for architecture search have been proposed, considering various aspects such as the search space, selection criterion, and training tricks. SNAS [27] has discussed it from a statistical perspective with however a minor to moderate performance improvement. The search efficiency has also been improved by sampling a portion of the search space during each update in training. A perturbation-based selection scheme has been proposed in [34], as the magnitude of architecture parameters is believed to be inadequate as a selection criterion. P-DARTS [35] has adopted operation dropout and regularization on skip-connections. From the procedure side to delay a quick short cut aggregation, it has also divided the search stage into multiple stages and progressively adds more depth than DARTS. PC-DARTS [36] samples a proportion of channels to reduce the bias of operation selection and enlarge the batch size as well. GDAS [18] searches the architecture with one operation sampled at a time. Other approaches apply differentiable methods on much larger search spaces with sampling techniques to save memory and avoid model transfer [7], [12]. We will see that these variants of the differentiable architecture search method are actually complementary to our approach that advances DARTS on the purely algorithmic side by mobilizing weights. Moreover, many works [7], [12], [16], [37]–[39] manage to balance latency with the performance of the model to enhance the efficiency of the model. Despite the broad use of differentiable methods in the works we have mentioned, one may wonder how DARTS and its variants beat random search. A detailed comparison in [40] has elaborated the advantage of DARTS in accuracy and efficiency compared with random search.

B. SEARCH FOR WIDTH AND CHANNEL PRUNING

Differentiable search method has contributed to a wide range of tasks other than topological architecture search. TAS [20] searches for the width of each layer, i. e. number of channels, by learning the optimal one from the aggregation of several candidate feature maps via a differentiable method and sampling. FasterSeg [16] searches for the cell operations and layer width, as well as the multi-resolution network path over the semantic segmentation task. These works of searching width are closely related to channel pruning, which means pruning redundant channels from the convolutional layers. Among numerous methods to prune redundant channels [24], [41]–[45], a classical approach is to apply group LASSO [46] on the weights to identify unimportant channels. The weights in each channel form one group, and the magnitude of each group is measured by ℓ_2 norm of its weights. The network is trained by minimizing a loss function penalized by the ℓ_1 norm of these magnitudes from all groups. The channels are pruned based on thresholding their norms. Selecting good thresholds as hyperparameters for different channels can be laborious for deep networks. On the other hand, channel selection is intrinsically a network architecture issue. It is debatable if thresholding by weight magnitudes is always meaningful [47].

Another approach of channel pruning [21], [22] involves assigning a channel scaling (scoring) factor to each channel, which is a learnable parameter independent of the weights. In the training process, the factors and the weights are learned jointly, and the channels with low scaling factors are pruned. After that, the optimal weights of the pruned network are adjusted by one more stage of fine-tuning. In terms of channel scaling factors, the channel pruning problem becomes a special case of neural architecture search. Besides this formulation, there are several pruning methods based on

NAS. AMC [37] has defined a reward function and pruned the channels via reinforcement learning. MetaPruning [48] generates the best pruned model and weights from a meta network.

III. METHODOLOGY

In this section, we introduce the RARTS formulation, its iterative algorithm and convergence properties. RARTS is different from all the differentiable algorithms we have mentioned, in that it puts forward a relaxed formulation of a single level problem which benefits from both data splitting and network splitting.

A. DATA SPLITTING AND NETWORK SPLITTING

As pointed out in DARTS [14] and MiLeNAS [15], when learning the architecture parameter α, splitting training and validation data should be taken into account to avoid overfitting. However, we have discussed that the bilevel formulation (1) and training algorithm of DARTS may lead to several issues: unknown convergence, low efficiency and the unfair selection of operations. Therefore, we follow the routine of train-validation data splitting, but want to formulate a single level problem, in contrast to DARTS and MiLeNAS. First, if we use (w, α), the pair of weight and architecture parameters in Eq. (2) to represent a network, what we propose to do is to further relax the network weights w via splitting a network copy denoted by (y, α). We call (y, α) and (w, α) the primary and auxiliary networks, which share the same architecture α and the same dimensions as weight tensors, but can have different weight initialization.

Next, a primary loss $L_v(y, \alpha)$ is computed with parameters (y, α) fed on data D_v, while an auxiliary loss $L_t(w, \alpha)$ is computed with parameters (w, α) fed on data D_t. Note that the computation of the auxiliary loss $L_t(w, \alpha)$ is the same as that of DARTS. The difference is that the primary loss is computed on the primary network (y, α), instead of (w, α). Now we present the single level objective of our relaxed architecture search (RARTS) framework. With a ℓ_2 penalty on the distance between w and y, the two loss functions are combined through the following relaxed Lagrangian $L = L(y, w, \alpha)$ of Eq. (2):

$$L := L_v(y, \alpha) + \lambda L_t(w, \alpha) + \frac{1}{2}\beta \|y - w\|_2^2, \tag{4}$$

where λ and β are hyperparameters controlling the penalty scale and the learning process. We will see in the search algorithm that the penalty term enables the two networks to exchange information and cooperate to search the architecture which they share together. This technique of splitting w and y is called *network splitting*, which is also inspired by some previous work [49]. In their work, splitting of variables is able to approximate a non-smooth minimization problem via an algorithm of combined closed-form solutions and gradient descent.

Since various NAS approaches discover architectures of inconsistent sizes or FLOPS, it has made the comparison through different methods unfair, because larger models are likely to have better performance but low efficiency. Many NAS methods have adopted latency as a model constraint [7], [16]. To control model size, we follow the technique of approximating the model latency with the sum of latency from all the operations [16], and add the approximated latency to the loss function as a penalty. Since each component of the latency tensor (denoted by Lat) is the latency amount associated with a candidate operation, the dimension of Lat is the same as that of α. Therefore, we provide an alternative objective which is penalized by the latency of the model:

$$\begin{aligned} L := {} & L_v(y, \alpha) + \lambda L_t(w, \alpha) + \frac{1}{2}\beta \|y - w\|_2^2 \\ & + \langle \text{Softmax}(\alpha), \text{Lat} \rangle, \end{aligned} \tag{5}$$

where the bracket is inner product.

B. RARTS ALGORITHM

We minimize the relaxed Lagrangian $L(y, w, \alpha)$ in (4) by iteration on the three variables in an alternating way to allow individual and flexible learning schedules for the three variables. Similar to Gauss-Seidel method in numerical linear algebra [50], we use updated variables immediately in each step and obtain the following three-step iteration:

$$\begin{aligned} w^{t+1} &= w^t - \eta_w^t \nabla_w L(y^t, w^t, \alpha^t) \\ y^{t+1} &= y^t - \eta_y^t \nabla_y L(y^t, w^{t+1}, \alpha^t) \\ \alpha^{t+1} &= \alpha^t - \eta_\alpha^t \nabla_\alpha L(y^{t+1}, w^{t+1}, \alpha^t). \end{aligned} \tag{6}$$

With explicit gradient $\nabla_{w,y}\|y - w\|_2^2$, we have:

$$\begin{aligned} w^{t+1} &= w^t - \lambda \eta_w^t \nabla_w L_t(w^t, \alpha^t) - \beta \eta_w^t (w^t - y^t) \\ y^{t+1} &= y^t - \eta_y^t \nabla_y L_v(y^t, \alpha^t) - \beta \eta_y^t (y^t - w^{t+1}) \\ \alpha^{t+1} &= \alpha^t - \lambda \eta_\alpha^t \nabla_\alpha L_t(w^{t+1}, \alpha^t) \\ &\quad - \eta_\alpha^t \nabla_\alpha L_v(y^{t+1}, \alpha^t). \end{aligned} \tag{7}$$

To minimize the Lagrangian (5), the first two steps are the same as Eq. (7) since the latency only depends on α. The third step becomes:

$$\begin{aligned} \alpha^{t+1} = {} & \alpha^t - \lambda \eta_\alpha^t \nabla_\alpha L_t(w^{t+1}, \alpha^t) \\ & - \eta_\alpha^t \nabla_\alpha L_v(y^{t+1}, \alpha^t) \\ & - \eta_\alpha^t \nabla_\alpha \langle \text{Softmax}(\alpha^t), \text{Lat} \rangle. \end{aligned} \tag{8}$$

Note that the update of α in Eq. (7) involves both L_t and L_v, which is similar to the second-order DARTS but *without the mixed second derivatives*. The first-order DARTS uses $\nabla_\alpha L_v$ only in this step. In the previous section, we have discussed the architecture collapse issue of DARTS, i.e., selecting to many skip-connections. A possible reason why DARTS may lead to architecture collapse is that its architecture parameters converge more quickly than the weights in the convolutional layers. That means, when DARTS selects architecture parameters, it tends to select skip-connection operations, since the convolutional layers are not trained well.

Algorithm 1 Relaxed Architecture Search (RARTS)

Input: the number of iterations N, the hyperparameters λ and β, a learning rate schedule $(\eta_w^t, \eta_u^t, \eta_\alpha^t)$, initialization of the weight parameters w^0, u^0 and the architecture parameters α^0.

Output: α^*, the architecture we want

Split the dataset D into two subsets D_p and D_a.

for $t = 0, 1, \ldots, N$ **do**

Compute L_p and L_a on D_p and D_a, respectively, and then compute L using Eq. (4)

Update the parameters via gradient descent:

$w^{t+1} = w^t - \eta_w^t \nabla_w L(y^t, w^t, \alpha^t)$

$y^{t+1} = y^t - \eta_y^t \nabla_y L(y^t, w^{t+1}, \alpha^t)$

$\alpha^{t+1} = \alpha^t - \eta_\alpha^t \nabla_\alpha L(u^{t+1}, w^{t+1}, \alpha^t)$

end

The fact that first-order DARTS has only used one of the two data splits to train the weights, makes the training of convolutional layers worse. For RARTS, we make use of both L_t and L_v to update the weight parameters w and y in the first two steps of Eq. (7). In the third step of Eq. (7), both L_t and L_v are also used to update the shared architecture α. In this way, the architecture is learned better, as more data are involved during training. If $y = w$ is enforced in Eq. (7) e.g. through a multiplier, RARTS essentially reduces to first-order MiLeNAS [15]. However, relaxing to $y \neq w$ has its advantages of having more generality and robustness as it is optimized on two networks with different but related weights. In contrast, MiLeNAS trains the network weigts on the training data D_t only, and suffers from the same convergence issue as first-order DARTS (Section III-D). We summarize the RARTS algorithm in Algorithm 1.

C. CONVERGENCE ANALYSIS

Suppose that L_t and L_v both satisfy Lipschitz gradient property, or there exist positive constants L_1 and L_2 such that $(z = (y, \alpha), z' = (y', \alpha'))$:

$$\|\nabla_z L_v(z) - \nabla_z L_v(z')\| \leq L_1 \|z - z'\|, \quad \forall (z, z'),$$

which implies:

$$L_v(z) - L_v(z') \leq \langle \nabla_z L_v(z'), (z - z') \rangle + \frac{L_1}{2} \|z - z'\|^2,$$

for any (z, z'); similarly $(\zeta = (w, \alpha), \zeta' = (w', \alpha'))$:

$$\|\nabla_\zeta L_t(\zeta) - \nabla_\zeta L_t(\zeta')\| \leq L_2 \|\zeta - \zeta'\|, \quad \forall (\zeta, \zeta'),$$

which implies:

$$L_t(\zeta) - L_t(\zeta') \leq \langle \nabla_\zeta L_t(\zeta'), (\zeta - \zeta') \rangle + \frac{L_2}{2} \|\zeta - \zeta'\|^2,$$

for any (ζ, ζ').

Theorem 1: Suppose that the loss functions L_t and L_v satisfy Lipschitz gradient property. If the learning rates η_w^t, η_y^t and η_α^t are small enough depending only on the Lipschitz constants as well as (λ, β), and approach nonzero limit at large t, the Lagrangian function $L(y, w, \alpha)$ is descending on the iterations of (7). If additionally the Lagrangian L is lower bounded and coercive (its boundedness implies that of its variables), the sequence (y^t, w^t, α^t) converges sub-sequentially to a critical point $(\bar{y}, \bar{w}, \bar{\alpha})$ of $L(y, w, \alpha)$ obeying the equilibrium equations:

$$\begin{aligned} &\lambda \nabla_w L_t(\bar{w}, \bar{\alpha}) + \beta(\bar{w} - \bar{y}) = 0, \\ &\nabla_y L_v(\bar{y}, \bar{\alpha}) + \beta(\bar{y} - \bar{w}) = 0, \\ &\lambda \nabla_\alpha L_t(\bar{w}, \bar{\alpha}) + \nabla_\alpha L_v(\bar{y}, \bar{\alpha}) = 0. \end{aligned} \tag{9}$$

If the loss is penalized by latency as in (5), the last equilibrium equation becomes:

$$\begin{aligned} &\lambda \nabla_\alpha L_t(\bar{w}, \bar{\alpha}) + \nabla_\alpha L_v(\bar{y}, \bar{\alpha}) \\ &\quad + \nabla_\alpha \langle \text{Softmax}(\bar{\alpha}), \text{Lat} \rangle = 0. \end{aligned} \tag{10}$$

Proof: We only need to prove for the loss (5) and the iterations (8), as the loss (4) is its special case when Lat $= 0$. We notice the latency penalty function $\langle \text{Softmax}(\alpha^t), \text{Lat} \rangle$ also satisfies the Lipschitz gradient property. This is because

$$\begin{aligned} \nabla_\alpha \text{Softmax}(\alpha^t) = &\ \text{diag}(\text{Softmax}(\alpha^t)) \\ &- \text{Softmax}(\alpha^t) \otimes (\text{Softmax}(\alpha^t))', \end{aligned}$$

and hence all the first and second derivatives of $\langle \text{Softmax}(\alpha^t), \text{Lat} \rangle$ are bounded uniformly regardless of α^t. Applying Lipschitz gradient inequalities on L_v and L_t, we have:

$$\begin{aligned} &L(y^{t+1}, w^{t+1}, \alpha^{t+1}) - L(y^t, w^t, \alpha^t) \\ &\quad = L_v(y^{t+1}, \alpha^{t+1}) + \lambda L_t(w^{t+1}, \alpha^{t+1}) \\ &\qquad + \frac{\beta}{2} \|y^{t+1} - w^{t+1}\|^2 + \langle \text{Softmax}(\alpha^{t+1}), \text{Lat} \rangle \\ &\qquad - L_v(y^t, \alpha^t) - \lambda L_t(w^t, \alpha^t) - \frac{\beta}{2} \|y^t - w^t\|^2 \\ &\qquad - \langle \text{Softmax}(\alpha^t), \text{Lat} \rangle \\ &\quad \leq \langle \nabla_{y,\alpha} L_v(y^t, \alpha^t), (y^{t+1} - y^t, \alpha^{t+1} - \alpha^t) \rangle \\ &\qquad + \frac{L_1}{2} \|(y^{t+1} - y^t, \alpha^{t+1} - \alpha^t)\|^2 \\ &\qquad + \lambda \langle \nabla_{w,\alpha} L_t(w^t, \alpha^t), (w^{t+1} - w^t, \alpha^{t+1} - \alpha^t) \rangle \\ &\qquad + \frac{L_2}{2} \|(w^{t+1} - w^t, \alpha^{t+1} - \alpha^t)\|^2 \\ &\qquad + \frac{\beta}{2} (\|y^{t+1} - w^{t+1}\|^2 - \|y^t - w^t\|^2) \\ &\qquad + \langle \nabla_\alpha \langle \text{Softmax}(\alpha^t), \text{Lat} \rangle, \alpha^{t+1} - \alpha^t \rangle \\ &\qquad + \frac{L_3}{2} \|\alpha^{t+1} - \alpha^t\|^2. \end{aligned}$$

Substituting for the (w, y)-gradients from the iterations (8), we continue:

$$\begin{aligned} &L(y^{t+1}, w^{t+1}, \alpha^{t+1}) - L(y^t, w^t, \alpha^t) \\ &\quad \leq -(\eta_y^t)^{-1} \\ &\qquad \cdot \langle y^{t+1} - y^t + \beta \eta_y^t (y^t - w^{t+1}), y^{t+1} - y^t \rangle \\ &\qquad + \langle \nabla_\alpha L_v(y^t, \alpha^t) + \lambda \nabla_\alpha L_t(w^t, \alpha^t), \alpha^{t+1} - \alpha^t \rangle \\ &\qquad + \lambda (-\lambda \eta_w^t)^{-1} \end{aligned}$$

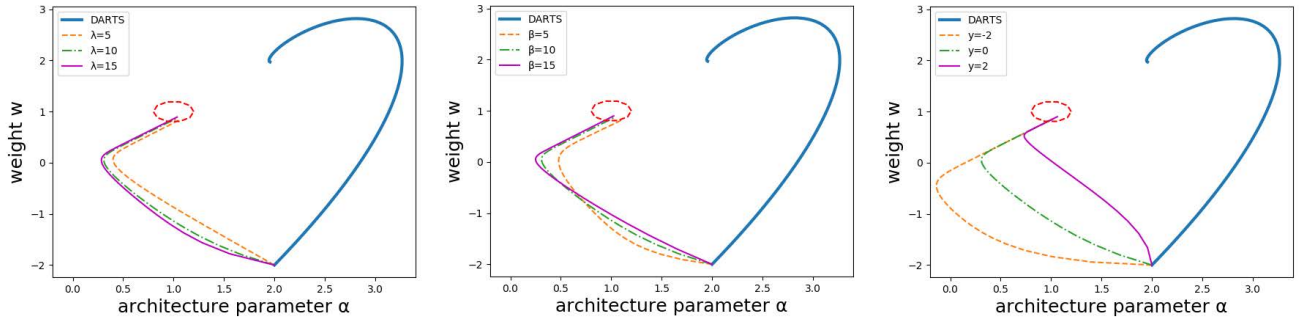


FIGURE 1. Learning trajectories of RARTS approach the global minimal point (1, 1) of the solvable model at suitable values of λ, β and y_0 ($\lambda = 10$ in middle/right subplots, $\beta = 10$ in left/right subplots, $y_0 = 0$ in left/middle subplots), compared with that of the baseline (first-order DARTS).

$$
\begin{aligned}
&\cdot \langle w^{t+1} - w^t + \beta\, \eta_w^t\, (w^t - y^t),\, w^{t+1} - w^t\rangle \\
&\quad + \frac{L_1}{2}\|y^{t+1} - y^t\|^2 + \frac{L_1 + L_2 + L_3}{2}\|\alpha^{t+1} - \alpha^t\|^2 \\
&\quad + \frac{L_2}{2}\|w^{t+1} - w^t\|^2 \\
&\quad + \frac{\beta}{2}(\|y^{t+1} - w^{t+1}\|^2 - \|y^t - w^t\|^2) \\
&\quad + \langle \nabla_\alpha \langle \mathrm{Softmax}(\alpha^t), \mathrm{Lat}\rangle, \alpha^{t+1} - \alpha^t\rangle \rangle \\
&= (-(\eta_y^t)^{-1} + L_1/2)\,\|y^{t+1} - y^t\|^2 \\
&\quad + (-(\eta_w^t)^{-1} + L_2/2)\,\|w^{t+1} - w^t\|^2 \\
&\quad - \beta\langle y^t - w^{t+1}, y^{t+1} - y^t\rangle \\
&\quad - \beta\langle w^t - y^t, w^{t+1} - w^t\rangle \\
&\quad + \frac{\beta}{2}(\|y^{t+1} - w^{t+1}\|^2 - \|y^t - w^t\|^2) \\
&\quad + \langle \nabla_\alpha L_v(y^t, \alpha^t) + \lambda \nabla_\alpha L_t(w^t, \alpha^t), \alpha^{t+1} - \alpha^t\rangle \\
&\quad + \frac{L_1 + L_2 + L_3}{2}\|\alpha^{t+1} - \alpha^t\|^2 \\
&\quad + \langle \nabla_\alpha \langle \mathrm{Softmax}(\alpha^t), \mathrm{Lat}\rangle, \alpha^{t+1} - \alpha^t\rangle \rangle. \qquad (11)
\end{aligned}
$$

We note the following identity

$$
\begin{aligned}
&\|y^{t+1} - w^{t+1}\|^2 \\
&\quad = \|y^{t+1} - w^t + w^t - w^{t+1}\|^2 \\
&\quad = \|y^{t+1} - w^t\|^2 + 2\langle y^{t+1} - w^t, w^t - w^{t+1}\rangle \\
&\qquad + \|w^t - w^{t+1}\|^2,
\end{aligned}
$$

where

$$
\begin{aligned}
&\|y^{t+1} - w^t\|^2 \\
&\quad = \| - w^t + y^t - y^t + y^{t+1}\|^2 \\
&\quad = \|y^t - w^t\|^2 + 2\langle y^t - w^t, y^{t+1} - y^t\rangle + \|y^{t+1} - y^t\|^2.
\end{aligned}
$$

Upon substitution of the above in the right hand side of (11), we find that:

$$
\begin{aligned}
&L(y^{t+1}, w^{t+1}, \alpha^{t+1}) - L(y^t, w^t, \alpha^t) \\
&\quad \le (-(\eta_y^t)^{-1} + L_1/2 + \beta/2)\,\|y^{t+1} - y^t\|^2 \\
&\qquad + (-(\eta_w^t)^{-1} + L_2/2 \\
&\qquad + \beta/2)\,\|w^{t+1} - w^t\|^2 \\
&\qquad + \beta\langle w^{t+1} - w^t, y^{t+1} - y^t\rangle \\
&\qquad + \beta\langle y^{t+1} - y^t, w^t - w^{t+1}\rangle \\
&\qquad + \langle \nabla_\alpha L_v(y^t, \alpha^t) \\
&\qquad + \lambda \nabla_\alpha L_t(w^t, \alpha^t), \alpha^{t+1} - \alpha^t\rangle \\
&\qquad + \frac{L_1 + L_2 + L_3}{2}\|\alpha^{t+1} - \alpha^t\|^2 \\
&\qquad + \langle \nabla_\alpha \langle \mathrm{Softmax}(\alpha^t), \mathrm{Lat}\rangle, \alpha^{t+1} - \alpha^t\rangle \rangle.
\end{aligned}
$$

The β-terms cancel out. Substituting for the α-gradient from the iterations (8), we get:

$$
\begin{aligned}
&L(y^{t+1}, w^{t+1}, \alpha^{t+1}) - L(y^t, w^t, \alpha^t) \\
&\quad \le (-(\eta_y^t)^{-1} + L_1/2 + \beta/2)\,\|y^{t+1} - y^t\|^2 \\
&\qquad + (-(\eta_w^t)^{-1} + L_2/2 + \beta/2)\,\|w^{t+1} - w^t\|^2 \\
&\qquad + (-(\eta_\alpha^t)^{-1} + \frac{L_1 + L_2 + L_3}{2})\|\alpha^{t+1} - \alpha^t\|^2 \\
&\qquad + \langle \nabla_\alpha L_v(y^t, \alpha^t) - \nabla_\alpha L_v(y^{t+1}, \alpha^t), \alpha^{t+1} - \alpha^t\rangle \\
&\qquad + \lambda\langle \nabla_\alpha L_t(w^t, \alpha^t) - \nabla_\alpha L_t(w^{t+1}, \alpha^t), \alpha^{t+1} - \alpha^t\rangle
\end{aligned}
$$

where the last two inner product terms are upper bounded by:

$$
(1 + \lambda)\, L_4\, (\|y^t - y^{t+1}\| + \|w^t - w^{t+1}\|)\, \|\alpha^{t+1} - \alpha^t\|,
$$

for positive constant $L_4 := \max(L_1, L_2)$. It follows that:

$$
\begin{aligned}
&L(y^{t+1}, w^{t+1}, \alpha^{t+1}) - L(y^t, w^t, \alpha^t) \\
&\quad \le \left[-(\eta_y^t)^{-1} + \frac{L_1}{2} + \frac{\beta}{2} + (1+\lambda)\,\frac{L_4}{2}\right]\|y^{t+1} - y^t\|^2 \\
&\qquad + \left[-(\eta_w^t)^{-1} + \frac{L_2}{2} + \frac{\beta}{2} + (1+\lambda)\,\frac{L_4}{2}\right]\|w^{t+1} - w^t\|^2 \\
&\qquad + \left[-(\eta_\alpha^t)^{-1} + \frac{L_1 + L_2 + L_3}{2} + (1+\lambda)\,\frac{L_4}{2}\right] \\
&\qquad \cdot \|\alpha^{t+1} - \alpha^t\|^2. \qquad (12)
\end{aligned}
$$

If

$$
\begin{aligned}
\eta_y^t &< \frac{1}{2}\left[\frac{L_1}{2} + \frac{\beta}{2} + (1+\lambda)\,\frac{L_4}{2}\right]^{-1} := c_1, \\
\eta_w^t &< \frac{1}{2}\left[\frac{L_2}{2} + \frac{\beta}{2} + (1+\lambda)\,\frac{L_4}{2}\right]^{-1} := c_2, \\
\eta_\alpha^t &< \frac{1}{2}\left[\frac{L_1 + L_2 + L_3}{2} + (1+\lambda)\,\frac{L_4}{2}\right]^{-1} := c_3,
\end{aligned}
$$

L is descending along the sequence (y^t, w^t, α^t). For $c_4 = \frac{1}{2}\min\{c_1^{-1}, c_2^{-1}, c_3^{-1}\}$, it follows from (12) that:

$$c_4\|(y^{t+1}-y^t, w^{t+1}-w^t, \alpha^{t+1}-\alpha^t)\|^2 \leq L(y^t, w^t, \alpha^t) - L(y^{t+1}, w^{t+1}, \alpha^{t+1}) \to 0$$

as $t \to +\infty$, implying that

$$\lim_{t\to\infty} \|(y^{t+1}-y^t, w^{t+1}-w^t, \alpha^{t+1}-\alpha^t)\| = 0.$$

Since L is lower bounded and coercive, $\|(y^t, w^t, \alpha^t)\|$ are uniformly bounded in t. Let $(\eta_w^t, \eta_y^t, \eta_\alpha^t)$ tend to non-zero limit at large t. Then (y^t, w^t, α^t) sub-sequentially converges to a limit point $(\bar{y}, \bar{w}, \bar{\alpha})$ satisfying the equilibrium system (9) or (10). □

D. A SOLVABLE BILEVEL MODEL

We compare a few differentibale methods through an example [14] which has an analytical solution. Regardless of the latency penalty, we consider quadratic functions $L_v = \alpha w - 2\alpha + 1, L_t = w^2 - 2\alpha w + \alpha^2$ for the bilevel problem (1). Therefore, the solution to the inner level problem is:

$$w^*(\alpha) = \arg\min_w L_t(w, \alpha) = \alpha.$$

Then $L_v(w^*(\alpha), \alpha) = \alpha^2 - 2\alpha + 1$, and the global minimizer of this bilevel problem is $(w^*, \alpha^*) = (1, 1)$. However, the equilibrium equations of first-order DARTS is:

$$\nabla_w L_t(\bar{w}, \bar{\alpha}) = 0$$
$$\nabla_\alpha L_v(\bar{w}, \bar{\alpha}) = 0,$$

which gives a spurious equilibrium $(\bar{w}, \bar{\alpha}) = (2, 2)$. The equilibrium equations of first-order MiLeNAS is:

$$\nabla_w L_t(\bar{w}, \bar{\alpha}) = 0$$
$$\nabla_\alpha L_t(\bar{w}, \bar{\alpha}) + \lambda \nabla_\alpha L_v(\bar{w}, \bar{\alpha}) = 0,$$

which also results in the spurious equilibrium $(\bar{w}, \bar{\alpha}) = (2, 2)$.

On the other hand, RARTS can approximate the correct minimizer $(w^*, \alpha^*) = (1, 1)$ better. Note that both L_v and L_t satisfy the Lipschitz gradient property, which implies the descent of Lagrangian L by the proof of Theorem 1. If $\lambda > 1/2, \beta > 3/2, L$ is bounded and coercive, which follows from an eigenvalue analysis of linear system (7) and is observed in computation. Hence, Theorem 1 can be applied to this example, and the equilibrium system (9) reads:

$$\lambda(2\bar{w} - 2\bar{\alpha}) + \beta(\bar{w} - \bar{y}) = 0, \quad (13)$$
$$\bar{\alpha} + \beta(\bar{y} - \bar{w}) = 0, \quad (14)$$
$$\lambda(-2\bar{w} + 2\bar{\alpha}) + \bar{y} - 2 = 0. \quad (15)$$

Adding (13) and (14) gives: $\bar{w} = \frac{2\lambda-1}{2\lambda}\bar{\alpha}$, which together with (15) determines $(\bar{\alpha}, \bar{w}, \bar{y})$ uniquely: $(\bar{\alpha}, \bar{w}, \bar{y}) = (\frac{4\beta\lambda}{4\beta\lambda-\beta-2\lambda}, \frac{4\beta\lambda-2\beta}{4\beta\lambda-\beta-2\lambda}, \frac{4\beta\lambda-2\beta-4\lambda}{4\beta\lambda-\beta-2\lambda})$, if $4\beta\lambda - \beta - 2\lambda \neq 0$. At $\lambda = \beta = 15$, $(\bar{\alpha}, \bar{w}, \bar{y}) \approx (1.053, 1.018, 0.947)$ where *global convergence holds for the whole RARTS sequence*. The learning dynamics starting from $(\alpha_0, w_0, y_0) = (2, -2, y_0)$, is reproduced in Fig. 1, along with three learning curves from RARTS as the parameters (λ, β) and the initial value y_0 vary. In Fig. 1a, $\beta = 10, y_0 = 0$. In Fig. 1b, $\lambda = 10, y_0 = 0$. In Fig. 1c, $\lambda = \beta = 10$. In all experiments, the learning rates are fixed at 0.01. For a range of (λ, β) and y_0, we see that our learning curves enter a small circle around (1, 1), while first-order DARTS converges to the spurious point.

IV. EXPERIMENTS

We show by a series of experiments how RARTS works efficiently for different tasks: the search for topology and the search for width, on various datasets and search spaces.

A. SEARCH FOR TOPOLOGY

For the hyperparameters and settings like learning rate schedules, number of epochs for CIFAR-10 and the transfer learning technique for ImageNet, we follow those of DARTS [14]. We also consider the results on CIFAR-10 and CIFAR-100 for NATS-Bench [30], which is another benchmark search space.

1) COMPARISONS ON CIFAR-10

The CIFAR-10 dataset consists of 50,000 training images and 10,000 test images [17]. These 3-channel images of 32×32 resolutions are allocated to 10 object classes evenly. For the architecture search task on CIFAR-10, the D_t and D_v data we have used are random non-overlapping halves of the original training data, the same as DARTS. The settings for searching topology with RARTS follows those of DARTS. That is, batch size = 64, initial weight learning rate = 0.025, momentum = 0.9, weight decay = 0.0003, initial alpha learning rate = 0.0003, alpha weight decay = 0.001, epochs = 50. For the stage of training, batch size = 96, learning rate = 0.025, momentum = 0.9, weight decay = 0.0003 [14]. For each cell (either normal or reduction), 8 edges are selected, with 1 out of 8 candidate operations selected for each edge (see Fig. 2). Besides the standard ℓ_2 regularization of the weights, we also adopt the latency penalty. The latency regularization loss is *weighted* so that it is balanced with other loss terms. Typically, if we increase the latency weight, the model we find will be smaller in size. The latency term Lat for each operation is measured via PyTorch/TensorRT [16], and thus it depends on the devices we use. For the current search, the latency weight is 0.002 so that the model size is comparable to those in prior works. The final latency loss is the weighted sum of the latency from each operation, where the weights are the architecture parameters.

As shown in Table 1, the search cost of RARTS is 1.1 GPU days, far less than that of the second-order DARTS. The test error of RARTS is 2.65%, outperforming the 3.00% of the first-order DARTS and the 2.76% of the second-order DARTS. It should also be pointed out that the model found by RARTS has 3.2M parameters, which is smaller than the 3.3M model found by DARTS. Moreover, RARTS outperforms other recent differentiable methods in accuracy and search cost at comparable model size. We also notice that the variance of RARTS performance is lower than that of

TABLE 1. Comparison of DARTS, RARTS and other methods on CIFAR-10 based network search. DARTS-1/2 stands for DARTS 1st/2nd-order, SNAS-Mi/Mo stands for SNAS plus mild/moderate constraints. Note that faster search times also depend on speed and memory capacity of local machines used. The V100 column indicates whether the model is trained on high-end Tesla V100 GPUs or not. Each run of our experiment is conducted on a single GTX 1080 Ti GPU. The numbers in the parentheses indicate the search GPU days of DARTS on our machine. Average of 5 runs. ⋄ These runs are conducted on our machine.

Method	Test Error (%)	Para. (M)	V100	Search GPU Days
Random Baseline [14]	3.29 ± 0.15	3.2	✗	4
AmoebaNet-B [10]	2.55 ± 0.05	2.8	✗	3150
SNAS-Mi [27]	2.98	2.9	✗	1.5
SNAS-Mo [27]	2.85 ± 0.02	2.8	✗	1.5
DARTS-1 [14]	3.00 ± 0.14	3.3	✗	1.5 (0.7)
DARTS-2 [14]	2.76 ± 0.09	3.3	✗	4 (3.1)
GDAS [18]	2.82	2.5	✓	0.2
ProxylessNAS [7]	2.08	5.7	✓	4.0
FairDARTS [19]	2.54 ± 0.05	3.3	✓	0.4
FairDARTS [19]	2.94 ± 0.05 ⋄	3.2	✗	0.3
P-DARTS [35]	2.50	3.4	✓	0.3
PC-DARTS [36]	2.57 ± 0.07	3.6	✓	0.1
PC-DARTS [36]	2.71 ⋄	2.9	✗	0.1
MiLeNAS [15]	2.80 ± 0.04	2.9	✓	0.3
MiLeNAS [15]	2.51 ± 0.11	3.9	✓	0.3
RARTS	2.65 ± 0.07	3.2	✗	1.1

TABLE 2. Comparison of the latency for the models found under different hyperparameters. The setting of batch size = 64, learning rate = 3×10^{-4}, weight decay = 1×10^{-3} is consistent with the settings of DARTS and other DARTS variants, and is selected to be our baseline setting.

Latency Weight	Batch Size	Learning Rate	Weight Decay	Latency (ms)
2×10^{-3}	64	3×10^{-4}	1×10^{-3}	21.7
2×10^{-2}	64	3×10^{-4}	1×10^{-3}	12.4
2×10^{-4}	64	3×10^{-4}	1×10^{-3}	23.4
2×10^{-3}	64	3×10^{-4}	1×10^{-3}	21.7
2×10^{-3}	64	3×10^{-3}	1×10^{-3}	21.0
2×10^{-3}	64	3×10^{-5}	1×10^{-3}	23.1
2×10^{-3}	64	3×10^{-4}	1×10^{-3}	21.7
2×10^{-3}	64	3×10^{-4}	1×10^{-4}	21.3
2×10^{-3}	64	3×10^{-4}	1×10^{-2}	22.9
2×10^{-3}	64	3×10^{-4}	1×10^{-3}	21.7
2×10^{-3}	32	3×10^{-4}	1×10^{-3}	20.1
2×10^{-3}	16	3×10^{-4}	1×10^{-3}	16.5

DARTS. RARTS has also *arrested architecture collapse* and only selected one skip-connection, as shown in Fig. 2. We are aware that different values of hyperparameters in the RARTS search stage may impact the latency of the models found by RARTS. Table 2 has listed the latency of several models with different hyperparameters. Here we use the baseline setting of latency weight $= 2 \times 10^{-3}$, batch size $= 64$, learning rate $= 3 \times 10^{-4}$, weight decay $= 1 \times 10^{-3}$. We change the value of one hyperparameter and keep the others the same during each experiment, so that we can see how sensitive the resulting latency is to a specific hyperparameter. First, the result shows that a small batch size of 16 can impact the model's latency, whereas a batch size of 32 or 64 can lead to similar latency. This is a positive phenomenon, since we prefer larger batch size as it requires less training time. Among the other hyperparameters, it is clear that the only factor that could cause a significant difference is the latency weight. A latency weight of 2×10^{-2} is so large that its model has only 60% latency compared with the baseline. The model's latency is not sensitive to the other hyperparameters, as the latency is around 22.0, and varies within 10% only. This finding is beneficial, since we can fix the latency level via fixing the latency weight and find the model with the best accuracy among the models of similar latency level via tuning the other hyperparameters.

2) COMPARISONS ON ImageNet

ImageNet [28], [29] is composed of over 1.2 million training images and 5,000 test images from 1,000 object classes. The architecture which is built of the cells learned on CIFAR-10 is transferred to be learned on ImageNet-1000, producing the results in Table 3. Even if our experiments are performed on a GTX 1080 Ti whose maximum memory allows only a batch size of 128, our 25.9% error rate outperforms those of DARTS and SNAS (batch size 128), and is also comparable to those of GDAS (batch size 128) and MiLeNAS. MiLeNAS among

TABLE 3. Transfer to ImageNet: test error comparison of DARTS, RARTS and other methods on local machines resp. The V100 column indicates whether the model is trained on high-end Tesla V100 GPUs or not. The larger GPU memory can support larger batch size, which leads to better accuracy and training efficiency on ImageNet. The Direct column indicates if the model is searched directly on ImageNet without transfer-learning. The direct search tends to be more accurate but costs more computational resources.

Method	Top-1 (%)	Top-5 (%)	Parameters (M)	V100	Direct
SNAS [27]	27.3	9.2	4.3	✗	✗
DARTS [14]	26.7	8.7	4.7	✗	✗
GDAS [18]	26.0	8.5	5.3	✓	✗
ProxylessNAS [7]	24.9	7.5	7.1	✓	✓
FairDARTS [19]	24.9	7.5	4.8	✓	✗
FairDARTS [19]	24.4	7.4	4.3	✓	✓
P-DARTS [35]	24.4	7.4	4.9	✓	✗
PC-DARTS [36]	25.1	7.8	5.3	✓	✗
PC-DARTS [36]	24.2	7.3	5.3	✓	✓
MiLeNAS [15]	25.4	7.9	4.9	✓	✗
RARTS	25.9	8.3	4.7	✗	✗

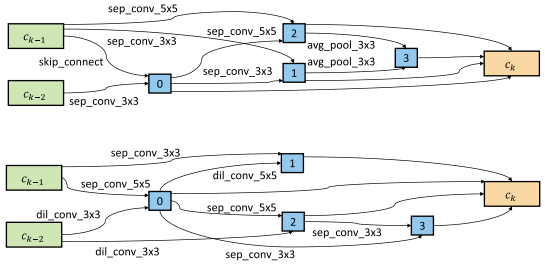


FIGURE 2. The architecture of the normal (top) and reduction (bottom) cells found by RARTS. This architecture contains only one skip connection. The last four edges are simply concatenated together to construct the next cell. So there is no search along these edges, following the convention of DARTS [14].

some other algorithms in Table 2 have been implemented on Tesla V100 with batch size 1024, a much higher end hardware than that in our experiments. This partly explains its lower accuracy occurrence (2.80) on CIFAR-10 but higher accuracy after transfer to ImageNet. Typically ImageNet is trained better on larger GPU's because of the larger batch size. ProxylessNAS has obtained high accuracy on both CIFAR-10 and ImageNet, but their models are much larger than the other methods. It has avoided transfer learning as the training cost is reduced via path sampling. Inheriting the building blocks from DARTS and ProxylessNAS, FairDARTS has penalized the neutral (close to 0.5) architecture parameters, but its high accuracy also benefits from the relaxation on the search space. Their normal cells contain less than 8 operations since the operations with architecture parameters lower than a preset threshold are eliminated. This explains their smaller model size and comparable accuracy. P-DARTS has devised a progressive method to increase the depth of search. Their work shows that deeper cells have better capability of representation, which is also an improvement on the search space. PC-DARTS as a sampling method has achieved the least searching cost and can be trained directly on ImageNet. These methods are complementary to our work which is purely on the differentiable search algorithm without modifying the search space of DARTS.

TABLE 4. Test errors of DARTS vs. RARTS on NATS-Bench search space. The results of DARTS on NATS-Bench are from [30]. Ratio = the number of skip-connections over the number of total operations in the discovered architecture.

Dataset	Method	Error (%)	Ratio (%)
CIFAR-10	DARTS-1	40.16	100
	DARTS-2	34.62	100
	RARTS	**11.48**	**0**
CIFAR-100	DARTS-1	38.74	38.9
	DARTS-2	39.51	38.9
	RARTS	**32.37**	**0**

3) COMPARISONS ON NATS-BENCH

For NATS-Bench, one has to search a block of 6 nodes from the search space of 5 different operations, including zero, skip-connection, 3×3 average pooling, 1×1 convolution or 3×3 convolution [30]. Therefore, it includes 15,625 different candidate architectures and any DARTS style methods can be adapted easily to its search space. NATS-Bench has measured each architecture's performance under the same training settings, and hence fair comparisons can be made between the discovered architectures since no further evaluation is needed on the local machines. In our experiments, we set batch size = 64, initial weight learning rate = 0.025, momentum = 0.9, weight decay = 0.0005, initial alpha learning rate = 0.0003, alpha weight decay = 0.001, number of epochs = 100. Table 4 presents the search results of DARTS vs. RARTS on NATS- Bench. RARTS has surpassed both DARTS-1 and DARTS-2 in accuracy by more than 20% on CIFAR-10 and 6% on CIFAR-100. Besides its success in accuracy, RARTS has totally escaped from the architecture collapse issue, i. e., the architectures found by RARTS from NATS-Bench contain no skip-connections. On the contrary, both architectures found by DARTS-1 and DARTS-2 contain 100% and 38.9% (average of 3 runs) skip-connections on CIFAR-10 and CIFAR-100, respectively. It is clear that too many skip-connections resulting in architecture collapse will impact the performance of the models greatly.

B. SEARCH FOR WIDTH

To search the width of the architecture (number of channels in convolutional layers), we follow the settings of Network Slimming [21], by introducing scoring parameters α to measure channel importance. Denote the original feature map by $F_{i,j}$ and define the new feature map $\tilde{F}_{i,j} = \alpha_{i,j} F_{i,j}$, where (i, j) are the layer and channel indices. Multiplying a channel of output feature map by α is equivalent to multiplying the convolutional kernels connecting to this output feature map by the same α. We prune a channel if the corresponding α is 0 or very small. The α_{ij}'s are learnable architecture parameters independent of channel weights, and hence is considered to have similar roles to the architecture parameters in the case of searching topological architecture.

Although such treatment of scoring parameters is much like that in Network Slimming [21], we point out that the single level formulation of RARTS and the training algorithm to learn those scoring parameters are novel. The first difference is that Network Slimming trains both weight and architecture parameters on the whole (training and validation) data, unlike DARTS or RARTS, without using either dataset splitting or network splitting. Another key difference between RARTS pruning and Network Slimming is in the search algorithm, i.e., Network Slimming trains the weights and the architecture jointly in one step, while RARTS trains them in a three-step iteration. Moreover, Network Slimming has used batch normalization weights as the scoring parameters. We point out that we could still define such a set of learnable architecture parameters α, even if the batch normalization operation is not contained in the architecture.

We also compare RARTS with TAS [20], which is another width search method based on differentiable NAS, relying on both continuous relaxation via feature maps of various sizes and *model distillation*. The first difference is on how the channel scoring parameters are applied to the feature maps. For TAS, the channel parameters are treated as probabilities of candidate feature maps, smoothed by Gumbel-Softmax. Then a subset of feature maps is sampled to alleviate the high memory costs. RARTS is much simpler in its formulation, as it is a dot product of the channel parameters with the filter to be pruned. The second key difference is the use of a training technique called Knowledge Distillation (KD) [51] by TAS to improve accuracy. There are some other NAS based methods for width search, or channel pruning [37], [48] mentioned in Section II-B. Noting that our formulation of the problem and the criterion to evaluate results are different, we emphasize that out progress is in fusion of a new search algorithm and the width search task.

When using RARTS to search for width, we follow the hyperparameters and settings of Network Slimming as well. That is, learning rate $= 0.1$, weight decay $= 0.0001$, epochs $=$ 160 [21]. In Table 5, RARTS outperforms the un-pruned baseline, Network Slimming (NS) and TAS [20] by over 10% error reduction on CIFAR-10. While TAS does not offer an option to specify the pruning ratio of channels (PRC), the pruning ratio of FLOPs is around 30% for NS (40% PRC), RARTS (40% PRC) and TAS. So the comparison is fair. On CIFAR-100, RARTS still leads NS at the same PRC. The gap is smaller as the baseline network is less redundant. Our experimental results reveal that the accuracy of TAS with KD is lower than (on CIFAR-10) or similar to (on CIFAR-100) that of RARTS, while TAS without the training technique like KD is 2% worse [20]. This supports the fact that RARTS works better as a differentiable method for width search, without regard to any other training tricks. Apart from the comparisons with the above methods, we also consider a pruning task for comparing DARTS and RARTS, which can be viewed as an ablation study of RARTS on the width search task. For this task, we prune MobileNetV2 [52] on a randomly sampled 20-class subset of ImageNet-1000, with ℓ_1 regularization but unfixed pruning ratio. The pruning ratio can be learned automatically by the strong regularization term, as many of the architecture parameters are simply zero. Table 6 shows that RARTS also beats both random pruning and DARTS in accuracy. Even though the 2nd DARTS obtains a higher sparsity, it sacrifices the accuracy.

TABLE 5. Application of RARTS to ResNet-164 (baseline, 1.7 M parameters) channel pruning on CIFAR-10 and CIFAR-100, in comparison with the baseline, TAS and network slimming. the numbers in the parentheses indicate the pruning ratio of channels (PRC). For NS and RARTS, PRC is fixed at 40% or 60%. NS = network slimming.

Data	Method	Test Error (%)
CIFAR-10	Baseline [21]	5.42
	TAS [20]	6.00
	NS (40% PRC) [21]	5.08
	RARTS (40% PRC)	**4.58**
	NS (60% PRC) [21]	5.27
	RARTS (60% PRC)	**4.90**
CIFAR-100	Baseline [21]	23.37
	TAS [20]	22.24
	NS (40% PRC) [21]	22.87
	RARTS (40% PRC)	**22.64**
	NS (60% PRC) [21]	23.91
	RARTS (60% PRC)	**23.26**

TABLE 6. Application of RARTS to mobileNetV2 pruning on the ImageNet-R dataset (a randomly sampled subset of ImageNet-1000, with 20 object classes), compared with the baseline, random pruning, 1st and 2nd-order DARTS. Here random pruning means that we zero out channels randomly in accordance with the pruning ratio of RARTS. Average of 5 runs. PRC = the average pruning ratio of channels over the pruned layers. We note that the PRC can be high because the dataset is much smaller.

Method	Test Error. (%)	PRC (%)
Baseline	12.3 ± 1.4	-
Random Pruning	12.0 ± 1.1	71.2 ± 1.9
DARTS-1	10.1 ± 2.0	69.0 ± 0.9
DARTS-2	9.8 ± 1.7	72.6 ± 2.0
RARTS	$\mathbf{8.2 \pm 1.9}$	71.2 ± 1.9

V. CONCLUSION

We have developed RARTS, a novel relaxed differentiable method for neural architecture search. We have proved its convergence theorem and compared it with DARTS

on an analytically solvable model. Thanks to the design of data and network splitting, RARTS has achieved high accuracy and search efficiency over the state-of-the-art differentiable methods, especially DARTS, with a wide range of experiments, including both topology search and width search. These results support RARTS to be a more reliable and robust differentiable neural architecture search tool for various datasets and search spaces. In future work, we plan to incorporate search space sampling and regularization techniques to accelerate RARTS (as seen in several recent variants of DARTS) for broader applications in deep learning.

ACKNOWLEDGMENT

The authors would like to thank the associate editor and the anonymous referees for their careful reading and helpful feedback, which improved the presentation of the paper and also would like to thank Dr. Shuai Zhang and Dr. Jiancheng Lyu for wonderful discussions related to the project.

REFERENCES

[1] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*.

[2] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.

[3] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7036–7045.

[4] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 82–92.

[5] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 1997–2017, 2019.

[6] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–34, May 2022.

[7] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," 2018, *arXiv:1812.00332*.

[8] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, F.-F. Li, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 19–34.

[9] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," 2018, *arXiv:1802.03268*.

[10] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4780–4789.

[11] D. Stamoulis, R. Ding, D. Wang, D. Lymberopoulos, B. Priyantha, J. Liu, and D. Marculescu, "Single-path NAS: Designing hardware-efficient ConvNets in less than 4 hours," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Cham, Switzerland: Springer, 2019, pp. 481–497.

[12] B. Wu, K. Keutzer, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, and Y. Jia, "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10734–10742.

[13] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 544–560.

[14] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," 2018, *arXiv:1806.09055*.

[15] C. He, H. Ye, L. Shen, and T. Zhang, "MiLeNAS: Efficient neural architecture search via mixed-level reformulation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11993–12002.

[16] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang, "FasterSeg: Searching for faster real-time semantic segmentation," 2019, *arXiv:1912.10917*.

[17] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[18] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1761–1770.

[19] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair darts: Eliminating unfair advantages in differentiable architecture search," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 465–480.

[20] X. Dong and Y. Yang, "Network pruning via transformable architecture search," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 760–771.

[21] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2736–2744.

[22] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proc. ECCV*, 2018, pp. 304–320.

[23] K. Bui, F. Park, S. Zhang, Y. Qi, and J. Xin, "Improving network slimming with nonconvex regularization," *IEEE Access*, vol. 9, pp. 115292–115314, 2021.

[24] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2074–2082.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[26] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[27] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic neural architecture search," 2018, *arXiv:1812.09926*.

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[30] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-bench: Benchmarking NAS algorithms for architecture topology and size," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3634–3646, Jul. 2022.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[32] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Ann. Oper. Res.*, vol. 153, no. 1, pp. 235–256, Sep. 2007.

[33] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *Proc. ICML*, 2018, pp. 1568–1577.

[34] R. Wang, M. Cheng, X. Chen, X. Tang, and C.-J. Hsieh, "Rethinking architecture selection in differentiable NAS," 2021, *arXiv:2108.04392*.

[35] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1294–1303.

[36] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "PC-DARTS: Partial channel connections for memory-efficient architecture search," 2019, *arXiv:1907.05737*.

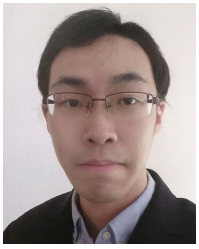
[37] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: AutoML for model compression and acceleration on mobile devices," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 784–800.

[38] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2820–2828.

[39] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen, P. Vajda, and J. E. Gonzalez, "FBNetV2: Differentiable neural architecture search for spatial and channel dimensions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12965–12974.

[40] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Proc. Uncertainty Artif. Intell.*, 2020, pp. 367–377.

[41] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," 2016, *arXiv:1607.03250*.
[42] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*.
[43] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1389–1397.
[44] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5058–5066.
[45] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," 2018, *arXiv:1810.05270*.
[46] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Roy. Stat. Soc., B, Stat. Methodol.*, vol. 68, no. 1, pp. 49–67, 2006.
[47] J. Ye, L. Xin, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," 2018, *arXiv:1802.00124*.
[48] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K.-T. Cheng, and J. Sun, "MetaPruning: Meta learning for automatic neural network channel pruning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3296–3305.
[49] T. Dinh and J. Xin, "Convergence of a relaxed variable splitting method for learning sparse neural networks via ℓ_1, ℓ_0, and transformed-ℓ_1 penalties," in *Proc. SAI Intell. Syst. Conf.* Cham, Switzerland: Springer, 2020, pp. 360–374.
[50] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.
[51] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[52] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.



FANGHUI XUE received the B.S. degree in mathematics from Fudan University, Shanghai, China, in 2016, the M.S. degree in analytics and mathematical risk management from Georgia State University, Atlanta, GA, in 2018, and the Ph.D. degree in mathematics from the University of California at Irvine, Irvine, CA, in May 2022. His research interests include deep learning and computer vision, with a focus on AutoML and constructing efficient neural networks.

YINGYONG QI received the Ph.D. degree in speech and hearing sciences from The Ohio State University, in 1989, and the Ph.D. degree in electrical and computer engineering from the University of Arizona, in 1993. He held a faculty member position with the University of Arizona, from 1989 to 1999. He was a Visiting Scientist with the Research Laboratory of Electronics, Massachusetts Institute of Technology, from 1995 to 1996, and a Visiting Scientist with the Visual Computing Laboratory of Hewlett Packard, Palo Alto, in 1998. He is currently a Senior Director of Technology at Qualcomm and a Researcher with the Department of Mathematics, University of California at Irvine. He has published over 100 scientific articles and U.S. patents during his tenure at university and industry. His research interests include speech processing, computer vision, and machine learning. He received Klatt Memorial Award in Speech Science from the Acoustical Society of America, in 1991, the First Award from the National Institute of Health, in 1992, and the AASFAA Outstanding Faculty Award from the University of Arizona, in 1998. More recently, he led a team winning the 3rd Place of IEEE Low Power Image Recognition Competition, sponsored by Google at CVPR 2019.

JACK XIN received the Ph.D. degree in mathematics from the Courant Institute of Mathematical Sciences, New York University, in 1990. He was a Faculty Member of the University of Arizona, from 1991 to 1999, and The University of Texas at Austin, from 1999 to 2005. He is currently a Chancellor's Professor in mathematics at UC Irvine. His research interests include applied analysis and computational methods and their applications in multi-scale problems and data science. He is a fellow of Guggenheim Foundation, American Mathematical Society, American Association for the Advancement of Science, and the Society for Industrial and Applied Mathematics. He was a recipient of Qualcomm Faculty Award (2019–2022).