ISTA-NAS: Efficient and Consistent Neural Architecture Search by Sparse Coding

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neural architecture search (NAS) aims to produce the optimal sparse solution from a high-dimensional space spanned by all candidate connections. Current gradient-based NAS methods commonly ignore the constraint of sparsity in the search phase, but project the optimized solution onto a sparse one by post-processing. As a result, the dense super-net for search is inefficient to train and has a gap with the projected architecture for evaluation. In this paper, we formulate neural architecture search as a sparse coding problem. We perform the differentiable search on a compressed lower-dimensional space that has the same validation loss as the original sparse solution space, and recover an architecture by solving the sparse coding problem. The differentiable search and architecture recovery are optimized in an alternate manner. By doing so, our network for search at each update satisfies the sparsity constraint and is efficient to train. In order to also eliminate the depth and width gap between the network in search and the target-net in evaluation, we further propose a method to search and evaluate in one stage under the target-net setting. When training finishes, architecture variables are absorbed into network weights. Thus we get the searched architecture and optimized parameters in a single run. In experiments, our two-stage method on CIFAR-10 requires only 0.05 GPU-day for search. Our one-stage method produces state-of-the-art performances on both CIFAR-10 and ImageNet at the cost of only evaluation time.

1 Introduction

Current NAS studies can be mainly categorized into reinforcement learning-based [1, 52, 50, 53, 3], evolution-based [36, 35, 25, 30, 13], Bayesian optimization-based [22, 51], and gradient-based methods [26, 43, 4, 46]. Most reinforcement learning and evolution-based algorithms suffer from huge computational cost, while gradient-based methods are simple to implement.

In gradient-based methods, an over-parameterized super-net is constructed to cover all candidate connections. Different architectures are sub-graphs of the super-net by weight sharing [34]. Thus the aim of search is to determine a sparse solution from the high-dimensional architecture space. Liu *et al.* propose a differentiable search framework, DARTS [26], by introducing a set of architecture variables jointly optimized with the network weights. After search, the architecture variables are projected to be sparse by only keeping the top-2 strongest connections for each intermediate node as the target-net for evaluation. This method is the basis of many follow-up studies [43, 4, 8, 44].

However, we note that the architecture variables do not satisfy the sparsity constraint during search. There is a gap between the dense super-net in search and the sparse target-net in evaluation. From the perspective of optimization, solutions should be constrained in the feasible region at each iteration, as adopted in the Projected Gradient Descent (PGD) [31] and proximal algorithms [32]. Otherwise, the converged solution may be far from the optimal one in the feasible region. It explains the fact that there is a poor correlation between the performances of super-net for search and target-net for

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

evaluation in DARTS [43, 6, 45]. A high-performance super-net does not necessarily produce a good architecture after projection onto the sparsity constraint. Besides, the dense super-net covering all candidate connections is inefficient to train due to its huge computational and memory cost.

In some follow-up studies [43, 12, 42, 6], the Gumbel Softmax strategy [21, 29] is adopted to enforce sparsification. Nevertheless, the super-net still contains the whole graph which is different from the derived architecture in target-net and does not reduce the computational and memory consumption. In ProxylessNAS [4] and NASP [46], only sampled or projected connections are active for each edge to reduce the memory consumption and search cost. But the active paths of super-net still deviate from the target-net as there is no connection for some edges in the target-net. For single-path architecture search, DSNAS [19] proposes a one-stage method to eliminate the inconsistency between super-net and target-net brought by two-stage methods, while there is yet no single-stage solution to DARTS-based architecture where the dimension of candidate connections is relatively higher.

Inspired by the observation that architecture variables in DARTS should have a sparse structure that can be well-represented by a compact space, in this paper, we propose to formulate NAS as a sparse coding problem, named ISTA-NAS. We construct an equivalent compressed search space where each point corresponds to a sparse solution in the original space. We perform gradient-based search in the compressed space with the sparsity constraint inherently satisfied, and then recover a new architecture by the sparse coding problem, which can be efficiently solved by well-developed methods, such as the iterative shrinkage thresholding algorithm (ISTA) [11]. The differentiable search and architecture recovery are conducted in an alternate way, so at each update, the network for search is sparse and efficient to train. After convergence, there is no need of projection onto sparsity constraint by post-processing and the searched architecture is directly available for evaluation.

In previous studies [26, 43, 44], the super-net in search is dense, and thus has a smaller depth and width than the target-net setting due to limited memory. However, the number of cells and channels has a significant effect on the search result [45, 8]. In order to also eliminate these gaps between super-net in search and target-net in evaluation, we develop a one-stage framework where search and evaluation share the same super-net under the target-net settings, such as depth, width and batchsize. One-stage ISTA-NAS begins to search with all batch normalization (BN) layers frozen. When a termination condition is satisfied, architecture variables cease to change and one sparse architecture is searched and fixed. BN layers are now trainable and other network weights continue to be optimized. After training, architecture variables are absorbed into the parameters of BN layers, and we get the searched architecture and all optimized parameters in a single run with only evaluation time.

We list the contributions of this paper as follows:

- We formulate neural architecture search as a sparse coding problem and propose ISTA-NAS, which performs the differentiable search on an equivalent compressed space and recovers its corresponding sparse architecture in an alternate manner. The sparsity constraint is inherently satisfied at each update so the search is more efficient and consistent with evaluation.
- We further develop a one-stage ISTA-NAS that incorporates search and evaluation in a single framework under the target-net settings. The network for search has no gap in terms of depth, width and even training batchsize with the derived architecture for evaluation.
- In experiments, the two-stage ISTA-NAS finishes search on CIFAR-10 in only 0.05 GPU-day. And the one-stage version produces state-of-the-art performances in a single run on CIFAR-10 or directly on ImageNet at the cost of only evaluation time. Code will be released.

2 Related Work

Neural architecture search (NAS) has been proposed to spare the manual efforts in traditional networks [37, 38, 17, 20] and benefit related computer vision tasks, such as object detection [33, 14, 40, 9], and semantic segmentation [7, 24]. Current gradient-based search methods widely adopt a two-stage strategy. In the first stage for search, the dense super-net covers all candidate connections and has to decrease the depth and width. As a result, it has a gap with the target-net for evaluation in the second stage, and the two stages correlate poorly. Several studies have been proposed for this problem.

Reducing the gap. To derive an architecture that is close to the one during search, Gumbel Softmax [43, 12, 6] or sparsity regularization [48] are adopted to enforce sparse architecture variables. But the whole graph still needs to be stored and computed, which is inefficient to train for the dense super-net. Some studies use a sparse super-net by only keeping sampled or projected connections active [4, 46].

P-DARTS [8] reduces the depth gap by gradually dropping connections and increasing depths. Even if these studies improve the correlation between the two stages, their super-nets only approach to but do not strictly satisfy the sparsity constraint. A post-processing is still needed to derive the target-net. Our method differs from theses studies in that we keep the sparsity constraint inherently satisfied at each update, and do not need the projection as post-processing.

One-stage NAS. DSNAS [19] proposes a one-stage search method to circumvent the gap brought by two-stage methods. The architecture variables and network weights are simultaneously optimized. However, it is proposed for single-path architecture search [16, 47, 42]. We show that ISTA-NAS also enables one-stage search. As a comparison, our method is developed for DARTS-based architecture space, which has a higher dimension because candidate connections for each intermediate node come from different operations of multiple previous nodes.

We note that a study [10] introduces compressed sensing into NAS. But the method is based on a meta-learning algorithm, instead of our differentiable architecture search.

3 ISTA-NAS

In this section, we first make a brief review of the differentiable architecture search and sparse coding. Then we build their relation and formulate NAS as a sparse coding problem. Finally, we introduce our two-stage and one-stage ISTA-NAS algorithms, respectively.

3.1 Preliminaries on Differentiable Neural Architecture Search

The search space of a cell in DARTS-based method is formed as a directed acyclic graph (DAG) consisting of n ordered nodes $\{x_1, x_2, \cdots, x_n\}$ and their edges $\mathcal{E} = \{e^{(i,j)} | 1 \leq i < j \leq n\}$. Each edge has K candidate operations from $\mathcal{O} = \{o_1, o_2, \cdots, o_K\}$, such as identity, max-pooling, and 3×3 separable convolution. With a binary variable $z_k^{(i,j)} \in \{0, 1\}$ to indicate whether the corresponding connection is active, we have the intermediate node x_j as:

$$x_j = \sum_{i=1}^{j-1} \sum_{k=1}^{K} z_k^{(i,j)} o_k(x_i) = \mathbf{z}_j^T \mathbf{o}_j, \tag{1}$$

where $\mathbf{z}_j \in \{0, 1\}^{(j-1)K}$ and $\mathbf{o}_j \in \mathbb{R}^{(j-1)K}$ denote the vectors formed by $z_k^{(i,j)}$ and $o_k(x_i)$, respectively. The goal of NAS is to solve the following constrained bi-level optimization problem:

$$Z^* = \underset{Z}{\operatorname{argmin}} \, \mathcal{L}_{val}(\mathcal{N}(W^*, Z)), \tag{2}$$

$$W^* = \underset{W}{\operatorname{argmin}} \, \mathcal{L}_{train}(\mathcal{N}(W, Z)), \tag{3}$$

$$s.t. \quad \|\mathbf{z}_j\|_0 = s_j, \; 1 < j \leq n, \tag{4}$$

where $Z = \{\mathbf{z}_j\}_{j=2}^n$, W is the weights of super-net $\mathcal{N}$, and s_j denotes the sparseness of node j. Since the architecture variables $\mathbf{z}_j$ are binary and thus stubborn to optimize in a differentiable way, current studies adopt the continuous relaxation by $z_k^{(i,j)} = \exp(\alpha_k^{(i,j)}) / \sum_k \exp(\alpha_k^{(i,j)})$ and optimize $\alpha_k^{(i,j)}$ as the trainable variables [26, 43, 8, 44]. In the search phase, the sparsity constraint Eq. (4) is not considered. $\alpha_k^{(i,j)}$ and W are optimized in an alternate manner by solving Eq. (2) and Eq. (3), respectively. After search, $\mathbf{z}_j$ is projected onto the sparsity constraint Eq. (4) by only keeping the top-2 strongest dimensions for node j, *i.e.*, $s_j = 2, \forall \, 1 < j \leq n$.

It is shown that current studies widely ignore the sparsity constraint Eq. (4) during search, and the sparse architecture variables $\mathbf{z}_j$ are derived by post-processing. This may cause an invalid search process because there is few correlation between the optimal solutions inside and outside the feasible region. Constraints should be satisfied at each step of optimization [31, 32]. In addition, the dense super-net $\mathcal{N}(W, Z)$ covers all candidate connections in search. As a result, it is inefficient to train, and has to adopt a small depth, width and training batchsize due to limited memory, which introduces the gap of depth, width and batchsize with the target-net in evaluation.

3.2 Preliminaries on Sparse Coding

Sparse coding serves as the basis of many machine learning applications, such as image denoising and super-resolution (see [49] and references therein). It aims to recover a sparse signal $\mathbf{z} \in \mathbb{R}^n$ from

its compressed observation $\mathbf{b} \in \mathbb{R}^m$:

$$\mathbf{b} = \mathbf{A}\mathbf{z} + \epsilon, \tag{5}$$

where $\epsilon \in \mathbb{R}^m$ is the additive noise, $\mathbf{A} \in \mathbb{R}^{m \times n}$ is an over-complete measurement matrix and we have $m < n$. The sparsity constraint of $\mathbf{z}$ is non-convex and challenging to deal with. A popular approach is to use the ℓ_1-norm as the convex surrogate, known as the LASSO formulation [41]:

$$\min_{\mathbf{z}} \frac{1}{2}\|\mathbf{A}\mathbf{z} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{z}\|_1, \tag{6}$$

where λ is the regularization parameter. Many well-developed algorithms are proposed to solve the problem Eq. (6), such as the iterative shrinkage thresholding algorithm (ISTA) [11]:

$$\mathbf{z}^{(t+1)} = \eta_{\lambda/L}\left(\mathbf{z}^{(t)} - \frac{1}{L}\mathbf{A}^T\left(\mathbf{A}\mathbf{z} - \mathbf{b}\right)\right), \quad t = 0, 1, \cdots, \tag{7}$$

where L is taken as the largest eigenvalue of $\mathbf{A}^T\mathbf{A}$, and η_θ is the component-wise soft-thresholding operator defined as: $\eta_\theta(x) = \text{sign}(x)\max(|x| - \theta, 0)$. ISTA is a proximal gradient method applied to Eq. (6) and has an $O(1/t)$ convergence rate with a proper λ [2].

3.3 Formulate Differentiable NAS as Sparse Coding

We also relax the binary constraint of $\mathbf{z}_j$ and let $\mathbf{z}_j \in \Omega(\mathbf{z}_j) = \{\mathbf{z}_j \in \mathbb{R}^{(j-1)K} : \|\mathbf{z}_j\|_0 \leq s_j\}$, where s_j is the sparseness in Eq. (4). It is hard for current methods to search $\mathbf{z}_j$ directly in $\Omega(\mathbf{z}_j)$. If there is an equivalent search process constructed on a compressed space $\Omega(\mathbf{b}_j) = \mathbb{R}^{m_j}$, where $m_j < (j-1)K$, we can perform the differentiable search on the lower-dimensional space without the sparsity constraint Eq. (4), and then recover a $\mathbf{z}_j$ in $\Omega(\mathbf{z}_j)$ by sparse coding. We have $\mathbf{b}_j$ by a measurement matrix $\mathbf{A}_j \in \mathbb{R}^{m_j \times (j-1)K}$ as $\mathbf{b}_j = \mathbf{A}_j\mathbf{z}_j$. Assuming that $\mathbf{E}_j$ is a residual matrix of $\mathbf{A}_j$, such that $\mathbf{A}_j^T\mathbf{A}_j - \mathbf{E}_j = \mathbf{I}$, we introduce a network defined on $\Omega(\mathbf{b}_j)$ by:

$$\begin{aligned}\mathcal{N}(W, Z) :\rightarrow x_j = \mathbf{z}_j^T\mathbf{o}_j &= \mathbf{z}_j^T(\mathbf{A}_j^T\mathbf{A}_j - \mathbf{E}_j)\mathbf{o}_j = (\mathbf{A}_j\mathbf{z}_j)^T(\mathbf{A}_j\mathbf{o}_j) - \mathbf{z}_j^T\mathbf{E}_j\mathbf{o}_j \\ &= (\mathbf{b}_j^T\mathbf{A}_j - [\mathbf{z}_j(\mathbf{b}_j)]^T\mathbf{E}_j)\mathbf{o}_j :\rightarrow \mathcal{N}(W, B)\end{aligned} \tag{8}$$

where x_j and $\mathbf{o}_j$ are defined in Eq. (1), B denotes the architecture variables in the network $\mathcal{N}(W, B)$, *i.e.*, $B = \{\mathbf{b}_j\}_{j=2}^n$, and $\mathbf{z}_j(\mathbf{b}_j)$ is deemed as a function of $\mathbf{b}_j$ in $\mathcal{N}(W, B)$. We note that $\mathcal{N}(W, B)$ and $\mathcal{N}(W, Z)$ have the same propagation for node $x_j, \forall\, 1 < j \leq n$ when $\mathbf{b}_j = \mathbf{A}_j\mathbf{z}_j$.

Supposing there is only one architecture variable (*i.e.*, $n = 2$), we remove the index j for simplicity. We consider that $\mathbf{A}$ satisfies the restricted isometry property (RIP) [5] with the constant δ_{2s}, which is the smallest $\delta \in (0, 1)$ such that $(1-\delta)\|\mathbf{z}\|_2^2 \leq \|\mathbf{A}\mathbf{z}\|_2^2 \leq (1+\delta)\|\mathbf{z}\|_2^2$ for all $2s$-sparse $\mathbf{z}$. Then for any $\mathbf{b} \in \Omega(\mathbf{b})$, there is at most one s-sparse $\mathbf{z} \in \Omega(\mathbf{z})$ that satisfies $\mathbf{A}\mathbf{z} = \mathbf{b}$. When δ meets a more strict condition, an exact s-sparse recovery of $\mathbf{z}$ is guaranteed via ℓ_1-norm minimization [5]. Here we simply assume that the solution derived by $\mathbf{z} = \text{argmin}_{\mathbf{z}} \frac{1}{2}\|\mathbf{A}\mathbf{z} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{z}\|_1$ is s-sparse and satisfies $\mathbf{A}\mathbf{z} = \mathbf{b}$, which is possible with λ small enough. Then we have the following proposition:

Proposition 1. *Assume that* $\mathbf{A}$ *satisfies the RIP with its constant* δ_{2s} *and the exact* s*-sparse solution* $\mathbf{z}^*$ *is recovered by* $\text{argmin}_{\mathbf{z}} \frac{1}{2}\|\mathbf{A}\mathbf{z} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{z}\|_1$ *and satisfies* $\mathbf{A}\mathbf{z} = \mathbf{b}$*. We have that* $\mathbf{z}^*$ *is the optimal solution of network* $\mathcal{N}(W, \mathbf{z})$ *if and only if* $\mathbf{b}^* = \mathbf{A}\mathbf{z}^*$ *is the optimal solution of* $\mathcal{N}(W, \mathbf{b})$.

Proof. For sufficiency, suppose that there exists another s-sparse $\mathbf{z}' \neq \mathbf{z}^*$, such that $\mathcal{L}(\mathcal{N}(W, \mathbf{z}')) < \mathcal{L}(\mathcal{N}(W, \mathbf{z}^*))$. Then $\mathbf{b}' = \mathbf{A}\mathbf{z}' \neq \mathbf{b}^*$ due to the RIP of $\mathbf{A}$. Because the two networks $\mathcal{N}(W, \mathbf{b})$ and $\mathcal{N}(W, \mathbf{z})$ have the same propagation by Eq. (8), we have $\mathcal{L}(\mathcal{N}(W, \mathbf{b}')) = \mathcal{L}(\mathcal{N}(W, \mathbf{z}')) < \mathcal{L}(\mathcal{N}(W, \mathbf{z}^*)) = \mathcal{L}(\mathcal{N}(W, \mathbf{b}^*))$, which is in conflict with $\mathbf{b}^* = \mathbf{A}\mathbf{z}^*$ is the optimal solution of $\mathcal{N}(W, \mathbf{b})$. The necessity can be derived in the same way, which concludes the proof. □

It is shown that a proper measurement matrix $\mathbf{A}$ builds the connection between the original and the compressed spaces. The optimal solution in $\Omega(\mathbf{z})$ can be searched by optimization in $\Omega(\mathbf{b})$. Thus we have our problem formulated as:

$$\mathbf{z}_j = \text{argmin}_{\mathbf{z}} \frac{1}{2}\|\mathbf{A}_j\mathbf{z} - \mathbf{b}_j\|_2^2 + \lambda\|\mathbf{z}\|_1, \quad 1 < j \leq n, \tag{9}$$

$$\begin{cases} B^* = \text{argmin}_B \mathcal{L}_{val}(\mathcal{N}(W^*, B)), \\ W^* = \text{argmin}_W \mathcal{L}_{train}(\mathcal{N}(W, B)), \end{cases} \tag{10}$$

Algorithm 1 Two-stage ISTA-NAS (for search only)

Input: Initialize the network weights W of the whole super-net $\mathcal{N}(W, B)$ and architecture variables $\mathbf{b}_j \in \mathbb{R}^{m_j}$ for each intermediate node $1 < j \leq n$. Sample $\mathbf{A}_j \in \mathbb{R}^{m_j \times (j-1)K}, \forall 1 < j \leq n$.

1: **while** *not converged* **do**
2: Recover $\mathbf{z}$ by solving Eq. (9) with ISTA. Keep the top-s strongest magnitudes and set other dimensions as zeros. The support set $\mathcal{S}(\mathbf{z}) = \{i|\mathbf{z}(i) \neq 0\}$;
3: Derive a sub-graph $\mathcal{N}(W_{(\mathcal{S})}, B)$ of the super-net by only propagating the dimensions in $\mathcal{S}$;
4: Update network weights $W_{(\mathcal{S})}$ by descending $\nabla_{W_{(\mathcal{S})}} \mathcal{L}_{train}(\mathcal{N}(W_{(\mathcal{S})}, B))$;
5: Update architecture variables $\mathbf{b}$ by descending $\nabla_{\mathbf{b}} \mathcal{L}_{val}(\mathcal{N}(W_{(\mathcal{S})}, B))$;
6: **end while**

Output: Produce a sparse architecture for evaluation according to the final $\mathcal{S}(\mathbf{z})$.

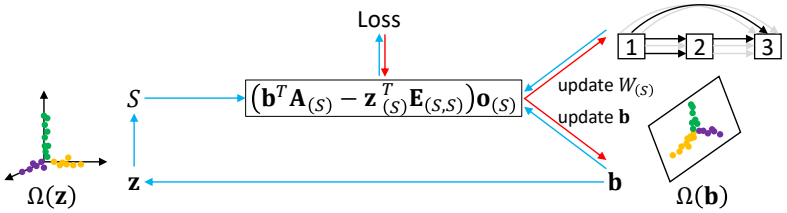

Figure 1: An illustration of our search process. The blue arrows denote the forward propagation, while the red ones are backward updates. $\Omega(\mathbf{z})$ and $\Omega(\mathbf{b})$ are the original and the compressed spaces, respectively. $\mathbf{z}$ lies near the axises because it is sparse. The black arrows in the network represent the corresponding connections that are active in the current $\mathcal{S}$, while the gray ones do not belong to $\mathcal{S}$.

where $B = \{\mathbf{b}_j\}_{j=2}^n$ is the trainable variables in the network $\mathcal{N}(W, B)$. B and W are optimized by the differentiable NAS without the sparsity constraint Eq. (4). $Z = \{\mathbf{z}_j\}_{j=2}^n$ is recovered by the sparse coding problem in Eq. (9), which inherently produces a sparse architecture.

3.4 Two-stage ISTA-NAS

The benefit of introducing sparse coding into NAS is that we can utilize the sparsity for more efficient search. In implementation, the solution of Eq. (9) may not lie in $\Omega(\mathbf{z})$ and strictly satisfy the sparsity constraint. We keep the top-s strongest magnitudes and set other dimensions as zeros. Let $\mathcal{S}(\mathbf{z})$ denote the support set of $\mathbf{z}$, *i.e.*, $\mathcal{S}(\mathbf{z}) = \{i|\mathbf{z}(i) \neq 0\}$ and $|\mathcal{S}| = s$, where i is the i-th dimension of $\mathbf{z}$ and s is the sparseness. Then the network propagation in Eq. (8) is equivalent to:

$$x = \mathbf{z}^T\mathbf{o} = \mathbf{z}_{(\mathcal{S})}^T\mathbf{o}_{(\mathcal{S})} = \mathbf{z}_{(\mathcal{S})}^T\left(\mathbf{A}_{(\mathcal{S})}^T\mathbf{A}_{(\mathcal{S})} - \mathbf{E}_{(\mathcal{S},\mathcal{S})}\right)\mathbf{o}_{(\mathcal{S})} = \left(\mathbf{b}^T\mathbf{A}_{(\mathcal{S})} - \mathbf{z}_{(\mathcal{S})}^T\mathbf{E}_{(\mathcal{S},\mathcal{S})}\right)\mathbf{o}_{(\mathcal{S})}, \quad (11)$$

where $\mathbf{z}_{(\mathcal{S})}$ denotes the elements of $\mathbf{z}$ indexed by $\mathcal{S}$, $\mathbf{A}_{(\mathcal{S})}$ denotes the columns of $\mathbf{A}$ indexed by $\mathcal{S}$, and $\mathbf{E}_{(\mathcal{S},\mathcal{S})}$ denotes the rows and columns of $\mathbf{E}$ indexed by $\mathcal{S}$.

We now introduce our two-stage ISTA-NAS outlined in Algorithm 1. The two-stage pipeline is consistent with current studies [26, 43, 8, 44]. A super-net with the whole DAG is constructed for search, after which the searched sparse architecture is for evaluation with a larger depth and width. We first initialize the network weights W and architecture variables B of the super-net. Each $\mathbf{b}$ corresponds to a sparse $\mathbf{z}$ by solving Eq. (9) with ISTA. Then we keep the top-s strongest magnitudes and derive its support set $\mathcal{S}$, which indicates a sparse architecture. The network only propagates the connections in $\mathcal{S}$ by Eq. (11), and then updates $W_{(\mathcal{S})}$ and $\mathbf{b}$, respectively. The optimization of Eq. (10) is performed in an alternate way, which is the same as the bi-level scheme in current differentiable NAS methods. As a comparison, $\mathcal{S}$ dynamically changes in our search, so the sparsity constraint is inherently satisfied at each update. We do not need a projection onto the target-net constraint by post-processing. Besides, the network $\mathcal{N}(W_{(\mathcal{S})}, B)$ in our search is sparse and much more efficient to train. An illustration of how we perform our search process is shown in Figure 1.

3.5 One-stage ISTA-NAS

Since the super-net of ISTA-NAS is as sparse as the target-net, our method is friendly to memory and is able to adopt the larger depth and width in target-net. If we can use one network for both search and evaluation under the target-net setting, the gap of depth, width, and even batch size in two-stage

Algorithm 2 One-stage ISTA-NAS (for search and evaluation)

Input: Initialize $\mathcal{N}(W, B)$ with depth, width, and batch size in the target-net setting. γ and β of BN layers in all candidate operations are frozen and initialized as 1 and 0. $search\_flag := True$.

while *not converged* **do**
  if $search\_flag$ **then**
    Perform the Line 2 and Line 3 of Algorithm 1; $\mathbf{z}^{new} := \mathbf{z}$;
  end if
  if $search\_flag$ **and** $\|\mathbf{z}^{new} - \mathbf{z}^{old}\| \leq \epsilon$ **then**
    $\gamma.requires\_grad := True$; $\beta.requires\_grad := True$; $search\_flag := False$;
  end if
  Update network weights $W_{(\mathcal{S})}$ by descending $\nabla_{W_{(\mathcal{S})}} \mathcal{L}_{train}(\mathcal{N}(W_{(\mathcal{S})}, B))$;
  if $search\_flag$ **then**
    Update architecture variables $\mathbf{b}$ by descending $\nabla_{\mathbf{b}} \mathcal{L}_{train}(\mathcal{N}(W_{(\mathcal{S})}, B))$; $\mathbf{z}^{old} := \mathbf{z}^{new}$;
  end if
end while
Update the parameters of BN layers by Eq. (12);

Output: Produce a sparse architecture and its optimized parameters.

methods will be eliminated. To this end, we develop a one-stage method, where search and evaluation are finished in a single run, after which we get both architecture and its optimized parameters.

In Eq. (11), the architecture variables $\mathbf{b}$ and matrix $\mathbf{A}$ decide the coefficient of each connection for differentiable search. We need to combine these coefficients with network weights W as the final optimized parameters. Considering the parameterized operations for search, such as 3×3 separable convolution, 5×5 dilation convolution, end up with a batch normalization (BN) layer, we also add BN layers after non-parametric operations, including identity and all pooling operations.

The one-stage ISTA-NAS is outlined in Algorithm 2. At the beginning, the weight and bias of BN layers, *i.e.,* γ and β, are frozen and initialized as 1 and 0, respectively, so they will not affect the search process. Different from the two-stage method, $W_{(\mathcal{S})}$ and $\mathbf{b}$ share the same loss using the full train set (no validation set is split out). We introduce a termination condition, such that when $\mathbf{z}$ of two neighboring iterations are close, *i.e.*, $\|\mathbf{z}^{new} - \mathbf{z}^{old}\| \leq \epsilon$, we stop the optimization of $\mathbf{b}$. Then $\mathbf{z}$ and $\mathcal{S}$ no longer change so a sparse architecture is searched and fixed. The coefficient of each connection do not change accordingly. The weight γ and bias β of BN layers are now enabled to optimize. When training finishes, the coefficients are absorbed into γ, $\beta \in \mathbb{R}^s$ as[1]:

$$\hat{\gamma} = \left(\mathbf{b}^T \mathbf{A}_{(\mathcal{S})} - \mathbf{z}_{(\mathcal{S})}^T \mathbf{E}_{(\mathcal{S},\mathcal{S})}\right) \circ \gamma; \quad \hat{\beta} = \left(\mathbf{b}^T \mathbf{A}_{(\mathcal{S})} - \mathbf{z}_{(\mathcal{S})}^T \mathbf{E}_{(\mathcal{S},\mathcal{S})}\right) \circ \beta; \tag{12}$$

where $\circ$ is the element-wise multiplication, and $\hat{\gamma}$, $\hat{\beta} \in \mathbb{R}^s$ are updated parameters that keep the trained network accuracy unchanged. In this way, we get the searched architecture and its optimized parameters in a single run of Algorithm 2.

4 Experiments

We analyze the improved efficiency and correlation of the two-stage and one-stage ISTA-NAS, and then compare our search results on CIFAR-10 and ImageNet with state-of-the-art methods. **All searched architectures are visualized in the supplementary material**.

4.1 Implementation Details

Our setting of search and evaluation is consistent with the convention in current studies [26, 43, 8, 44]. Please see the full description of our implementation details in the supplementary material. For our two-stage ISTA-NAS, the super-net is composed of 6 normals cells and 2 reduction cells. Each cell has 6 nodes. The first two nodes are input nodes output from the previous two cells. As convention, each intermediate node keeps two connections after search, so the sparseness $s_j = 2$ in our method. We adopt the Adam optimizer for $\mathbf{b}_j$ and SGD for network weights W. We use the released tool MOSEK with CVX [15] to efficiently solve Eq. (9). For our single-stage ISTA-NAS, we adopt the target-net setting of the evaluation stage. The network is stacked by 18 normal cells and 2 reduction cells. Other details can be seen in the supplementary material.

[1] γ and β are viewed as vectors in $\mathbb{R}^s$ formed by all active connections to the same node.

	Bs.	Mem.	Search Cost
DARTS (1st order)	64	9.1 G	0.70 day
PC-DARTS	256	11.6 G	0.14 day
ISTA-NAS	64	1.9 G	0.15 day
ISTA-NAS	256	5.5 G	0.05 day
ISTA-NAS	512	10.5 G	0.03 day

Table 1: Search cost of DARTS, PC-DARTS, and two-stage ISTA-NAS. "Bs." and "Mem." denote the batchsize and its corresponding memory consumption. Search cost is tested on a GTX 1080Ti GPU.

	Kendall τ
DARTS (1st order)	-0.36
PC-DARTS	-0.21
Two-stage ISTA-NAS	0.43
One-stage ISTA-NAS	0.57

Table 2: Kendall τ correlation of search and evaluation performances in DARTS, PC-DARTS, and ISTA-NAS. The one-stage ISTA-NAS is measured by its converged accuracy and retraining without the optimization of $\mathbf{b}$.

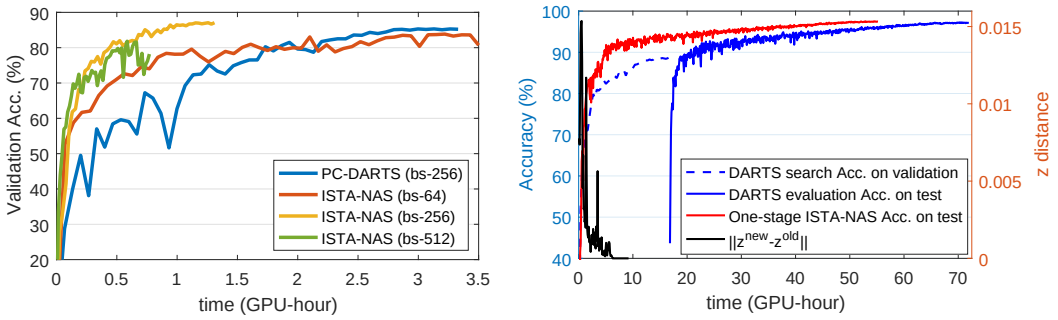

Figure 2: (left) The super-net accuracies of search in PC-DARTS and two-stage ISTA-NAS; (right) The search and evaluation processes of two-stage DARTS and one-stage ISTA-NAS.

4.2 Efficiency and Correlation Improved by ISTA-NAS

ISTA-NAS inherently satisfies the sparsity constraint at each update, so significantly improves the search efficiency. As shown in Table 1, we re-implement DARTS and PC-DARTS using their released codes on CIFAR-10. When our two-stage ISTA-NAS uses a batchsize of 64, the search cost is on par with PC-DARTS, which uses a batchsize of 256 and consumes much more GPU memory. Two-stage ISTA-NAS supports a batchsize of 512, which leads to a search cost of only 0.03 GPU-day, about 20 times faster than DARTS. Their search processes are depicted in Figure 2 (left). The learning rates are also enlarged by the same times as batchsize. It is shown that ISTA-NAS with a batchsize of 256 has the most stable search accuracy on validation. Thus we adopt the 256 batchsize and 0.1 learning rate of W for our two-stage search result on CIFAR-10. As for the one-stage ISTA-NAS, its accuracy on test is shown in Figure 2 (right). We also visualize the distance of $\mathbf{z}_j$ of two neighboring iterations averaged by all intermediate nodes j. It is observed that the distance of neighboring $\mathbf{z}$ decays fast. In about 10 GPU-hour, the termination condition is satisfied for all intermediate nodes, after which only network weights in the searched architecture get optimized. We have the architecture searched and optimized in a single run, which saves the search cost compared to the two-stage DARTS.

We also test the correlation between search and evaluation by the Kendall τ metric [23], which ranges from -1 to 1 and evaluates the rank correlation of data pairs. If $\tau = -1$, the ranking order is reversed, while if the ranking is identical, τ will be 1. The full introduction of this metric is appended in the supplementary material. We implement DARTS, PC-DARTS and two-stage ISTA-NAS for 8 times on CIFAR-10 with different seeds and calculate their Kendall τ metrics based on the super-net accuracies in search and the target-net accuracies in evaluation. We measure the metric of one-stage ISTA-NAS using its converged accuracy and the one retrained for the same epochs without optimizing $\mathbf{b}$. It is shown that both two-stage and one-stage ISTA-NAS have positive correlation scores, and the one-stage ISTA-NAS has the best correlation. The Kendall τ metric is still not close to 1, because accuracies on CIFAR-10 are not so distinguishable, and are easily affected by random noise.

4.3 Search Results on CIFAR-10

The search results of two-stage and one-stage ISTA-NAS on CIFAR-10 are shown in Table 3. We see that two-stage ISTA-NAS achieves a state-of-the-art error rate of 2.54% within only 0.05 GPU-day. The search cost is reduced to a half of the number in PC-DARTS and NASP, which are the fastest approaches for DARTS-based search space before ISTA-NAS. The one-stage ISTA-NAS unifies the

Methods	Test Error (%)	Params (M)	Cost (GPU-day)		Search Method
			search	eval.	
DenseNet-BC [20]	3.46	25.6	-	-	manual
NASNet-A + cutout [53]	2.65	3.3	1800	3.2	RL
ENAS + cutout [34]	2.89	4.6	0.5	3.2	RL
AmoebaNet-B +cutout [35]	2.55±0.05	2.8	3150	-	evolution
NAONet-WS [27]	3.53	3.1	0.4	-	NAO
DARTS (2nd order) + cutout [26]	2.76±0.09	3.3	4.0	2.3	gradient
SNAS (moderate) + cutout [43]	2.85±0.02	2.8	1.5	2.2	gradient
P-DARTS+cutout [8]	2.50	3.4	0.3	2.9	gradient
NASP + cutout [46]	2.83±0.09	3.3	0.1	-	gradient
PC-DARTS + cutout [44]	2.57±0.07	3.6	0.1	3.1	gradient
Two-stage ISTA-NAS + cutout	2.54±0.05	3.32	**0.05**	2.0	gradient
One-stage ISTA-NAS + cutout	**2.36**±0.06	3.37	**2.3**		gradient

Table 3: Search results on CIFAR-10 and comparison with state-of-the-art methods. Cost is tested on a GTX 1080Ti GPU. The evaluation cost is calculated by us with their searched architectures in the same experimental setting. The cost of one-stage ISTA-NAS is the time spent in a single run.

Methods	Test Err. (%)		Params (M)	Flops (M)	Cost (GPU-day)		Search Method
	top-1	top-5			search	eval.	
Inception-v1 [38]	30.2	10.1	6.6	1448	-	-	manual
MobileNet [18]	29.4	10.5	4.2	569	-	-	manual
ShuffleNet 2× (v2) [28]	25.1	-	∼5	591	-	-	manual
MnasNet-92 [39]	25.2	8.0	4.4	388	-	-	RL
AmoebaNet-C [35]	24.3	7.6	6.4	570	3150	-	evolution
DARTS (2nd order) [26]	26.7	8.7	4.7	574	4.0	3.6×8	gradient
SNAS [43]	27.3	9.2	4.3	522	1.5	3.3×8	gradient
P-DARTS [8]	24.4	7.4	4.9	557	0.3	3.6×8	gradient
ProxylessNAS (ImgNet) [4]	24.9	7.5	7.1	465	8.3	-	gradient
PC-DARTS (ImgNet) [44]	24.2	7.3	5.3	597	3.8	3.9×8	gradient
One-stage ISTA-NAS (C-10)	25.1	7.7	4.78	550	2.3	3.4×8	gradient
One-stage ISTA-NAS (ImgNet)	**24.0**	**7.1**	5.65	638	4.2×8		gradient

Table 4: Search results on ImageNet and comparison with state-of-the-art methods. Cost is tested on eight GTX 1080Ti GPUs. "ImgNet" denotes it is directly searched on ImageNet. Otherwise, it is searched on CIFAR-10 and then transfered to ImageNet for evaluation.

search and evaluation in a single run of 2.3 GPU-day, which is smaller than the total cost of most two-stage methods. Albeit its cost is larger than the two-stage ISTA-NAS, the performance gets better due to its improved correlation brought by direct search in the evaluation setting.

4.4 Search Results on ImageNet

We use one-stage ISTA-NAS for experiments on ImageNet. As shown in Table 4, the architecture searched on CIFAR-10 has a top-1 error rate of 25.1%, which is competitive considering its parameters and search time. When the method is directly performed on ImageNet, we have a top-1/5 error rates of 24.0%/7.1%, which surpasses the best known performance to date of DARTS-based search space on ImageNet. Still, the total cost is lower than most two-stage methods.

5 Conclusion

In this paper, we formulate the NAS problem as optimizing a sparse solution from a high-dimensional space spanned by all candidate connections, and introduce a more efficient and consistent search method, named ISTA-NAS. The differentiable search is performed on a compressed space and the sparse solution is recovered by ISTA. The sparsity constraint is inherently satisfied at each update, which makes the search more efficient and consistent with the architecture for evaluation. We further develop a one-stage method that unifies the search and evaluation in a single run. Experiments verify the improved efficiency and correlation of two-stage and one-stage ISTA-NAS. The searched architectures surpass the state-of-the-art performances on both CIFAR-10 and ImageNet.

6 Broader Impact

Our work proposes a new perspective to formulate the NAS problem and develops two algorithms that have better efficiency and correlation. The positive impacts are obvious. First, better architectures may be searched for some problems, so the practical application of neural network to the corresponding area can be fostered. It benefits industry because the products or services with more satisfactory performance can be employed. Second, NAS has been a resource demanding task. Our method saves much memory and time cost, which is friendly to environment and easy to use for practitioners. Finally, automation is believed as a complementary tool to human experts. We think that a good NAS method could bring researchers expertise in understanding neural architectures.

References

[1] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcemen learning. In *ICLR*, 2017.

[2] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[3] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu. Path-level network transformation for efficient architecture search. In *ICML*, volume 80, pages 678–687. PMLR, 2018.

[4] H. Cai, L. Zhu, and S. Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019.

[5] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005.

[6] J. Chang, Y. Guo, G. MENG, S. XIANG, C. Pan, et al. Data: Differentiable architecture approximation. In *NeurIPS*, pages 874–884, 2019.

[7] L.-C. Chen, M. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *NeurIPS*, pages 8699–8710, 2018.

[8] X. Chen, L. Xie, J. Wu, and Q. Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, pages 1294–1303, 2019.

[9] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun. Detnas: Backbone search for object detection. In *NeurIPS*, pages 6638–6648, 2019.

[10] M. Cho, M. Soltani, and C. Hegde. One-shot neural architecture search via compressive sensing. *arXiv preprint arXiv:1906.02869*, 2019.

[11] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

[12] X. Dong and Y. Yang. Searching for a robust neural architecture in four gpu hours. In *CVPR*, pages 1761–1770, 2019.

[13] T. Elsken, J. H. Metzen, and F. Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. In *ICLR*, 2019.

[14] G. Ghiasi, T.-Y. Lin, and Q. V. Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, pages 7036–7045, 2019.

[15] M. Grant and S. Boyd. Cvx: Matlab software for disciplined convex programming, version 2.1, 2014.

[16] 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. *arXiv preprint arXiv:1904.00420*, 2019.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[19] S. Hu, S. Xie, H. Zheng, C. Liu, J. Shi, X. Liu, and D. Lin. Dsnas: Direct neural architecture search without parameter retraining. *arXiv preprint arXiv:2002.09128*, 2020.

[20] G. Huang, Z. Liu, L. van der Maaten, and K. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[21] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[22] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing. Neural architecture search with bayesian optimisation and optimal transport. In *NeurIPS*, pages 2016–2025, 2018.

[23] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[24] 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 *CVPR*, pages 82–92, 2019.

[25] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. In *ICLR*, 2018.

[26] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. In *ICLR*, 2019.

[27] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu. Neural architecture optimization. In *NeurIPS*, pages 7816–7827, 2018.

[28] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, pages 116–131, 2018.

[29] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[30] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, et al. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pages 293–312. Elsevier, 2019.

[31] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[32] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[33] J. Peng, M. Sun, Z.-X. ZHANG, T. Tan, and J. Yan. Efficient neural architecture transformation search in channel-level for object detection. In *NeurIPS*, pages 14290–14299, 2019.

[34] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

[35] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In *AAAI*, volume 33, pages 4780–4789, 2019.

[36] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin. Large-scale evolution of image classifiers. In *ICML*, pages 2902–2911. JMLR. org, 2017.

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[39] 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 *CVPR*, pages 2820–2828, 2019.

[40] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection. *arXiv preprint arXiv:1911.09070*, 2019.

[41] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[42] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, pages 10734–10742, 2019.

[43] S. Xie, H. Zheng, C. Liu, and L. Lin. Snas: stochastic neural architecture search. In *ICLR*, 2019.

[44] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. In *ICLR*, 2020.

[45] A. Yang, P. M. Esperança, and F. M. Carlucci. Nas evaluation is frustratingly hard. In *ICLR*, 2020.

[46] Q. Yao, J. Xu, W.-W. Tu, and Z. Zhu. Efficient neural architecture search via proximal iterations. In *AAAI*, 2020.

[47] S. You, T. Huang, M. Yang, F. Wang, C. Qian, and C. Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. *arXiv preprint arXiv:2003.11236*, 2020.

[48] X. Zhang, Z. Huang, and N. Wang. You only search once: Single shot neural architecture search via direct sparse optimization. *arXiv preprint arXiv:1811.01567*, 2018.

[49] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang. A survey of sparse representation: algorithms and applications. *IEEE access*, 3:490–530, 2015.

[50] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu. Practical block-wise neural network architecture generation. In *CVPR*, June 2018.

[51] H. Zhou, M. Yang, J. Wang, and W. Pan. BayesNAS: A Bayesian approach for neural architecture search. In *ICML*, volume 97, pages 7603–7613. PMLR, 2019.

[52] B. Zoph and Q. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

[53] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, June 2018.