Geometry-Aware Gradient Algorithms for Neural Architecture Search

Liam Li[*1], **Mikhail Khodak**[*2], **Maria-Florina Balcan**[2], **and Ameet Talwalkar**[1,2]
[1]Determined AI, [2]Carnegie Mellon University, [*]denotes equal contribution
me@liamcli.com,khodak@cmu.edu,ninamf@cs.cmu.edu,talwalkar@cmu.edu

Abstract

Recent state-of-the-art methods for neural architecture search (NAS) exploit gradient-based optimization by relaxing the problem into continuous optimization over architectures and shared-weights, a noisy process that remains poorly understood. We argue for the study of single-level empirical risk minimization to understand NAS with weight-sharing, reducing the design of NAS methods to devising optimizers and regularizers that can quickly obtain high-quality solutions to this problem. Invoking the theory of mirror descent, we present a geometry-aware framework that exploits the underlying structure of this optimization to return sparse architectural parameters, leading to simple yet novel algorithms that enjoy fast convergence guarantees and achieve state-of-the-art accuracy on the latest NAS benchmarks in computer vision. Notably, we exceed the best published results for both CIFAR and ImageNet on both the DARTS search space and NAS-Bench-201; on the latter we achieve near-oracle-optimal performance on CIFAR-10 and CIFAR-100. Together, our theory and experiments demonstrate a principled way to co-design optimizers and continuous relaxations of discrete NAS search spaces.

Introduction

Neural architecture search has become an important tool for automating machine learning (ML) but can require hundreds of thousands of GPU-hours to train. Recently, *weight-sharing* approaches have achieved state-of-the-art performance while drastically reducing the computational cost of NAS to just that of training a single *shared-weights network* (Pham et al. 2018; Liu, Simonyan, and Yang 2019). Methods such as DARTS (Liu, Simonyan, and Yang 2019), GDAS (Dong and Yang 2019), and many others (Pham et al. 2018; Zheng et al. 2019; Yang et al. 2020; Xie et al. 2019; Liu et al. 2018; Laube and Zell 2019; Cai, Zhu, and Han 2019; Akimoto et al. 2019; Xu et al. 2020) combine weight-sharing with a continuous relaxation of the discrete search space to allow cheap gradient updates, enabling the use of popular optimizers. However, despite some empirical success, weight-sharing remains poorly understood and has received criticism due to (1) rank-disorder (Yu et al. 2020; Zela, Siems, and Hutter 2020; Zhang et al. 2020; Pourchot, Ducarouge, and Sigaud 2020), where the shared-weights performance is a poor surrogate of standalone performance, and (2) poor results on recent benchmarks (Dong and Yang 2020; Zela et al. 2020).

Motivated by the challenge of developing simple and efficient methods that achieve state-of-the-art performance, we study how to best handle the goals and optimization objectives of NAS. We start by observing that weight-sharing subsumes architecture hyperparameters as another set of learned parameters of the shared-weights network, in effect extending the class of functions being learned. This suggests that a reasonable approach towards obtaining high-quality NAS solutions is to study how to regularize and optimize the empirical risk over this extended class. While many regularization approaches have been implicitly proposed in recent NAS efforts, we focus instead on the question of optimizing architecture parameters, which may not be amenable to standard procedures such as SGD that work well for standard neural network weights. In particular, to better-satisfy desirable properties such as generalization and sparsity of architectural decisions, we propose to constrain architecture parameters to the simplex and update them using exponentiated gradient, which has favorable convergence properties due to the underlying problem structure. Theoretically, we draw upon the mirror descent meta-algorithm (Nemirovski and Yudin 1983; Beck and Teboulle 2003) to give convergence guarantees when using any of a broad class of such *geometry-aware* gradient methods to optimize the weight-sharing objective; empirically, we show that our solution leads to strong improvements on several NAS benchmarks. We summarize these contributions below:

1. We argue for studying NAS with weight-sharing as a single-level objective over a structured function class in which architectural decisions are treated as learned parameters rather than hyperparameters. Our setup clarifies recent concerns about rank disorder and makes clear that proper regularization and optimization of this objective is critical to obtaining high-quality solutions.
2. Focusing on optimization, we propose to improve existing NAS algorithms by re-parameterizing architecture parameters over the simplex and updating them using

exponentiated gradient, a variant of mirror descent that converges quickly over this domain and enjoys favorable sparsity properties. This simple modification—which we call the **Geometry-Aware Exponentiated Algorithm (GAEA)**—is easily applicable to numerous methods, including first-order DARTS (Liu, Simonyan, and Yang 2019), GDAS (Dong and Yang 2019), and PC-DARTS (Xu et al. 2020).

3. To show correctness and efficiency of our scheme, we prove polynomial-time stationary-point convergence of block-stochastic mirror descent—a family of geometry-aware gradient algorithms that includes GAEA—over a continuous relaxation of the single-level NAS objective. To the best of our knowledge these are the first finite-time convergence guarantees for gradient-based NAS.
4. We demonstrate that GAEA improves upon state-of-the-art methods on three of the latest NAS benchmarks for computer vision. Specifically, we beat the current best results on NAS-Bench-201 (Dong and Yang 2020) by 0.18% on CIFAR-10, 1.59% on CIFAR-100, and 0.82% on ImageNet-16-120; we also outperform the state-of-the-art on the DARTS search space (Liu, Simonyan, and Yang 2019), for both CIFAR-10 and ImageNet, and match it on NAS-Bench-1Shot1 (Zela et al. 2020).[1]

Related Work. Most optimization analyses of NAS show monotonic improvement (Akimoto et al. 2019), asymptotic guarantees (Yao et al. 2020), or bounds on auxiliary quantities disconnected from any objective (Noy et al. 2019; Nayman et al. 2019; Carlucci et al. 2019). In contrast, we prove polynomial-time stationary-point convergence on a single-level objective for weight-sharing NAS, so far only studied empirically (Xie et al. 2019; Li et al. 2019). Our results draw upon the mirror descent meta-algorithm (Nemirovski and Yudin 1983; Beck and Teboulle 2003) and extend recent nonconvex convergence results (Zhang and He 2018) to handle alternating descent. While there exist related results (Dang and Lan 2015) the associated guarantees do not hold for the algorithms we propose. Finally, we note that a variant of GAEA that modifies first-order DARTS is related to XNAS (Nayman et al. 2019), whose update also involves exponentiated gradient; however, GAEA is simpler and easier to implement.[2] Furthermore, the regret guarantees for XNAS do not relate to any meaningful performance measure for NAS such as speed or accuracy, whereas we guarantee convergence on the ERM objective.

The Weight-Sharing Optimization Problem

In supervised ML we have a dataset T of labeled pairs (x, y) drawn from a distribution $\mathcal{D}$ over input/output spaces X and Y. The goal is to use T to search a function class H for $h_{\mathbf{w}} : X \mapsto Y$ parameterized by $\mathbf{w} \in \mathbb{R}^d$ that has low expected test loss $\ell(h_{\mathbf{w}}(x), y)$ when using x to predict the associated y on unseen samples drawn from D, as measured by some loss $\ell : Y \times Y \mapsto [0, \infty)$. A common way to do so is by approximate (regularized) empirical risk minimization (ERM), i.e. finding $\mathbf{w} \in \mathbb{R}^d$ with the smallest average loss over T, via some iterative method $\texttt{Alg}$, e.g. SGD.

Benefits and Criticisms of Weight-Sharing for NAS

NAS is often viewed as hyperparameter optimization on top of $\texttt{Alg}$, with each architecture $a \in \mathcal{A}$ corresponding to a function class $H_a = \{h_{\mathbf{w},a} : X \mapsto Y, \mathbf{w} \in \mathbb{R}^d\}$ to be selected by using validation data $V \subset X \times Y$ to evaluate the predictor obtained by fixing a and doing approximate ERM over T:

$$\min_{a \in \mathcal{A}} \sum_{(x,y) \in V} \ell(h_{\mathbf{w}_a, a}(x), y) \quad \text{s.t.} \quad \mathbf{w}_a = \texttt{Alg}(T, a) \tag{1}$$

Since training individual sets of weights for any sizeable number of architectures is prohibitive, weight-sharing methods instead use a single set of shared weights to obtain validation signal about many architectures at once. In its most simple form, RS-WS (Li and Talwalkar 2019), these weights are trained to minimize a *non-adaptive* objective, $\min_{\mathbf{w} \in \mathbb{R}^d} \mathbb{E}_a \sum_{(x,y) \in T} \ell(h_{\mathbf{w}_a, a}(x), y)$, where the expectation is over a fixed distribution over architectures $\mathcal{A}$. The final architecture a is then chosen to maximize the outer (validation) objective in (1) subject to $\mathbf{w}_a = \mathbf{w}$. More frequently used is a *bilevel* objective over some continuous relaxation Θ of the architecture space $\mathcal{A}$, after which a valid architecture is obtained via a discretization step $\texttt{Map} : \Theta \mapsto \mathcal{A}$ (Pham et al. 2018; Liu, Simonyan, and Yang 2019):

$$\begin{aligned} \min_{\theta \in \Theta} \quad & \sum_{(x,y) \in V} \ell(h_{\mathbf{w},\theta}(x), y) \\ \text{s.t.} \quad & \mathbf{w} \in \arg\min_{\mathbf{u} \in \mathbb{R}^d} \sum_{(x,y) \in T} \ell(h_{\mathbf{u},\theta}(x), y) \end{aligned} \tag{2}$$

This objective is not significantly different from (2), since $\texttt{Alg}(T, a)$ approximately minimizes the empirical risk w.r.t. T; the difference is replacing discrete architectures with relaxed *architecture parameters* $\theta \in \Theta$, w.r.t. which we can take derivatives of the outer objective. This allows (2) to be approximated via alternating gradient updates w.r.t. $\mathbf{w}$ and θ. Relaxations can be *stochastic*, so that $\texttt{Map}(\theta)$ is a sample from a θ-parameterized distribution (Pham et al. 2018; Dong and Yang 2019), or a *mixture*, in which case $\texttt{Map}(\theta)$ selects architectural decisions with the highest weight in a convex combination given by θ (Liu, Simonyan, and Yang 2019). We overview this in more detail in the appendix[3].

While weight-sharing significantly shortens search (Pham et al. 2018), it draws two main criticisms:

- Rank disorder: this describes when the rank of an architecture a according to the validation risk evaluated with fixed shared weights $\mathbf{w}$ is poorly correlated with the one using

[1] Code to obtain these results has been made available here: https://github.com/liamcli/gaea_release

[2] XNAS code does not implement search and, as with previous efforts (Li et al. 2019, OpenReview), we cannot reproduce results after correspondence with the authors. XNAS's best architecture achieves an average test error of 2.70% under the DARTS evaluation, while GAEA achieves 2.50%. For details see the appendix.

[3] A full version of the paper including the appendix can be found here: https://openreview.net/pdf?id=MuSYkd1hxRP

"standalone" weights $\mathbf{w}_a = \texttt{Alg}(T, a)$. This causes sub-optimal architectures to be selected after shared weights search (Yu et al. 2020; Zela, Siems, and Hutter 2020; Zhang et al. 2020; Pourchot, Ducarouge, and Sigaud 2020).
- Poor performance: weight-sharing can converge to degenerate architectures (Zela et al. 2020) and is outperformed by regular hyperparameter tuning on NAS-Bench-201 (Dong and Yang 2020).

Single-Level NAS as a Baseline Object of Study

Why are we able to apply weight-sharing to NAS? The key is that, unlike regular hyperparameters such as step-size, *architectural* hyperparameters directly affect the loss function without requiring a dependent change in the model weights $\mathbf{w}$. Thus we can distinguish architectures without retraining simply by changing architectural decisions. Besides enabling weight-sharing, this point reveals that the goal of NAS is perhaps better viewed as a regular learning problem over an extended class $H_{\mathcal{A}} = \bigcup_{a \in \mathcal{A}} H_a = \{h_{\mathbf{w},a} : X \mapsto Y, \mathbf{w} \in \mathbb{R}^d, a \in \mathcal{A}\}$ that subsumes the architectural decisions as parameters of a larger model class, an unrelaxed "supernet." The natural approach to solving this is by approximate empirical risk minimization, e.g. by approximating continuous objective below on the right using a gradient algorithm and passing the output θ through $\texttt{Map}$ to obtain a valid architecture:

$$\begin{aligned} &\min_{\mathbf{w} \in \mathbb{R}^d, a \in \mathcal{A}} \sum_{(x,y) \in T} \ell(h_{\mathbf{w},a}(x), y) && \text{discrete (unrelaxed) supernet (NAS ERM)} \\ &\min_{\mathbf{w} \in \mathbb{R}^d, \theta \in \Theta} \sum_{(x,y) \in T} \ell(h_{\mathbf{w},\theta}(x), y) && \text{continuous relaxation (supernet ERM)} \end{aligned} \tag{3}$$

Several works have optimized this single-level objective as an alternative to bilevel (2) (Xie et al. 2019; Li et al. 2019). We argue for its use as the baseline object of study in NAS for three reasons:

1. As discussed above, it is the natural first approach to solving the statistical objective of NAS: finding a good predictor $h_{\mathbf{w},a} \in H_{\mathcal{A}}$ in the extended function class over architectures and weights.
2. The common alternating gradient approach to the bilevel problem (2) is in practice very similar to alternating block approaches to ERM (3); as we will see, there are established ways of analyzing such methods for the latter objective, while for the former convergence is known only under very strong assumptions such as uniqueness of the inner minimum (Franceschi et al. 2018).
3. While less frequently used in practice than bilevel, single-level optimization can be very effective: we use it to achieve new state-of-the-art results on NAS-Bench-201 (Dong and Yang 2020).

Understanding NAS as single-level optimization—the usual deep learning setting—makes weight-sharing a natural, not surprising, approach. Furthermore, for methods—both single-level and bilevel—that adapt architecture parameters during search, it suggests that we need not worry about rank disorder as long as we can use optimization to find a single feasible point that generalizes well; we explicitly do *not* need a ranking. Non-adaptive methods such as RS-WS still do require rank correlation to select good architectures after search, but they are explicitly *not* changing θ and so have no variant solving (3). The single-level formulation thus reduces search method design to well-studied questions of how to best regularize and optimize ERM. While there are many techniques for regularizing weight-sharing—including partial channels (Xu et al. 2020) and validation Hessian penalization (Zela et al. 2020)—we focus on the second question of optimization.

Geometry-Aware Gradient Algorithms

We seek to minimize the (possibly regularized) empirical risk $f(\mathbf{w}, \theta) = \frac{1}{|T|} \sum_{(x,y) \in T} \ell(h_{\mathbf{w},\theta}(x), y)$ over shared-weights $\mathbf{w} \in \mathbb{R}^d$ and architecture parameters $\theta \in \Theta$. Assuming we have noisy gradients of f w.r.t. $\mathbf{w}$ or θ at any point $(\mathbf{w}, \theta) \in \mathbb{R}^d \times \Theta$—i.e. $\tilde{\nabla}_{\mathbf{w}} f(\mathbf{w}, \theta)$ or $\tilde{\nabla}_{\theta} f(\mathbf{w}, \theta)$ satisfying $\mathbb{E}\tilde{\nabla}_{\mathbf{w}} f(\mathbf{w}, \theta) = \nabla_{\mathbf{w}} f(\mathbf{w}, \theta)$ or $\mathbb{E}\tilde{\nabla}_{\theta} f(\mathbf{w}, \theta) = \nabla_{\theta} f(\mathbf{w}, \theta)$, respectively—our goal is a point where f, or at least its gradient, is small, while taking as few gradients as possible. Our main complication is that architecture parameters lie in a constrained, non-Euclidean domain Θ. Most search spaces $\mathcal{A}$ are product sets of categorical decisions—which operation $o \in O$ to use at edge $e \in E$—so the natural relaxation is a product of $|E|$ $|O|$-simplices. However, NAS methods often re-parameterize Θ to be unconstrained using a softmax and then SGD or Adam (Kingma and Ba 2015). Is there a better parameterization-algorithm co-design? We consider a *geometry-aware* approach that uses mirror descent to design NAS methods with better properties depending on the domain; a key desirable property is to return *sparse* architectural parameters to reduce loss from post-search discretization.

Background on Mirror Descent

Mirror descent has many formulations (Nemirovski and Yudin 1983; Beck and Teboulle 2003; Shalev-Shwartz 2011); the *proximal* starts by noting that, in the unconstrained case, an SGD update at a point $\theta \in \Theta = \mathbb{R}^k$ given gradient estimate $\tilde{\nabla} f(\theta)$ with step-size $\eta > 0$ is equivalent to

$$\theta - \eta \tilde{\nabla} f(\theta) \;=\; \operatorname*{arg\,min}_{\mathbf{u} \in \mathbb{R}^k} \; \eta \tilde{\nabla} f(\theta) \cdot \mathbf{u} \;+\; \frac{1}{2} \|\mathbf{u} - \theta\|_2^2 \tag{4}$$

Here the first term aligns the output with the gradient while the second (proximal) term regularizes for closeness to the previous point as measured by the Euclidean distance. While the SGD update has been found to work well for unconstrained high-dimensional optimization, e.g. deep nets, this choice of proximal regularization may be sub-optimal over a constrained space with sparse solutions. The canonical such setting is optimization over the unit simplex, i.e. when $\Theta = \{\theta \in [0, 1]^k : \|\theta\|_1 = 1\}$. Replacing the ℓ_2-regularizer in Equation 4 by the relative entropy $\mathbf{u} \cdot (\log \mathbf{u} - \log \theta)$, i.e. the KL-divergence, yields the exponentiated gradient (EG) update ($\odot$ denotes element-wise product):

$$\begin{aligned} &\theta \odot \exp(-\eta \tilde{\nabla} f(\theta)) \\ &\quad \propto \quad \operatorname*{arg\,min}_{\mathbf{u} \in \Theta} \; \eta \tilde{\nabla} f(\theta) \cdot \mathbf{u} \;+\; \mathbf{u} \cdot (\log \mathbf{u} - \log \theta) \end{aligned} \tag{5}$$

Algorithm 1: Block-stochastic mirror descent optimization of a function $f : \mathbb{R}^d \times \Theta \mapsto \mathbb{R}$.

Input: initialization $(\mathbf{w}^{(1)}, \theta^{(1)}) \in \mathbb{R}^d \times \Theta$, strongly-convex DGF $\phi : \Theta \mapsto \mathbb{R}$, number of iterations $T \geq 1$, step-size $\eta > 0$
for iteration $t = 1, \ldots, T$ **do**
  sample $b_t \sim \text{Unif}\{\mathbf{w}, \theta\}$ // randomly select update block
  if block $b_t = \mathbf{w}$ **then**
    $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \tilde{\nabla}_{\mathbf{w}} f(\mathbf{w}^{(t)}, \theta^{(t)})$ // SGD update to shared weights
    $\theta^{(t+1)} \leftarrow \theta^{(t)}$ // no update to architecture params
  else
    $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)}$ // no update to shared weights
    $\theta^{(t+1)} \leftarrow \arg\min_{\mathbf{u} \in \Theta} \quad \eta \tilde{\nabla}_{\theta} f(\mathbf{w}^{(t)}, \theta^{(t)}) \cdot \mathbf{u} \; + \; D_{\phi}(\mathbf{u} || \theta^{(t)})$ // update architecture params
Output: $(\mathbf{w}^{(r)}, \theta^{(r)})$ for $r \sim \text{Unif}\{1, \ldots, T\}$ // return random iterate

Note that the full EG update is obtained by ℓ_1-normalizing the l.h.s. It is well-known that EG over the k-dimensional simplex requires only $\mathcal{O}(\log k)/\varepsilon^2$ iterations to achieve a function value ε-away from optimal (Beck and Teboulle 2003), compared to the $O(k/\varepsilon^2)$ guarantee of gradient descent. This nearly dimension-independent iteration complexity is achieved by choosing a regularizer—the KL divergence—well-suited to the underlying geometry—the simplex. More generally, mirror descent is specified by a distance-generating function (DGF) ϕ that is strongly-convex w.r.t. some norm. ϕ induces a *Bregman divergence* $D_{\phi}(\mathbf{u}||\mathbf{v}) = \phi(\mathbf{u}) - \phi(\mathbf{v}) - \nabla\phi(\mathbf{v}) \cdot (\mathbf{u} - \mathbf{v})$ (Bregman 1967), a notion of distance on Θ that acts as a regularizer in the mirror descent update:

$$\arg\min_{\mathbf{u} \in \Theta} \quad \eta \tilde{\nabla} f(\theta) \cdot \mathbf{u} \; + \; D_{\phi}(\mathbf{u}||\theta) \tag{6}$$

For example, to recover SGD (4) we set $\phi(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|_2^2$, which is strongly-convex w.r.t. the Euclidean norm, while EG (5) is recovered by setting $\phi(\mathbf{u}) = \mathbf{u} \cdot \log \mathbf{u}$, strongly-convex w.r.t. the ℓ_1-norm.

Block-Stochastic Mirror Descent

In the previous section we saw how mirror descent can perform better over certain geometries such as the simplex. However, in weight-sharing we are interested in optimizing over a hybrid geometry containing both the shared weights in an unconstrained Euclidean space and the architecture parameters in a non-Euclidean domain. Thus we focus on optimization over two blocks: shared weights $\mathbf{w} \in \mathbb{R}^d$ and architecture parameters $\theta \in \Theta$, the latter associated with a DGF ϕ that is strongly-convex w.r.t. some norm $\|\cdot\|$. In NAS a common approach is to perform alternating gradient steps on each domain; for example, both ENAS (Pham et al. 2018) and first-order DARTS (Liu, Simonyan, and Yang 2019) alternate between SGD on the shared weights and Adam on architecture parameters. This approach is encapsulated in the block-stochastic algorithm described in Algorithm 1, which at each step chooses one block at random to update using mirror descent (recall that SGD is a variant) and after T steps returns a random iterate. Algorithm 1 generalizes the single-level variant of both ENAS and first-order DARTS if SGD is used to update θ instead of Adam, with some mild caveats: in practice blocks are picked cyclically and the algorithm returns the last iterate, not a a random one. To analyze the convergence of Algorithm 1 we first state some regularity assumptions on the function:

Assumption 1. *Suppose ϕ is strongly-convex w.r.t. some norm $\|\cdot\|$ on a convex set Θ and the objective function $f : \mathbb{R}^d \times \Theta \mapsto [0, \infty)$ satisfies the following:*

1. **γ-relatively-weak-convexity:** *$f(\mathbf{w}, \theta) + \gamma\phi(\theta)$ is convex on $\mathbb{R}^d \times \Theta$ for some $\gamma > 0$.*
2. **gradient bound:** *$\mathbb{E}\|\tilde{\nabla}_{\mathbf{w}} f(\mathbf{w}, \theta)\|_2^2 \leq G_{\mathbf{w}}^2$ and $\mathbb{E}\|\tilde{\nabla}_{\theta} f(\mathbf{w}, \theta)\|_*^2 \leq G_{\theta}^2$ for some $G_{\mathbf{w}}, G_{\theta} \geq 0$.*

The second assumption is a standard bound on the gradient norm while the first is a generalization of smoothness that allows all smooth and some non-smooth non-convex functions (Zhang and He 2018).

Our aim will be to show (first-order) ε-stationary-point convergence of Algorithm 1, a standard metric indicating that it has reached a point with no feasible descent direction, up to error ε; for example, in the unconstrained Euclidean case an ε-stationary-point is simply one where the gradient has squared-norm $\leq \varepsilon$. The number of steps required to obtain such a point thus measures how fast a first-order method terminates. Stationarity is also significant as a necessary condition for optimality.

In our case Θ may be constrained and so the gradient may never be small, thus necessitating a measure other than gradient norm. We use *Bregman stationarity* (Zhang and He 2018), which measures stationary at a point $(\mathbf{w}, \theta)$ using the Bregman divergence between the point and its proximal map $\text{prox}_{\lambda}(\mathbf{w}, \theta) = \arg\min_{\mathbf{u} \in \mathbb{R}^d \times \Theta} \lambda f(\mathbf{u}) + D_{\ell_2, \phi}(\mathbf{u}||\mathbf{w}, \theta)$ for some $\lambda > 0$:

$$\Delta_{\lambda}(\mathbf{w}, \theta) = \frac{D_{\ell_2,\phi}(\mathbf{w}, \theta || \text{prox}_{\lambda}(\mathbf{w}, \theta)) + D_{\ell_2,\phi}(\text{prox}_{\lambda}(\mathbf{w}, \theta) || \mathbf{w}, \theta)}{\lambda^2} \tag{7}$$

Here $\lambda = \frac{1}{2\gamma}$ and the Bregman divergence $D_{\ell_2,\phi}$ is that of the DGF $\frac{1}{2}\|\mathbf{w}\|_2^2 + \phi(\theta)$ that encodes the geometry of the joint optimization domain over $\mathbf{w} \in \mathbb{R}^d$ and θ; note that the dependence of the stationarity measure on γ is standard (Dang and Lan 2015; Zhang and He 2018).

To understand why reaching a point $(\mathbf{w}, \theta)$ with small Bregman stationarity is a reasonable goal, note that the proximal operator prox_{λ} has the property that its fixed points, i.e.

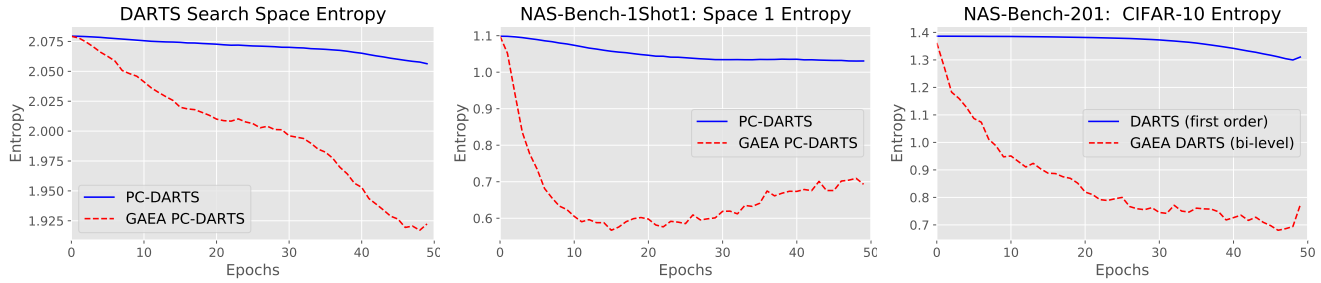

Figure 1: **Sparsity:** Evolution over search phase epochs of the average entropy of the operation-weights for GAEA and approaches it modifies when run on the DARTS search space (left), NAS-Bench-1Shot1 Search Space 1 (middle), and NASBench-201 on CIFAR-10 (right). GAEA reduces entropy much more quickly, allowing it to quickly obtain sparse architecture weights. This leads to both faster convergence to a single architecture and a lower loss when pruning at the end of search.

those satisfying $(\mathbf{w}, \theta) = \text{prox}_\lambda(\mathbf{w}, \theta)$, correspond to points where f has no feasible descent direction. Thus measuring how close $(\mathbf{w}, \theta)$ is to being a fixed point of prox_λ—as is done using the Bregman divergence in (7)—is a good measure of how far away the point is from being a stationary point of f. Finally, note that if f is smooth, ϕ is Euclidean, and Θ is unconstrained—i.e. if we are running SGD over architecture parameters as well—then $\Delta_{\frac{1}{2\gamma}} \leq \varepsilon$ implies an $O(\varepsilon)$-bound on the squared gradient norm, recovering the standard definition of ε-stationarity. More intuition on proximal operators can be found in Parikh and Boyd (2013), while further details on Bregman stationarity and how it relates to notions of convergence can be found in Zhang and He (2018).

The following result shows that Algorithm 1 needs polynomially many iterations to finds a point $(\mathbf{w}, \theta)$ with ε-small Bregman stationarity in-expectation:

Theorem 1. *Let $F = f(\mathbf{w}^{(1)}, \theta^{(1)})$ be the value of f at initialization. Under Assumption 1, if we run Algorithm 1 for $T = \frac{16\gamma F}{\varepsilon^2}(G_{\mathbf{w}}^2 + G_\theta^2)$ iterations with step-size $\eta = \sqrt{\frac{4F}{\gamma(G_{\mathbf{w}}^2+G_\theta^2)T}}$ then $\mathbb{E}\Delta_{\frac{1}{2\gamma}}(\mathbf{w}^{(r)}, \theta^{(r)}) \leq \varepsilon$. Here the expectation is over the randomness of the algorithm and gradients.*

The proof in the appendix follows from single-block analysis (Zhang and He 2018) and in fact holds for the general case of any number of blocks associated to any set of strongly-convex DGFs. Although there are prior results for the multi-block case (Dang and Lan 2015), they do not hold for nonsmooth Bregman divergences such as the KL divergence needed for exponentiated gradient.

Thus Algorithm 1 returns an ε-stationary-point given $T = \mathcal{O}(G_{\mathbf{w}}^2 + G_\theta^2)/\varepsilon^2$ iterations, where $G_{\mathbf{w}}^2$ bounds the squared ℓ_2-norm of the shared-weights gradient $\tilde{\nabla}_{\mathbf{w}}$ and G_θ^2 bounds the squared magnitude of the architecture gradient $\tilde{\nabla}_\theta$, as measured by the *dual norm* $\|\cdot\|_*$ of $\|\cdot\|$. Only the last term G_θ is affected by our choice of DGF ϕ. The DGF of SGD is strongly-convex w.r.t. the ℓ_2-norm, which is its own dual, so $G_{\mathbf{w}}^2$ is defined via ℓ_2. However, for EG the DGF $\phi(\mathbf{u}) = \mathbf{u} \cdot \log \mathbf{u}$ is strongly-convex w.r.t. the ℓ_1-norm, whose dual is ℓ_∞. Since the ℓ_2-norm of a k-dimensional vector can be $\sqrt{k}$ times its ℓ_∞-norm, picking this DGF can lead to better bound on G_θ and thus on the number of iterations.

A Geometry-Aware Exponentiated Algorithm

Equipped with these single-level guarantees, we turn to designing methods that can in-principle be applied to both the single-level and bilevel objectives, seeking parameterizations and algorithms that converge quickly and encourage favorable properties; in particular, we focus on returning architecture parameters that are *sparse* to reduce loss due to post-search discretization. EG is often considered to converge quickly to sparse solutions over the simplex (Bradley and Bagnell 2008; Bubeck 2019), which makes it a natural choice for the architecture update. We thus propose **GAEA**, a **Geometry-Aware Exponentiated Algorithm** in which operation weights on each edge are constrained to the simplex and trained using EG; as in DARTS, the shared weights $\mathbf{w}$ are trained using SGD. GAEA can be used as a simple, principled modification to the many NAS methods that treat architecture parameters $\theta \in \Theta = \mathbb{R}^{|E|\times|O|}$ as real-valued "logits" to be passed through a softmax to obtain mixture weights or probabilities for simplices over the operations O. Such methods include DARTS, PC-DARTS (Xu et al. 2020), and GDAS (Dong and Yang 2019). To apply GAEA, first re-parameterize Θ to be the product set of $|E|$ simplices, each associated to an edge $(i,j) \in E$; thus $\theta_{i,j,o}$ corresponds directly to the weight or probability of operation $o \in O$ for edge (i,j), not a logit. Then, given a stochastic gradient $\tilde{\nabla}_\theta f(\mathbf{w}^{(t)}, \theta^{(t)})$ and step-size $\eta > 0$, replace the architecture update by EG:

$$\tilde{\theta}^{(t+1)} \leftarrow \theta^{(t)} \odot \exp\left(-\eta \tilde{\nabla}_\theta f(\mathbf{w}^{(t)}, \theta^{(t)})\right) \quad \text{multiplicative update}$$

$$\theta_{i,j,o}^{(t+1)} \leftarrow \frac{\tilde{\theta}_{i,j,o}^{(t+1)}}{\sum_{o' \in O} \tilde{\theta}_{i,j,o'}^{(t+1)}} \;\; \forall\, o \in O,\; \forall\, (i,j) \in E \quad \text{simplex projection} \tag{8}$$

These two simple modifications, *re-parameterization* and *exponentiation*, suffice to obtain state-of-the-art results on several NAS benchmarks. Note that to obtain a bilevel algorithm we simply replace the gradient w.r.t. θ of the training loss with that of the validation loss.

GAEA is equivalent to Algorithm 1 with $\phi(\theta) = \sum_{(i,j)\in E} \sum_{o \in O} \theta_{i,j,o} \log \theta_{i,j,o}$, which is strongly-convex w.r.t. $\|\cdot\|_1/\sqrt{|E|}$ over the product of $|E|$ $|O|$-simplices.

Table 1: **DARTS:** Comparison with SOTA NAS methods on the DARTS search space, plus three results on different search spaces with a similar number of parameters reported at the top for comparison. All evaluations and reported performances of models found on the DARTS search space use similar training routines; this includes auxiliary towers and cutout but no other modifications, e.g. label smoothing (Müller, Kornblith, and Hinton 2019), AutoAugment (Cubuk et al. 2019), Swish (Ramachandran, Zoph, and Le 2017), Squeeze & Excite (Hu, Shen, and Sun 2018), etc. The specific training procedure we use is that of PC-DARTS, which differs slightly from the DARTS routine by a small change to the drop-path probability; PDARTS tunes both this and batch-size. Our results are averaged over 10 random seeds. Search cost is hardware-dependent; we used Tesla V100 GPUs. For more details see the appendix.

	CIFAR-10 Error		**Search Cost**	**ImageNet Error**		**Search Cost**	
Search Method (source)	Best	Average	(GPU Days)	top-1	top-5	(GPU Days)	method
NASNet-A* (Zoph et al. 2018)	-	2.65	2000	26.0	8.4	1800	RL
AmoebaNet-B* (Real et al. 2019)	-	2.55 ± 0.05	3150	24.3	7.6	3150	evolution
ProxylessNAS* (2019)	2.08	-	4	24.9	7.5	8.3	gradient (WS)
ENAS (Pham et al. 2018)	2.89	-	0.5	-	-	-	RL (WS)
RS-WS† (Li and Talwalkar 2019)	2.71	2.85 ± 0.08	0.7	-	-	-	random (WS)
ASNG (Akimoto et al. 2019)	-	2.83 ± 0.14	0.1	-	-	-	gradient (WS)
SNAS (Xie et al. 2019)	-	2.85 ± 0.02	1.5	27.3	9.2	1.5	gradient (WS)
DARTS (1st)† (2019)	-	3.00 ± 0.14	0.4	-	-	-	gradient (WS)
DARTS (2nd)† (2019)	-	2.76 ± 0.09	1	26.7	8.7	4.0	gradient (WS)
PDARTS (Chen et al. 2019)	2.50	-	0.3	24.4	7.4	0.3	gradient (WS)
PC-DARTS† (Xu et al. 2020)	-	2.57 ± 0.07	0.1	24.2	7.3	3.8	gradient (WS)
GAEA PC-DARTS† (ours)	2.39	2.50 ± 0.06	0.1	24.0	7.3	3.8	gradient (WS)
PC-DARTS† (Xu et al. 2020)	(search on CIFAR-10, train on ImageNet)			25.1	7.8	0.1	gradient (WS)
GAEA PC-DARTS† (ours)	(search on CIFAR-10, train on ImageNet)			24.3	7.3	0.1	gradient (WS)

* Search space/backbone differ from the DARTS setting; we show results for networks with a comparable number of parameters.

† For fair comparison to other work, we show the search cost for training the shared-weights network with a single initialization.

The dual is $\sqrt{|E|}\|\cdot\|_\infty$, so if $G_{\mathbf{w}}$ bounds the shared-weights gradient and we have an entry-wise bound on the architecture gradient then GAEA reach ε-stationarity in $\mathcal{O}(G_{\mathbf{w}}^2 + |E|)/\varepsilon^2$ iterations. This can be up to a factor $|O|$ improvement over SGD, either over the simplex or the logit space. In addition, GAEA encourages sparsity in the architecture weights by using a multiplicative update over simplices and not an additive update over $\mathbb{R}^{|E|\times|O|}$. Obtaining sparse architecture parameters is critical for good performance, both for the mixture relaxation, where it alleviates the effect of discretization on the validation loss, and for the stochastic relaxation, where it reduces noise when sampling architectures.

Empirical Results using GAEA

We evaluate GAEA on three different computer vision benchmarks: the large and heavily studied search space from DARTS (Liu, Simonyan, and Yang 2019) and two smaller oracle evaluation benchmarks, NAS-Bench-1Shot1 (Zela et al. 2020), and NAS-Bench-201 (Dong and Yang 2020). NAS-Bench-1Shot1 differs from the others by applying operations per node instead of per edge, while NAS-Bench-201 differs by not requiring edge-pruning. Since GAEA can modify a variety of methods, e.g. DARTS, PC-DARTS (Xu et al. 2020), and GDAS (Dong and Yang 2019), on each benchmark we start by evaluating the GAEA variant of the current best method on that benchmark. We show that despite the diversity of search spaces, GAEA improves upon this state-of-the-art across all three. Note that we use the same step-size for GAEA variants of DARTS/PC-DARTS and do not require weight-decay on architecture parameters. We defer experimental details and hyperparameter settings to the appendix and release all code, hyperparameters, and random seeds for reproducibility.

Convergence and Sparsity of GAEA

We first examine the impact of GAEA on convergence and sparsity. Figure 1 shows the entropy of the operation weights averaged across nodes for a GAEA-variant and its base method across the three benchmarks, demonstrating that it decreases much faster for GAEA-modified approaches. This validates our expectation that GAEA encourages sparse architecture parameters, which should alleviate the mismatch between the continuously relaxed architecture parameters and the discrete architecture returned. Indeed, we find that post-search discretization on the DARTS search space causes the validation accuracy of the PC-DARTS supernet to drop from 72.17% to 15.27%, while for GAEA PC-DARTS the drop is only 75.07% to 33.23%; note that this is shared-weights accuracy, obtained *without* retraining the final network. The numbers demonstrate that GAEA both (1) achieves better supernet optimization of the weight-sharing objective and (2) suffers less due to discretization.

GAEA on the DARTS Search Space

Here we evaluate GAEA on the task of designing CNN cells for CIFAR-10 (Krizhevksy 2009) and ImageNet (Russakovsky et al. 2015) by using it to modify PC-DARTS (Xu et al. 2020), the current state-of-the-art method. We follow

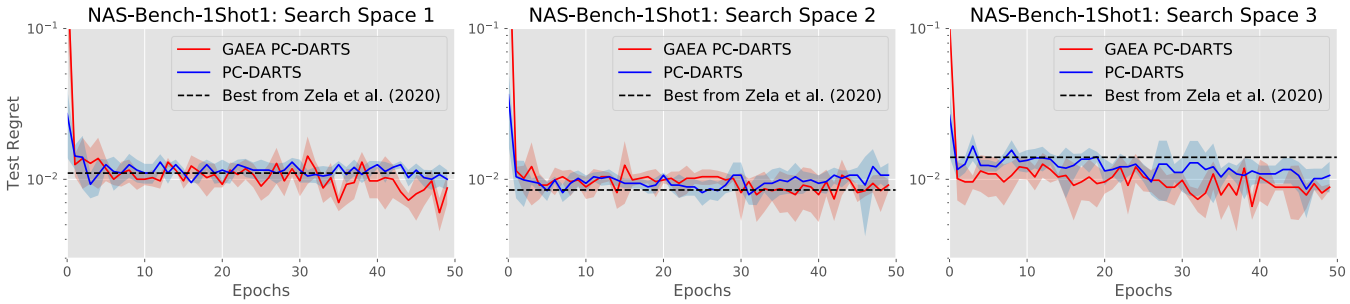

Figure 2: **NAS-Bench-1Shot1:** Online comparison of PC-DARTS and GAEA PC-DARTS in terms of the test regret at each epoch of shared-weights training, i.e. the difference between the ground truth test error of the proposed architecture and that of the best architecture in the search space. The dark lines indicate the mean of four random trials and the light colored bands $\pm$ one standard deviation. The dashed line is the final regret of the best weight-sharing method according to Zela, Siems, and Hutter (2020); note that in our reproduction PC-DARTS performed better than their evaluation on spaces 1 and 3.

the same three stage process used by both DARTS and RS-WS for search and evaluation. Table 1 displays results on both datasets and demonstrates that GAEA's parameterization and optimization scheme improves upon PC-DARTS. In fact, GAEA PC-DARTS outperforms all search methods except ProxylessNAS, which uses 1.5 times as many parameters on a different search space. Thus we improve the state-of-the-art on the DARTS search space. To meet a higher bar for reproducibility on CIFAR-10, in the appendix we report "broad reproducibility" (Li and Talwalkar 2019) by repeating our pipeline with new seeds. While GAEA PC-DARTS consistently finds good networks when selecting the best of four independent trials, multiple trials are required due to sensitivity to initialization, as is true for many approaches (Liu, Simonyan, and Yang 2019; Xu et al. 2020).

On ImageNet, we follow Xu et al. (2020) by using subsamples containing 10% and 2.5% of the training images from ILSVRC-2012 (Russakovsky et al. 2015) as training and validation sets, respectively. We fix architecture parameters for the first 35 epochs, then run GAEA PC-DARTS with step-size 0.1. All other hyperparameters match those of Xu et al. (2020). Table 1 shows the final performance of both the architecture found by GAEA PC-DARTS on CIFAR-10 and the one found directly on ImageNet when trained from scratch for 250 epochs using the same settings as Xu et al. (2020). GAEA PC-DARTS achieves a top-1 test error of 24.0%, which is state-of-the-art performance in the mobile setting when excluding additional training modifications, e.g. those in the caption. Additionally, the architecture found by GAEA PC-DARTS for CIFAR-10 and transferred achieves a test error of 24.2%, comparable to the 24.2% error of the one found by PC-DARTS directly on ImageNet. Top architectures found by GAEA PC-DARTS are depicted in the appendix.

GAEA on NAS-Bench-1Shot1

NAS-Bench-1Shot1 (Zela et al. 2020) is a subset of NAS-Bench-101 (Ying et al. 2019) that allows benchmarking weight-sharing methods on three search spaces over CIFAR-10 that differ in the number of nodes considered and the number of input edges per node. Of the weight-sharing methods benchmarked by Zela et al. (2020), we found that PC-DARTS achieves the best performance on 2 of 3 search spaces, so we again evaluate GAEA PC-DARTS here. Figure 2 shows that GAEA PC-DARTS consistently finds better architectures on average than PC-DARTS and thus exceeds the performance of the best method from Zela et al. (2020) on 2 of 3 search spaces. We hypothesize that the benefits of GAEA are limited here due to the near-saturation of NAS methods. In particular, existing methods obtain within 1% test error of the top network in each space, while the latters' test errors when evaluated with different initializations are 0.37%, 0.23% and 0.19%, respectively.

GAEA on NAS-Bench-201

NAS-Bench-201 has one search space on three datasets—CIFAR-10, CIFAR-100, and ImageNet-16-120—that includes 4-node architectures with an operation from $O = \{$none, skip connect, 1x1 convolution, 3x3 convolution, 3x3 avg pool$\}$ on each edge, yielding 15625 possible networks. Table 2 reports a subset of results from Dong and Yang (2020) alongside GAEA approaches, showing that GDAS is the best previous weight-sharing method. Our reproduced results for GDAS are slightly worse than published ones but confirm its position. We evaluate GAEA GDAS and find that it achieves better results on CIFAR-100 than our reproduced runs and similar results on the other two.

Since we are interested in improving upon the reported results, we also investigate the performance of GAEA applied to first-order DARTS. We evaluate GAEA DARTS with both single-level (ERM) and bilevel optimization; recall that in the latter case we optimize architecture parameters w.r.t. the validation loss and the shared weights w.r.t. the training loss, whereas in ERM there is no data split. GAEA DARTS (ERM) achieves state-of-the-art performance on all three datasets, exceeding the test accuracy of both weight-sharing and traditional hyperparameter tuning by a wide margin. GAEA DARTS (bilevel) performs worse but still exceeds all other methods on CIFAR-100 and ImageNet-16-120. The result thus also confirms the relevance of studying the single-level case to understand NAS; notably, the DARTS (ERM) baseline also improves substantially upon DARTS (bilevel) baseline.

Table 2: **NAS-Bench-201:** Results are separated into those for weight-sharing methods that were reported by Dong and Yang (2020) (top), our reproductions and GAEA-modifications of weight-sharing methods (middle), and those for traditional hyperparameter optimization methods that were reported by Dong and Yang (2020) (bottom).

	Search (seconds)	CIFAR-10 (test)	CIFAR-100 (test)	ImageNet-16-120 (test)
RSPS	7587	87.66 ± 1.69	58.33 ± 4.34	31.14 ± 3.88
DARTS	35781	54.30 ± 0.00	15.61 ± 0.00	16.32 ± 0.00
SETN	34139	87.64 ± 0.00	59.05 ± 0.24	32.52 ± 0.21
GDAS	31609	93.61 ± 0.09	70.70 ± 0.30	41.71 ± 0.98
GDAS (reproduced)	27923*	93.52 ± 0.15	67.52 ± 0.15	40.91 ± 0.12
GAEA GDAS	16754*	93.55 ± 0.13	70.47 ± 0.47	40.91 ± 0.12
DARTS (bilevel)	10683*	54.30 ± 0.00	15.32 ± 0.00	28.96 ± 10.22
GAEA DARTS (bilevel)	7930*	91.63 ± 2.57	71.87 ± 0.57	45.69 ± 0.56
DARTS (ERM)	18112*	84.39 ± 3.82	51.26 ± 6.14	31.35 ± 7.46
GAEA DARTS (ERM)	9061*	$\mathbf{94.10 \pm 0.29}$	$\mathbf{73.43 \pm 0.13}$	$\mathbf{46.36 \pm 0.00}$
REA	N/A	93.92 ± 0.30	71.84 ± 0.99	45.54 ± 1.03
RS	N/A	93.70 ± 0.36	71.04 ± 1.08	44.57 ± 1.25
REINFORCE	N/A	93.85 ± 0.37	71.71 ± 1.09	45.25 ± 1.18
BOHB	N/A	93.61 ± 0.52	70.85 ± 1.28	44.42 ± 1.49
ResNet	N/A	93.97	70.86	43.63
Optimal	N/A	94.37	73.51	47.31

* Search cost measured on NVIDIA P100 GPUs.

Conclusion

In this paper we take an optimization-based view of NAS, arguing that the design of good NAS algorithms is largely a matter of successfully optimizing and regularizing the supernet. In support of this, we develop GAEA, a simple modification of gradient-based NAS that attains state-of-the-art performance on several computer vision benchmarks while enjoying favorable speed and sparsity properties. We believe that obtaining high-performance NAS algorithms for a wide variety of applications will continue to require a similar co-design of search space parameterizations and optimization methods, and that our geometry-aware framework can help accelerate this process. In particular, most modern NAS algorithms search over products of categorical decision spaces, to which our approach is directly applicable. More generally, as the field moves towards more ambitious search spaces, e.g. full-network topologies or generalizations of operations such as convolution or attention, these developments may result in new architecture domains for which our work can inform the design of appropriate, geometry-aware optimization methods.

References

Akimoto, Y.; Shirakawa, S.; Noshinari, N.; Uchida, K.; Saito, S.; and Nishida, K. 2019. Adaptive stochastic natural gradient method for one-shot neural architecture search. In *Proceedings of the 36th International Conference on Machine Learning*.

Beck, A., and Teboulle, M. 2003. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters* 31:167–175.

Bradley, D. M., and Bagnell, J. A. 2008. Differentiable sparse coding. In *Advances in Neural Information Processing Systems*.

Bregman, L. M. 1967. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics* 7:200–217.

Bubeck, S. 2019. Five miracles of mirror descent. Lectures on Some Geometric Aspects of Randomized Online Decision Making.

Cai, H.; Zhu, L.; and Han, S. 2019. ProxylessNAS: Direct neural architecture search on target task and hardware. In *Proceedings of the 7th International Conference on Learning Representations*.

Carlucci, F. M.; Esperança, P. M.; Singh, M.; Yang, A.; Gabillon, V.; Xu, X.; Chen, Z.; and Wang, J. 2019. MANAS: Multi-agent neural architecture search. arXiv.

Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*.

Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; and Le, Q. V. 2019. AutoAugment: Learning augmentation strategies from data. In *Proceedings of the IEEE Conference on Conference on Computer Vision and Pattern Recognition*.

Dang, C. D., and Lan, G. 2015. Stochastic block mirror descent methods for nonsmooth and stochastic optimization. *SIAM Journal on Optimization* 25:856–881.

Dong, X., and Yang, Y. 2019. Searching for a robust neural architecture in four GPU hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Dong, X., and Yang, Y. 2020. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *Pro-*

ceedings of the 8th International Conference on Learning Representations.

Franceschi, L.; Frasconi, P.; Salzo, S.; Grazzi, R.; and Pontil, M. 2018. Bilevel programming for hyperparameter optimization and meta-learning. In *Proceedings of the 35th International Conference on Machine Learning*.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Krizhevksy, A. 2009. Learning multiple layers of features from tiny images. Technical report.

Laube, K. A., and Zell, A. 2019. Prune and replace NAS. In *Proceedings of the IEEE International Conference on Machine Learning and Applications*.

Li, L., and Talwalkar, A. 2019. Random search and reproducibility for neural architecture search. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Li, G.; Zhang, X.; Wang, Z.; Li, Z.; and Zhang, T. 2019. StacNAS: Towards stable and consistent differentiable neural architecture search. arXiv.

Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.-J.; Fei-Fei, L.; Yuille, A.; Huang, J.; and Murphy, K. 2018. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision*.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable architecture search. In *Proceedings of the 7th International Conference on Learning Representations*.

Müller, R.; Kornblith, S.; and Hinton, G. E. 2019. When does label smoothing help? In *Advances in Neural Information Processing Systems*.

Nayman, N.; Noy, A.; Ridnik, T.; Friedman, I.; Jin, R.; and Zelnik-Manor, L. 2019. XNAS: Neural architecture search with expert advice. In *Advances in Neural Information Processing Systems*.

Nemirovski, A., and Yudin, D. 1983. *Problem Complexity and Method Efficiency in Optimization*. Wiley.

Noy, A.; Nayman, N.; Ridnik, T.; Doveh, S.; Friedman, I.; Giryes, R.; and Zelnik-Manor, L. 2019. ASAP: Architecture search, anneal and prune. arXiv.

Parikh, N., and Boyd, S. 2013. Proximal algorithms. *Foundations and Trends in Optimization* 1(3):123–231.

Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient neural architecture search via parameter sharing. In *Proceedings of the 35th International Conference on Machine Learning*.

Pourchot, A.; Ducarouge, A.; and Sigaud, O. 2020. To share or not to share: A comprehensive appraisal of weight-sharing. arXiv.

Ramachandran, P.; Zoph, B.; and Le, Q. V. 2017. Searching for activation functions. arXiv.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision* 115(3):211–252.

Shalev-Shwartz, S. 2011. Online learning and online convex optimization. *Foundations and Trends in Machine Learning* 4(2):107–194.

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2019. SNAS: Stochastic neural architecture search. In *Proceedings of the 7th International Conference on Learning Representations*.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2020. PC-DARTS: Partial channel connections for memory-efficient architecture search. In *Proceedings of the 8th International Conference on Learning Representations*.

Yang, Z.; Wang, Y.; Chen, X.; Shi, B.; Xu, C.; Xu, C.; Tian, Q.; and Xu, C. 2020. CARS: Continuous evolution for efficient neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Yao, Q.; Xu, J.; Tu, W.-W.; and Zhu, Z. 2020. Efficient neural architecture search via proximal iterations. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*.

Ying, C.; Klein, A.; Christiansen, E.; Real, E.; Murphy, K.; and Hutter, F. 2019. NAS-Bench-101: Towards reproducible neural architecture search. In *Proceedings of the 36th International Conference on Machine Learning*.

Yu, K.; Sciuto, C.; Jaggi, M.; Musat, C.; and Salzmann, M. 2020. Evaluating the search phase of neural architecture search. In *Proceedings of the 8th International Conference on Learning Representations*.

Zela, A.; Elsken, T.; Saikia, T.; Marrakchi, Y.; Brox, T.; and Hutter, F. 2020. Understanding and robustifying differentiable architecture search. In *Proceedings of the 8th International Conference on Learning Representations*.

Zela, A.; Siems, J.; and Hutter, F. 2020. NAS-Bench-1Shot1: Benchmarking and dissecting one-shot neural architecture search. In *Proceedings of the 8th International Conference on Learning Representations*.

Zhang, S., and He, N. 2018. On the convergence rate of stochastic mirror descent for nonsmooth nonconvex optimization. arXiv.

Zhang, Y.; Lin, Z.; Jiang, J.; Zhang, Q.; Wang, Y.; Xue, H.; Zhang, C.; and Yang, Y. 2020. Deeper insights into weight sharing in neural architecture search. arXiv.

Zheng, X.; Ji, R.; Tang, L.; Zhang, B.; Liu, J.; and Tian, Q. 2019. Multinomial distribution learning for effective neural architecture search. In *Proceedings of the IEEE International Conference on Computer Vision*.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.